HPI Future SOC Lab: Proceedings 2014

Christoph Meinel, Andreas Polze, Gerhard Oswald, Rolf Strotmann, Ulrich Seibold, Bernhard Schulzki (Hrsg.)

Universität Potsdam

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

HPI Future SOC Lab:
Proceedings 2014

Christoph Meinel | Andreas Polze | Gerhard Oswald | Rolf Strotmann |
Ulrich Seibold | Bernhard Schulzki

HPI Future SOC Lab

Proceedings 2014

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Hasso-Plattner-Institut 2016
https://hpi.de/

Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam
Tel.: +49-(0)331 5509-0 7 / Fax: +49-(0)331 5509-325
E-Mail: hpi-info@hpi.de

Das Manuskript ist urheberrechtlich geschützt.

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URN urn:nbn:de:kobv:517-opus4-86271
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-86271

Contents

Prototype of an In-Memory Business Intelligence Solution for the Support of Forecasting of Energy Load Demand – Project Report –

Witold Abramowicz Wioletta Sokolowska Tymoteusz Hossa Jakub Opalka
Monika Kaczmarek
Department of Information Systems
Faculty of Informatics and Electronic Economy
Poznan University of Economics
Al. Niepodleglosci 10, 61-875 Poznan, Poland
firstname.lastname @kie.ue.poznan.pl

Abstract

This report gives an insight into activities performed in the field of forecasting of energy load demand and energy generation from renewable sources using the computational power of SAP HANA.
It presents a short overview of attempts undertaken with the aim to build an analytical solution using SAP HANA and Crystal Reports that would support business analysts working in the energy sector.
The report provides information on the project main objectives, used HPI Future SOC Lab resources, findings as well as next steps envisioned.

1. Introduction

The implementation of Smart Grids concepts enables EU countries to switch to renewable energy sources (RES), analyse energy data on a real-time basis and thus, allows for increased market interactions and the emergence of new market players. The customers are becoming not only subjects of energy provider's activities, but they may also become first-class citizens - participants in different processes.
Moreover, the increasing RES utilization (especially intermittent ones - wind, sun) and the variability in the consumption/production patterns affects the grid security and market operations, as e.g., real-time trading, billing, load balancing become more complex.
Therefore, the role of business analysts becomes more and more important as they are responsible for preparing various forecasts ans simulations, and based on the identified trends, are making decisions directly affecting the stability of the given energy sector or the entire market. A rational decision-making process as well as preparing various forecasts and simulations requires however, a large number of data as well as an adequate analytical tool.

The vast amount of data that organizations operating in the energy sector should gather, store and process, entails a set of new requirements towards the analytical solutions used by organizations, as the currently used predictive analysis tools are becoming inefficient and insufficient. These requirements have become drivers for the development of the in-memory computing paradigm [2], which enables the creation of applications running advanced queries and performing complex transactions on very large sets of data in a much faster and scalable way than the traditional solutions.

The main aim of the research performed by our team, which encompasses also the activities undertaken within this project, is to examine the analytical possibilities of the in-memory computing solution, on the example of SAP HANA, and their possible applications. In order to do that we apply SAP HANA and its components to the challenge of forecasting of the energy demand and energy generation from renewable resources.

The current project is a continuation of two projects *Quasi Real-Time Individual Customer Based Forecasting of Energy Load Demand Using In Memory Computing* and *Forecasting of Energy Load Demand and Energy Production from Renewable Sources using In-Memory Computing* run previously under HPI Future SOC Lab. It is to combine the already prepared building blocks together to form a working prototype of Business Intelligence solution that would support business entities in their decision making process. Business analysts working in the energy sector are faced with challenging decisions regarding planning of energy generation (own sources) and energy acquisition (on the market) in order to satisfy the predicted energy demand.

High performance computing of forecasting models in the energy sector is possible with SAP HANA, as it has been shown in the previous projects, but working with code and queries is too technical for any practical business use. Therefore, our aim was to build an environment that would wrap algorithms and programs written by our team to provide flexible tool and enable business entities to apply different forecasting models, dynamic visualizations and that they could make modifications to existing approaches by introducing new variables or changing the parameters. The undertaken attempts and achieved results are reported in this document.

The document is organized as follows. First, the project aims and the project plan is shortly presented. Then, the Future SOC Lab Resources used are pointed and some technical details are given. Next, the obtained results are shortly summarized. The document concludes with final remarks and an outlook on the future work is given.

2. Project Aims

As already mentioned the project reported in this document is a part of a cycle of undertakings aiming at building an analytical solution using SAP HANA for support of business analysts in the energy sector.

In order to make rational decisions, business analysts need accurate and up-to-date forecasts for both energy load demand and energy generation from renewable sources.

Within previous projects, the focus was placed on the analytical possibilities of SAP HANA and the possible application of PAL procedures and R integration. However, to take full advantage of the platform's computational power, not only the PAL and R aspects of SAP HANA need to be mastered, but also the understanding of SAP HANA reporting capabilities need to be achieved. Therefore, the main project aim was to create an easy to use prototype of user interface using SAP HANA and Crystal Reports to provide energy analysts with a tool that would dynamically respond to their needs.

The short synopsis of the project's main scenario presented in a condensed form follows:

Elevator Pitch style presentation of the main scenario.

FOR: An energy sector analyst
WHO: wants to estimate the forecasts of energy demand and energy generation from renewable energy for an artificially defined area
THE: prototype of an in-memory Business Intelligence solution using SAP HANA and Crystal Reports
IS A: tool that allows for preparing a pre-defined set of forecasting models and reports
UNLIKE: other commercial solutions that enable users to choose only strictly static models that are based only on a small number of variables
OUR PROJECT: enables any energy analyst, without prior programming knowledge, to calculate and compare, in a dynamic manner, various energy forecasts (energy demand and RES generation) for an artificially defined area.

The additional goal of the project was to acquire and analyse additional energy data, design and implement new forecasting methods, evaluate efficiency and performance of different computational strategies, as well as examine various reporting capabilities of SAP HANA and Crystal Reports.

One of the research hypothesis that we focused on, was that the gathered data, both on the load demand and generation, should make it possible to carry out the market simulations even for the artificially defined area.

3. Future SOC Lab Resources Used

During the project we accessed a standard physical machine with SAP HANA Instance (12) together with SAP HANA Predictive Analysis Library (PAL) and combined with Rserve for more advanced predictive analyses. Thus, we gained a possibility to aggregate and process millions of rows of data at sub-second speed.

In addition, in order to equip a business analyst with desired reporting possibilities *Crystal Reports* was utilised. Thanks to that it became possible to prepare various reports that aggregate and perform calculations on huge volumes of data returning only a small result set suited to the business analyst's needs.

Throughout the project different data was used:

- data on energy load demand obtained from a major Polish energy distributor, which we already used within the previous project run under the HPI Future SOC Lab;
- newly acquired data on electricity generation from renewable sources - namely solar data and wind data complemented with additional meteorological data.

The volume of the newly gathered solar and wind generation data is about nine million readings. This data is described by eighteen different variables, starting from generated power and time (up to 5 seconds) to the meteorological readings like solar radiation or wind speed.

As already mentioned, within the phase of our project addressed by this report, we used both SAP HANA Predictive Analysis Library [3] as well as R scripts. SAP HANA Predictive Analysis Library (PAL) was used both to build univariate time series forecasting models as well as more advanced forecasting models. As SAP HANA offers a way to incorporate the R code directly into the SQL Script [4] [1], the R script procedure was used to implement more sophisticated models, which are currently not supported by PAL.

Throughout the project we uploaded the data into SAP HANA as column tables. Then, we rearranged the data for our experiments, creating auxiliary tables, columns and views. The experiments conducted so far focused on evaluating SAP HANA capabilities for time series forecasting, data manipulation and storage, as well as organizing the code.

The following experiments were run on SAP HANA using the aforementioned resources:

- Computing the summary forecast by summing all individual time series and calculating the (single) forecast over the summarized data,
- Calculating individual forecasts for all customers, then summarizing the fore-casted values to compute the summary forecast,
- Implementing different forecasting models - among others Holt-Winters exponential smoothing (single and double), linear regression, non-linear regression (exponential and power).
- Calculating forecasts of energy generation from solar and wind farms (on individual panel/turbine level) using both R and modified PAL procedures,
- Comparing the forecasting error between different scenarios,
- Comparing different forecasting models.

In order to define the best parameters for forecasting energy generation we decided to calculate correlation between variables. We created appropriate input and output tables and decided to incorporate R code directly into the SQL Script. The example of code that was used to examine the correlation between wind the turbine power and other meteorological variables (6 of them) is presented below:

```
DROP PROCEDURE WINDCOR;
CREATE PROCEDURE WINDCOR (IN
x "WIND_CORE1", OUT y "WIND_COR")
LANGUAGE RLANG AS BEGIN
w <- cor(cbind(x[4],x[6],x[8],
x[9],x[10],x[11],x[12]))
y <- as.data.frame(w)
END;
```

The usage of the Future SOC Lab resources allowed us to implement the desired solution and fulfil the defined project goals.

4. Findings

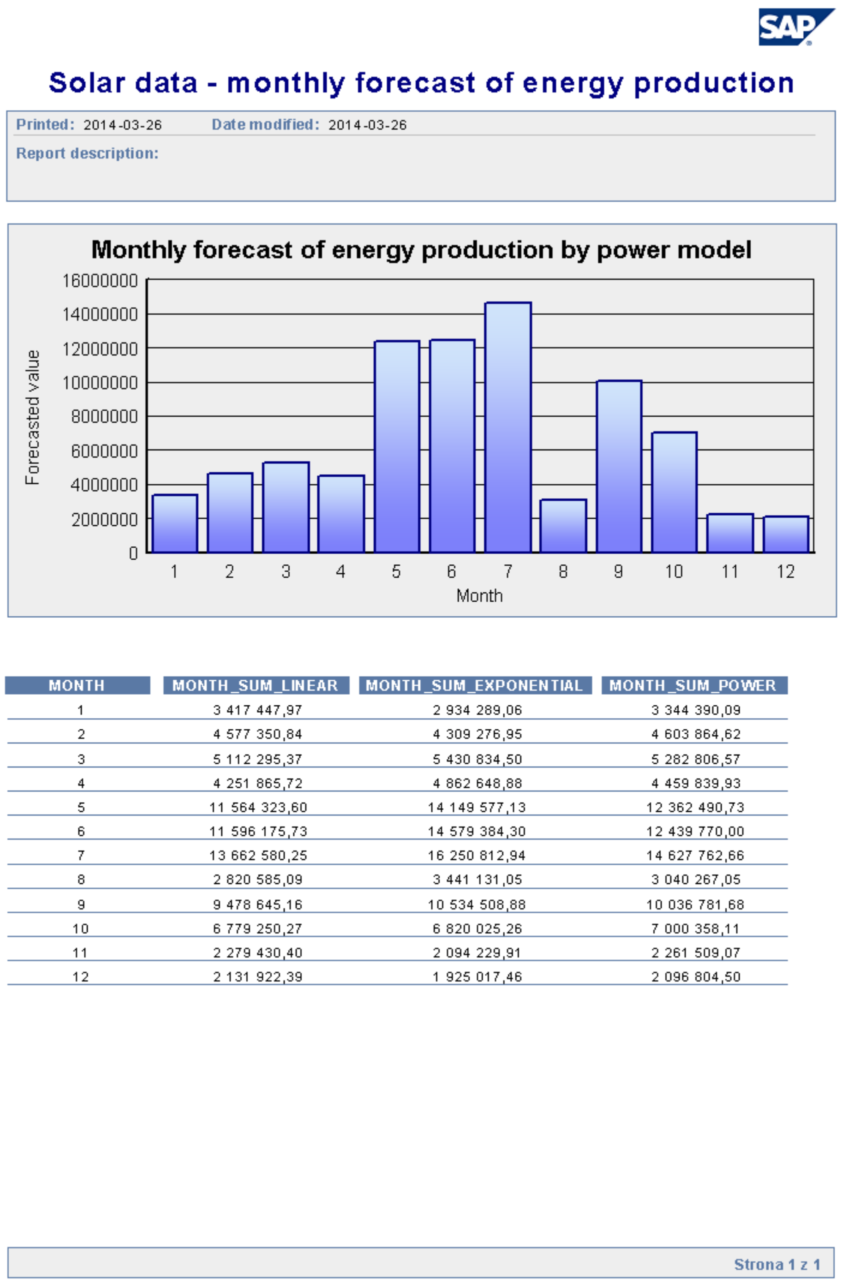
Within the project we have prepared a number of scripts and procedures aiming at creating various forecasts of energy load and energy generation within the auxiliary defined sector (see previous section). Next the integration with Crystal Reports took place.
In order to satisfy the information needs of business analysts, a number of reports that present different forecast dimensions have been prepared. Reports were created using Crystal Reports, based on the developed procedures and forecasting models from SAP HANA. Among others the following reports were prepared:

- Reports on the energy generation from the solar farm:
  - monthly and daily forecasts of energy production;
  - forecast of energy production for specific day or month;
  - hourly forecast of energy production (3 different views for 3 different models);
- Reports on energy generation from the wind farm:
  - monthly, daily and hourly forecasts of energy production (both for individual turbine and whole farm);
- Reports on forecasted load demand in the given sector:
  - monthly and daily forecasts of energy consumption for all users.

As already mentioned, for all enumerated reports' types we have prepared and implemented stored procedures using SAP HANA Modeler.
Through the use of the proposed approach the analyst is able to modify reports just by using procedure parameters, that can be changed within Crystal Reports (created and validated procedure is immediately available in Crystal Reports tool). Moreover, the analyst can modify the contents of the report (both the graphics layer and, to some extent, on the merits). He can refresh the report and introduce a new parameter value. Then, without changing the layout, there is a change in the presented data. An example of an SQL code for the procedure, which allows to create a report describing ”monthly forecast of energy production” is presented below:

```
SELECT TO_CHAR(DATE, 'MONTH_YYYY')
AS MONTH, SUM(FORECAST_LINEAR)
AS MONTH_SUM_LINEAR, SUM(FORECAST_EXPONENTIAL)
AS MONTH_SUM_EXPONENTIAL, SUM(FORECAST_POWER)
AS MONTH_SUM_POWER
FROM "TMH"."FORECAST_SOLAR_ALL"
WHERE TO_CHAR(DATE, 'YYYY_MM') LIKE :yyyy_mm
GROUP BY TO_CHAR(DATE, 'MONTH_YYYY');
```

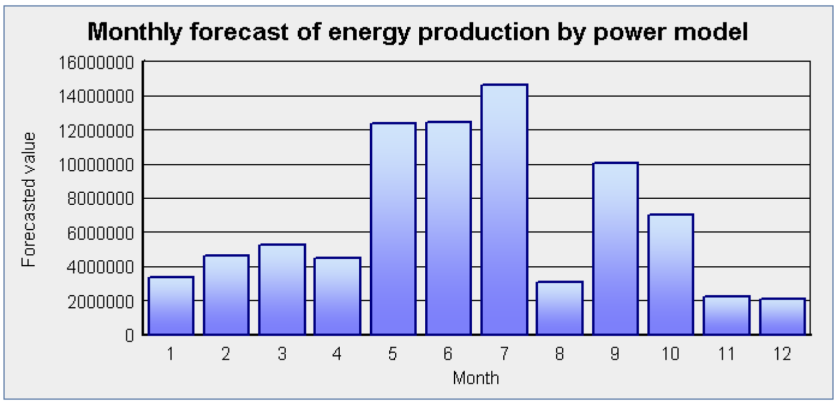
Figure 1. Report example



Solar data - monthly forecast of energy production

Printed: 2014-03-26 **Date modified:** 2014-03-26

Report description:



MONTH	MONTH_SUM_LINEAR	MONTH_SUM_EXPONENTIAL	MONTH_SUM_POWER
1	3 417 447,97	2 934 289,06	3 344 390,09
2	4 577 350,84	4 309 276,95	4 603 864,62
3	5 112 295,37	5 430 834,50	5 282 806,57
4	4 251 865,72	4 862 648,88	4 459 839,93
5	11 564 323,60	14 149 577,13	12 362 490,73
6	11 596 175,73	14 579 384,30	12 439 770,00
7	13 662 580,25	16 250 812,94	14 627 762,66
8	2 820 585,09	3 441 131,05	3 040 267,05
9	9 478 645,16	10 534 508,88	10 036 781,68
10	6 779 250,27	6 820 025,26	7 000 358,11
11	2 279 430,40	2 094 229,91	2 261 509,07
12	2 131 922,39	1 925 017,46	2 096 804,50

Strona 1 z 1

An example of a report prepared with the use of Crystal Reports and SAP HANA is presented in Figure 1. We have also performed run-time analysis. The examples of run-time for different procedures are as follows:

- Solar data - for 141 495 rows:
  - PAL Linear Regression: 17.665 seconds;
  - PAL Nonlinear regression - exponential linearization: 21.551 seconds;
  - PAL Nonlinear regression - power type linearization: 20.761 seconds;
- Load data - for 6445 rows:
  - Holt-Winters in R: summary forecast using the bottom-up approach by data and hour: 8.125 seconds;
  - Holt-Winters in R: individual forecasts for each prosumer: 15:15.625 minutes 1024*6445 rows
- Wind data - for 6448 rows:
  - Correlation using implemented R procedure: around 855 ms;

– PAL Linear Regression: 6 seconds;

– R Linear Regression: 4 seconds.

The achieved speed depends on a variety of factors like the extent to which the HANA model has been optimized, the type of queries used, and the volume of data that has been requested etc. By appropriate optimization of the above mentioned factors the obtained speed can be substantially improved, however, even the currently achieved values are much better than the efficiency of the computation of the currently used solutions in the energy sector.
SAP HANA together with Crystal Reports enabled to built a solution allowing to carry out various market simulations for the artificially defined area exhibiting very good performance and providing required information for the needs of rational decision making process!

5. Conclusions and Next Steps

Within the described project, taking advantage of the previously achieved outcomes, the dashboard like solution was developed to equip business analysts with the up-to-date prognosis allowing them to make better decisions.
With the analytical and computational experiments conducted we are now ready to extend the scope of analysis for the needs of rational decision-making by focusing not only on the structured (further development of more sophisticated forecasting models and methods using both PAL and R), but also on unstructured data and expanding the possibilities of previously created working prototype of Business Intelligence solution. The most obvious example of the unstructured data analysis, available in large quantity, is the automated analysis of various Internet portals and forums in order to identify relevant information e.g., to get to know opinions of current and future clients on the energy provider and on the provider's offer.
In parallel, we will continue our work on the analysis of the structured data, and continue the work on the forecasting models (both energy load and energy generation). The main goal is to improve the accuracy of the so far proposed solutions. The already developed models will be further extended and adjusted to form a hybrid solution allowing taking advantage of the strengths of each individual approach. In addition, in order to fully satisfy the information needs of business analysts the further manipulation (e.g., aggregation and disaggregation) of the obtained results is necessary (e.g. providing forecasts on various granularity levels in different time spans). Thus, we will continue to use PAL and R capabilities of SAP HANA, together with reporting capabilities to develop the required solution.
We want to verify whether by combining information from various data sources, both internal as well as external ones, and employing the analytical and computational power of SAP HANA both for the structured and unstructured data, business analysts will be equipped with a tool, which by providing relevant information, will allow to decrease the uncertainty connected with the decision making process.

References

[1] Y. Aragon. *Séries temporelles avec R. Méthodes et cas*. Springer, Collection Pratique R, 1st edition, 2011.

[2] H. Plattner and A. Zeier. *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, Berlin Heidelberg, 2011.

[3] SAP. SAP HANA Predictive Analysis Library (PAL) Reference. 2012.

[4] SAP. SAP HANA R Integration Guide. 2013.

Full Text processing using SAP HANA

(Author)
Jevgenij Jakunschin
University of Wismar
JevJaku@gmail.com

(Supervisor)
Prof. Dr.-Ing. Antje Düsterhöft
University of Wismar
Antje.Duesterhoeft@hs-wismar.de

Abstract

The primary goal of the project is the evaluation of No-SQL and SQL databases (SAP HANA, Oracle, HBASE), while focusing on different indexing options, full text search types, search accuracy, performance and possible semantic and information modeling options. This includes the generation of an adequate test data collection (Project Gutenberg, Twitter), functionality tests and the evaluation of syntax and semantic modeling capabilities.

1 Introduction

In today‟s environment of rapidly evolving technologies and database systems, with the boom of NO-SQL databases and the rising problem of the "big data" issue, new projects are rapidly created, merged, changed and even aborted.

This project is evaluated as part of the of the University of Wismar, that is specializing on linguistic and voice processing technology and is looking into new database approaches as possible support for the already present applications.

The goal of this project is the comparison and evaluation of SQL and No-SQL environments. The primary focus is set on full-text functionality and performance. The SAP HANA environment was chosen because of the array of full text processing methods, the in-memory database nature and a diversity of multithreading optimizations. This project evaluates multiple criteria including performance, full-text functionality, system stability and format compatibility. The test data is extracted from two different sources: a merged table of over 15000 books and a twitter extraction application.

2 Project state

The project follows the following concept:

The first step is to collecting a set of full text-test data. It is imperative to use data of different text type and length to test different indexing strategies and cause sufficient pressure on the tested databases management systems.

This project, as part of a master thesis, seeks to test out the following databases management systems:

Oracle - The Oracle database is a popular relational database system with a high variety of full-text features, a highly optimized data structures and the capability of processing "„big data"‟ information.

SAP HANA - The In-memory database SAP HANA features several full-text processing methods, both row and column store storage capabilities, multithreading optimizations and data mining and analytic modules.

HBASE - The HBASE environment is a popular No-SQL database system that is highly customizable and can provide full-text processing features by expanding it with SolR or Lucene.

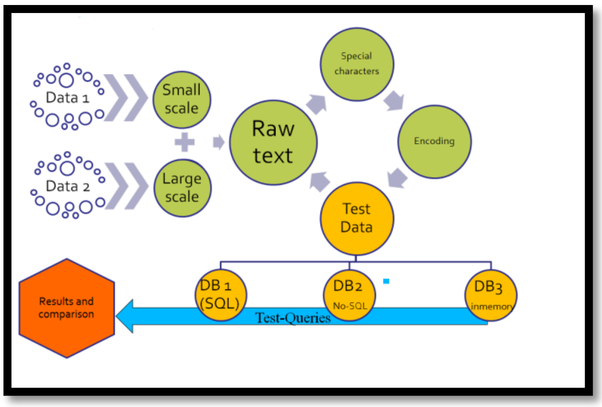


Figure 1 - Project Concept

Next the full-text files are formatted and adjusted to fit the standards for each database.

This step includes format conversion, encoding changes, special character removal, uppercase handling, insert of markers for recall/precision testing and adding meta-data rows.

The project then follows through a series of different data tests to acquire statistics on the performance and precision of different database types.

Finally the tests are evaluated, considering the different database structures, hardware dependencies and other factors and use cases are suggested for the different database type.

2.1 Input data and additional applications.

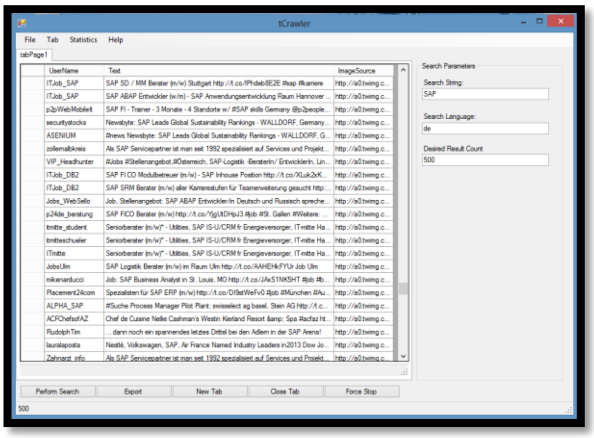
A large amount of full text data is required in order to cause sufficient measurable pressure on the databases.

A big database of books and articles (initially in *.txt format) with irregular file sizes has been prepared in order to perform such tests.

The books have been collected, from the Project Gutenberg (http://www.gutenberg.org/) database. Mostly English books in .txt format only have been selected for the tests. Overall a database of 11.531 files, with a total size of 4.34 gigabytes has been collected and prepared.

Afterwards these files have been run through a special application to remove conflicting special characters, HTML tags, adapt encoding, restrict line length and fix some other formatting problems that might be encountered during the tests. The program also merges files into Microsoft Excel Comma Separated Values (CSV) format to prepare them for the upload into the database.

The application also provides the possibility to split and merge files, in order to test the optimal table length and storage strategy.



	UserName	Text	ImageSource
	ITJob_SAP	SAP SD / MM Berater (m/w) Stuttgart http://t.co/lPhdeb8E2E #sap #karriere	http://a0.twimg.c...
	ITJob_SAP	SAP ABAP Entwickler (w/m) - SAP Anwendungsentwicklung Raum Hannover ...	http://a0.twimg.c...
	p2pWebMobileIt	SAP FI - Trainer - 3 Monate - 4 Standorte w/ #SAP skills Germany @p2people...	http://a0.twimg.c...
	securitystocks	Newsbyte: SAP Leads Global Sustainability Rankings - WALLDORF, Germany...	http://a0.twimg.c...
	ASENIUM	#news Newsbyte: SAP Leads Global Sustainability Rankings - WALLDORF, G...	http://a0.twimg.c...
	zollernalbkreis	Als SAP Servicepartner ist man seit 1992 spezialisiert auf Services und Projekt...	http://a0.twimg.c...
	VIP_Headhunter	#Jobs #Stellenangebot,#Österreich, SAP-Logistik -BeraterIn/ EntwicklerIn, Lin...	http://a0.twimg.c...
	ITJob_DB2	SAP FI CO Modulbetreuer (m/w) - SAP Inhouse Position http://t.co/XLuk2xK...	http://a0.twimg.c...
	ITJob_DB2	SAP SRM Berater (m/w) aller Karrierestufen für Teamerweiterung gesucht http:...	http://a0.twimg.c...
	Jobs_WebSello	Job, Stellenangebot: SAP ABAP Entwickler/in Deutsch und Russisch spreche...	http://a0.twimg.c...
	p24de_beratung	SAP FICO Berater (m/w) http://t.co/YjgUtDHpJ3 #job #St. Gallen #Weitere: ...	http://a0.twimg.c...
	itmitte_student	Seniorberater (m/w)* - Utilities, SAP IS-U/CRM fr Energieversorger, IT-mitte Ha...	http://a0.twimg.c...
	itmitteschueler	Seniorberater (m/w)* - Utilities, SAP IS-U/CRM fr Energieversorger, IT-mitte Ha...	http://a0.twimg.c...
	ITmitte	Seniorberater (m/w)* - Utilities, SAP IS-U/CRM fr Energieversorger, IT-mitte Ha...	http://a0.twimg.c...
	JobsUlm	SAP Logistik Berater (m/w) im Raum Ulm http://t.co/AAHEHkFYUr Job Ulm	http://a0.twimg.c...
	mikenarducci	Job: SAP Business Analyst in St. Louis, MO http://t.co/JAxS1NK5HT #job #b...	http://a0.twimg.c...
	Placement24com	Spezialisten für SAP ERP (m/w) http://t.co/DI9stWeFv0 #job #München #Au...	http://a0.twimg.c...
	ALPHA_SAP	#Suche Process Manager Pilot Plant: swisselect ag basel, Stein AG http://t.c...	http://a0.twimg.c...
	ACFChefsofAZ	Chef de Cuisine Nelle Cashman's Westin Kierland Resort &amp; Spa #acfaz ht...	http://a0.twimg.c...
	RudolphTim	... dann noch ein spannendes letztes Drittel bei den Adlern in der SAP Arena!	http://a0.twimg.c...
	lauralaposta	Nestlé, Volkswagen, SAP, Air France Named Industry Leaders in2013 Dow Jo...	http://a0.twimg.c...
	Zahnarzt_info	Als SAP Servicepartner ist man seit 1992 spezialisiert auf Services und Projekt...	http://a0.twimg.c...

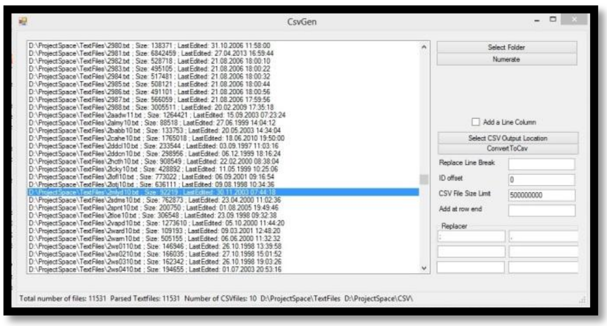
Search Parameters
Search String: SAP
Search Language: de
Desired Result Count: 500

500

Figure 2 - Twitter crawler

The project also contains an application to quickly gather full text data from twitter for special tests, such as fuzzy search efficiency, performance and precision. Initially the data from both applications is (initially) stored in a 3-column CSV file, consisting out of an ID-key value, a line-value for splitting long texts and the text field.

Additional fields for evaluations and tests can be dynamically added (line numbers, book numbers, titles, authors, twitter author…) depending on the requirements and data types.



```
D:\ProjectSpace\TextFiles\2980.txt ; Size: 138371 ; LastEdited: 31.10.2006 11:58:00
D:\ProjectSpace\TextFiles\2981.txt ; Size: 6842459 ; LastEdited: 27.04.2013 16:59:44
D:\ProjectSpace\TextFiles\2982.txt ; Size: 528718 ; LastEdited: 21.08.2006 18:00:10
D:\ProjectSpace\TextFiles\2983.txt ; Size: 495105 ; LastEdited: 21.08.2006 18:00:22
D:\ProjectSpace\TextFiles\2984.txt ; Size: 517481 ; LastEdited: 21.08.2006 18:00:32
D:\ProjectSpace\TextFiles\2985.txt ; Size: 508121 ; LastEdited: 21.08.2006 18:00:44
D:\ProjectSpace\TextFiles\2986.txt ; Size: 491101 ; LastEdited: 21.08.2006 18:00:56
D:\ProjectSpace\TextFiles\2987.txt ; Size: 566059 ; LastEdited: 21.08.2006 17:59:56
D:\ProjectSpace\TextFiles\2988.txt ; Size: 3005511 ; LastEdited: 20.02.2009 17:35:18
D:\ProjectSpace\TextFiles\2aadw11.txt ; Size: 1264421 ; LastEdited: 15.09.2003 07:23:24
D:\ProjectSpace\TextFiles\2almy10.txt ; Size: 88518 ; LastEdited: 27.06.1999 14:04:12
D:\ProjectSpace\TextFiles\2babb10.txt ; Size: 133753 ; LastEdited: 20.05.2003 14:34:04
D:\ProjectSpace\TextFiles\2cahe10.txt ; Size: 1765018 ; LastEdited: 18.06.2010 19:50:00
D:\ProjectSpace\TextFiles\2ddcl10.txt ; Size: 233544 ; LastEdited: 03.09.1997 11:03:16
D:\ProjectSpace\TextFiles\2ddcn10.txt ; Size: 298956 ; LastEdited: 06.12.1999 18:16:24
D:\ProjectSpace\TextFiles\2hcth10.txt ; Size: 908549 ; LastEdited: 22.02.2000 08:38:04
D:\ProjectSpace\TextFiles\2icky10.txt ; Size: 428892 ; LastEdited: 11.05.1999 10:25:06
D:\ProjectSpace\TextFiles\2lofl10.txt ; Size: 773022 ; LastEdited: 06.09.2001 09:16:54
D:\ProjectSpace\TextFiles\2lotj10.txt ; Size: 636111 ; LastEdited: 09.08.1998 10:34:36
D:\ProjectSpace\TextFiles\2mlyd10.txt ; Size: 92219 ; LastEdited: 30.11.2003 07:44:18
D:\ProjectSpace\TextFiles\2sdms10.txt ; Size: 762873 ; LastEdited: 23.04.2000 11:02:36
D:\ProjectSpace\TextFiles\2sprt10.txt ; Size: 200750 ; LastEdited: 01.08.2005 19:49:46
D:\ProjectSpace\TextFiles\2tioe10.txt ; Size: 306548 ; LastEdited: 23.09.1998 09:32:38
D:\ProjectSpace\TextFiles\2vapd10.txt ; Size: 1273610 ; LastEdited: 05.10.2000 11:44:20
D:\ProjectSpace\TextFiles\2ward10.txt ; Size: 109193 ; LastEdited: 09.03.2001 12:48:20
D:\ProjectSpace\TextFiles\2warn10.txt ; Size: 505155 ; LastEdited: 06.06.2000 11:32:32
D:\ProjectSpace\TextFiles\2ws0110.txt ; Size: 146946 ; LastEdited: 26.10.1998 13:39:58
D:\ProjectSpace\TextFiles\2ws0210.txt ; Size: 166035 ; LastEdited: 27.10.1998 15:01:52
D:\ProjectSpace\TextFiles\2ws0310.txt ; Size: 162342 ; LastEdited: 26.10.1998 19:03:26
D:\ProjectSpace\TextFiles\2ws0410.txt ; Size: 194655 ; LastEdited: 01.07.2003 20:53:16
```

Add a Line Column [ ]
Replace Line Break:
ID offset: 0
CSV File Size Limit: 500000000
Add at row end:
Replacer: : | ,

Total number of files: 11531 Parsed Textfiles: 11531 Number of CSVfiles: 10 D:\ProjectSpace\TextFiles D:\ProjectSpace\CSV\

Figure 3 – Gutenberg to Csv Converter

In addition, a "troublemaker" application has been designed to create, recorded edits in different files in order to test the search performance, precision and recall of the fuzzy search algorithms.

2.2 Running tests

The first step was finding all common text retrieval techniques between the three systems.

Methods exclusive to one database were also tested, but usually the critical methods are the one most database management systems support. Such methods are for instance: "„normal select/search"‟, "„full text exact search"‟, "„full text fuzzy search"‟, "„full text Boolean search"‟, "„wildcard search"‟.

Next, the test data is imported into all tested database systems. Multiple tables of the test data are created in order to apply different indexing strategies. Each full text retrieval technique is used with each table in each of the three systems.

Once the fastest pair is selected on each system, we test the selected pair with several parameters - multi threading efficiency (eg. data import), argument influence (eg. fuzzy search), and different indexing strategies

Finally the most effective methods and configurations are double-checked (with a different data set) and compared to the other systems and approaches.

A major problem that creates additional complexity is the hardware difference. Since the SAP HANA software runs on dedicated in-memory optimized hardware it‟s hard to create a direct comparison, with the much weaker Oracle and HBASE hardware, without considering the price and performance difference.

Instead this thesis focuses on relative numbers, tendencies and practical use cases.

The current state of the project is most of the tests being completed. Currently the results are being verified, compared and evaluated.

The following tests were already completed with all three databases and will be included in the presentation:

- Basic select performance
- Exact search performance
- Boolean search performance
- Wildcard search performance
- Theoretical differences
- Import speed

3 Next steps

The project now focuses on the following tasks and tests:

- Fuzzy search precision/recall
- Use Case evaluation
- Linguistic functions
- Alternative indexing strategies

Afterwards a suggestion on the optimal use cases for each different database management system will be presented and a comprehensive statistic comparison table will be created, with some supporting charts for each single system.

4 Related work

The project is not directly based on another work, but draws some base knowledge and inspiration from some other works.

The presentation "„How to Compare NoSQL Databases: Determining True Performance and Recover ability Metrics For Real-World Use Cases"‟ from the No-SQL Matters conference 2013 by Benjamin Engber"‟ compares the performance and stability of 4 No-SQL databases, while also providing insight on their backup strategies and statistics and database behavior in the case a server fails and is used as an example for a good system comparison project.

The book: "Ähnlichkeitssuche in Multimedia-Datenbanken, Retrieval, Suchalgorithmen und Anfagebehandlung"‟ by the Oldenburg publisher provides a wide variety of information, base knowledge and comparison ideas used during the evaluation of different systems.

A lot of measurement and performance retrieval techniques are based on Oracle‟s and SAP HANA‟s approaches and existing methods.

Raising the Power of Ensemble Techniques - Follow up & Extension Project

David Müller, Christoph M. Friedrich, Christoph Engels
University of Applied Sciences and Arts Dortmund, Department of Computer Science
Emil-Figge-Str. 42, D-44227 Dortmund
david.mueller@fh-dortmund.de, christoph.friedrich@fh-dortmund.de,
christoph.engels@fh-dortmund.de

Abstract

Ensemble methods (like random forests, quantile forests, gradient boosting machines and variants) have demonstrated their outstanding behavior in the domain of data mining techniques.

This project focuses on an implementation of an ensemble method on SAP HANA to combine a powerful environment with a fully developed data mining algorithm. The implemented functions are integrated into the Data Scientist Prototype, a workflow tool in HANA Studio, for building workflows with function nodes.

1 Project Idea

In the first FSOC Lab period the University of Applied Sciences and Arts Dortmund successfully addressed the topic *Data Mining on SAP HANA* with their project *Raising the power of Ensemble Techniques*. The project idea was to compare different opportunities, which enable the usage of predictive analytical techniques on SAP HANA [8].

SAP is offering the Predictive Analytical Library (PAL), which contains more than 40 well-known algorithms in the fields of classification analysis, association analysis, data preparation, outlier detection, cluster analysis, time series analysis, link prediction and others [16].

In the previous project very accurate predictions could be achieved by using PAL [10]. On the other hand performance problems for certain functions in combination with special datasets occurred, as the PAL implementation is relatively new and programmers have not taken the full potential of the HANA architecture [4]. Furthermore, no ensemble methods were part of the comprehensive selection of algorithms offered by PAL, yet [16].

On this basis a follow up project has been started in order to write and implement an ensemble method on HANA, to utilize its powerful capabilities for CPU-intensive algorithms [9].

Why Ensemble Methods?

Predictive statistical data mining has evolved further over the recent years and remains a steady field of active research. The latest research results provide new data mining methods which lead to better results in model identification and behave more robustly especially in the domain of predictive analytics. Most analytic business applications lead to improved financial outcomes directly, for instance demand prediction, fraud detection and churn prediction [1,2,7,11,12,18]. Even small improvements in prediction quality lead to enhanced financial effects. Therefore the application of new sophisticated predictive data mining techniques enable business processes to leverage hidden potentials and should be considered seriously.

Especially for classification tasks ensemble methods (like random forests) show powerful behavior [5,6,17] which includes that

- they exhibit an excellent accuracy,
- they scale up and are parallel by design,
- they are able to handle
    - thousands of variables,
    - many valued categories,
    - extensive missing values,
    - badly unbalanced data sets,
- they give an internal unbiased estimate of test set error as primitives are added to ensemble,
- they can hardly overfit,
- they provide a variable importance and
- they enable an easy approach for outlier detection.

Why SAP HANA?

SAP HANA is a “flexible, data-source-agnostic toolset […] that allows you to hold and analyze massive volumes of data in real time” [3]. It enhances data processing by sophisticated technologies like Massive Parallel Processing (MPP), in-memory computing, columnar data storage and others [3,13,14,15]. Through this project the powerful capabilities of SAP HANA shall be exploited to gain fast processing of CPU-intensive predictive calculations.

Project Goal and Strategy

The overall project idea is to implement a random forest on HANA. Therefore different languages are considered. Furthermore, the result shall be integrated into the Data Scientist Prototype, a workflow tool in SAP HANA Studio. In the Data Scientist Prototype function nodes can be dragged and dropped and sequences of comprehensive analytical functions can be created, to simplify the usage of PAL and R algorithms (see appendix 2).

The follow up & extension project consists of following milestones:

- Understand functionality and opportunities of the Data Scientist Prototype.
- Consider possible languages to implement a random forest on SAP HANA. Determine the range of functionality of Llang, a SAP internal language.
- Construct a concept for random forest and its prediction.
- Implement random forest and create a random forest training node within the Data Scientist Prototype. Implement and create a prediction node.
- Create a complete workflow, comprising:
    - data loading,
    - creating a test and training set,
    - running the random forest,
    - prediction and
    - determine accuracy of prediction.

2 Used Future SOC Lab Resources

For this project a HANA environment (HW and SW) with the latest PAL distribution is needed. For implementing functional nodes in the Data Scientist Prototype, an access to a HANA Studio with the integrated workflow tool is necessary. Therefore, a fully prepared Windows server with a pre-installed HANA Studio can be used.

3 Findings and Impacts on Project

Impacts on the project and its results are listed in this chapter, as well as the project findings.

3.1 Usage of PAL Functions

PAL functions are written in C++ and are property of SAP AG. Access to the source code cannot be granted to the project team during this project period [4]. Therefore the coding of the decision tree has to be done from scratch.

3.2 Language

Two languages can be used to build analytical nodes in the Data Scientist Prototype, R language and L language. R must be executed on a dedicated R server and therefore, data must be transferred between those servers. L on the other hand is processed directly in the core of SAP HANA and takes advantage of the HANA capabilities. Hence L is chosen as the programming language for the implementation of the random forest algorithm.

Unfortunately it is not possible to use SQL or SQLScript in combination with L yet. This would lead to significant performance advantages, as a lot of columns oriented functions are used in the random forest algorithm.

3.3 Parallelization

A random forest can be executed highly parallel and the Data Scientist Prototype offers the possibility of parallelization. This advantage can’t be taken in this project period, as the given infrastructure doesn’t contain the newest version of the Data Scientist Prototype, which is mandatory for parallelization.

3.4 Performance

For small data sets the performance times are similar to the PAL implementation. For larger data sets the PAL implementation delivers better performance results (see appendix 1).

There are three reasons for this performance issue:

- The random forest method is implemented from scratch in a new language. There are some opportunities to improve the performance of this algorithm. An option is to use pointer methods pointing on selected rows instead of building new tables, in which selected rows are inserted.
- Time intensive iterative L commands are used instead of fast SQL commands for selecting data and doing projections on tables. The usage of SQLScript would deliver better performance results.
- Parallelization can’t be used, as access to the newest version of the Data Scientist Prototype in not possible in this project period.

3.5 Prediction Accuracy

The prediction results are satisfying and the implemented algorithm runs reliably. Depending on the data set, parameters and the selection of test and training data, the prediction accuracy can either be better or worse compared to the PAL C4.5. decision tree (see appendix 1).

4 Final Results / Deliveries

4.1 Decision Tree

An implementation of a decision tree is mandatory to create a random forest. Therefore a decision tree node and a prediction node for decision trees is provided (see appendix 2).

4.2 Random Forest

The main contribution of this project is the random forest implementation. A random forest node is delivered as well as a prediction node for ensemble methods (see appendix 3).

4.3 Confusion Matrix

Beside the random forest and the decision tree node a third node is realized. This node delivers a Confusion Matrix, which evaluates the prediction accuracy (see appendix 4). The implementation is realized by using R.

5 Next steps

There are a lot of opportunities to use the project results for further improvements.

On the one hand, the whole implementation needs to be optimized. Especially the usage of pointer like data structures should be taken into account, which would improve many parts of the source code.

On the other hand more options to work on SAP HANA should be considered, to exploit the full potential of this architecture:

- The newest version of the Data Scientist Prototype is a mandatory prerequisite which includes
  - parallelism and
  - container nodes.
- The applicability of SQLScript is an optional feature, but would have big impact on the performance.
- The feasibility to work with C++ on SAP HANA would open doors to new and enriching implementations of predictive analytical functions.

Beside the performance improvement there are some options to optimize the algorithm and its prediction quality, for example different approaches for identifying the best split for numeric attributes or implementing a post pruning by regarding a validation data set.

6 Conclusion

The ensemble technique is implemented successfully in this project period and the majority of project goals are accomplished. But there are still a lot of opportunities to optimize the implementation with respect to performance and accuracy of prediction. These outcomes are basis for a follow-up project, in which especially the performance of the ensemble algorithm is to be optimized.

7 References

[1] R. E. Banfield; R.E., et. al.: “A Comparison of Decision Tree Ensemble Creation Techniques”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 1 (2007).

[2] S. Benkner, A. Arbona, G. Berti, A. Chiarini, R. Dunlop, G. Engelbrecht, A. F. Frangi, C. M. Friedrich, S. Hanser, P. Hasselmeyer, R. D. Hose, J. Iavindrasana, M. Köhler, L. Lo Iacono, G. Lonsdale, R. Meyer, B. Moore, H. Rajasekaran, P. E. Summers, A. Wöhrer und S. Wood: „@neurIST Infrastructure for Advanced Disease Management through Integration of Heterogeneous Data, Computing, and Complex Processing Services“, DOI:10.1109/TITB.2010.2049268, IEEE Transactions on Information Technology in BioMedicine, 14(6), Pages 1365 - 1377, (2010).

[3] B. Berg, P. Silvia: “SAP HANA An Introduction”, 2nd edition, GalileoPress, Boston (2013).

[4] J.-H. Böse, SAP Innovation Center Potsdam, personal communication, Aug. 2013.

[5] L. Breiman: „RF / tools – A Class of Two-eyed Algorithms“, SIAM Workshop, (2003), URL: http://www.stat.berkeley.edu/~breiman/siamtalk2003.pdf , accessed on 11.03.2014.

[6] L. Breiman: “Random Forests”, (1999), URL: http://www.stat.berkeley.edu/~breiman/randomforests-rev.pdf, accessed on 11.03.2014.

[7] C. Engels: „Basiswissen Business Intelligence.“, W3L Verlag, Witten (2009).

[8] C. Engels, C. Friedrich: „Proposal - Raising the power of Ensemble Techniques“, Proposal to summer 2013 period at the HPI Future Lab, (2013).

[9] C. Engels, C. Friedrich: „Proposal - Follow up & extension activities to *the Raising the power of Ensemble Techniques* project “, Proposal to winter 2013 period at the HPI Future Lab, (2013).

[10] C. Engels, C. Friedrich, D. Müller: „Report - Raising the power of Ensemble Techniques“, Report to summer 2013 period at the HPI Future Lab, (2013).

[11] C. Engels; W. Konen: „Adaptive Hierarchical Forecasting”. Proceedings of the IEEE-IDACCS 2007 Conference, Dortmund (2007).

[12] J. Friedman: „Computational Statistics & Data Analysis”, Volume 38, Issue 4, 28 February 2002, Pages 367–378, (2002), URL: http://dx.doi.org/10.1016/S0167-9473(01)00065-2, accessed on 11.03.2014.

[13] J. Haun, et al.: “Implementing SAP HANA”, 1st edition, Galileo Press, Boston (2013).

[14] R. Klopp: “Massively Parallel Processing on HANA”, (2013), URL: http://www.saphana.com/community/blogs/blog/2013/04/22/massively-parallel-processing-on-hana, accessed on 11.03.2014.

[15] SAP AG: “SAP HANA Developer Guide (document version: 1.0 – 27.11.2013, SPS 07)”, (2013), URL: http://help.sap.com/hana/SAP_HANA_Developer_Guide_en.pdf, accessed on 11.03.2014.

[16] SAP AG: “What´s New? SAP HANA SPS 07 - SAP HANA Application Function Library (AFL)”, (2013), URL: http://www.saphana.com/servlet/JiveServlet/download/4267-1-12720/What%C2%B4s%20New%20SAP%20HANA%20SPS%2007%20-%20AFL%20Predictive.pdf, accessed on 11.3.2014

[17] G. Seni, J. Elder: “Ensemble Methods in Data Mining”, Morgan & Claypool, San Rafael, California (2010).

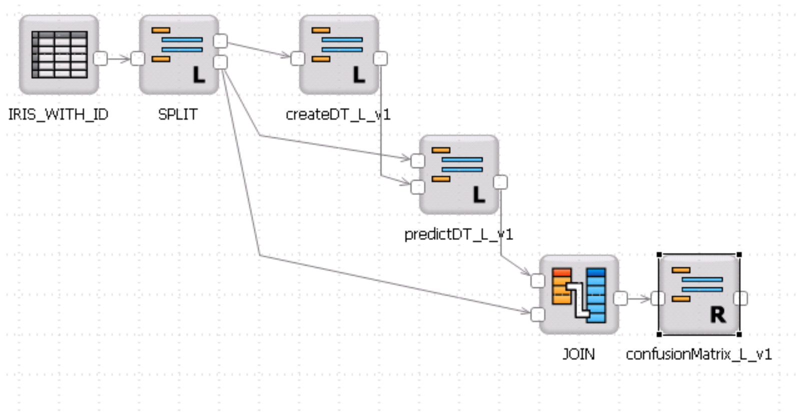
[18] G. Üstünkar; S. Özögür-Akyüz; G. W. Weber; C. M. Friedrich und Y. A. Son, „Selection of Representative SNP Sets for Genome-Wide Association Studies: A Metaheuristic Approach“, DOI:10.1007/s11590-011-0419-7, Optimization Letters, Volume 6(6), Seite 1207-1218, (2012)

Appendix:

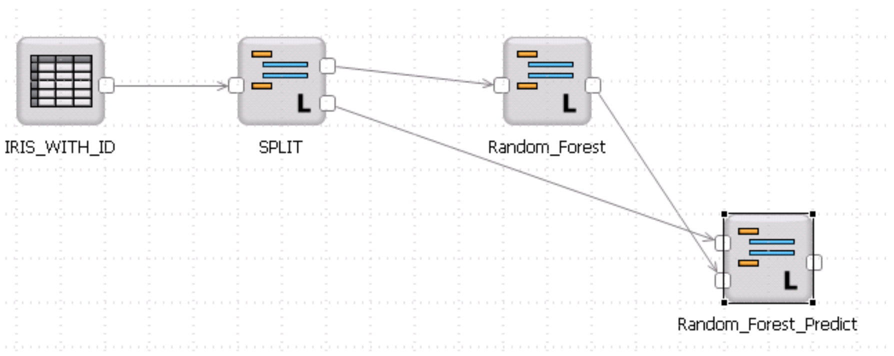
Appendix 1: Test Results - Decision Tree with PAL and L

Dataset	Llang / PAL	Create Decision Tree			Predict Decision Tree			
		Data Quantity	Parameter	Performance	Data Quantity	Parameter	Performance	Accuracy
Iris (150)	PAL	95	THREAD = 16 SPLIT MODEL = 1 PMML EXPORT = 2 MIN_REC = 2	95 ms	55	THREAD = 16	34 ms	0.94
	Llang	95	Max_Tree_Size = 100 MIN_REC = 2	62 ms	55	-	23 ms	1
KRKOPT – chess data (28.056)	PAL	17.675	THREAD = 16 SPLIT MODEL = 1 PMML EXPORT = 2 MIN_REC = 2	7.08 sec	10.381	THREAD = 16	2.23 sec	0.58
	Llang	17.675	Max_Tree_Size = 30 MIN_REC = 2	18.18 sec	10.381	-	14.79 sec	0.55
	Llang	17.675	Max_Tree_Size = 30 MIN_REC = 10	10.17 sec	10.381	-	3.06 sec	0.52
Pokerhand (1.025.010)	PAL	645.756	THREAD = 16 SPLIT MODEL = 1 PMML EXPORT = 2 MIN_REC = 2	1:18 min	379.254	THREAD = 16	22 sec	0.92
	Llang	20.000 (subset)	Max_Tree_Size = 30 MIN_REC = 2	41 sec	379.254	-	5:27 min	0.61
	Llang	50.000 (subset)	Max_Tree_Size = 15 MIN_REC = 10	3:44 min	379.254	-	6:51 min	0.64
	Llang	100.000 (subset)	Max_Tree_Size = 15 MIN_REC = 4	11:01 min	379.254	-	14:44 min	0.71
Connect4 (67.557)	PAL	42.560	THREAD = 16 SPLIT MODEL = 1 PMML EXPORT = 2 MIN_REC = 2	18.41 sec	24.997	THREAD = 16	986 ms	0.77
	Llang	20.000 (subset)	Max_Tree_Size = 30 MIN_REC = 2	1.55 min	24.997	-	8.6 sec	0.73
	Llang	20.000 (subset)	Max_Tree_Size = 10 MIN_REC = 20	0.45 min	24.997	-	1.9 sec	0.74
Optical_Rec (5620)	PAL	3.540	THREAD = 16 SPLIT MODEL = 1 PMML EXPORT = 2 MIN_REC = 2	3.7 sec	2080	THREAD = 16	562 ms	0.61
	Llang	1.000 (subset)	Max_Tree_Size = 64 MIN_REC = 2	11 sec	2080	-	215 ms	0.82

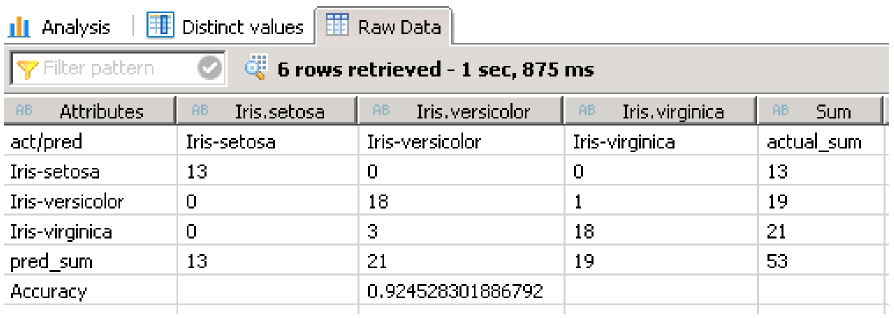
Appendix 2: Decision Tree Workflow in Data Scientist Prototype



Appendix 3: Random Forest Workflow in Data Scientist Prototype



Appendix 4: Result of Confusion Matrix in Data Scientist Prototype



6 rows retrieved - 1 sec, 875 ms

Attributes	Iris.setosa	Iris.versicolor	Iris.virginica	Sum
act/pred	Iris-setosa	Iris-versicolor	Iris-virginica	actual_sum
Iris-setosa	13	0	0	13
Iris-versicolor	0	18	1	19
Iris-virginica	0	3	18	21
pred_sum	13	21	19	53
Accuracy		0.924528301886792		

In-Memory Computing in Context of Smart Metering

Robert Wehlitz
Leipzig University
Information Systems Institute
Grimmaische Str. 12
04109 Leipzig
robert.wehlitz@uni-leipzig.de

Andrej Werner
Leipzig University
Information Systems Institute
Grimmaische Str. 12
04109 Leipzig
andrej.werner@uni-leipzig.de

Abstract

The European Union's Third Energy Package requires that at least 80 % of the European consumers shall be equipped with intelligent metering devices by 2020. As a result of this demand, the number of installed smart meters will increase as the amount of transmitted data in the upcoming years. This particularly refers to the transmission of energy consumption values, which are typically gathered by smart meters every 15 minutes. Therefore, the actors within the energy value chain must be able to handle such large amounts of data by use of advanced information and communication technology (ICT). In this paper, we present the preliminary results of implementing use cases for storing, processing and visualising smart meter data on top of SAP HANA.

1 Introduction

In September 2012, the research group Smart Energy IT Systems (SEITS) at the Information Systems Institute at Leipzig University initiated the research project *10.000 Smart Meters in the Model Region Leipzig*. In the course of this project we investigate how smart metering processes could become more efficient due to process optimisation. For this purpose, the researchers collaborate with a local meter operator. The contractor was charged to implement the research platform and thus rolled out more than 1,000 smart meters in the city of Leipzig for the first time.

One primary objective of the project is to gain new insights concerning the roll-out itself, and management of smart metering systems. Against this backdrop, smart meter processes within the energy value chain will be investigated out of a meter operator's perspective (see also [8]). Furthermore, we develop prototype-based approaches that lead to an efficient process support by use of advanced information and communication technology (ICT).

Accordingly, the objective of this paper is to describe our concept of a Smart Meter Data Map (SMDM), which supports meter operators in running a smart metering infrastructure and developing individual energy data service for their customers. This shall be achieved with the storage, processing and visualisation of large amounts of data transmitted by smart meters.

At first, we will give a brief introduction to smart metering in general and outline a main reason for the future need for advanced ICT regarding the administration and management of energy data. Then, the different development aspects, e.g. the considered use cases or the architecture, of the SMDM prototype are described. Finally, the preliminary results will be summarised and an outlook of further research interests will be given.

2 Smart Metering Basics

In the following, the term smart metering stands for the processes of automated capturing, transmission, administration and management of energy consumption and production data [1][3][4][5][9]. From a business perspective, it could be understood as a management process that develops innovative business models and increases the company's value through applied smart measuring technique and advanced ICT [7].

The current enacted obligations for installing smart meters in Germany are legally fixed in §21c of the *Energiewirtschaftsgesetz* and do comprise[1]

- New buildings,
- Existing buildings undergoing major renovations,
- Consumers with an annual consumption of more than 6,000 kWh,
- Producers with new energy generators that have a capacity of 7 kW or more.

The majority of the already conducted smart meter roll-out projects in Germany took place within pilot studies with a relatively small amount of devices [1].

[1]The obligations depend on the respective technical feasibility.

This is partially attributed to the uncertainty regarding the expectable return of investment [6]. However, the German government is increasingly forcing a massive roll-out. A cost-benefit analysis conducted by *Ernst&Young* also recommends an expansion of the previously mentioned mandatories for ensuring the economic efficiency of smart metering as a whole [2]. In consequence the number of installed devices will also increase as the amount of transmitted consumption data in the upcoming years.

This data is typically captured every 15 minutes, which accounts for 96 data records for one tariff register are delivered by a single smart meter per day. Considering the fact that intelligent metering devices enable the application of multiple tariffs, the number of captured and transmitted consumption values might be far larger.

Number of smart meters	Number of data records			
	Daily	Weekly	Monthly	Yearly
100	$96×10^2$	$627×10^2$	$2,880×10^2$	$35,040×10^2$
1,000	$96×10^3$	$627×10^3$	$2,880×10^3$	$35,040×10^3$
10,000	$96×10^4$	$627×10^4$	$2,880×10^4$	$35,040×10^4$
100,000	$96×10^5$	$627×10^5$	$2,880×10^5$	$35,040×10^5$
1,000,000	$96×10^6$	$627×10^6$	$2,880×10^6$	$35,040×10^6$

Table 1: Number of data records over time

Finally, as Table 1 suggests, the actors within the energy industry must be capable of handling a large amount of data by use of advanced ICT to benefit from the information contained.

3 Smart Meter Data Map

Our objective is to make a part of this information available for meter operators to support them running their smart meter infrastructure and developing individual energy data services for their customers. We want to investigate innovative technologies that are capable of processing millions of data records in real-time at best. Therefore, we are cooperating with the Future SOC Lab of the Hasso-Plattner-Institut (HPI) that provides us with free access to a SAP HANA instance. By using this appliance, our concept of a SMDM is being implemented. The SMDM is a web mapping service that visualises smart meter data depending on spatial data, whereby the relevant information can be accessed via an easy-to-use graphical user interface (GUI).

3.1 Preconditions

The smart meters that were installed in the course of the research project *10.000 Smart Meters in the Model Region Leipzig* are equipped with communication modules that use the General Packet Radio Service (GPRS) network for data transmission. A subcontractor is responsible for the remote reading of the meters and sends us the current gathered consumption data via e-mail once a day. We implemented a daily scheduled background job that automatically fetches these e-mails from the inbox, extracts the files contained and writes the data records into a MySQL database.

For testing our prototype, we selected an anonymised set[2] of 12,441,071 data records from December 2013 and transferred it onto SAP HANA. Additionally, an address list[3] containing 1,023 smart meter locations within the city of Leipzig was imported as well.

3.2 Considered Use Cases

We currently consider three use cases. The first use case addresses the visualisation of daily consumption values with regard to streets, buildings and households on a web map. Thereby, meter operators are able to analyse the consumption behaviour of their customers, in order to develop individual energy services for them.

Another use case is the identification of weak points within the smart meter infrastructure regarding the data transmission via GPRS. This kind of remote reading is susceptible to various influence factors. We noticed the daily data delivered by the subcontractor is incomplete because not all smart meters were reachable at any time. Thus, it might be useful for meter operators to know which locations occasionally cause problems, so that the decision-making for fault-clearing actions can be accelerated.

The third considered use case refers to the visualisation of consumption over time. The respective consumption behaviour of households or companies is classified using specific boundary values. By defining a certain time period, meter operators are able to demonstrate the customers' trend in consumption that might be part of advisory services.

3.3 Architecture

Regarding the implementation of the SMDM prototype, we use SAP HANA Studio which consists of the integrated development environment *eclipse* extended by plug-ins.

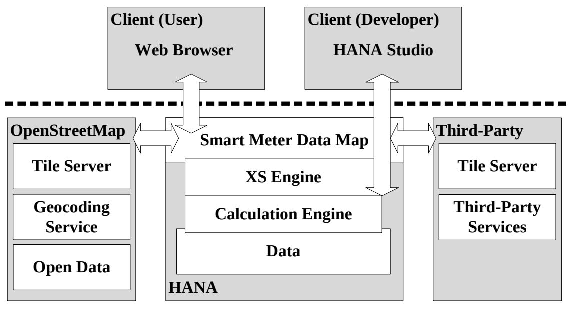


Figure 1: Architecture overview

[2]The data set does not contain any person-specific data.

[3]The address list does not contain any person-specific data.

The gathered consumption values as well as the address data are stored in the SAP HANA in-memory database. Considering the data processing, graphical calculation views are being used. These views are executed by the calculation engine. The calculated results are made available for the SMDM front-end by web services. These are based on server-side JavaScript that is interpreted by the XS engine. As shown in Figure 1, the SMDM, whose JavaScript code runs on the client-side, is built on top of SAP HANA.

For integrating map data into our web application, we decided to use the resources from the *OpenStreetMap* project because it seems to have fewer restrictions regarding the terms of use than *Google Maps*. It provides us with so called tiles. Tiles are graphical map sections that are dynamically loaded into a web page object through JavaScript calls. Services that allow us to transform address into spatial data and to add further information, e.g. street names or shop locations, to the map are available as well. Furthermore, because of the open architecture, the SMDM might be extended by third-party services.

3.4 Implementation

As one of the first steps, we defined the database schema as well as the table structure for the consumption and address data to be stored. A web service using resources of the *OpenStreetMap* project for gathering the latitude and longitude of all smart meter locations in our database was implemented afterwards.

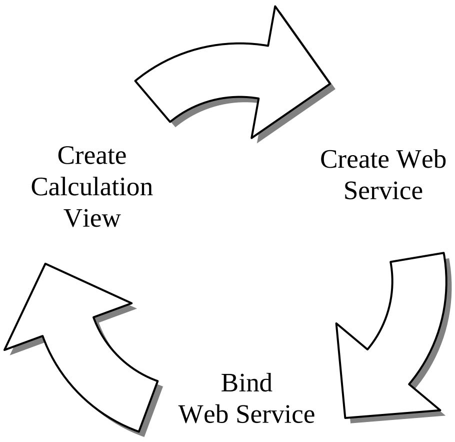
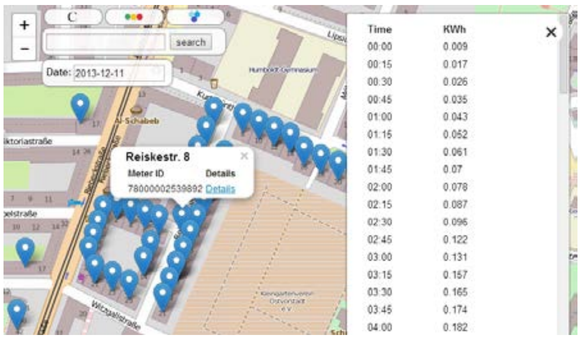


Figure 2: Use case implementation process

Then, the cyclic implementation process, as shown in Figure 2, was performed for the first time. We started with creating a calculation view to obtain the daily consumption values for a specific date with regard to streets, buildings and households. Thereby, we used the graphical modeller within the SAP HANA Studio. When the calculation view was finished, the corresponding web service could be created by use of server-side JavaScript. The web service calls the graphical calculation view using the parameters *latitude*, *longitude* and *date*. The results are returned as JavaScript Object Notation (JSON) objects. These JSON objects could henceforth easily processed by the SMDM front-end. Subsequent to the implementation of use case one, the process described above was iterated for the remaining use cases.



Time	KWh
00:00	0.009
00:15	0.017
00:30	0.026
00:45	0.035
01:00	0.043
01:15	0.052
01:30	0.061
01:45	0.07
02:00	0.078
02:15	0.087
02:30	0.096
02:45	0.122
03:00	0.131
03:15	0.157
03:30	0.165
03:45	0.174
04:00	0.182

Figure 3: Screenshot of the SMDM front-end

The SMDM front-end (illustrated in Figure 3) is developed by means of a lightweight JavaScript library called *Leaflet*. This library is used very often for integrating map data from the *OpenStreetMap* project into web pages. It allows the dynamic loading of map tiles and provides a lot of interactive features such as zooming or setting markers. For adding our own GUI elements, we fall back on the jQuery library that also provides useful functionalities, e.g. a date picker, to make the web application more comfortable to use.

3.5 Results

In the course of the SMDM development, we are able to get familiar with some of the concepts and technologies of SAP HANA. Thereby, the appliance proves to fit our needs. We test the capabilities of the in-memory database by a set of 12,441,071 data records that turned out to be no challenge for the system. The asynchronous web service requests for fetching the data from the database are fulfilled in a millisecond range. Only the web service for geocoding the address data of all smart meter locations within the city of Leipzig took an unexpectedly long time. We find out this problem is caused by an external service called *Nominatim* that we use from the *OpenStreetMap* project. It takes six seconds to obtain the latitude and longitude for each 100 addresses. This results into an overall time of about one minute. Since the geocoding is an initial and no critical task, we decided to further use *Nominatim* as geocoding service. Finally, considering the preliminary results, we conclude that in-memory technologies such as SAP HANA provide the necessary capability to develop suitable software applications for performing large-scale smart metering services.

4 Conclusion and Outlook

Since the mandatories for smart meter installations in Germany will be expanded in the near future, the number of installed devices will also subsequently increase as the amount of captured and transmitted consumption data. Thus, the actors within the energy industry must be capable of handling such large amounts of data. This can only be achieved by use of advanced ICT that enables to store, process and visualise the smart meter data, in order to utilise the information contained.

The research group SEITS cooperates with the HPI Future SOC Lab to investigate in which ways inmemory computing could be applied in the context of smart metering. Therefore, as a first step, the researchers developed and prototypically implemented the extensible concept of a SMDM. This web application currently supports meter operators in running a smart meter infrastructure and developing individual energy data services for their customers.

The preliminary results and experiences gained by the researchers indicate SAP HANA provides the necessary technologies and tools for developing suitable applications that enable large-scale smart metering services.

We would be pleased to continue the cooperation with the HPI Future SOC Lab in summer 2014 to further develop existing use cases and to consider new ones. In this regard, we aim to stronger concentrate on the integration of third-party services. For instance, it could be possible to involve weather data for analysing the correlation to consumption behaviour. Another interesting use case might be to consider the local position and status of telecommunication providers' radio installations. With regard to the smart meter locations, metrics and recommendations could be derived that support meter operators in detecting and preventing regular GPRS data transmission failures.

Acknowledgement

The research project *10.000 Smart Meters in the Model Region Leipzig* is funded by the European Regional Development Fund (ERDF) and the Free State of Saxony (Sächsische Aufbaubank – SAB). During the development of the SMDM, the researchers were provided with a SAP HANA instance by the HPI Future SOC Lab.

References

[1] BEAMA Limited: European Smart Metering Alliance – Final Report, http://www.eaci-projects.eu/iee/page/Page.jsp?op=project_detail&prid=1564, accessed: 27/03/2014.

[2] Ernst&Young GmbH: Kosten-Nutzen-Analyse für einen flächendeckenden Einsatz intelligenter Zähler. (Cost-Benefit Analysis for the Comprehensive Use of Smart Metering Systems), http://www.bmwi.de/DE/Mediathek/publikationen,did=586064.html, accessed: 27/03/2014.

[3] U.C.C. Jagstaidt, J. Kossahl, L.M. Kolbe: Smart Metering Information Management. Business & Information Systems Engineering, vol. 3, no. 5, pp. 323-326. Springer Gabler (2011).

[4] B. Neenan, R.C. Hemphill: Societal Benefits of Smart Metering Investments. The Electricity Journal, vol. 21, no. 8, pp. 32-45. Elsevier, 2008.

[5] J.C.P. Kester, M.J.G. Burgos, J. Parsons: Smart Metering Guide - Energy Saving and the Customer, http://www.ecn.nl/docs/library/report/2011/o11004.pdf, accessed: 27/03/2014.

[6] K. Lohnert: Beschleunigung der Transformation vom Energieversorger zum Energiedienstleister. (Accelerate the Transformation from Energy Utilities to Energy Service Providers) In: Aichele, C., Doleski, O.D. (eds.) Smart Meter Rollout – Praxisleitfaden zur Ausbringung intelligenter Zähler. pp. 75-103. Springer Vieweg, Wiesbaden (2013).

[7] O. Schaloske: Möglichkeiten zur Erschließung von Effizienzpotentialen durch Smart Metering. (Possibilities for the Development of Efficiency Potentials by means of Smart Metering), GRIN Verlag, Munich (2010).

[8] R. Wehlitz, A. Werner, B. Franczyk: SMIM – A Cloud-based Approach for the Digitisation of Smart Meter Installation Processes. Accepted paper at International Conference on Business Information Systems, 28.-30. April 2014, Taichung, Taiwan.

[9] K.S.K. Weranga, S. Kumarawadu, D.P. Chandima: Smart Metering Design and Applications. Springer, Singapore (2014).

Integration of a VEE-Framework into SAP HANA

Jan-Patrick Weiß
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
jan-patrick.weiss@uni-oldenburg.de

Benjamin Reinecke
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
benjamin.reinecke@uni-oldenburg.de

Marco Lucht
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
marco.lucht@uni-oldenburg.de

Jad Asswad
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
jad.asswad@uni-oldenburg.de

Jan Hendrik Wege
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
jan-hendrik.wege@uni-oldenburg.de

Christoph Walther
University of Oldenburg
Department of Computer Science
Uhlhornsweg 84
D-26129 Oldenburg
christoph.walther@uni-oldenburg.de

Abstract

The energy market is changing by legislation so by 2020 smart meters will be installed in most households. Therefore the IT systems of energy companies have to adjust to deal with the growing amount of measured data. With smart metering it is possible to measure the consumption near real time. This allows companies to create new business use cases. They can for example predict the usage of their grids and close short time contracts with suppliers to meet the demands [1][2]. Furthermore there are new possibilities for analysis and assurance of data quality. In order to assure a defined integrity, possible missing or false values of the available measuring data have to be corrected. The project deals with these problems of data quality and handling of these massive data. To ensure integrity and the best quality of data, a VEE-Framework (Validation, Estimation, Editing) has taken place. The VEE-Framework has been implemented into SAP HANA to benefit from the efficiency of in-memory-database technology.

1. Definition of the project

The research project “Integration of a VEE-Framework into SAP HANA” is done in cooperation with CX4U AG. The main idea of the project is to access the Smart Gateway (provided by CX4U) with SAP HANA and to process analysis and predictions of the data within the in-memory database. In the used demonstration system, data of 15,000 smart meters are simulated. Each smart meter generates 768 bytes of data every day which leads to 330 gigabytes of data in an accounting period. It is very important to analyze these data very fast and easily change the rules of analysis depending on the origin of the data. Often there are missing or incorrect values and through the analysis these data get corrected. Each customer and use case has a different rule which defines the methods to correct the data. This function is part of a VEE-Framework in which the data get validated, estimated and edited. The VEE-Framework has been implemented into SAP HANA and within a proof of concept it has been proven how the Smart Gateway and SAP HANA can communicate with each other.

2. Solution alternatives

In the course of the proof of concept, the research led to a variety of different alternatives. The most considered alternatives are provided from SAP itself and listed as SAP Data Provisioning Technologies (Figure 1). Those Technologies are represented by SAP Sybase Event Stream Processor, SAP HANA Smart Data Access and SAP HANA Replication Technologies. The replication Technologies are also divided into four other technologies according to their replication functionality and they are represented by SAP Data Services as ETL-Based Replication, SAP HANA Direct Extractor Connection (DXC) as Extractor-Based Data Acquisition, SAP Sybase Replication Server (SRS) as Log-Based Replication and finally SAP Landscape Transformation Replication Server (SLT) as Trigger-Based Replication [3].

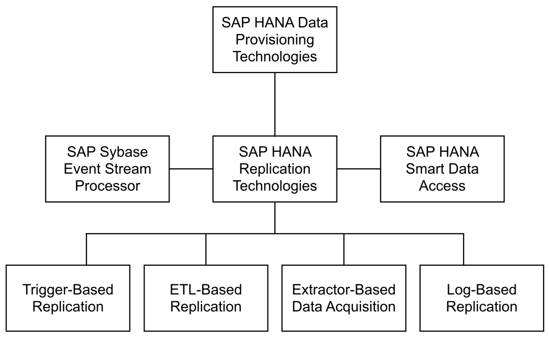

SAP HANA Data Provisioning Technologies
SAP Sybase Event Stream Processor
SAP HANA Replication Technologies
SAP HANA Smart Data Access
Trigger-Based Replication
ETL-Based Replication
Extractor-Based Data Acquisition
Log-Based Replication

Figure 1. Data Provisioning Technologies

Source: Own figure

SLT, as one of the best provisioning technologies provided from SAP, allows the exchange of data between SAP systems and other SAP or non-SAP systems. With SLT it is possible to continue to use existing system and benefit from the increased speed of SAP HANA. By using the SLT Replication Server, only the tables which are relevant for the data delivery are processed, so that the data exchange remains efficient.

Besides SAP solutions, two other alternatives was considered. One of them is Secondary Database Connection (SDBC). SDBC is used in case real time data processing is not required. This method uses replications of specific data of a database to load it into SAP HANA database. HANA does not act as an alternative of the existing database, furthermore it is an add-on whose main focus is to speed up the reading of data from existing ABAP-Systems. This increased speed is also the main advance of this technology. There are currently three existing alternatives: SDBC with open SQL, native SQL and native SQL - ADBC (ABAP database connectivity). SDBC with native SQL - ADBC provides the ABAP database connectivity concept. By using HANA it is important, that ADBC also provides access to non-data-dictionary-products.

The second alternative is SAP HANA Extended Application Services (XS). The XS-Engine has a high integration level within the SAP HANA itself. With only a simple OData Interface on the HANA side and a HTTP-Request from the NetWeaver side the XS-Engine can fulfill the purpose of the project and assure the connectivity between the two systems in addition to processing the data locally through its server side JavaScript capabilities.

3. Selection criteria

In order to select the suitable solution among the different alternatives, a set of criteria is set to evaluate them. The criteria assess the alternatives according to their efficiency and effectiveness concerning the replication or the communication between SAP HANA and the source system and the transformation capabilities between both of them.

One of the most important aspects is the ability to handle the data at real-time or according to scheduled intervals in a manner that preserve the benefit of using the high speed of SAP HANA in processing metering data. Furthermore, the supporting capabilities of the method are examined to determine the supported databases and the ability to support non-SAP-based systems. On the other hand, the prerequisites of each alternative are listed in order to define the system requirements required to implement the method. The transformation functionalities during the replication or the communication process play a significant role in the evaluation beside the pricing information that help to choose the best alternative within an affordable cost and even better with no cost or extra expenses like the alternative implemented in the project.

The selection criteria are listed as follow:

- Real Time, Interval or Scheduled
- Data Sources
- System Requirements
- Supporting non-SAP-based Systems
- Transformation Functionalities
- Costs

4. Implementation

After evaluating different implementation alternatives, the SAP HANA Extended Application Services (XS-Engine) have been chosen as the suitable solution for this particular project. Among all other alternatives, the XS-Engine fulfils the needed functionalities and provides a real-time performance without any extra expenses or products.

The first step was to create an OData service on the HANA side, so that the Netweaver side could insert data into and get data from the HANA database. At the same time, a functionality on the Netweaver is developed in order to read the OData interface. Afterwards the data had to be processed within the HANA system and therefore several scripts are implemented, which offered the needed VEE-functionality. In addition to the scripts which execute the VEE functions, a web interface had to be developed for managing and calling those implemented functions. The web interface is programmed using HTML and Javascript. Finally it was possible to request data from the OData interface into the Netweaver using a HTTP request. Moreover, the VEE algorithms were able to be used to process the data in the HANA database.

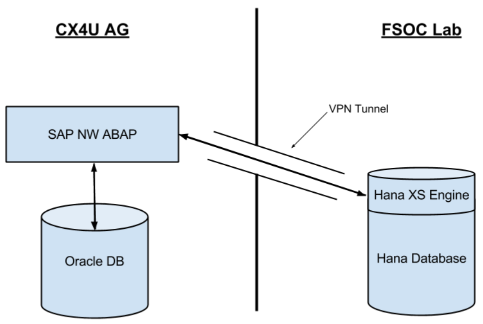


Figure 2. Systemarchitecture
Source: Own figure

5. Results

As a summarization it can be said that the main objectives of this project were two points. First, proving the communication ability between the two separated systems withing a proof of concept. This has been done on a conceptual basis by comparing different implementation methods with each other as well as selecting the most appropriate one and implementing it into a prototypical communication sample.
Second, implementing the VEE-framework which has also been described and evaluated in a conceptual manner, inside the given SAP HANA architecture.
By achieving these goals the results of this project now provide the basic functionality of the VEE-vision and can therefore be seen as a fundamental contribution for further steps of the CX4U in order to handle the existing and future challenges inside the energy and gas market.
Also there has been a huge interest in this particular topic from representatives from different industries as the project group presented its results at the CeBIT 2014 in Hannover, Germany. The feedback which has been gathered at this exhibition shows that the subject of dealing with big data in near real-time or in real-time scenarios is and will become more and more important in the future.

6. Further Steps and Outlook

The project has shown that it is possible to handle metering data inside SAP HANA that comes from a specific source system. As this has been a prototypical solution for the desired context, there have to be additional implementations regarding robustness and range of functions. Also specific standards like encrypting the data transfer have to be examined, especially since they come from statutory requirements. Also the set of methods used within the separate VEE-process steps has to be extended in order to achieve customer specific test scenarios.
Considering the increasing amount of assessed data in the near future it seems obvious to deal with software solutions that offer an accelerated performance in handling huge amounts of data. Now that the communication ability has been proven for the productive system at CX4U and SAP HANA, the local management needs to decide if and when they will migrate to SAP HANA.

References

[1] C. Aichele. Innovativ smart meter für die energiemärkte der zukunft nutzen, 2013.
[2] Verbraucherzentrale_Bundesverband. Energiemarkt im wandel. tagung von der verbraucherzentrale bundesverband und der bundnetzagentur, 2013.
[3] SAP AG. SAP HANA Technical Operations Manual, 2013.

Next Generation Operational Business Intelligence

exploring the example of the bake-off process

Alexander Gossmann
Research Group Information Systems
University of Mannheim
Schloss
68131 Mannheim
agossman@mail.uni-mannheim.de

Abstract

Large retail organizations have to plan customer demands accurately, to achieve customer satisfaction and loyalty. The primary objective is to avoid out-of-shelf situations. On the other hand, losses of perished goods, especially in case of fresh food, have to be minimized. The handling of the trade-off between availability and loss can be dramatically improved by a real-time analytic system. The challenge is to analyze large amounts of data (big data), typically derived from the transactions in the retail process, enhanced by external data, like weather and holidays. Different management groups require specific information with short response times at reasonable costs. Transferred to the retail domain, local store managers are focused on operational decision making, while top management requires a view on the business at a glance.
Both requirements rely on transactional data, whereas the analytic views on this data differ completely. Thus different data mining capabilities in the underlying software system are targeted, especially related to processing masses of transactional data.
The examined software system is a SAP HANA in-memory appliance, which satisfies the aforementioned divergent analytic capabilities, as will be shown in this work.

1 Introduction (Project Idea)

Operational Business Intelligence is becoming an increasingly important in the field of Business Intelligence, which traditionally was targeting primarily strategic and tactical decision making [1]. The main idea of this project is to show that reporting requirements of all organizational levels (operational and strategic) can be fulfilled by an agile, highly effective data layer, by processing directly operative data. The reason for such architecture is a dramatically decreased complexity in the domain of data warehousing, caused by the traditional ETL process [2]. This requires a powerful and flexible abstraction level of the data layer itself, as well as the appropriate processability of huge amounts of transactional data.
The SAP HANA appliance software is currently released in SPS 07. Important peripheral technologies have been integrated, such as the SAP UI5 Presentation Layer and the SAP Extended Application Services, a lightweight Application Layer. This project proves the tremendous possibilities offered by this architecture which allows a user centric development focus.
This report is organized in the following chapters. The first chapter provides a general overview of the explored use case. In the second chapter the used resources will be explained. The third and fourth chapters contain the current project status and the findings. This Document concludes with an outlook on the future work in the field.

2 Use Case

This project is observing a use case in the field of fast moving goods of a large discount food retail organization. Specifically, the so called bake-off environment is taken into account. Bake-off units reside in each store and are charged with pre-baked pastries based on the expected demand. The trade-off between product availability and loss hereby is extremely high.
From the management point of view, the following user group driven requirements exist: On the one hand, placing orders in the day to day business requires accurate and automated data processing, to increase the quality of the demand forecast. On the other hand, strategic decision makers need a flexible way to drill through the data on different aggregation levels, to achieve a fast reaction time to changing market conditions.
The observation period of two years is considered. The basic population consists of fine grained, minute wise data for thousands of bake-off units, providing all facts related to the bakery process.

2.1 Store Level Requirments

On the store level, the store manager will be supported with matters regarding daily operational demands. Primarily for order recommendations, a certain amount of historical data is taken into account to satisfy the appropriate statistical calculation on time series. Additionally location related and environmental information increases the accuracy of the forecasting model. Environmental variables, like historical weather and holidays, are considered in correlation with historical process data to improve the forecast model. Furthermore, forecasted weather data and upcoming holidays are taken into account for ex-ante-anta in order to improve the prediction. The appropriate store manager is processing model fitting and operational data analysis ad hoc and on demand.

2.2 Corporate Level Requirements

On the corporate level a 'bird's eye view' is the starting point, where highly aggregated key figures indicate business success or problems. These measures deliver information on a very high level, whereas the reasons for the appearance of these indicators can vary strongly. For accurate decisions, it is tremendously important to drill down to the line level, to indicate the reasons for certain business patterns. As the strategic reporting is based on one common data foundation of operational data, navigation to the line level is implicated. It is important that the system is having user satisfying response times, allowing the exploration of a huge amount of data. The application provides the detection of certain patterns and correlations for a more complex classification. For example, the daily availability is analyzed based on certain thresholds, provided by minute wise real time data. To sum up, real-time enabled reporting on strategic level allows reactions on market changes to reach an unprecedented level of effectiveness.

3 Project set up

This chapter illustrates the used technology. After a listing of the architectural resources the appropriate implementation domains will be described in more detail.

3.1 Used Resources

As stated in the introduction, the used architecture is based on the SAP HANA Appliance Software SPS07 [3].
The presentation layer is built upon the HTML5 based framework SAP UI5. The communication with the SAP HANA In-Memory database and user handling is established through SAP Extended Application Services (XS Engine). Data intensive calculations and data querying are handled by the appropriate APIs in the database, such as the calculation engine (CE), the SQL engine, the Application Function Library (AFL), and particularly the Predictive Analytics Library (PAL) [4].
Additionally, the newly introduced development language River will be used, to create an abstraction from the HANA artifacts and enable a higher flexibility of the implementation process. For time series analysis the Rserve based R integration is used. The data load of CSV formatted transactional data, as well as the data replication and 3rd party are implemented in Java and imported through the JDBC API. The considered 3rd party data consists of weather data, as well as school and public holidays.

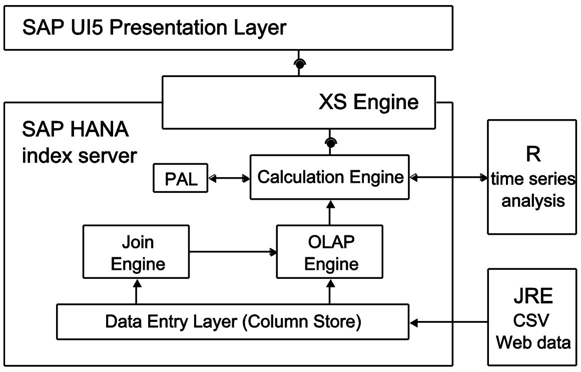


Figure 1: Architecture

The used architecture, as described in the following chapters, is summarized in Figure 1.

3.2 SAP Front End

As stated above the SAP UI5 constitutes the presentation layer. The Model View Controller pattern is being conducted for front-end implementation. For web and mobile versions of the application, two different view variants are implemented.
The entry point for a specific user group is the login screen, whereas the different management roles are distinguished by specific HANA user roles. The user groups are differentiated into the strategic, tactical, and operational management role. The strategic and tactical roles are showing the same reports, restricted by the related aggregation level. On the operational level, completely different reports are provided and are mainly focused on daily analysis. Additionally, order recommendations for the next three days are visualized. Each report relies on one associated calculation view, described later in the back end section. The selection parameter invoked by a user are handled by OData services, with the belonging data binding, or manually by SQL Script calls.

3.3 SAP HANA Back End

The HANA in-memory database is the core technology of this investigation. In the following section, the data model will be shortly discussed.
The data entry layer consists of two main fact tables. One fact table contains daily aggregated sales related

key figures. The second fact table consists of minutely wise measures, derived from the bakery process.
This fact table has an expected cardinality of approximately two billion records. In the current implementation this table has rounded 800 million records for the first testing runs. A hash partitioning policy, based on stores is conducted here, in regards to the expected limit of 2 billion records per table. Several master data tables contain information about stores, regions, products, and holidays. Historical weather data is stored in an appropriate table, whereas weather forecasts will be stored separately and merged daily into the historical data table. All tables are implemented as column tables.
Upon this data entry layer several attribute views are implemented, building up the product, store and regional dimensions. The time dimension is based on the generated time table with minutely level of granularity (M_TIME_DIMENSION), provided by HANA standardly.
The two analytical views contain the fact tables, whereas the daily based fact table is additionally enhanced by the weather and holiday dimensions. Based on this multidimensional data model eight calculation views are implemented, to satisfy user reporting scenarios about availability, loss, and sales on tactical and strategic level. Additionally one calculation view provides reporting needs on operational level, showing the relevant process information of the current and previous days.
For more sophisticated data mining on the strategic level, as well as data preprocessing of time series data, PAL is used [5]. Specifically the linear regression model function is used to draw trends of dynamically aggregated sales data over time. Further the anomaly detection function is used for outlier detection in daily sales data.

3.4 Peripheral technology

The load of historic and transactional data is handled by a proprietary Java import module, using the JDBC API. The reason for this implementation mainly relies on huge amount of heterogeneous CSV format- ted files. Approximately two hundred thousand different types of CSV files have been imported into the HANA database. Therefore, a special bulk load strategy has been used, especially in spite of the insert properties of column oriented tables in the entry layer. Furthermore, historic weather data as well as weather forecast and holiday data is loaded via the JDBC interface of the import module.

Holidays

Both school and public holidays have been downloaded for the past two years, and until the year 2015 from the online portal 'Schulferien.org'. The data is available in the iCal format, and covers all dates for the different states of Germany. These files were loaded into the HANA index server, after conversion into CSV format, using the appropriate build in wizard.

Weather

The historical weather data has been imported from the web weather API ‘wonderground.com’. For model training of the forecast module, the corresponding time interval values of daily, city wise consolidated store data was called from the API. This results in approximately one million JSON files (one file corresponds to one data record), generated by the REST interface, afterwards converted into CSV format and loaded via JDBC of the import module.

Forecast

The demand forecast requirements are primarily developed using the R environment. The appropriate time series are generated on demand, and invoked by a store manager who is responsible for one’s store. As stated in the previous section, the time series are being preprocessed in advance by the PAL frame- work primarily for performance reasons.
The important outlier detection and handling have been additionally implemented in the R environment, as here more advanced algorithms are available in the R community. Furthermore, two different forecast models have been utilized for comparison reasons. The ARIMA (Auto Regressive Integrated Moving Average) model as well as the ANN (Artificial Neuronal Network) based model have been observed.

3.5 Development environment

The eclipse based HANA Studio is used as the main IDE for the development. In addition to the newly introduced SPS05 features, regarding the ‘HANA development’ perspective, the Java import module is implemented as well.
For usability reasons the following implementation strategy of the R environment has been utilized: Each developer uses a local R runtime for coding R script and model testing. The appropriate time series data is supplied through the ODBC interface. After finalizing a model in R, it is transformed into the HANA environment using the RLANG extension in SQL Script [5].
All artifacts, including java classes, java script, UI5 artifacts and R script have been set under version control with git [6].
The prototype has been completely redesigned regarding the SAP HANA components. The SAP HANA repository has been used for this purpose, to store all relevant design time artifacts like:

- hdb tables
- hdb roles
- procedures

4 Findings

This chapter contains findings on technological as well as on the process level. The findings will be explained analogous to the outline of the previous chapter. In conclusion the outcome of this project will be summarized.

4.1 SAP Front End

Through the tight integration of the controller and model layer, the presentation layer profits of the advantage of a high abstraction level. The data binding feature of the OData services is especially beneficial for strategic and tactical reporting. Hereby flexible data navigation for the top management user is provided, by selecting free time intervals and breaking down into different products, regions, or stores. Never the less, the store management invokes an ad hoc data mining and forecasting capability by calling a SQL Script procedure through a Java Script DB connection call.

For parameterization of the calculation views the following limitations exist:

- exclusively input parameters are used, instead of variables for performance reasons
- for input parameters, no ranges are supported and graphical calculation views require additional filter expressions
- character based date parameters work with the OData interface (thus no type safety is provided, implicit cast)

4.2 SAP HANA Back End

In the previous chapter (2.3) the data model has been explained. The biggest column based table contains two years of minutely based transactional data. It has been partitioned by regions. The response times of the appropriate calculation view calls are absolutely satisfying . Nevertheless, the following main restrictions have been experienced which are listed by the appropriate domain:

Predictive Analytics Library

- usability of PAL functions is inconvenient and non-transparent
- restrictive parameterization policy
- very limited exception handling

The restriction in the design time usability especially in the case of PAL, compromises the performance experience of the data analysis.

The AFL framework is in a relatively early stage of maturity and in this project context, only few functions could be utilized. The major functionality in the area of time series analysis has been conducted in the R environment, as stated in the next section.

4.3 Forecast

The demand forecast for each store is calculated on demand. The appropriate time series is generated and sent, together with the belonging weather and holiday information to the R runtime. Hence the sent data frame to R contains daily related time series derivate of additional environmental data to the historic sales data for a certain pastry and store.

Time Series Preprocessing (Outlier Adjustment)

The majority of given outliers belongs to the class of additive outliers, due to public holiday, related store closing. The effect is even more significant, the longer a closing period is. Here the precedent open business date shows an abnormal high characteristic. Other outlier classes are by far less significant or cannot be assigned directly to events. Different outlier handling strategies have been tested and implemented, and will be investigated in further proceedings.

ARIMA based forecast

An automated ARIMA model has been implemented in R. The used package is mainly the package 'forecast' [7] available at CRAN (Comprehensive R Archive Network [8]). The automated ARIMA fitting algorithm ‘auto.arima()’[9] has been utilized for this project purposes, which is based on the Hyndman et al algorithm [10]. Specifically seasonality, non-stationarity, and time series preprocessing (see outlier handling) required manually coded model adjustment. All additional predictor variables like holidays and weather information could be processed auto-matically, passed by the ‘xreg’ matrix parameter.

ANN based forecast

Alternatively, to the ARIMA approach, an Artificial Neuronal Network model has been implemented and is especially for capturing automatically nonlinear time series shapes. As expected in the retail context, ANN is supposed to deliver more accurate forecast results [11]. In this use case the ‘RSNNS’ [12] (Stuttgarter Neural Nets Simulator [13]) package has been utilized. Similarly to the ARIMA model (see above), the independent variables, primarily the daily sales an all additional related variables are used for model fitting.

Summary Forecast with R

One major design change has been made; the R runtime has been transferred to another server. As the previous solutions has been running together with the HANA instance on a virtual machine with 64 cores the parallelized ARIMA based forecast used all avail-able resources on the Linux server. This is not recommended by SAP, as it could harm the processes of the HANA instance itself. Thus, an R runtime on a separated server is obligatory. It can be stated that the performance behaves nearly inversely proportional to the number of cores for the ARIMA algorithm, proposed above. Nevertheless, different loading and presentation strategies are required, to provide user acceptance in response times. For instance, asynchronous XSJS calls could be per- formed, to avoid persisting trained models. This is especially true for ANN algorithms which are only tenuous parallelizable.

5 Conclusion

The built prototype was expected to satisfy the reporting requirements of the different stakeholders of information consumption. Although the data analysis capabilities differ throughout the organizational roles of managers, all human recipients expect short response times of a system. With the usage of the SAP HANA appliance software this challenging task could be achieved.

From the development perspective, previously not known effectiveness could be achieved. As all reporting and predictive analytics requirements rely on only a few physical tables, the main effort consists in providing different views on this data. Even more complex measure calculations, like availability and some regression analysis, are processed on the fly. This is a completely new way of designing a reporting system. Compared to traditional ETL based data warehousing tools this saves a lot of manual effort in the loading process. However, this does not imply that the effort for implementing the business logic disappears, merely that the programming paradigm is straightforward. The capability of providing demand forecasts based on long time series intervals for thousands of stores and different products particularly supports operational decision makers on the day to day business. This could not, or only very difficultly, be achieved with traditional disk based data warehouse approaches, focused on aggregated measures. In this prototype, forecast algorithms are performed on demand.

6 Outlook

The upcoming work will focus on four key areas. The first area focuses on enhancing the existing outbound connectivity by implementing routines which automatically load the external data. For this purpose the newly introduced XS Job Scheduling feature of HANA will be used. As the performance of the analysis of data is highly influenced by the quality of the implemented model, the second area focuses on the testing and improvement of the existing analysis models. Having a growing project with growing complexity, it is inevitable to standardize and abstract different objects of the project. For this purpose the third area focuses on the utilizing of River language, as part of SPS07, which has recently been introduced. By using River, the amount of time needed to implement new artefacts and business logic in general can be reduced significantly, which enables test cases that are more flexible.

For example new user inputs can be considered very fast in the business logic, when building upon River enabling a more in-time application development. In order to be able to use the major benefits of the in memory technology and the analysis models a high-class graphical interface is needed. This requirement is met by the fourth area which focuses on the implementation of a powerful user interface. In the previous project period, a desktop and mobile version of the application has been implemented by using the UI5 framework. Due to the high abstraction of the data interfaces a new framework has come into focus, which enables the research on additional areas like for example the adaption of the user interface, depending on the type of mobile device, or the adaption, depending on the operating system. This research will be done, by using additionally the SenchaTouch framework. To provide further information on the applicability of the developed models in the fresh food industry, an additional use case will be observed. Fresh vegetables and fruits have a sell-by date of only a few days.

References

[1] C. White: The Next Generation of Business Intelligence: Operational BI. *DM Review Magazine*. Sybase 2005

[2] H. Plattner: A common database approach for OLTP and OLAP using an in-memory column database. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.* ACM, 2009.

[3] SAP HANA Developer Guide. http://help.sap.com/hana/SAP_HANA_Developer_Guide_en.pdf, (accessed:19th of March 2014)

[4] SAP HANA Predictive Analysis Library (PAL) Reference. help.sap.com/hana/hana_dev_pal_en.pdf, (accessed:19th of March 2014)

[5] SAP HANA R Integration Guide. help.sap.com/hana/hana_dev_r_emb_en.pdf, (accessed:19th of March 2014)

[6] http://git-scm.com/ (accessed:19th of March 2014)

[7] http://cran.r-project.org/web/packages/forecast/forecast.pdf (accessed:19th of March 2014)

[8] http://cran.r-project.org/ (accessed:19th of March 2014)

[9] http://otexts.com/fpp/8/ (accessed:19th of March 2014)

[10] Hyndman, Rob J., and Yeasmin Khandakar. Automatic Time Series for Forecasting: The Forecast Package for R. *No. 6/07. Monash University, Department of Econometrics and Business Statistic*s, 2007.

[11] Doganis, P., Alexandridis, A., Patrinos, P., & Sarimveis, H. (2006). Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *Journal of Food Engineering,* 75(2), 196-204.

[12] http://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf (accessed:19th of March 2014)

[13] http://www.ra.cs.uni-tuebingen.de/SNNS/ (accessed:19th of March 2014)

Using SAP ERP and SAP BW on SAP HANA: A mixed workload approach

- Research in progress report -

Galina Koleva
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
galina.koleva@in.tum.de

Robert Meyer
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
robert.meyer@in.tum.de

Jonas Hueber
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
jonas.hueber@in.tum.de

Sonja Hecht
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
sonja.hecht@in.tum.de

Helmut Krcmar
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
krcmar@in.tum.de

Abstract

The purpose of our research project is to evaluate the mixed workload of SAP HANA. In a first step we have performed performance comparisons of SAP ERP running on SAP HANA and IBM DB2 with the help of the Rational Performance Tester from IBM. This tool emulates virtual concurrent users to simulate database load. The first results of this comparison is presented in this progress report. In order to be able to further compare the mixed workload of SAP HANA we have applied for a project period extension.[1]

1 Introduction

Unlike common relational database systems that rely on disk storage, an in-memory database (IMDB) stores data within the main memory. This is not a completely new concept as IMDBs have been around since the 1980s (e.g. TimesTen) [1]. But with today's growing capacities, diminishing latencies and sinking costs of DRAM it is possible to render large enterprise applications solely on main-memory-resident data. Along with this comes an increase in computing power through multi-core architectures, which can improve performance by parallelizing computations. Besides leveraging multi-core parallelism with multi-processor systems, technologies like quick path interconnect (QPI) or hyper threading further support the advancements that enable in-memory database systems to promise performance enhancements and real-time data processing [2].

Common ERP systems rely heavily on Online Transaction Processing (OLTP). OLTP is characterized by departments saving data tuples in row-stores and performing small transactional database updates or data retrieval operations. Analytical and financial planning applications that refer to Online Analytical Processing (OLAP) were separated into own systems to prevent OLTP systems to be throttled back from time consuming and complex queries [3]. The discussion (as in [2], [3], [4]) of handling both OLAP and OLTP workloads

[1] Please refer to the project extension application: Using SAP ERP and SAP BW on SAP HANA: A mixed workload approach

in one system supported by a column oriented organization of data led to the questions whether in-memory databases are capable of handling ad hoc queries on transactional data in real-time and to which extent ERP systems can benefit from IMDB technology keeping the recent release of SAP ERP on SAP HANA in mind.

2 Load Testing SAP Solutions

An answer to these questions can be found by performance testing. Performance testing targets non-functional requirements like response times and transaction throughput and captures risks to system availability that cannot be revealed by functional testing.

2.1 Classification

While the whole process of simulating users and generating workload is called performance testing, the term incurs different characteristics depending on the operation purpose. Single user testing thereby defines a best-case scenario which isn't influenced by simultaneous operations from multiple users and therefore considered to be the first step of performance testing and can be used in early development stages. Conducting the test and its KPIs with a single user enables the task of load testing with multiple users that simulate real-world workload scenarios under realistic operating conditions. Its aim is to pre-test the behavior of the system with a realistic maximum load that is likely to be reached after the release. If the simulation of users and the workload aims to find the limits by exceeding the systems' boundaries, the test is called a stress test. Additionally, this category of volume tests targets solely the throughput of mass data.

2.2 Methodology

As Helfen/Trauthwein state in [5], the aim of load testing SAP solutions is to cover the risks of system failures caused by overload, incomplete time-critical processes, unmet service-level agreements (SLAs) and bad user acceptance, which results in several possible test goals. Addressing these goals by load testing enterprise SAP solutions requires a structured three-step approach at minimum. The general phase model for SAP performance test projects includes the phases of planning, performing the load test (execution phase), performing the stress test (optional) and completing [5]. By using IBM Rational Performance Tester (RPT) as the test tool we were able to refine this approach with features (section 3) that guarantee the close-grained measuring of KPIs, which relate to our load testing goals (section 4.1).

2.3 KPIs

Traditionally, the approach of performance testing focuses mainly on end to end response times as requirements. Optimization activities then try to reduce time spent in the specific components. Cheng states in [6] that although this is easy to understand and proven to be effective for performance related customer support cases, KPI driven tests can detect possible issues even if they don't occur in the measured data. The term "KPI driven" means that the processes of test execution and system optimization iterate until the measured results reach the defined KPIs. This approach is necessary since SAP customers have different customized and modified installations adapted to their individual business processes. It is considered impossible to create test cases that reveal all possible performance issues on all possible hardware and software configurations. Proper KPIs give information about the performance as seen from the user as well as the system. The more accurate KPIs are measured, the more they support the optimization process with the resulting performance improvements which otherwise would be impossible to verify. This means that the accuracy percentage of a measured KPI is directly proportional to the verified improvement percentage, an optimization process can gain. Furthermore, the measured results of a KPI should be reproducible and give hints to possible optimizations.

3 IBM Rational Performance Tester

The RPT is a tool to accomplish the tasks of performance and load testing and is part of the de facto standard in application quality management solutions. The primary field of application for RPT is to provide a scripting-free environment able to automate load testing of HTTP-, Citrix-, Socket API-, SIP-, Siebel-, and SAP-Applications. With its SAP extension it is possible to analyze SAP applications by recording a test via SAP GUI, which is afterwards split into editable SAP transactions. It is possible to manipulate these transactions by adding custom code or datapools in order to provide data variation and regression throughout multiple virtual users (VUs). The use of a RPT Agent Controller on distributed agent systems supports the flexible emulation of large user populations to generate user load. Furthermore, the SAP extension includes test execution features like advanced scheduling, real-time reporting and monitoring and a scalable execution engine to successfully measure the system's performance under load. The RPT thereby enhances the execution phase and completion phase by providing the steps of test creation, test manipulation, test validation, workload emulation, schedule execution and the evaluation of results.

4 Test Case

The aim of the load tests conducted within the scope of this project is the comparison of SAP ERP on SAP HANA and SAP ERP on IBM DB2 regarding their performance. The test system based on SAP HANA is provided by the Hasso-Plattner-Institut (HPI) via the

HPI Future SOC Lab. The IBM DB2 system is provided by the Chair for Information Systems of the Technische Universität München (TUM) and the SAP University Competence Center (SAP UCC), which also includes the test laboratory and licenses required for the IBM Rational Performance Tester environment.

4.1 Test Goals and Planning Phase

Load testing a product from a black box perspective comes with certain constraints that define the comparability and the limitations in which the project operates:

- There is no legacy data available from the specific systems under test like existing load profiles or previous test results. This can be seen as a load test from scratch, which results in the freedom of selecting processes which have to comply only with the context and extent of the project's work and the underlying fictional test scenario of simulating a small business workload.
- The systems under test resides in different locations. This does not pose a threat to the comparability because latencies are even and data throughput only limits high volume tests, the workload is not allowed to be affected by network bandwidth. Therefore the load test cannot be high volume and its success is influenced by emulating a minimum workload that does not reveal bottlenecks caused by hardware sizing rather than a realistic maximum load.
- Although the structure and data from the Global Bike Incorporation (GBI) study is migrated into both databases, there is no transactional data available. Due to the missing data, no high profile or long run processes (like dunning runs) could be executed. Therefore the created SAP performance tests should rely on providing variable data on their own and working with results of previous tests in a recursive manner by being executed sequentially.

The load test planning profile, which covers these constraints, is illustrated in Table 1. SAP screen request response times and succeeded SAP elements as KPIs reflect user acceptance from an external point of view. The SAP screen request response time measures the delay between a server request submitted by the user via SAP GUI and the moment the server responds and is used to compare the selected load test runs regarding their performance. A test run is considered successful if all virtual users are finished with an error margin less than 1% regarding the overall succeeded SAP elements. Reaching these requirements has to be verified in three independent runs with the same execution schedule. Both SAP ERP instances run the GBI case studies developed by the University Competence Centers (UCC) and used by more than 1200 Universities.

Item	Property
Test goal	Performance comparison based on user acceptance
KPIs	SAP screen request response time [ms]
	Succeeded SAP elements [%]
Completion	Three times successfully executed compound test runs
	Workload according to business scenario with 20 simultaneously active VUs
	Overall percentage of succeeded SAP elements with error margin less than 1%
Existing data	Internal SAP GBI data
	1000 created dialog users for testing
Test data	Variable data via datapools
	Tool supported substitution of datapool candidates
Business scenario	Small business
	20 simultaneously active users
	Business processes according to GBI
Tests	Mix of transactions that trigger database read and write operations
	Make use of variable data from datapools
	Self-contained
Workload	Distributed workload via agents
	All users active from the beginning
Tool	IBM Rational Performance Tester
	IBM RPT Extension for SAP Solutions
	IBM RPT 100 Virtual Tester Packs

Table 1: Load Test Planning Profile

To execute this case studies 1000 student users already exist in the system. Data needed during test runs will be provided by datapools. Their access mode is defined in the execution schedule and guarantees data correlation if it is necessary. The workload should reflect a small enterprise with peaks of 20 simultaneously active users. The tests should contain transactions of different departments to ensure diversity.

They make use of data variation and offer variety in their transactional behavior. The execution schedule distributes the workload to ensure that measured results are not influenced by possible hardware bottlenecks of the host machine. The RPT supports all requirements for the conducted load test with the use of the SAP extension and additional licenses for virtual users.

4.2 Execution Phase

4.2.1 Test Environment

The mobile test laboratory consists of one host and four distributed agent workstations using static IPv4 addresses within the same local area network. The host which runs the workbench and is used to create, execute, control and monitor the test runs is a 64-Bit Windows 7 laptop computer and consist of an Intel Core 2 Duo CPU with 2.53GHz and 4GB of DDR3 memory. The distributed agents which emulate the workload also run Windows 7 64-Bit and consist of an Intel i5 processor and 4GB of DDR3 memory each. With the expected workload of 20 simultaneously active users and equal distribution one agent has to run 5 SAP GUI instances to simulate the polled workload.

The software on the host system includes installations of Java Development Kit (version 7 and higher), SAP GUI with enabled scripting (version 7.30 patch level 6), RPT (version 8.5.1), SAP Java Connector libraries and a Rational License Key Server (FLEXlm server). Besides a permanent license for the RPT the manually integrated licenses consist of an IBM RPT Extension for SAP Solutions and five RPT 100 Virtual Tester Packs which grant SAP performance tests with 500 simultaneously active users. To ensure the integrity of all services the Windows Firewall was deactivated and the ports for the MajorDomo service used to communicate with agents were forwarded.

The agent systems were prepared with the Java Development Kit (version 7 and higher), the SAP GUI with enabled scripting (version 7.30 patch level 6) and the RPT Load Generation Agent. The agents were configured to constantly poll for work from the host workbench.

The connection to the SAP ERP on SAP HANA system provided by the HPI requires the use of OpenVPN which was installed and configured on all machines. Although the measured KPIs are not influenced by moderate latency variability as mentioned previously, testing the latencies by sending packets to both SAP ERP instances indicated an equal distribution. However, running tests on the system using OpenVPN required patching the Windows hosts file. By manually mapping the host workstation to the dynamic IP address given successful FLEXlm license checks could be guaranteed.

4.2.2 Benchmark

The load test project consists of nine self-contained tests. Every test represents one or more processes of a single business department within the GBI. Tests can have datapool candidates which are substituted by variable data during a test run. A test starts with the SAP [SESSION_MANAGER] which requires login information regarding client, user, password and language details. The client value is changed manually when switching between the two SAP systems under test. Although password and language are also fixed values, the user name is substituted by the datapool variable [User] each time a test is run by a new virtual user. Then the specified transactions are executed by entering the stated transaction codes into the SAP Easy Access field. After finishing a unique course of actions, each test ends with pressing the log off button (Shift + F3) and confirming the security query. The nine tests are described in detail in table 2. Each transaction contains several recorded SAP Set, SAP Get, SAP Call and SAP Sequence elements, which playback the recorded scenarios and have to be successful regarding the KPIs mentioned previously. The SAP elements reflect a mix of queries with INSERT, UPDATE, DELETE, SELECT / SELECT* behavior and analytical processes. To ensure data correlation and that every agent-simulated virtual user gets a set of unique data, the opening mode of the resulting datapool was set to “segmented” (per machine). The sequential access of each row guarantees that all allocated data is used since the parameter “Fetch only once per user” was set to true and the number of rows given equals the amount of virtual users. Also wrapping when the last row is reached was deactivated.

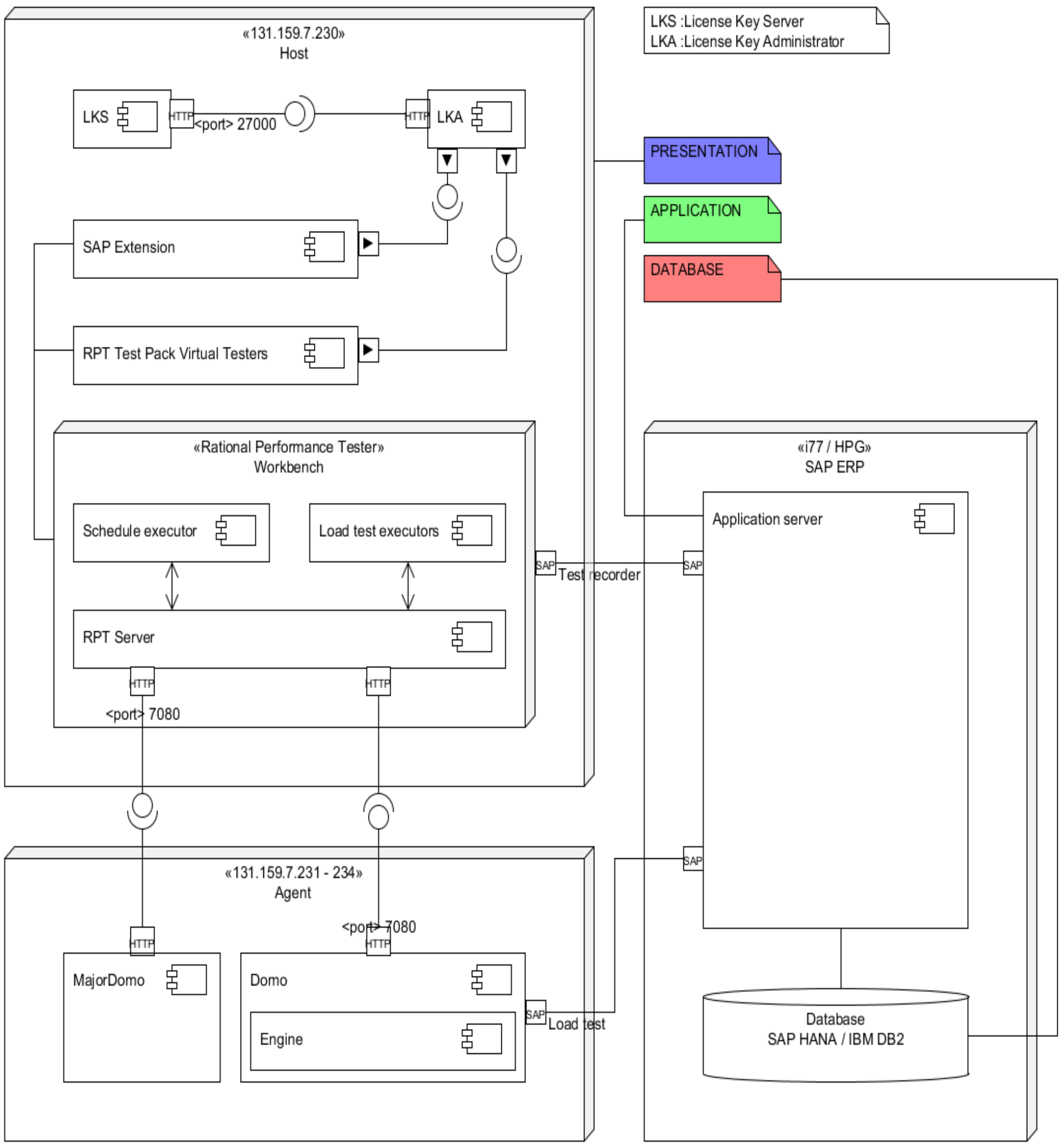


Figure 1: Test Environment

4.2.3 Workload

Within this project, various schedules with different total user group sizes and configuration parameters were executed for debugging and testing purposes. The workload profile illustrated by Table 1 includes 20 simultaneously active users which all start from the beginning without any delay caused by change rate or settle time. Each of the four user groups consists of one agent that runs 25% of the total user amount, so that each agent runs 5 SAP GUI instances. As defined by the stage duration, a run is only considered complete if all users successfully complete all process steps. The think time was set to a maximum value of 4 seconds which includes selection and typing. Since one multi user run executes each test only once, the elapsed runtime was about 15 minutes. This enabled reducing the sampling interval to 5 seconds which otherwise would cause possible performance problems when conducting long term load tests. All statistic, error, failure and warning logs were enabled while unneeded features like resource monitoring and response time breakdown (not available for SAP solutions) were deactivated. Since KPIs and performance requirements were observed separately, the performance requirements feature of the RPT was also deactivated.

4.3 First Result Analysis

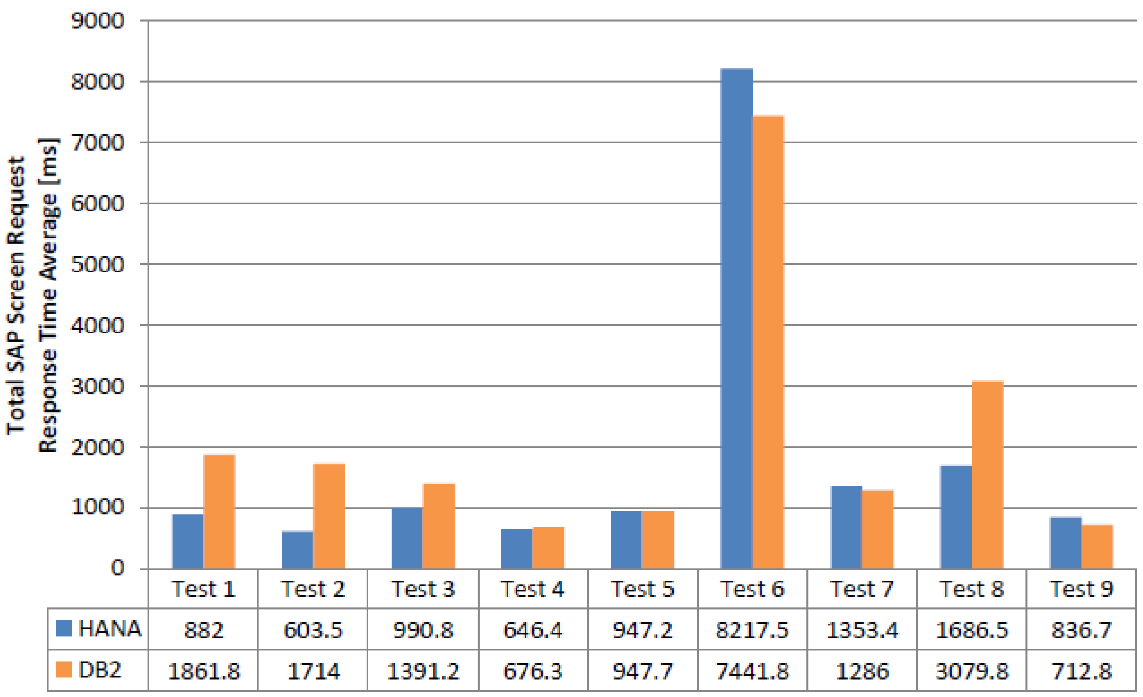
Different tests runs were executed to perform the comparison. Table 2 shows an overview over the individual tests devised for the analysis.

Test	Description	Used transactions
Test 1	The test changes the street number in the master records of an existing customer	[FD02] Change Customer (Accounting)
Test 2	The test uses the request value help to search for a range of values and displays two customer information tabs about an existing business partner	[XD03] Display Customer (Centrally)
Test 3	The test creates a complete purchase order for 60 off-road helmets per virtual user	[ME21N] Create Purchase Order
Test 4	The test shows a multilevel bill of materials (BOM) for a specific Deluxe Touring Bike (black) with two different views	[CS12] Explode BOM: Multilevel BOM
Test 5	The test displays a specific chart of accounts, an account list and searches for a specific G/L Account which is displayed with different tabs afterwards	[S_ALR_87012326] Chart of Accounts, [S_ALR_87012328] G/L Account List, [FS00] Edit G/L Account Centrally
Test 6	The test browses through the standard hierarchy of a specific controlling area and shows the labor cost element and the line items of the internal services cost center	[OKENN] Display Standard Hierarchy, [KA03] Display Cost Element, [KSB1] Display Actual Cost Line Items for Cost Centers
Test 7	The test searches for the organizational unit hierarchy of the Global Bike Group and expands its staff assignment structure further to a specific position	[PPOSE] Organization and Staffing Display
Test 8	The test displays the status of all storage bins in a specific warehouse.	[LX03] Bin Status Report
Test 9	The test creates an individual project for each virtual user	[CJ20N] Project Builder

Table 2: Overview of the performed workload tests

The main significance criteria during comparison were increases or decreases of response times, which exceeded a SAP screen request response time of 100ms.

First results showed that with an average of 451.9ms, the DB2 test run produced slightly better results than the HANA test run with 476.1ms. However, this was caused by a 31.37% higher overall standard deviation for all HANA results and is therefore not representative. Figure 2 illustrates the distribution of all average SAP screen request response times grouped by the tests executed. Grouping all results by their corresponding single tests makes the averages more resilient against outliers, which required summing up all measured average SAP screen request response times for each individual test. The chart shows significantly better results for the HANA test run in 4 out of 9 tests (Test 1, 2, 3 and 8). While 3 out of 9 tests (Test 4, 5 and 7) present only slight or no improvements at all, DB2 significantly outperforms HANA in 2 out of 9 tests (Test 6 and 9). However, this was again relativized in favor of SAP HANA by further analyzing the results. The total of all grouped averages further support this trend since HANA (16164ms) was able to reduce the response time by 15.42% compared to DB2 (19111.4ms). This includes an average of 283.58ms (HANA) and 335.29ms (DB2) for all test averages. Further work is necessary with more than 20 concurrent virtual users performing the test cases.



	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
🟦 HANA	882	603.5	990.8	646.4	947.2	8217.5	1353.4	1686.5	836.7
🟧 DB2	1861.8	1714	1391.2	676.3	947.7	7441.8	1286	3079.8	712.8

Figure 2: Total SAP Screen Request Time Averages Grouped by Tests

5 Conclusion

Following a refined three step approach, the planning phase revealed that executing a black-box test from an external point of view comes with constraints that dictate a certain load test planning profile. As a result, the performance of both systems was compared under the goal of user acceptance which was reflected by the percentage of succeeded SAP elements and the SAP Screen Request Response Times. The execution phase presented detailed information about the test environment, test structure, test implementation and schedule of the emulated workload. The test included multiple self-contained tests of different business domains and was carried out by four distributed agents which emulated a realistic minimum load of 20 simultaneously active users. The results of the multi user test showed, that both test runs were able to finish successfully with a percentage of succeeded SAP elements that complied with the error margin of 1%. The completion phase verified these circumstances and analyzed the measured SAP screen request response times. The results showed that under the given circumstances, the system based on SAP HANA was able to reduce the total of all test-wise grouped average SAP screen request response times by 15,42% compared to the results of SAP ERP on IBM DB2. Further analyzing these results supported this trend and showed that 4 out of 9 tests were in favor of HANA while 3 tests were considered insignificant and 2 tests were in favor of DB2. Future work intends to repeat the load test with more than 20 simultaneous users in order to enrich the current results. Furthermore, we plan to conduct similar test cases to compare the performance of SAP BW systems based on DB2 and HANA databases. The last research step would then consist of deploying a SAP BW alongside with a SAP ERP on the same host system (and respective database) in order to test the performance of a concurrently running mixed (OLAP + OLTP) workload on both database technologies. The workloads from the prior research phases will be reused in this last step. In order to be able to complete this agenda, we applied for a project period extension.

References

[1] DeWitt, D.J.; Katz, R.H.; Olken, F.; Shapiro, L.D.; Stonebraker, M.R.; Wood, D.A. (1984): Implementation techniques for main memory database systems. *Presented at: 1984 ACM SIGMOD international conference on Management of data*, Boston, Mas-sachusetts, p. 1-8.

[2] Plattner, H.; Zeier, A. (2012): In-Memory Data Management: Technology and Applications, Springer-Verlag, New York 2012.

[3] Plattner, H. (2009): A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database. *Presented at: 2009 ACM SIGMOD International Conference on Management of data*, Providence, Rhode Island, USA, p. 1-7.

[4] Krueger, J.; Grund, M.; Boissier, M.; Zeier, A.; Plattner, H. (2010): Data structures for mixed workloads in in-memory databases. *Presented at: 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT) 2010*, p. 394-399

[5] Helfen, M.; Trauthwein, H.M. (2011): Testing SAP Solutions. (2 Ed.), Galileo Press, Boston 2011.

[6] Cheng, X. (2008): Performance, Benchmarking and Sizing in Developing Highly Scalable Enterprise Software. In: *Performance Evaluation: Metrics, Models and Benchmarks* (Vol. 5119). Eds.: Kounev, S.; Gorton, I.; Sachs, K. Springer, Berlin Heidelberg, 2008, p. 174-190.

HPI Vulnerability Database: Integration and Analytics of Vulnerability Information

Report for the Project
"Towards an Integrated Platform for Simulating, Monitoring, and Analytics of SAP Software (Second Phase)"
in 2013 Fall at HPI FutureSoC Lab

Marian Gawron, Andrey Sapegin, Feng Cheng, Christoph Meinel
Hasso-Plattner-Institut
University of Potsdam
14482, Potsdam, Germany
{marian.gawron, andrey.sapegin, feng.cheng, meinel}@hpi.uni-potsdam.de

Abstract

A new data structure of vulnerability information to combine human understanding with machine-readability and automated analytical functions is proposed in the following. This standardized model is needed to integrate information of different sources of vulnerabilities. Additionally automated features, which take advantage from the comprehensive collection of vulnerability information, are used to aid humans in securing their infrastructure. We created a suitable data structure to allow insights in vulnerability details, as well as perform automated security analytics due to machine-readability of vulnerability information.

1 Introduction

Security in IT-Infrastructure is an important and complex topic nowadays, since affairs like spying out personal data, leaking user account data and several other recent problems point out security weaknesses in nearly every infrastructure The amount of nodes in a network, the number of programs on each individual host increase dramatically. Thus it is not feasible to maintain an overview about all possible security risks manually. Therefore a computer-aided solution is needed to allow analysis of security leaks of single computers and networks. In addition we do not want to lose the possibility of human readability of vulnerability information. So we created a data structure which is usable by humans and machine-readable at the same time.
The main contributions of this report are organized as follows.

- Creation of Vulnerability Attributes.
- Unification of Vulnerability Information from different Sources.
- Results and Achievements: In this section we explain the benefits from using FutureSOC Lab as well as the testing environment

2 Creation of Vulnerability Attributes

Vulnerabilities are represented as objects with several attributes, such as description, references, identifier, cvss-attributes [2], preconditions, and postconditions. These attributes need to be stored and are queried frequently during the analysis procedures. Thus an In-Memory technology is well suited for this issue. This technology allows a real-time processing in detecting vulnerabilities on single machines as well as on networks. Most of the attributes of a vulnerability are used to keep the possibility for human experts to receive information about vulnerabilities. The preconditions and postconditions are the main innovations of this data structure. These conditions are used for further analytical methods.
The requirement was to create a data model, which allows automatic reasoning of the existence of vulnerabilities. This characteristic is called preconditions, because those conditions have to be fulfilled to assure the existence of the vulnerability, e.g. certain programs has to be installed on the target system. Further other preconditions have to be met to exploit the vulnerability, e.g. the attacker has to be in the specified range. The automatic reasoning uses logic data structures, which is provided by the machine-readability of the conditions. Additionally the preconditions should also be available in a human readable format. Those two requirements, which were partly contradicting were figured out. The machine-readability requires structured format, whereas the usage of simple unstructured

text is appropriate for human understanding. Finally the solution is to create a set based structure which is able to fulfill all requirements in terms of machine-readability. These sets have a low complexity with self-explaining key words. Because of that it is possible that humans can understand the information as well. Those sets consist of subsets and/or primitive elements, which are connected with logical operators. The operator specifies whether all conditions of a set has to be fulfilled or if one is enough. The primitive elements are composed of key and value pairs to transform them to XML during the export.

2.1 Creation of preconditions

The conditions are created during the import phase for each vulnerability. The preconditions are generated based on the given information. For each vulnerability the programs of it are connected with an ”or” operator and appended to the preconditions. Further the given ranges, which are connected with an ”or” operator are appended as well. Finally an influence on active resources which is the program running on the system is the last part of the preconditions. In this case the influence on active resource has the value ”Input”. That influence describes the possibility of an attacker to produce input to the running program, which is a requirement to exploit the weakness of the program. The resulting set contains those subsets, which are linked with an ”and” operator. The preconditions are composed of the following necessary elements:

- To exploit the vulnerability at least one of the specified programs has to run on the target machines
- The attacker has to be in the given range of the target
- The attacker has to be able to send input to the machine and the running services

2.2 Creation of postconditions

After the preconditions are successfully created the postconditions, which specify the status after a successful exploitation, are generated out of the preconditions and the violation of security goals. Basically the focus lies on the impact of availability, confidentiality, and integrity. One can imagine that a violation of availability leads to an influence of the existence of the program, because the availability guarantees the existence of the service, which is offered. The confidentiality is significant to unauthorized read access of data. The violation of confidentiality enables the attacker to gain secret or critical data, which could be used for further attacks, e.g. login information of a target system. Nevertheless the most important issue is the integrity of data, since the modification of data in the system enables attackers to inject malicious code, which is executed with the rights of the service that runs on the target system. Hence a violation of integrity leads to read, write, and delete influence of data. Additionally the range of the attacker is changed if he is able to execute malicious code on the target machine, since he could use this machine as a proxy host for secondary attacks without the need to attack from outside of the network any more. Thus the attack range local is added to the postconditions in this situation. So postconditions consist of the following elements that specify the state of the target after a successful exploitation of its vulnerability:

- The specified programs that run on the target machine
- The influences of an attacker on those programs
- The possibility to access data on the target machine
- The influences on passive resources, which describes the art of data access
- The range an attacker has after a successful exploitation

A disadvantage of this structure is that the hierarchy increases fast. For example every CPE-ID [**?**], which identifies a program, is located in a subset of the condition set. So the programs, which are connected to the vulnerability, occur at the third level of the structure for the first time. This results in 3 lookups if a query for programs of a vulnerability has to be executed. Additionally every vulnerability has its own conditions. Hence the search for all vulnerabilities of a specified program results in a complex query. First every vulnerability is scanned if its conditions include the program, which leads to the mentioned three queries per vulnerability. Since one major requirement is to search for vulnerabilities with a specified program, another object to represent the programs is needed and has to be connected to the vulnerability. The other elements of a condition would not be queried frequently and the conditions already have the form which was required in terms of machine readability. Thus the conditions remain unchanged whereas an additional object, which holds the CPE-ID again, is created and linked to the vulnerability directly.
The present deployment of HPI-VDB [3] on the FutureSOC Lab has the following features listed below. The latest changes will be added as part of the next deployment phase throughout the next six months.

- Creation of conditions on the fly
- In-Memory based platform with up to 2 TB of main memory
- Multi-Core support with thousands of cores

- Multi-processing during import and CPE-ID checking
- Visualization of attack scenarios in networks and threats on single machines

3 Unification of Vulnerability Information from different Sources

Due to the requirement of multiple sources a suitable merge function has to be established. The basic idea is to check whether a new vulnerability is already listed in the database or whether it has to be created. If the vulnerability is listed an update of its characteristics is sufficient. If it is not listed it has to be integrated as a new vulnerability. The first idea is to check for the identifier. If a new vulnerability has a CVE-ID then one can search for it in the database, because the CVE-ID is commonly used by several databases like NVD [6] or CVE [5]. But considering the fact that vulnerabilities of multiple sources do not always have CVE-IDs, such as some vulnerabilities listed in the OSVDB [4], this method is not applicable for all possibilities. In this case it is possible that maybe a Secunia-ID or an OSVDB-ID [4] is the only identifier of a vulnerability. At this point the problem was fixed with the option to have any ID as an identifier, because the only requirement of the identifier is that it is a unique string. So if a vulnerability has no CVE-ID it could still be stored in the database using a different identifier, such as Secunia-ID. The main problem is to figure out if a vulnerability with a missing CVE-ID is a new vulnerability, which does not have a CVE-ID yet or if it is an already registered vulnerability where the CVE-ID is just missing in the source. An idea to deal with this issue was the usage of descriptions of the vulnerability. The textual description was meant to be unique for every vulnerability, since the vulnerabilities differ in their targets, violations, and affected programs, which are normally part of the textual description. This intention seems to have good chances to be proved with simple SQL queries to verify the uniqueness of descriptions of vulnerabilities. The SQL queries which were used to prove the intention are specified in the following. The queries were sent to the database containing 56700 vulnerabilities.

Listing 1. SQL queries to verify Uniqueness of descriptions of vulnerabilities

```
(1) select count (description)
    from vulnerabilities_vulnerability;
Result : 56700

(2) select count (distinct description)
    from vulnerabilities_vulnerability;
Result : 56208

(3) select count (*) from
    vulnerabilities_vulnerability
    group by description
    having count (*) >1;
Result : a list of counts of
identical descriptions:
11, 17, 2, 2, ..... ,64, 2, 11, 3, 2

(4) select group_concat ( distinct
    identifier separator ',') as id ,
    count (*) , description
    from vulnerabilities_vulnerability
    group by description
    having count (*) >1;
Result : a list of tuples
(list of identifiers , description
 and occurrences of it)
```

Query 1 shows the total amount of descriptions, which is equal to the number of vulnerabilities at the given time. The second query shows the amount of different descriptions in the database, which was expected to generate the same result. The difference of those numbers is produced by duplicate descriptions. The third and fourth query were executed to ensure that the duplicates are not only NULL values, which would mean that the description is just missing. Thus a list of duplicate descriptions could be produced with their occurrences and their individual identifiers from the fourth query. This list contains 492 vulnerabilities whose descriptions are not unique. These results are used to guarantee that the potential duplicates are real ones with different identifiers but identical descriptions. Finally it was proven that the identification of a vulnerability could not be done with the descriptions if CVE-IDs are missing or absent.

Another possible solution was to have a closer look at preconditions and postconditions of the new vulnerability. The conditions could be created temporarily and afterwards the system could search for identical existent conditions in the database. That means that an already listed vulnerability has to have the same preconditions and postconditions as the temporary ones. At first this strategy seemed successfully, since the affected programs and the violated security issues have to match. Thus a check whether all listed vulnerabilities have different preconditions and postconditions had to be performed. Therefore one iterates over the complete database which contains 56700 vulnerabilities. Afterwards a script iterates over all tuples of preconditions and postconditions of each vulnerability and checks the uniqueness. This analysis produced the result that there are duplicates, which means that different vulnerabilities have the same preconditions and the same postconditions. For example the vulnerabilities CVE-2013-0358 and CVE-2013-0352 have the same affected products and the same CVSS vector, which leads to identical CVSS characteristics [2].

This insight was confusing since it shows that different vulnerabilities have the same requirements and identical effects to a target system. So if all requirements (preconditions) of the vulnerability are the same and the impact on the target (postconditions) also match, it would be difficult to decide whether one of the vulnerabilities exists on a system or if both are present. Since the conditions match it is not possible to detect the major differences between the vulnerabilities with the current characteristics of the vulnerability. Thus the automated analysis of a system will detect both. Since neither the description nor the tuple of precondition and postcondition are sufficient to identify a vulnerability without its identifier, the next step was to combine both properties. The new idea was that if the preconditions and postconditions are identical as shown in the example of CVE-2013-0358 and CVE-2013-0352 an additional check for unique descriptions is done. Duplicates like the example could be solved and the vulnerabilities could be identified since the description of the vulnerabilities in this example is different. Thus the amount of duplicates of vulnerabilities which were identified by this method would be reduced dramatically. The problem which occurred was that some vulnerabilities could not even be identified now. The method to find duplicates with previously specified identification characteristics works on the whole database. The objects are handled one after the other and the characteristics, which have to be tested, and the identifier are stored in the memory. The identifier is used to manually examine the vulnerabilities if a duplicate is found and to create a list of duplicates. For each entry the precondition is concatenated with the postcondition. Then a check whether the concatenated conditions are already in the list is done, which results in duplicates in terms of conditions. If this check is positive another test for an exact match of description of the current item and corresponding description of the potential duplicate is performed. If both checks are positive the conditions are equal and the corresponding description, which is the description of the same vulnerability, are identical.
The vulnerabilities CVE-2011-0662 and CVE-2011-0666 have the same description, the same preconditions, and the same postconditions. So these two vulnerabilities differ only in their references:

- CVE-2011-0662 links to http://osvdb.org/71740
- CVE-2011-0666 links to http://osvdb.org/71742

However the identification with preconditions, postconditions, and description worked quite good, since the number of duplicates, which were found is 390. This amount of duplicates is low considering the total number of 56700 vulnerabilities. So the identification has an error rate of less than 1%. A possible solution to correct the identification would be to include the references to the possible identification characteristics. But if a vulnerability has the preconditions, postconditions, descriptions, and the references in common with another one there is no additional benefit to enrich the existing data. Thus it is only possible to accurately identify a vulnerability without its identifier by a complete list of its other characteristics.
At this point further research to integrate additional properties of vulnerabilities would be required to create a suitable identification of vulnerabilities. This identification could be used to achieve the capability of merging different sources without creating duplicates in the case of missing CVE-IDs. Additional the resulting duplicates, which differ only in references to other sources or documentations has to be examined more accurate. Because these vulnerabilities were checked by several experts with the result to create multiple entries of vulnerabilities in the database. So there has to be a difference since otherwise these vulnerabilities would not have been created with different IDs. Thus further investigation is needed to depict the differences and integrate the different characteristics in the data structure to consider them for future analysis.

4 Analytics

The machine-readable conditions of the vulnerabilities allow us to create an automated detection of vulnerabilities. The only information the system needs is a list of all installed applications and network configurations as detailed as possible. With this information we can predict all vulnerabilities which are present on the system. Therefore we check the preconditions of all vulnerabilities for matches with the system which is evaluated. Since we know the software and network configuration we can reason the fulfillment of each of the preconditions. If one set of preconditions is fulfilled the vulnerability is present on the system. The In-Memory technology enables a fast processing of this condition check. Furthermore the network configuration allows us to detect risks which are remotely exploitable which results in a higher risk, since the attacker could be located anywhere in the network. Finally we can show the results to the user who should perform counter measures against the identified security leaks.

5 Results and Achievements

We created a data structure of vulnerability information which could be investigated by humans and by automated tools. So we still provide the possibility for expert users to explore vulnerability information by themselves. In addition we are able to create analytical features for automated detection and correlation of vulnerabilities on single machines as well as in networks. We also merged information of multiple databases and investigated a possible identification of vulnerabilities without using the identifier. This method could be used in the case of absence of the identifier as well. This comparisons are quite intensive in terms of performance but the hardware of Future-SOC Lab allows fast processing for this complex tasks. We also created a testing environment with multiple honeypots to check the correctness of our security analytics. In this environment the honeypots are located in a network and scanned for vulnerabilities automatically. Afterwards all possible security risks are evaluated and possible attack points are highlighted. Due to the postconditions of the vulnerabilities it is possible to identify attacks with multiple steps. Thus we will also detect risks where one vulnerability is exploited to create the environmental factors that are needed for another vulnerability. This test environment allowed us to prove that the detected vulnerabilities are present and test whether an exploitation as it is described and detected by our security analytics is possible.

6 Conclusion

As we work to secure networks and single machines, we think the first step is to identify all possible security risks in a system. Additional the integration of information from multiple sources into one standardized data structure allows us to deal with comprehensive information and a complete list of registered vulnerabilities from at least one other source. The unification mechanism has to be tested further to reduce the error rate of vulnerability identification. The reasoning of vulnerability characteristics allows us to automate the detection of security risks. Thus it was shown that the analytical features benefit from our data structure and allow a fast scan of the system which is a major advantage for administrators since they can automatically check for security leaks now.

References

[1] Sebastian Roschke, Feng Cheng, Robert Schuppenies, and Christoph Meinel: Towards Unifying Vulnerability Information for Attack Graph Construction , in Proceedings of 12th Information Security Conference (ISC'09), Springer LNCS, Pisa, Italy, pp. 218-233 (September 2009)

[2] Peter Mell, Karen Scarfone and Sasha Romanosky *Common vulnerability scoring system* Security & Privacy, IEEE 4, 85–89 (2006)

[3] Hasso-Plattner-Institut *HPI Vulnerability Database* Available from: https://www.hpi-vdb.de accessed: 25.02.2014

[4] Open Source Vulnerability Database Available from: http://www.osvdb.org accessed: 27.02.2014

[5] Mitre Corporation *CVE List Main Page* Available from: http://nvd.nist.gov/ accessed: 27.02.2014

[6] National Institute of Standards and Technology *National Vulnerability Database* Available from: http://nvd.nist.gov/ accessed: 26.02.2014

[7] Mitre Corporation *Common Platform Enumeration* Available from: http://cpe.mitre.org/ accessed: 26.02.2014

Open Government Data Integration with Stratosphere on the FutureSOC 1000-core cluster

Arvid Heise
Hasso-Plattner-Institut
arvid.heise@hpi.uni-potsdam.de

Felix Naumann
Hasso-Plattner-Institut
felix.naumann@hpi.uni-potsdam.de

Abstract

Integrating data from multiple data sources enhances their value for businesses and organizations. In this project, we integrate large Open Government datasets to find interesting relationships between politicians and companies, such as potential cases of nepotism.
We devised data integration operators for the parallel data analysis framework Stratosphere, which we evaluate on the 1000-core compute cluster of HPI's Future SOC Lab. In particular, we compare the scale-up to the scale-out capabilities of Stratosphere and our implementations.

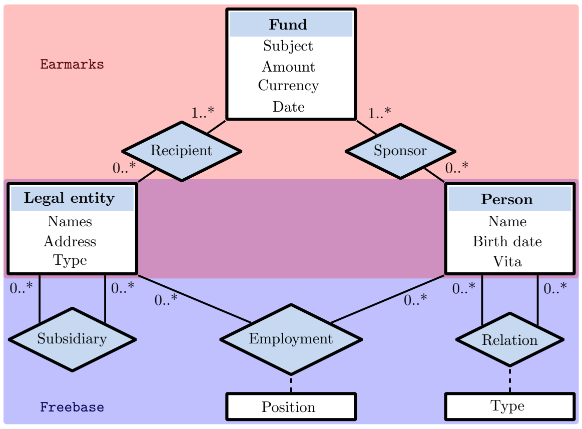


Figure 1. Integrated data model.

1. The power of integrated Open Government datasets

Data plays an important role in today's organizations - either directly as an asset, most prominently seen in Google, or as the driver for business decisions. Data is typically collected through several sources and applications, such as customer relations tables, sales reports, or data derived from suppliers. For high tech companies, the acquisition of data is, next to the acquisition of intellectual property and strategic reasons, the main motivation for buying other companies.
In this project, we integrate large Open Government datasets with our data integration operators implemented in Stratosphere [1]. A possible usage of the integrated dataset is to find interesting relationships between politicians and companies, such as potential cases of nepotism [2]. Figure 1 exemplarily shows the resulting data model of an integration of the US Earmarks[1] data source and Google's Freebase[2].
The first data source contains earmarks: Personal spending of a US congress member to an organization. We extract information about the receiving legal entity, the enacting congress member, and the fund itself.
To detect suspicious cases, we lack information about possible connections between the recipient and the sponsor. In Freebase, we find familiar relationships, employment records, and subsidiary information.
We now need to integrate the two data sources to find suspicious circular relationships: A congress member enacted an earmark that benefits a legal entity, at which the congress member or a relative is employed.

[1]`http://earmarks.omb.gov/earmarks-public/`
[2]`http://freebase.org`

2. Integration process

Data integration consists of several subtasks with different complexity, which addresses different types of heterogeneities. In the following we briefly present each subtask for our Open Government Data integration query. The complete workflow is summarized in Figure 2.

2.1. Data scrubbing

One of the most underestimated challenges in the integration of several data sets is often systematic heterogeneity on the value and record level.

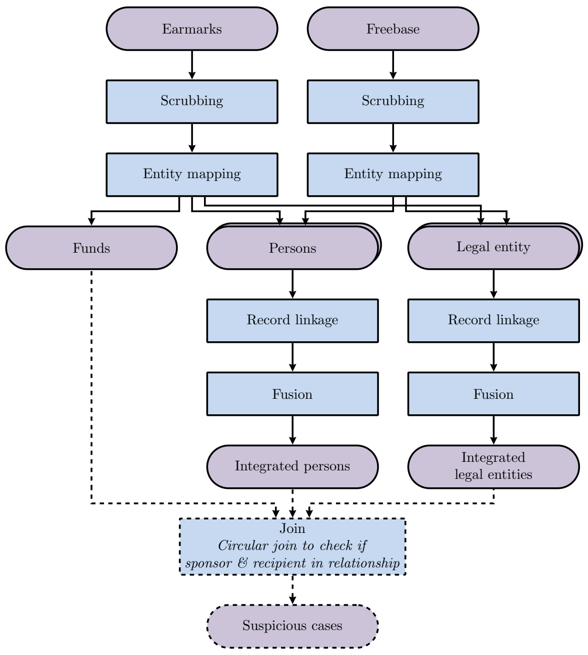


Figure 2. Data integration query for potential cases of nepotism

For example, differently abbreviated street names render matching techniques of records more complex as they need to be able to detect such differences and at the same time avoid being too lenient, which would results in many wrong matches.
Our data scrubbing operator allows users to declaratively define constraints on the values and repair functions that correct violations. For example in the script excerpt below, each politician needs to have an id, name, and party. Any record without id and name are discarded, while missing parties are repaired with a default value. Further, names are normalized with a custom function that honors the official titles given in a dictionary.

```
$pol_scrubbed = scrub $politicians
  with rules {
  id: required,
  name: [required,
    normalizeName(officialTitles)],
  party: required?: default(""),
};
```

2.2. Entity mapping

The next operator addresses schematic heterogeneities. The user declaratively specifies the mappings of attributes of the source relations and the target relations. Our framework then finds the minimal number of transformation operators that aligns the schemata accordingly.
For example, the previously normalized and split name is now assigned to separate attributes of a

```
$politician, $party = map entities of
    $p in $pol_scrubbed, $t in $tenure
  where $pol_scrubbed.party == $t.id
  into [
  entity $politician
    identified by $p._id
    with {
      firstName: $p.name[0],
      lastName: $p.name[2],
      worksFor: [{
        legalEntity: $t.party
      }]
    }...
];
```

person. Further, our operator supports nested expressions and arbitrary cardinalities as can be seen in the *worksFor* relationship to the parties.

2.3. Record linkage

The most compute-intensive and hardest part of the data integration is to find corresponding records across data sources. It usually comprises three tasks:

- Select candidate record pairs that have a high probability of representing the same real world entity.
- Apply sophisticated (string) similarity measures to the candidate to decide if it is an actual match.
- Cluster the matches to a consistent, transitively closed result. This step usually means the addition of more matches, but may also involve the deletion of borderline cases.

All three tasks can be seen in the following script. We define a weighted, composite similarity measure in the *where* clause, which compares first and last names as well as the first employers. We then provide a sorting key hint that results in a two pass sorted neighborhood method on the first and last name respectively. Finally, we specify the result should be transitively closed.

```
$persons = link records
    $c in $usCongressPersons,
    $f in $freebasePoliticians
  where
   (5*jaro($c.firstName,$f.firstName)+
    5*jaro($c.lastName,$f.lastName)+
    compareParty(
      $c.worksFor[0].legalEntity,
      $f.worksFor[0].legalEntity))/11
   > 0.75
  sort on [$c.firstName,$c.lastName]
  with window 20
  cluster with "transitive closure";
```

For current and future evaluations, we consider altogether three common candidate selection implementations that we parallelized:

- Naïve comparison based on the Cartesian product of both sources.
- Partitioning of the data on one or more attributes or parts thereof.
- Sorting on a calculated value from one or more attributes.

2.4. Data Fusion

Finally, for each cluster of representations of the same real-world entity, we want to obtain a single, consistent representation. Our data fusion operator enables users to declaratively specify conflict resolution functions for attributes. These functions decide which of the potentially conflicting values of the different representations to choose in the final representations.

```
$fused_persons = fuse $persons
  with weights {
     freebase: .7
  }
  with resolutions {
     *: [mostFrequent, longest, first],
     firstName: vote(&isNickNameOf),
     birth: smallest,
     worksFor: mergeDistinct
};
```

3. Intermediate results

We evaluated the scalability of our integration operators on the 1000-core compute cluster of the Future SOC Lab. The cluster consists of 25 Quanta QSSC-S4R, each having 4 Intel Xeon E7- 4870 @ 2.40GHz and 1024 GiB RAM. The nodes are connected through a 2 x Intel Corporation 82599EB 10-Gigabit network. Since local storage was unavailable for us, we simulated it with a 200 GiB Ramdisk.
We used Stratosphere v0.4 and the Cleansing package v0.1 and assigned 500 GiB to each task manager of Stratosphere.

3.1 Datasets

We use the datasets shown in Table 1. Datasets with more columns pose additional computational challenges due to the lenient and rather slow string comparison functions.

Dataset	Size	Tuples	Attributes
Freebase			
Politician	12.8 MiB	40250	7
Party	3.0 MiB	8097	6
Tenure	5.3 MiB	25811	6
Earmarks	21.4 MiB	43524	37

Table 1. Datasets characteristics

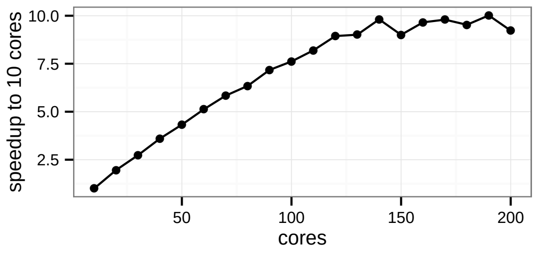


Figure 3. Speed up in comparison to 10 cores

3.2 Scaling overall parallelism

In the first experiments, we evaluated the overall scalability of the query on Stratosphere. We therefore executed the same query with a different degree of parallelism as shown in Figure 3.
With ten cores, the query took 3:57h to complete. Using 100 cores, the processing time went down to 31 min, resulting in a good speed-up of 7.6. However, further increase of the degree of parallelism did not yield significantly more benefit. Especially after 130 cores, the time stagnated.
Either the Stratosphere framework itself or our data integration operators hit the scalability limit for the given query size.

3.3 Scaling overall parallelism

In the next experiment, we distinguished between scale up and scale out. Usually, scale up should be better to a certain degree than scale out, because no network traffic is involved. Only after one node is overloaded, scaling out should be better.
The overall degree of parallelism (DOP) is the product of the number of worker threads and the total number of nodes used. We evaluated the same overall DOP by changing the number of nodes and adjusting the number of threads accordingly as shown in Table 2.
Surprisingly, the best overall runtime occurs with the highest number of nodes and thus probably the highest network traffic. The more we shifted the work towards individual nodes, the worse the performance. While the last experiment might also

Nodes	Threads	DOP	Time (s)
25	8	200	1542
10	20	200	3891
5	40	200	7796

Table 2. Same DOP but varying number of nodes and intranode DOP

be slower because of the fully used hyperthreaded CPUs, the more probable explanation is a suboptimal usage of local resources.

4. Next steps

Obviously, in the first step, the Stratosphere team needs to identify and fix the performance bottleneck when scaling up. Some of the problems might be alleviated when using larger data sets.
Nevertheless, we would also like to extend the previous experiments in the following three directions.

4.1 Advanced record linkage

We are currently verifying that our sophisticated, parallelized record linkage techniques produce the same result in our local test bed. Once we verified the correctness of our techniques, we would like to evaluate the scalability in comparison to the naïve approach.

4.2 Larger data sets

With the advanced techniques, it should be feasible to quickly integrate the US Spending data sets, which contains all enacted funds of the US. This data set comprises tens of millions of records each year and will help to evaluate the scalability of each operator.

4.3 Reordering of operators

One of the goals of Stratosphere is to provide an extensible, yet powerful data analytic system. Inspired by current DBMS, we would also like to reorder operators to gain performance.
To evaluate such an optimization, we would need to run several equivalent queries and measure their performance.

Acknowledgements

We would like to thank the Master students Tommy Neubert and Fabian Tschirschnitz that worked as part of the data integration team on the Stratosphere project.

References

[1] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/pacts: a programming model and execution framework for web-scale analytical processing. In *SoCC*, pages 119–130, 2010.

[2] A. Heise and F. Naumann. Integrating open government data with stratosphere for more transparency. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14(0):45–56, 2012.

SQL-based Data Profiling on SAP HANA in the HPI Future SOC Lab

Claudia Exeler[1] Thorsten Papenbrock[2] Felix Naumann[2]

[1] firstname.lastname@student.hpi.uni-potsdam.de
[2] firstname.lastname@hpi.uni-potsdam.de
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

Abstract

SQL is sometimes used for data profiling but is considered inferior to specialized algorithms. This report gives an overview of how different profiling tasks can be solved with SQL. The evaluation on SAP HANA in HPI's Future SOC Lab shows that for selected characteristics, SQL is equally good as or better than state-of-the art algorithms when executed on an in-memory column-store database. The more complex tasks, however favor algorithmic solutions.

1 Motivation

The aim of data profiling is to automatically collect statistics about large data sources, for which only little or no knowledge about the metadata is available. This project considers the following typical profiling characteristics:

- Minimal and maximal values
- Median value
- Value distributions
- Inclusion dependencies (INDs)
- Unique column combinations (UCCs)
- Functional dependencies (FDs)

For each of these characteristics, specialized algorithms exist. However, they can also be solved by using specialized SQL queries. The problem with previous SQL-based profiling solutions is their poor performance and the resulting inability to handle large amounts of data. However, this has only been evaluated on row-oriented databases yet. So we investigate whether SQL is a more appropriate solution when executed on in-memory column-oriented databases. SAP HANA as a column-store is well suited for analytical queries and might therefore execute typical profiling queries much more efficiently.
This project shows that SAP HANA as a representative of an in-memory column-store is indeed more suited for profiling than traditional row-based database systems, and that it executes selected profiling tasks faster than common algorithms, but is still outperformed on complex metrics.

2 SQL-based Profiling

This section gives an overview of SQL queries that solve the mentioned profiling tasks. Whereas the single column metadata can be calculated directly, for multi-column metadata like inclusion dependencies, unique column combinations and functional dependencies, one has to generate candidates and then check whether they hold. Thus, a driver program is necessary that generates the appropriate queries.

2.1. Single Column Statistics

For some single column metadata, such as minimum, maximum, and average, SQL provides predefined functions. They can thus be retrieved with the following simple query per column:

```
SELECT min(col1),
       max(col1),
       avg(col1)
FROM   table1;
```

Listing 1. SQL query for retrieving minimum maximum and average of a column

Only a single query is necessary to retrieve the minimum, maximum, and average of *all columns*, because the SELECT clause in Listing 1 can contain the expressions for all columns. Other metadata, such as a complete value distribution, are not supported by SQL and require a separate query for each column:

```
SELECT   count(*) AS frequency
FROM     table1
GROUP BY col1
ORDER BY col1;
```

Listing 2. SQL query for retrieving a histogram of a column

The above query returns a histogram ordered by the values in the inspected column; to sort the histogram by the values' frequencies, the last row needs to be `ORDER BY frequency`.

For calculating the median values, one query per column is needed as well:

```
SELECT   col1
FROM     table1
ORDER BY col1
LIMIT    1
OFFSET   x;
```

Listing 3. SQL query for retrieving the median of a column

The variable x is the index of the median in the sorted list and has to be calculated in beforehand once for each table using the total number of tuples n. For even n, this is $x = n/2$ for the lower median or $x = n/2+1$ for the upper median, and for odd numbers of tuples it is $x = (n+1)/2$.

2.2 Inclusion Dependencies

The following four queries validate whether the unary inclusion dependency $depCol \subseteq refCol$ holds. All queries can also be extended to validate multi-column INDs.

The most intuitive but slowest of all investigated queries uses the `NOT IN` operator. The IND holds if and only if the result (`unmatched`) is 0:

```
SELECT count(*) as unmatched
FROM   depTable
WHERE  depCol IS NOT NULL
AND    depCol NOT IN
       (SELECT refCol
        FROM   refTable)
LIMIT  1;
```

Listing 4. Validating an IND using Not In

The approach in Listing 5 uses the `JOIN` operator and actually consists of two queries per candidate. The IND holds if and only if `matched` is equal to `n`.

```
SELECT count(DISTINCT depCol) AS n
FROM   table;

SELECT count(*) AS matched
FROM   (SELECT DISTINCT depCol
        FROM   depTable)
JOIN   (SELECT DISTINCT refCol
        FROM   refTable)
ON      depCol = refCol;
```

Listing 5. Validating an IND using Join

In contrast to IND detection algorithms, this query does not use any clever mechanisms to stop the matching of two columns early on as soon as an inclusion is disproved by a counter-example. However, it has the advantage of using the join operator, which is highly optimized in most database systems.

The query in Listing 6 attempts to stop as soon as a non-matching value is found. This is advantageous when only few INDs exist.

```
SELECT count(*) AS unmatched
FROM
    (SELECT to_char(depCol)
     FROM   depTable
     WHERE  depCol IS NOT NULL
     EXCEPT
         (SELECT to_char(refCol)
          FROM refTable)
     LIMIT  1);
```

Listing 6. Validating an IND using Except

If the IND holds, `unmatched` is 0. The `to_char` conversion can be omitted if the columns have the same or comparable data types.

Another query that returns the same results uses the `NOT EXISTS` operator:

```
SELECT count(*) AS unmatched
FROM   depTable
WHERE  depCol IS NOT NULL
AND NOT EXISTS
       (SELECT 1
        FROM   refTable
        WHERE  depCol = refCol)
LIMIT  1;
```

Listing 7. Validating an IND using Not Exists

This query's execution is very similar to if a `LEFT OUTER JOIN` is used, which is why this variation is not listed separately.

2.3 Unique Column Combinations

To check if a column combination is unique, two major approaches exist: One uses the GROUP BY operator, and another is based on the DISTINCT operator. The queries are given for the example column combination $\{col1, col2, col3\}$:

```
SELECT   count(*) AS notUnique
FROM     table1
GROUP BY col1, col2, col3
HAVING   count(*) > 1
LIMIT    1;
```

Listing 8. Validating a UCC using Group By

If notUnique is 0, the candidate is unique, but if it is 1, a duplicate value has been found.

The DISTINCT approach needs the total number of rows in table for comparison, and the column combination is unique if the result of the query (distinctCount) is equal to the number of rows:

```
SELECT count(*) AS distinctCount
FROM
  (SELECT DISTINCT col1, col2, col3
   FROM   table1);
```

Listing 9. Validating a UCC using DISTINCT

For both approaches, one query can check the uniqueness of only one candidate. All single-column uniqueness checks, however, can be combined into a single query using the distinct count approach as follows:

```
SELECT count(*) AS totalCount,
       count(DISTINCT col1) AS c1,
       count(DISTINCT col2) AS c2,
       [...]
FROM   table1;
```

Listing 10. Validating all single-column UCCs in one query

Those columns where the distinct count (c1, c2, etc.) is equal to totalCount are unique. This idea can be extended to also validate multi-column candidates by concatenating the values, separated by a character that does not occur in any of the columns' values.

```
SELECT count(*) AS totalCount,
       count(DISTINCT col1) AS c1,
       count(DISTINCT
        CONCAT(col1,'#',col2)) AS c12,
       [...]
FROM   table1;
```

Listing 11. Using concatenation to validate all UCCs in one query

2.4 Functional Dependencies

For functional dependencies, we also identified two alternative SQL queries that validate whether an FD holds: One query that can check multiple FDs using distinct counts, and one that checks single candidates using the GROUP operator. The queries use the example left hand side $col1, col2$. Any other, arbitrary sized column combination can be used instead.
The first query delivers information on all columns that can be functionally dependent on $col1, col2$; it checks all possible right hand sides for one left hand side in one query:

```
SELECT   count(DISTINCT depCol1),
         count(DISTINCT depCol2),
         [...]
FROM     table1
GROUP BY col1, col2;
```

Listing 12. Validating mulitple FDs using distinct counts

The second query uses the GROUP operator. It can validate only one candidate FD but could abort early if a contradicting tuple is found. On HANA, this only leads to a noticeable speed-up if the result is very large otherwise.

```
SELECT   count(*) AS noFD
FROM     table1
GROUP BY col1, col2
HAVING   count(DISTINCT depCol) > 1
LIMIT    1;
```

Listing 13. Validating an FD using Group By

3 Experiments

To evaluate the performance of the SQL-based approaches, we executed the SQL queries and respective state-of-the art algorithms on comparable hardware for selected profiling tasks:

- Median: SQL query vs. Sorting

- INDs: All queries vs. SPIDER [1]
- UCCs: Group query vs. DUCC [2]

For sorting, we use a heapsort-based approach where each value that is read from the JDBC result is immediately inserted at the right position in the tree.
As mentioned in Section 2, candidates need to be generated in order to find INDs and UCCs. Since our goal was to compare rather naïve SQL-based profiling to sophisticated algorithms, we used a brute-force approach for candidate generation: For INDs, each pair of columns was considered unless they had contradicting datatypes, meaning one column had a numeric and the other a non-numeric type. This reflects the approach the SPIDER implementation takes. For UCCs, we use a very simple pruning mechanism: All column combinations are considered unless a subset has already been found to be unique.
Additionally, in order to evaluate HANA's profiling performance related to traditional row-based database systems, we compared HANA to that of a PostgreSQL instance on the same machine using the calculation of minimum and maximum of each column.
In all experiments we executed SQL queries on two data sets from different domains and with different numbers of rows and columns, namely:

- UniProt
  - Original data from life sciences domain
  - 1.1 GB
  - 539,166 rows with 221 attributes
- TPC-H Lineitems
  - Generated data form business domain
  - 750 MB
  - 6,001,215 rows with 16 attributes

For our experiments, we consider a scenario where the data of interest is available in a database. The user can thus use SQL to retrieve the desired metadata directly, or load all data via SQL and execute an algorithm on it.
We used the Future SOC's HANA-2 instance running on a Hewlett Packard DL980 G7 with 4x Xeon X7560 CPU, 1024 GB RAM, 2x 300 GB HDD, and 1280 GB SSD to store and access the data. The algorithms were executed on hardware that is comparable to the machine that the HANA is running on: a Fujitsu RX600S5 in the Future SOC, which has 4x Xeon X7550 CPU and 1024 GB RAM.
All queries were built and sent by a Java program using JDBC from within the HPI network. The driver program was also responsible for candidate generation. The time measured includes building and executing the queries as well as retrieving the results. The retrieval of necessary metadata such as the column and table names is considered preliminary work and not timed. For the algorithms, the measured time includes retrieving the data, building the appropriate data structures and executing the algorithm.

4 Results

As expected, HANA largely outperformed PostgreSQL for the calculation of minimum and maximum. This can be explained by the structure of a column store, where it is much more efficient to retrieve all values of one column than in a row store. The execution times are listed in Table 1.

	TPC-H	UniProt
PostgreSQL	34.66 s	75.12 s
HANA	0.49 s	1.84 s

Table 1. Execution Times for Minimum and Maximum Calculation

Table 2 shows that the HANA is also significantly faster for the median calculation than using the described tree-based sorting approach in Java.

	TPC-H	UniProt
SQL on HANA	17.3 s	48.8 s
Java	41.5 s	135.2 s

Table 2. Execution Times for Median Calculation

The execution times in Table 3 are the total time needed to generate and validate all IND candidates. The Not In query on UniProt did not return within 3 hours and was therefore aborted. Of the four SQL approaches, the Join is the fastest. For TPC-H, it is faster than SPIDER, but SPIDER performs much better on the UniProt data set. The reason for this lies in the nature of the approaches: The SQL approach does one check per candidate, and its execution time is therefore linear in the number of candidates. The SPIDER algorithm, on the other hand, checks all candidates simultaneously. Its overhead of sorting all columns and creating the appropriate data structures therefore pays off when many candidates exist, i.e. when many columns have to be considered.

	TPC-H	UniProt
Except query on HANA	263.9 s	269.0 min
Not In query on HANA	8808.5 s	–
Not Exists query on HANA	70.9 s	212.8 min
Join query on HANA	17.3 s	47.4 min
SPIDER in Java	30.2 s	0.7 min

Table 3. Execution Times for Inclusion Dependency Calculation

The value of specialized algorithms becomes most obvious when considering UCCs. A bottom-up approach with basic pruning as described in Section 3 using the Group By query from Listing 8 does not return within reasonable time for of UniProt (Table 4). After 7.5 hours, the execution was still validating candidates of size 3. The results emphasize the fact that the complexity of UCC discovery lies in the large number of candidates, and specialized algorithms are indispensable.

	TPC-H	UniProt (200 col.)
SQL on HANA	81.6 min	$>$ 450 min
DUCC in Java	60.3 min	2.7 min

Table 4. Execution Times for Unique Column Combinations Calculation

Since DUCC's strategy is mostly about an intelligent order of candidate validation that eliminates as many candidates as possible without actually validating them on the data, it can be combined with SQL as the validation strategy, so the two approaches do not directly contradict each other. Nevertheless, we proved that the naive bottom-up algorithm is inferior to more sophisticated algorithms even when the candidate validations are executed on a high-performance column-oriented database.
We also observed that the larger the UCC candidates grow, the longer HANA needs for each candidate validation, as shown in Figure 1. This can be explained by the characteristics of column stores: They are very fast in analyzing single columns, but putting the attributes of a tuple back together requires a greater effort.

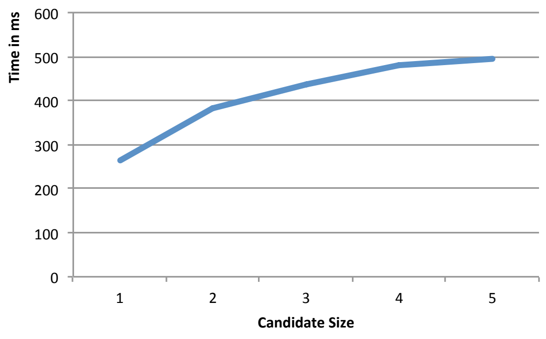


Figure 1. Average execution time for one UCC check on UniProt by candidate size

5 Conclusion

Overall, SQL becomes a much more considerable option when executed on an in-memory column-store like SAP HANA. It is especially useful for small data sets and simple metadata. For complex, multi-column metrics such as UCCs, however, smart algorithms remain superior, because they significantly reduce the number of candidates.
As next steps, we mainly want to evaluate the other profiling tasks. It would also be interesting to extend our experiments to other data sources. This allows to investigate which characteristics of the data influence the performance of the SQL-based approaches. For example, a data set where all columns are unique on their own, might lead to the bottom-up SQL approach being faster than DUCC. Furthermore, this would help to clarify in which situations SQL outperforms SPIDER for IND detection.

References

[1] J. Bauckmann, U. Leser, F. Naumann, and V. Tietz. Efficiently detecting inclusion dependencies. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1448–1450. IEEE, 2007.

[2] A. Heise, J.-A. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable discovery of unique column combinations. *Proceedings of the VLDB Endowment*, 7(4), 2013.

High-Performance In-Memory Genome Project

Matthieu-P. Schapranow Cindy Fähnrich
Hasso Plattner Institute
Enterprise Platform and Integration Concepts
August–Bebel–Str. 88
14482 Potsdam, Germany
schapranow|cindy.faehnrich@hpi.uni-potsdam.de

Abstract

Latest medical diagnostics, such as genome sequencing, generate increasing amounts of "big medical data". A variety of next-generation sequencing technologies reduced costs and improved quality for whole genome sequencing within the last decade. However, interpretation and analysis of generated raw genome data is still a time- and resource-intensive task taking up to weeks.

Our work focuses on the integration of medical data, e.g. acquired from latest next-generation sequencing technology, its systematical processing, and instant analysis for researchers and clinicians in the course of precision medicine. Thus, we developed specific software extensions for in-memory technology to enable processing and real-time analysis of fine-grained medical data within a single system. We share our research results on building a distributed in-memory computing platform for genome data processing, which enables instantaneous analysis of genome data. In this work, we present the technical foundation of our platform and concentrate on improving genome data processing pipelines with the help of distribution and in-memory technology.

1. Project Idea

The continuous progress in understanding relevant genomic basics, e.g. for treatment of cancer patients, collides with the tremendous amount of data that needs to be processed. Figure 1 provides a comparison of costs for sequencing and main memory modules on a logarithmic scale. Both graphs follow a steadily declining trend, which facilitates the increasing use of next-generation sequencing (NGS) for whole genome sequencing and In-Memory Database (IMDB) technology for data analysis. Latest NGS devices enable the processing of whole genome data within hours at reduced costs [1]. As a result, the time consumed for sequencing is meanwhile a comparable small portion of the time consumed by the complete workflow. Data processing and its analysis now consume a significantly higher portion of the time and accelerating them would affect the overall workflow duration. During this phase, the NGS output, i.e., short chunks of Deoxiribonucleic Acid (DNA) in digital format, need to be aligned to reconstruct the whole genome. Afterwards, variants compared to a reference, e.g., normal vs. pathologic tissue are identified during variant calling. The subsequent analysis of genome data builds on the list of those detected variants, e.g., to identify driver mutations for a medical finding [2].

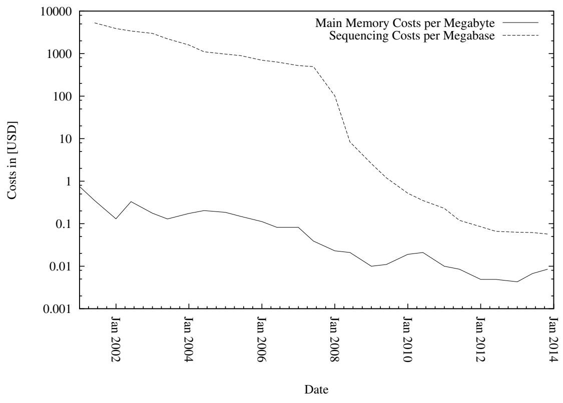


Figure 1. Costs for next-generation sequencing and main memory 2001-2014 adapted from [9, 10].

From a software engineering point of view, improving the analysis of genomic data is both a concrete research and engineering challenge. Combining knowledge of in-memory technology and of how to perform real-time analysis of huge amounts of data with concrete research questions of medical and biological experts is the aim of the High-Performance In-Memory Genome (HIG) project. Within the HIG project, we built an analysis platform providing tools for setting up and executing analysis pipelines, assessing their results, and combining these with scientific data from distributed data sources. We provide a modeling environment to create customized pipelines and choose from a range of ready-to-use third-party tools. We created a dedicated worker framework, which incorporates cloud computing to accelerate processing of genome data in a highly parallel manner across multiple computation nodes. For example, it involves alignment and variant calling of sequence data, which are CPU-bound tasks. We combine latest medical knowledge, such as variants for known diseases and genome annotations, from international data sources such as 1,000 genomes project or dbSNP [13].

2. High-Performance In-Memory Computing Platform

In the following, we share details about our system architecture and selected system components. We provide technical details regarding a highly parallel execution of Genome Data Processing Pipelines (GDPPs). This requires a dedicated worker framework as well as mechanisms for load balancing and prioritized task scheduling.

2.1. Architecture

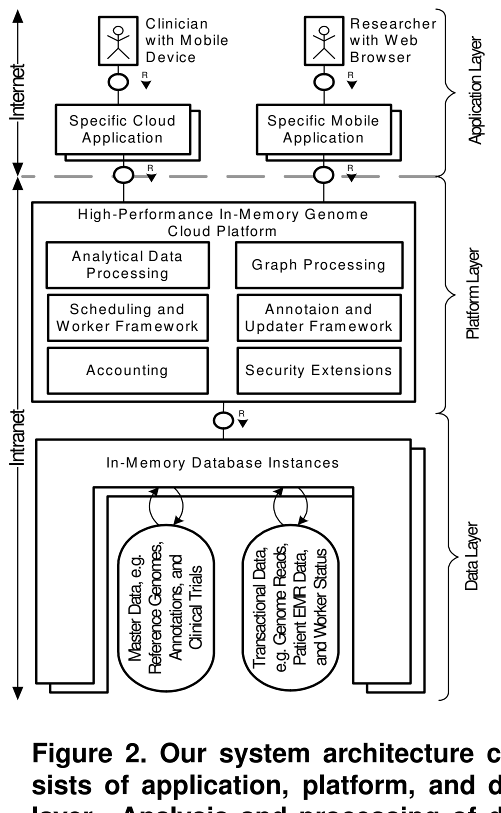

Internet
Intranet
Clinician with Mobile Device
Researcher with Web Browser
R
Specific Cloud Application
Specific Mobile Application
High-Performance In-Memory Genome Cloud Platform
Analytical Data Processing
Graph Processing
Scheduling and Worker Framework
Annotaion and Updater Framework
Accounting
Security Extensions
In-Memory Database Instances
Master Data, e.g. Reference Genomes, Annotations, and Clinical Trials
Transactional Data, e.g. Genome Reads, Patient EMR Data, and Worker Status
Application Layer
Platform Layer
Data Layer

Figure 2. Our system architecture consists of application, platform, and data layer. Analysis and processing of data is performed in the platform layer eliminating time-consuming data transfer.

Figure 2 depicts the software system architecture of our high-performance in-memory computing platform with application, platform, and data layer as Fundamental Modeling Concepts (FMC) block diagram [8]. Our platform combines data from various data sources, such as patient-specific data, genome data, and annotation data within a single system.

2.1.1 Application Layer

The application layer consists of special purpose applications to answer medical and research questions. We provide an Application Programming Interface (API) that can be consumed by various kinds of applications, such as web browsers or mobile applications. Figure 2 depicts the data exchange via asynchronous Ajax calls and JavaScript Object Notation (JSON) [3, 5]. As a result, accessing data and performing analyses is no longer limited to a specific location, e.g., desktop computer. Instead, all applications can be accessed via devices connected to the Internet, e.g., laptop, mobile phone, or tablet computer.

2.1.2 Platform Layer

The platform layer holds the complete process logic and consists of the IMDB system for enabling real-time analysis of genome data. The IMDB system is distributed across the FSOC cluster's 25 computing nodes with each having 1TB main memory. We developed specific extensions that support high-throughput processing of genome data and its real-time analysis. We established selected system components for parallel execution of pipeline model instances to enable high-throughput processing and prioritized scheduling of jobs within the worker framework. This framework also operates on the whole FSOC cluster, i.e., each of the nodes is used for parallel pipeline execution.

2.1.3 Data Layer

The data layer holds all required data for performing processing and analyzing of genomic data. As part of the IMDB system, the database is a landscape of several database instances that are distributed across the FSOC cluster. As a result, the data itself is equally distributed across all instances on the cluster. Our data can be distinguished in the two categories of master and transactional data [4]. For example, human reference genomes and annotation data are referred to as master data, whereas patient-specific NGS data, e.g., from the 1,000 genomes project, are referred to as transactional data [14]. Their analysis is the basis for gathering specific insights, e.g., individual genetic dispositions and to leverage personalized treatment decisions in course of precision medicine [7].

2.2. Parallel Execution of Pipelines

In this project, we focus on parallel execution of GDPPs and for that designed specific functionality within our platform. Our worker framework mentioned in Section 2.1.2 consists of multiple workers running on the cluster's distinct compute nodes. Each worker is directly connected to the database landscape to access their local portion of the database content. The task management is synchronized via the database by writing all tasks and their execution details into a shared database table. This table is frequently updated by the workers while they are processing a concrete task. Incorporating the database for these purposes reduces the complexity of the individual worker code because specific exception handling can be processed by the database, e.g., concurrent start of the identical task can be prevented by using built-in database locks.

2.3. Prioritized Task Scheduling

The execution of multiple GDPPs is coordinated by a single scheduler component. This component enables resource allocation and distribution of workload across our cluster of worker machines. The scheduler's internal state is permanently stored within the IMDB, e.g., for global communication, logging, and for maintaining statistics. The database also serves as consistent transaction log of scheduling decisions that enables controlled recovery in case of a system failure.
We implemented specific scheduling algorithms optimized for throughput that can take into account various aspects. For example, we incorporate the Shortest Task First (STF) scheduling policy to minimize turnaround time and maximize throughput [12]. Our STF scheduling policy is adapted to estimate the remaining execution time of all waiting tasks whenever a scheduling decision needs to be taken. The incorporated IMDB technology guarantees that the estimation can be processed in real-time and does not delay decision making significantly [11]. The developed scheduler component is very generic and can easily be adapted to fit individual requirements, e.g., to prioritize the execution of tasks from a department or to keep a processing reserve for the very important users. Furthermore, individual scheduling policies can be developed to change the behavior of the scheduling system. Each scheduling policy can incorporate various input data, e.g., details about the overall system load provided by the load balancer as described in Section 2.4.

2.4. Load Balancing

Running jobs and how they are assigned to individual workers influence the overall system load of all compute nodes. We implemented a load balancer that incorporates the current system status of all worker nodes. The configuration of each of the cluster's compute nodes, e.g., how many workers are running and how many CPU cores are available, is stored in the configuration database table. This information can also be used by the scheduler, e.g., to postpone the execution of long-running jobs when short-running jobs have just become available.

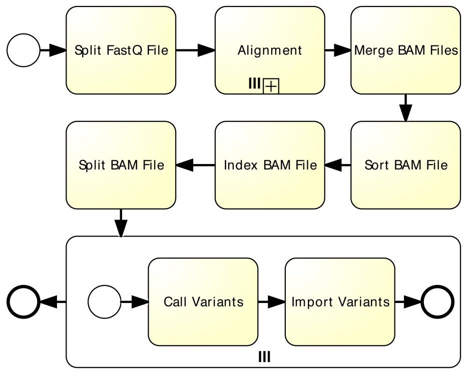


Figure 3. GDPP model of the general approach with file system as primary storage.

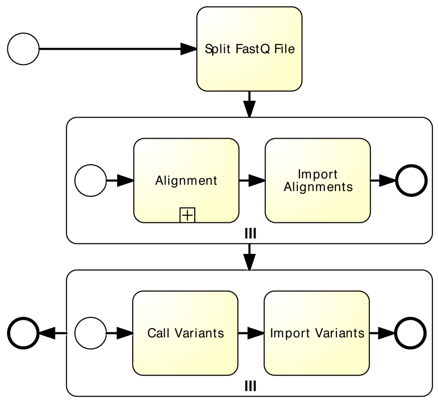


Figure 4. GDPP model incorporating an IMDB as primary storage.

3. Benchmark Setup

All benchmarks were performed on the FSOC cluster consisting of 25 identical compute nodes with a total of 1,000 cores. All compute nodes were equipped with Intel 520 series SSDs of 480 GB capacity combined using a hardware raid for local file operations [6]. The average throughput rate of the local SSDs was measured with 7.6GB/s cached reads and 1.4GB/s buffered disk reads. All nodes were interconnected via a Network File System (NFS) using dedicated 10 Gb/s Ethernet links and switches to share data between nodes.
We incorporated real NGS data for individual measurements, i.e., FASTQ files from the 1,000 genomes project [13]. We used the FASTQ file of patient HG00251 for our benchmarks. It consumed 160GB of disk space, consists of approx. 63 Gbp, approx. 695 M reads with 91 bp individual read length forming an average 20x coverage of the whole genome.
We implemented two GDPPs for our benchmarks. The

Experiment	Split Size	Primary Storage
A	1	File System
B	1	In-Memory Database
C	25	File System
D	25	In-Memory Database

Table 1. Experiment configurations.

Size $[Gbp]$	1.0	2.0	4.0	7.9	15.8
$t_A[s]$	409	690	1,256	2,305	4,931
$t_B[s]$	377	421	806	1,542	2,861
$t_C[s]$	330	529	860	1,427	3,443
$t_D[s]$	278	355	566	1,016	1,685
$R_B[\%]$	8	39	36	33	42
$R_C[\%]$	19	23	32	38	30
$R_D[\%]$	32	49	55	56	66

Table 2. Comparison of overall pipeline execution times.

first uses a file system as primary storage, i.e., intermediate results are stored on disk space. The distinct execution steps are modeled in Figure 3. On the contrary, the second pipeline as shown in Figure 4 uses an IMDB as primary storage. Our pipelines contain distinct parts for alignment and variant calling that are parallelized, e.g., by splitting up the input data. We used an alignment algorithm that operates directly within our IMDB system and was configured to use a maximum of 80 threads. Furthermore, the GDPP using a file system as primary storage requires extra file processing steps between alignment and variant calling.

We designed our benchmarks to compare the impact of the incorporated storage system and the level of parallelization on the overall execution time as outlined in Table 1. Exp. A and B were executed on a single compute node, while Exp. C and D were executed on all 25 compute nodes to evaluate the impact of a fully parallelized execution environment.

4. Results and Findings

In the following, we present and evaluate our obtained benchmark results. We measured the overall pipeline execution execution times t_x for Exp. x and derived the relative advance of execution time for Exp. x compared to Exp. A as $R_x = \frac{t_A - t_x}{t_A}$. Table 2 shows the overall pipeline execution times. Exp. A and B indicate runtime performances when running the pipelines on a single compute node. The measured execution times indicate that the use of the IMDB as primary storage can bring a performance improvement of up to 42 percent.

Exp. C and D as shown in Table 2 document the impact of the parameter `splits`, i.e., the number of

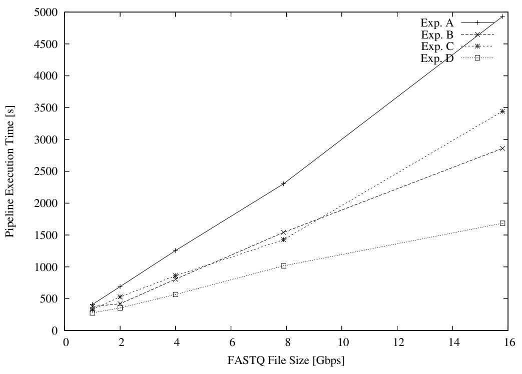


Figure 5. Development of overall execution times for varying file sizes and experiment setups.

distributed compute nodes used for parallel execution as introduced in our pipeline models. Parallel execution of selected pipeline steps reduces the overall execution time by up to 38 percent. Figure 5 illustrates execution time behavior for our GDPPs and different grades of parallelization. It clearly shows the improvements originating from parallelization and main memory as primary storage medium and that they increase for larger file sizes. In summary, execution time of a GDPP can be reduced by up to 66 percent.

Our conducted benchmarks verify the hypothesis that our system supports the parallel execution of intermediate process steps across multiple compute nodes, which results in an additional performance improvement compared to the execution on a single compute node. Our observed results show that the overall pipeline execution time correlates to the number of base pairs contained in the FASTQ file. However, the improvement of using 25 nodes is still below our expectation of a factor 25 since we also use traditional tools in the GDPP, e.g., SAMtools, which partially operate in a single threaded way.

Our results stress the benefits of using an IMDB for operating on intermediate results of the pipeline execution. The pipeline optimized for the IMDB no longer uses individual tools operating on files for specific process steps, such as sorting, merging, and indexing. In contrast, these operations are directly performed as an integral operation of the IMDB without the need to create intermediate files in the file system any longer.

5. Next Steps

We constantly aim at extending our system by new functionality and improve methods already applied. In addition, we search for new data sources, e.g., the second part of the 1,000 genomes project or European health statistics, to be integrated into our knowledge base and used in real-time analyses. With the help of our IMDB platform, we want to open up new possibil-

ities for combining relevant data.
We intend to set up additional GDPPs for more specific use cases, e.g., for targeted sequencing, and integrate a broader range of tools with improved performance. As a result, we want to benchmark our efforts and compare benefits and drawbacks of different pipeline setups.

References

[1] W. J. Ansorge. Next-Generation DNA Sequencing Techniques. *New Biotechnology*, 25(4):195–203, 2009.

[2] I. Bozic et al. Accumulation of Driver and Passenger Mutations during Tumor Progression. *Proceedings of the National Academy of Sciences of the United States of America*, 107(43):18545–50, Oct. 2010.

[3] D. Crockford. RFC4627: The application/json Media Type for JavaScript Object Notation (JSON). `http://www.ietf.org/rfc/rfc4627.txt` [retrieved: Mar 2, 2014], July 2006.

[4] T. K. Das and M. R. Mishra. A Study on Challenges and Opportunities in Master Data Management. *Int'l Journal of Database Mgmt Syst*, 3(2), May 2011.

[5] A. T. Holdener. *AJAX: The Definitive Guide*. O'Reilly, 1st edition, 2008.

[6] Intel Corporation. Intel Solid-State Drive 520 Series Product Specification. `http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/ssd-520-specification.pdf` [retrieved: Mar 2, 2014], Feb 2012.

[7] K. Jain. *Textbook of Personalized Medicine*. Springer, 2009.

[8] A. Knöpfel, B. Grone, and P. Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. John Wiley & Sons, 2006.

[9] J. C. McCallum. Memory Prices (1957-2013). `http://www.jcmit.com/memoryprice.htm` [retrieved: Mar 2, 2014], Feb 2013.

[10] National Human Genome Research Institute. DNA Sequencing Costs. `http://www.genome.gov/sequencingcosts/` [retrieved: Mar 2, 2014], Apr 2013.

[11] H. Plattner and M.-P. Schapranow, editors. *High-Performance In-Memory Genome Data Analysis: How In-Memory Database Technology Accelerates Personalized Medicine*. Springer-Verlag, 2014.

[12] A. S. Tanenbaum. *Modern Operating Systems*. Pearson Prentice Hall, 3rd edition, 2009.

[13] The 1000 Genomes Project Consortium. A Map of Human Genome Variation from Population-scale Sequencing. *Nature*, 467(7319):1061—1073, Oct. 2010.

[14] The Genome Reference Consortium. Genome Assemblies. `http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/data.shtml` [retrieved: Mar 2, 2014].

Distributed-memory Simulation of Seismic Events following Earthquakes

Fahad Khalid[a], Camilla Cattania[b], Andreas Polze[a]
[a]Hasso Plattner Institute for Software Systems Engineering, Potsdam, Germany
{fahad.khalid, andreas.polze}@hpi.uni-potsdam.de
[b]GFZ German Research Centre for Geosciences, Potsdam, Germany
camcat@gfz-potsdam.de

Abstract

The significance of effective forecasting techniques for predicting natural hazards is obvious; given the devastation caused by such disasters. Earthquakes constitute one category of such events. Numerical simulations of seismic events can be demanding both in terms of memory consumption and processing requirements. Therefore, in order to scale such simulations, it is imperative that distributed-memory implementations are available.

In this report we present our experience in porting a shared-memory parallel simulation developed at GFZ, to a distributed-memory architecture comprising of 1000 cores distributed across 25 nodes. Our distributed-memory simulation is based on the message passing paradigm implemented using MPI. We discuss the application design, as well as results obtained by running the application on the 1000-core cluster available in the FutureSOC Lab.

Our results show that the simulation scales up to 2000 processes. Moreover, our MPI based implementation makes it possible to extend the simulation to datasets that previously could not be computed at all due to heavy demands on both processing power and memory consumption.

1 Introduction

Earthquakes (seismic events) are caused by stresses that build up in the Earth's crust. Once an earthquake occurs, it induces changes to the stress; which can lead to subsequent events. The changes in stress that follow earthquakes can be simulated, making it possible to estimate the locations where subsequent earthquakes are more likely to occur.

Mathematical models for the rate of earthquake production [1] have been proposed and are currently in use. The current research at GFZ includes improving the predictive power and accuracy of the models with the help of simulations. A *C* language based shared-memory parallel (multicore CPU-based) simulation code – called *Coulomb Rate-State (CRS)* – was developed at GFZ that has enabled researchers to improve the existing mathematical models and devise better prediction strategies. However, performance of this implementation is constrained by the number of threads and the amount of main memory available in a single shared-memory machine.

In order to scale the simulation to larger problems and/or fine-grained models, it is important that the code be able to harness the power of multiple machines. This requirement naturally led us to this collaborative effort, where we successfully extended the existing simulation code to an MPI based distributed-memory implementation, specifically designed for execution on commodity clusters.

In the sections to follow, we present the parallelization strategy used in the distributed-memory implementation; the communication model employed, as well as performance analysis based on simulation runs on the *1000-core* cluster available in the FutureSOC Lab.

2 Simulation Design

The simulation code consists of several modules, where each computes complex mathematical models. In the interest of simplicity, we discuss only those aspects of the simulation that are important for the distributed-memory implementation. From this point onwards in the document, we will refer to the distributed-memory implementation as *CRS-MPI*.

CRS-MPI comprises of three major parts. Each of these is described in the following subsections.

2.1 File I/O and Broadcast

The first part of the simulation consists of several functions populating data structures with values read from input files. In the current *CRS-MPI* implementation, input files are read only by the root process. The values read by the root process are then *broadcast* to all other processes.

A more efficient solution would utilize a shared file system. However, since clusters with shared file systems are generally not available to the targeted end-users – Geophysicists –, we decided not to rely on the availability of a shared file system. The current implementation is capable of execution on commodity clusters supporting a minimal feature set.

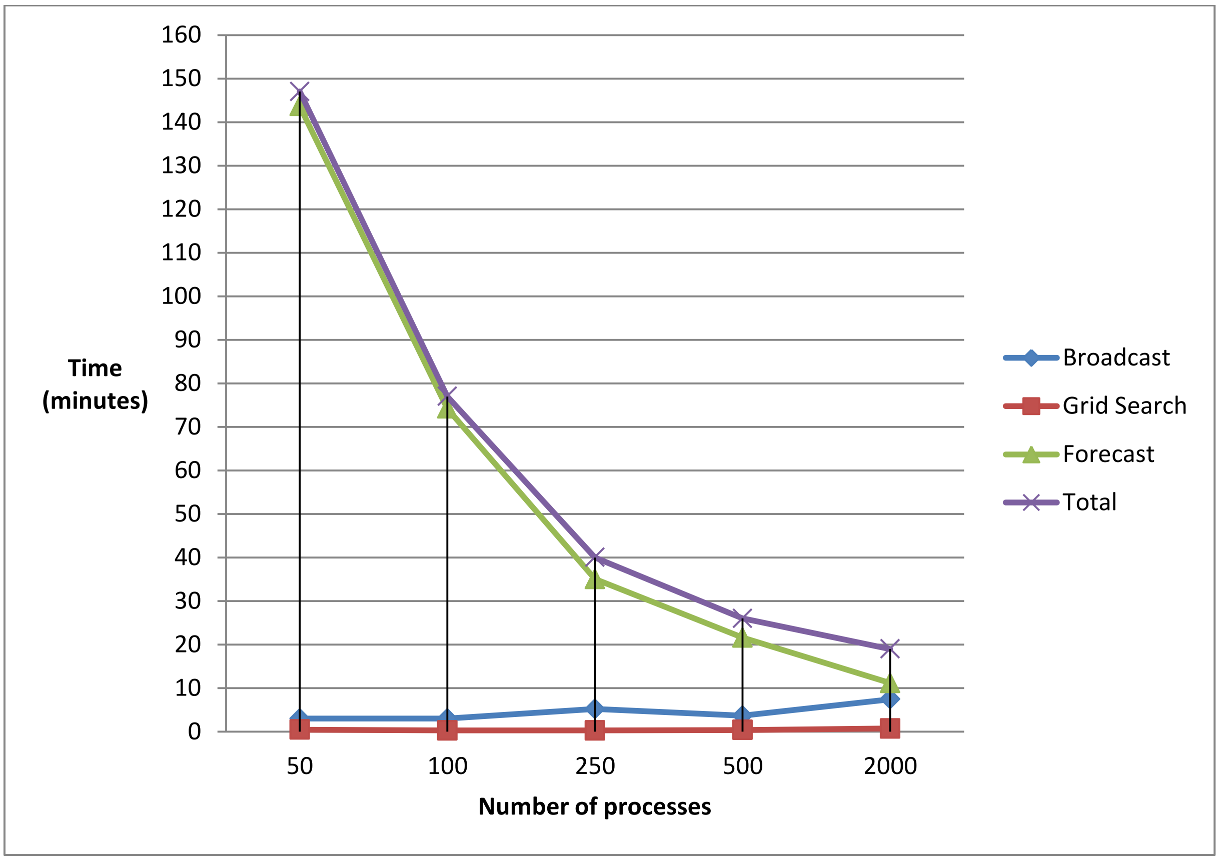


Figure 1: Time taken by different parts of the simulation, as well as the entire simulation when executed with different numbers of processes.

2.2 Grid Search

Grid search is an optimization algorithm used to find the optimal point on the grid. This is done by executing a *Monte Carlo* simulation encapsulated within a for-loop. In the original implementation, there is a dependency within the iterations of this for-loop that makes it a challenging target for loop parallelization. We managed to find a solution which removes this dependency, resulting in an embarrassingly parallel execution of the for-loop. Based on the rank, each MPI process executes only a certain unique chunk of iterations, saving the results in local variables. After all iterations have been completed, the local variables are reduced with the *sum* operation. After the reduction step, all processes share the same global values for all data structures.

2.3 Forecast

From the parallelization point of view, the *forecast* code is very similar to the *grid search* code. In this case as well, our solution removes a dependency between for-loop iterations, resulting in the *Map* [2] pattern for embarrassingly parallel processing of the loop iterations. Even though the penalization strategy is similar to the one employed for *grid search*; due to the computational complexity of the functions executed within the *forecast* loop iterations, *forecast* constitutes the most compute intensive part of the simulation.

3 Evaluation

In this section we present and analyze results obtained by running the simulation on the *1000-core* cluster with different numbers of processes. The same input data and parameters have been used in all simulation runs.

3.1 Test Environment

The machine used for testing is the *1000-core* cluster available in the FutreSOC Lab. The cluster consists of 25 nodes, where each node supports a maximum of 80 hardware threads when hyper-threading is enabled. Moreover, each node is equipped with 1 TB of RAM; making it a total of 25 TB distributed across all nodes. The cluster is homogenous, i.e., all nodes have the same hardware and software configuration. The code was compiled with GCC 4.7.3. During the tests, each node was running SUSE Enterprise 11 SP2 with Open MPI 1.7.4.

3.2 Results

Figure 1 plots execution times against the number of processes used for each simulation run. Results are plotted for each of the three major parts of the simulation, as well as for the entire simulation run. The

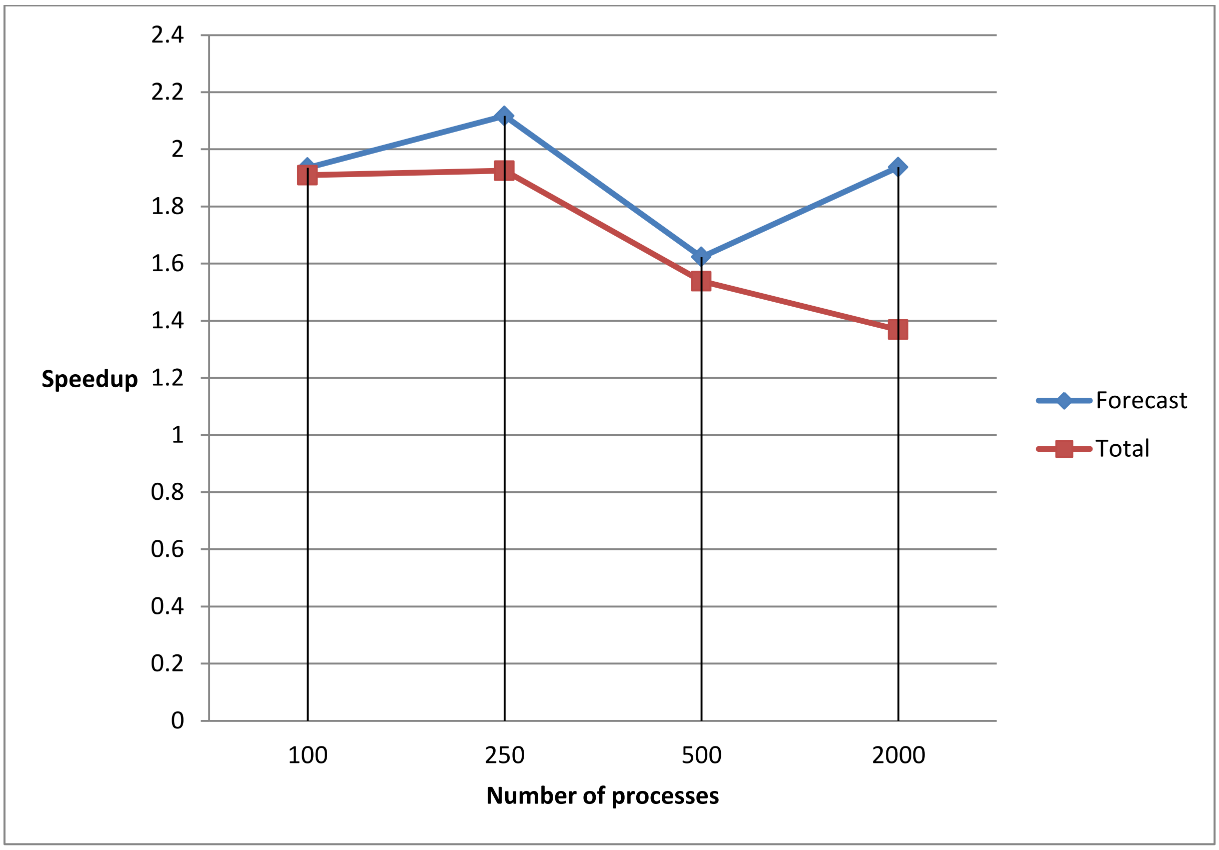


Figure 2: Speedup obtained with each increment in the number of processes. The first increment is from 50 processes to 100 processes.

total execution time was measured using the Linux *time* utility. The code was instrumented with *MPI_Wtime()* to calculate execution times for individual parts of the simulation.

It can be observed that the time taken by *grid search* does not vary significantly with the increasing number of processes. Also, *grid search* does not contribute significantly to the total execution time.

The time spent on *broadcast* does not change much up until the number processes is 500 (we believe the hump around 250 is an aberration caused by a temporary performance issue with the file system). However, going from 500 to 2000 processes increases the *broadcast* time to a value close to the time taken by *forecast*. This clearly indicates the need for optimizing the *broadcast* strategy, since with the current implementation *broadcast* will become a performance bottleneck for more than 2000 processes.

The time spent in *forecast* clearly dominates the total time taken by the simulation for up to 500 processes. However, the contribution from *broadcast* becomes more and more significant as *forecast* takes less and less time. Nevertheless, the current implementation clearly scales very well as far as reduction of *forecast* time is concerned.

Figure 2 shows the speedup achieved with each increment in the number of processes. In terms of the total execution time, the parallelization and communication overhead increases when moving from 250 to 500 processes. This results in a sub-linear speedup. When moving from 500 to 2000 processes, the *broadcast* time becomes so large that speedup is reduced even further. However, the resulting speedup is still beneficial and justified for the simulation. The total speedup achieved when moving from 50 to 2000 processes is approximately 7-fold.

The behavior of incremental speedup for *forecast* is a bit counterintuitive. We believe that the steep increase in speedup from 500 to 2000 processes is due to the much larger difference in this section, i.e., the number of processes increases by a factor of 4 instead of by a factor of 2 (as in the other cases). Nevertheless, this needs to be investigated further.

4 Conclusions

We have extended an existing seismic simulation code with distributed-memory parallelization using MPI. Success of the implementation has been shown with the help of empirical analysis. The results obtained are from execution of the simulation on the *1000-core* cluster available in the FutureSOC Lab.

Moreover, this project is an example of a successful interdisciplinary collaboration between the *Hasso Plattner Institute for Software Systems Engineering* and the *GFZ German Research Center for Geosciences*.

In the future, we intend to improve the simulation by adding the following features:

- Check-pointing for *CRS-MPI:* This will make it possible to pause and resume long running simulations without loss of data.
- Optimization of *broadcast* communication: As mentioned earlier, *broadcast* becomes a bottleneck when the number of processes exceeds a certain threshold. An efficient communication model can be employed to improve the performance of this *one-to-all* message transfer.

  However, the degree to which the communication graph can be optimized is constrained by the physical interconnect topology. This observation is not only relevant to our project, but for any HPC application running a large number of MPI ranks with a communication intensive phase.

  We intend to carry out in-depth investigation into this phenomenon – complemented with experiments – to find an optimal solution to the communication problem.
- Accelerator code: Porting parts of the simulation to accelerator architectures like GPUs and/or Intel Xeon Phi co-processors will make it possible to further improve the simulation performance.

References

1. Dieterich, J., “A constitutive law for rate of earthquake production and its application to earthquake clustering”. *Journal of Geophysical Research*, 1994. 99(B2): p. 2601-2618.
2. Kurt Keutzer, Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders. “A design pattern language for engineering (parallel) software: merging the PLPP and OPL projects”, 2010.

Simulation of Quantum Annealing of the Transverse Field Ising Model in Classical Parallel Hardware

Fabian Bornhofen, Fahad Khalid, and Andreas Polze
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
fabian.bornhofen@student.hpi.uni-potsdam.de,
{fahad.khalid, andreas.polze}@hpi.uni-potsdam.de

Abstract

Quantum Computing recently became commercially available when D-Wave launched their first chip in 2011. This chip – as well as its successor – utilizes a process called Quantum Annealing for solving a given problem. This process is radically different from the way conventional microprocessors operate, and therefore, poses new challenges in terms of programmability. Access to the D-Wave chip is eminently limited; thereby necessitating the development of an open source chip simulator, which would then be used to conduct research on suitable programming models for Quantum Annealing chips.
In this report we present the first steps taken toward the development of such a simulator and highlight the significance of utilizing parallel processing architectures for Quantum Annealing simulations. Based on existing research in statistical mechanics, we implemented the Path Integral Monte Carlo (PIMC) algorithm that simulates Quantum Annealing. We accelerated the process of accurate estimation of certain tuning parameters by running multiple Monte Carlo simulations in parallel. Results from running these simulations on the FutureSOC hardware are presented. In the next step, we intend to port our implementation to massively parallel accelerator architectures such as GPUs and/or Intel Xeon Phi Co-processors.

1 Motivation

Much of Quantum Computing research has focused on the conventional gate model [4, 2], where quantum analogs of classical logical gates are used to build computing circuits. Even programming models and programming languages have been proposed [5, 9, 12] for the gate model. Building such quantum computers however poses several technical challenges that scientists and engineers have yet to overcome. A major problem is that of *decoherence* [13], which has played an important role in limiting the commercial availability of gate model based quantum chips.
Quantum Annealing (QA) has recently attracted a lot of attention – even in mainstream media – due to D-Wave's progress in implementing QA in hardware. This has resulted in a chip that is much more resilient to decoherence. However, this type of quantum computation is radically different from "traditional" quantum computation. Moreover, the capabilities of the chip are limited to solving certain optimization problems.
Information on how to program the actual machine is relatively sparse. We aim at helping computer scientists without a profound background in physics understand the basic concepts of QA and the programming model of a computer based on QA. As a first step towards a classical simulator of a QA chip, we implemented a Monte Carlo algorithm that simulates QA of a 2D Ising model [7].

2 Programming as Optimization

Based on the discovery of Adiabatic Quantum Computation by Ed Farhi et al. [3], D-Wave have created a chip that performs the QA algorithm in hardware. It solves Quadratic Unconstrained Binary Optimization (QUBO) problems that have to be embedded into the 2D Ising model.
The 2D Ising model is a 2D lattice of so-called spin variables $s_i \in \{-1, 1\}$ (comparable to bits). Each s_i has an associated weight $h_i \in \mathbb{R}$. Nearest neighboring spins s_i, s_j are assigned a coupling strength $J_{i,j} \in \mathbb{R}$. At the boundaries of the lattice, the couplings wrap around. Fig. 1 shows a 3×3 spin lattice.
A vector $\mathbf{s} = (s_1, s_2, ..., s_N)$ of spin variables (spin configuration) is assigned an energy value using the Ising Hamiltonian function H:

$$H(\mathbf{s}) = -\sum_i h_i s_i - \sum_{<i,j>} J_{ij} s_i s_j$$

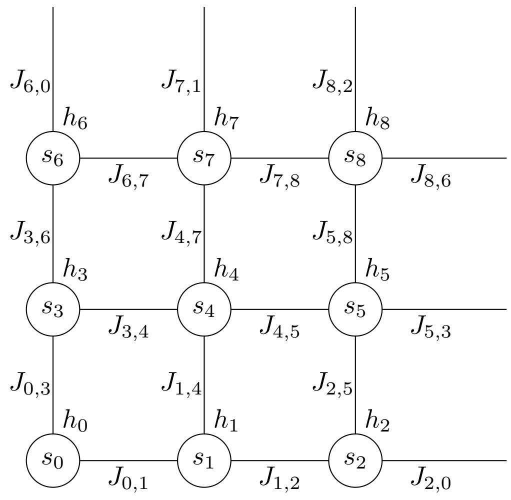


Figure 1. 3×3 Ising spin lattice with weighted spins and nearest-neighbor interactions that wrap around at the boundaries.

Given vectors $\mathbf{h} = (h_1, h_2, ..., h_N)$ and $\mathbf{J} = (J_{i,j})$, the spin configuration with minimum energy

$$\mathbf{s}_0 = \arg\min_{\mathbf{s}} H(\mathbf{s})$$

is called the ground state of the lattice. Finding the ground state for an arbitrary lattice is typically hard [1].

D-Wave claim their chip is capable of finding the ground state of an Ising lattice. Therefore, problem instances need to be transformed into a 2D Ising formulation [14]. This means that the problem has to be stated in terms of the Ising Hamiltonian H, i.e., vectors $\mathbf{h}$ and $\mathbf{J}$ need to be specified so that the ground state $\mathbf{s}$, once found, can be mapped to a solution of the problem.

3 Quantum Annealing

As aforementioned, the D-Wave chip is a Quantum Annealing device. This means that it will try to find the ground state of the problem instance in a manner resembling the physical process of annealing or its algorithmic equivalent, the Simulated Annealing [6] (SA) Markov chain Monte Carlo (MCMC) algorithm.

3.1 Simulated Annealing

Simulated Annealing is a well known heuristic method for exploring large search spaces. We quickly outline SA in order to make the analogy to QA obvious.

When solving an Ising model using SA, one would start out with a random spin configuration and a finite temperature T. In each Monte Carlo (MC) step, a random spin is flipped and the move is accepted if the resulting spin configuration has a lower energy E_2 than the previous one E_1, or if $\exp(-\frac{1}{T}\Delta E) > r$ with r randomly chosen from $[0, 1]$. After each step, T is decreased by a tunable ΔT.

3.2 Quantum Annealing

QA is conceptually similar to SA. Instead of decreasing the temperature, it decreases a transversal field Γ over time. T is usually fixed at a very small value in the neighborhood of 0 Kelvin. The energy of the system is given by:

$$H_Q = -\sum_{<i,j>} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z - \sum \Gamma\sigma_i^x$$

where σ_i^z and σ_i^x denote quantum mechanical spins of particles in the z or x direction. Initially, the field Γ is chosen to be very strong. It will force the spins to align in the x direction. For high values of Γ, this will be the ground state of H_Q. According to the adiabatic theorem [3], the system will remain in its respective ground state while Γ is reduced slowly enough. For $\Gamma \to 0$, its effect vanishes and the spins will align in the z direction and can finally be measured. The resulting vector $\mathbf{s} = (\sigma_1^z, \sigma_2^z, ..., \sigma_N^z,)$ is then a solution to the Ising formulation of the problem instance.

4 Path Integral Monte Carlo

In order to further study the programming model of the QA chip and experiment with it, we find it useful to be able to simulate the annealing algorithm for the 2D Ising model.

When reasoning about the Ising model in the quantum case, a single spin variable has to be represented by a *qubit* instead of a single classical bit. However, simulating quantum systems in classical hardware using exact calculations is computationally intractable. This is because each state of a system with N qubits is represented by a superposition of 2^N basis vectors. A common solution to approximate the dynamics of such a system is the application of Monte Carlo techniques such as Path Integral Monte Carlo [8, 11]. The PIMC algorithm is a way of approximating the behavior of QA using a purely classical MCMC algorithm.

The basic idea is to expand a d-dimensional quantum system into a $(d+1)$-dimensional system. This trick is called Suzuki-Trotter breakup. The additional dimension replicates the original spin system P times, with P being a tunable parameter of the algorithm (*Trotter number*). In the case of the Ising model, the transversal quantum field (Γ term) is modeled as interactions between the *replicas* – or *slices* – in the Suzuki-Trotter energy function:

$$H_{ST} = -\sum_{k=1}^{P}\Big(\sum_i h_i s_i^k + \sum_{<i,j>} J_{ij} s_i^k s_j^k + J_\perp \sum_i s_i^k s_i^{k+1}\Big)$$

with

$$J_\perp = -\frac{PT}{2} \ln \tanh \frac{\Gamma}{PT}$$

and

$$s_i^{P+1} = s_i^1$$

In H_{ST}, s_i^k denotes the i-th spin in the k-th replica. The $(d+1)$-dimensional system allows for conventional MCMC sampling of random walks. A Monte Carlo step consists of performing a local move, i. e. changing a spin and accepting the move if and only if the resulting Suzuki-Trotter energy of the system is lower than the previous one or a random number drawn uniformly from $[0,1]$ is larger than $\exp(-\Delta E * \beta)$ with $\beta = \frac{1}{T}$ being the inverse temperature. Previous research [8] suggests using global moves as well, where at a random position, spins in all replicas are flipped. The acceptance criterion remains the same.

```
begin
    $k$ := randomInt ($[0, P[$);
    $x$ := randomInt ($[0, GridWidth[$);
    $y$ := randomInt ($[0, GridHeight[$);
    $e_1$ := SuzukiTrotterEnergy(replicas);
    Flip bit at $(x, y)$ in replicas[$k$];
    $e_2$ := SuzukiTrotterEnergy(replicas);
    $r$ := random($[0, 1]$);
    $\beta := \frac{1}{T}$;
    if $e_1 < e_2$ and $r < exp\ (-\beta * (e_2 - e_1))$ then
        // reject move;
        Flip bit at $(x, y)$ in replicas[$k$];
    end
end
```

Algorithm 1: MakeLocalMove

This kind of sampling is closely related to the one used in SA in that it provides a probabilistic way of escaping from local optima. While in SA, T is slowly reduced from a finite value T_0 to 0, Quantum Annealing reduces Γ from Γ_0 to 0 while $T > 0$ at all times. $J_\perp$ increases quickly for $\Gamma \to 0$, making it less likely to accept worse configurations in a step and forcing all slices to agree on the same spin value at each position in the d-dimensional lattice.
A suggested improvement to Quantum Annealing is to use a hybrid QA/SA approach where T is also decreased to a small positive value [10].
In pure QA, P, $\Delta\Gamma$ and Γ_0 are tunable parameters. In the hybrid approach, ΔT and T_0 can be tuned as well. We implemented the PIMC with the possibility to optionally turn on hybrid behavior:

```
begin
    replicas := Array of $\mathbf{P}$ $\mathbf{N} \times \mathbf{M}$ spin
    configurations;
    for r : replicas do
        for spin : r do
            $s$ := random ($\{-1, 1\}$);
        end
    end
    while $t > 0$ and $\Gamma > 0$ do
        MakeLocalMove;
        MakeGlobalMove;
        if hybrid then
            $T := T - \Delta T$;
        end
        $\Gamma := \Gamma - \Delta\Gamma$;
    end
end
```

Algorithm 2: PIMC for simulated hybrid quantum/thermal annealing

Strictly speaking, in our implementation $\Delta\Gamma$ and ΔT depend on the number of MC steps that are to be performed. We then decrease Γ (and optionally T) accordingly in a linear fashion.

5 Implementation and First Experiments

We implemented a serial version of PIMC in C++ and checked the implementation against Ising lattices for which the ground state is easy to find. Specifically, we focused on instances with uniform h and J values. Besides finding suitable parameter settings for Γ we looked at how many MC steps would the algorithm require in order to find the ground state of a simple problem instance.
In cases where the resulting spin configuration is not the ground state, e.g., because the annealing schedule was too fast, the quality of the result is still measurable. $E_{res} = H(\mathbf{s}) - E_0$ denotes the residual energy of that configuration, with $E_0 = H(\mathbf{s}_0)$ being the true ground state energy.
In all of our experiments, we chose $h = 1$ and $J = -1$. The ground state of such a lattice with even edge lengths is a checkerboard pattern of $+1$ and -1 spins. The current implementation uses C++11 threads to run several Markov chains in parallel in order to generate more data.

5.1 Parameter Tuning

We tried different combinations of T and Γ in order to find out for which parameters QA performs best on a uniform 30×30 lattice with antiferromagnetic interactions ($h_i = 1, J_{i,j} = -1$) and $P = 20$. We ran 128 threads per configuration and used the mean of the residual energies to estimate the algorithm's performance.

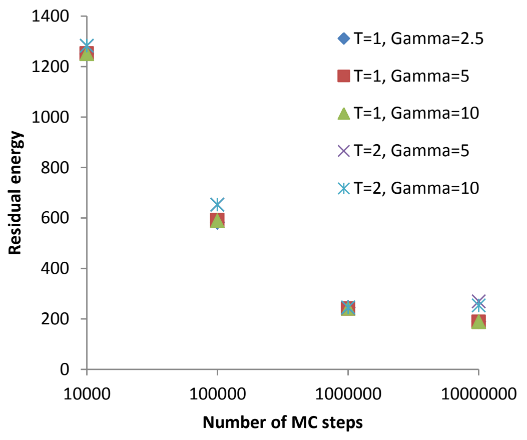


Figure 2. Residual energy in a 30×30 lattice after N Monte Carlo steps for different parameter configurations.

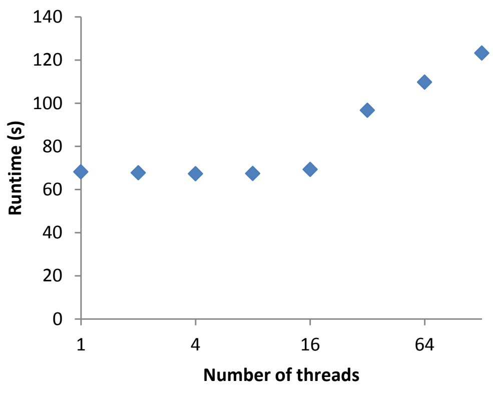


Figure 3. Total runtime when running N Markov chains in parallel.

Figure 2 shows that a low setting of T is favorable. In the experiments with $T = 1$, after 10^7 steps all settings for Γ yielded similar results. We have yet to examine whether the initial setting of Γ has a greater influence on harder problem instances.

5.2 Scalability

Our implementation runs several instances of the annealer in parallel and independently from each other. We expected to be able to run more Markov chains by simply adding more threads, up to the number of available hardware threads. The Hewlett Packard DL980 G7 has eight Intel Xeon X7560 processors with eight physical cores each; resulting in 64 cores and 128 hardware threads (with hyper-threading enabled).
The total runtime remains constant for $n = 1, 2, 4, 8, 16$ threads and then increases for $n = 32, 64, 128$ threads at a sub-linear rate (see Figure 3). Our current implementation is not optimized for efficient use of the cache hierarchy. We suspect this to be the cause of relative performance degradation when using more than 16 threads. In the upcoming FutureSOC session, this phenomenon will be thoroughly investigated.

6 Outlook and Parallelization

We have observed that the mere availability of a large number of cores and hardware threads does not guarantee optimal performance. For any application intended at efficient utilization of a large number of hardware threads, optimization of code for efficient cache utilization is imperative. In the future, we will implement NUMA awareness into our code.
Our central objective at this stage of our research is to make the implementation as efficient as possible so that it can solve larger or more difficult problems. For real-world applications, it is necessary to assess the algorithm's performance on non-uniform lattices, i. e. lattices with arbitrary h_i and $J_{i,j}$ values. Eventually, we would like to be able to embed instances of optimization problems from other domains onto the Ising model and solve them using QA.
During the upcoming FutureSOC project session, we will evaluate and implement parallelization strategies for PIMC on massively parallel accelerator architectures, particularly GPUs and possibly Intel Xeon Phi co-processors.

References

[1] B. A. Cipra. The Ising model is NP-complete.

[2] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. 400:97–117, 1985.

[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution, 2000.

[4] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

[5] S. J. Gay. Quantum programming languages: Survey and bibliography. *Mathematical Structures in Computer Science*, 16(4):581–600, 2006.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[7] W. Krauth. *Statistical Mechanics: Algorithms and Computations*. Oxford Master Series in Physics. Oxford University Press, UK, 2006.

[8] R. Martoňák, G. E. Santoro, and E. Tosatti. Quantum annealing by the path-integral monte carlo method: The two-dimensional random ising model. *Phys. Rev. B*, 66:094203, Sep 2002.

[9] R. Rdiger. Quantum programming languages: An introductory overview. *The Computer Journal*, 50(2):134–150, 2007.

[10] G. E. Santoro and E. Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General*, 39(36):R393, 2006.

[11] M. Sarjala, V. Petj, and M. Alava. Optimization in random field ising models by quantum annealing. J. Stat. Mech. (2006) P01008, 2005.

[12] P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[13] N. S. Yanofsky and M. A. Mannucci. *Quantum computing for computer scientists*. Cambridge: Cambridge University Press, 2008.

[14] W. G. M. Zhengbing Bian, Fabian Chudak and G. Rose. The ising model: teaching an old problem new tricks, 2011.

NUMA4HANA
A profiler to analyze the runtime behaviour of HANA on NUMA systems

Felix Eberhardt, Kai Schliewenz
Hasso-Plattner-Institut
Potsdam, Germany
felix.eberhardt@hpi.uni-potsdam.de
kai.schliewenz@hpi.uni-potsdam.de

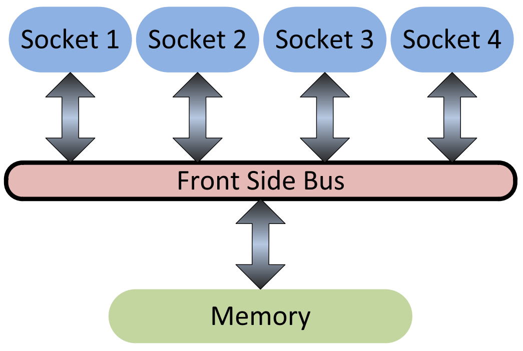
Frank Feinbube, Andreas Polze
Hasso-Plattner-Institut
Potsdam, Germany
frank.feinbube@hpi.uni-potsdam.de
andreas.polze@hpi.uni-potsdam.de

Abstract

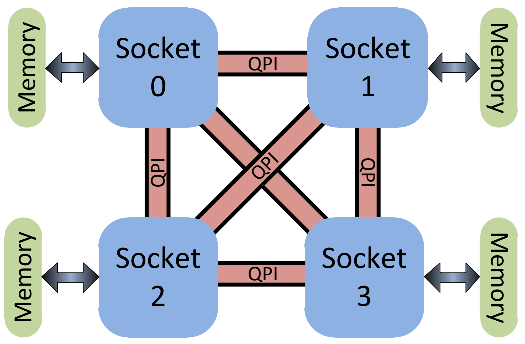
On Non-Uniform Memory Access (NUMA) systems, efficient thread and data placement is the key factor for high performance. Therefore information about the interaction between threads and memory objects is necessary. In this paper we present a profiler to analyse the runtime behaviour of an application on NUMA-based systems. The profiler is able to track the lifecycle of dynamically created objects in an application performing custom memory management. Gathered runtime data is analysed using SAP HANA. This enables new ways of combining obtained data to find patterns.

1 Introduction

The ever increasing core count on computer systems pushed Uniform Memory Access (UMA) designs to their performance limit. In an UMA-based system (see Figure 1(a)) all cores share a single memory bus. Thus only one core can access memory at a given time. On these systems the thread placement has a major impact on the overall performance. The data placement, if we do not consider caching, does not matter since the memory accesses from any core has the same latency. However, this design does not scale well with an increasing core count. Non Uniform Memory Access (NUMA) designs do not have a single shared memory bus. As shown in Figure 1(b) parts of the memory on NUMA-based systems are directly connected to the sockets. The sockets are linked with a fast interconnection network to communicate with each other [6, 9, 4]. The local attached memory can be accessed in parallel by the respective sockets. If data does not reside on the local attached memory of a socket, it can be transferred by another socket via the interconnection network. The NUMA design solves the core count scaling problem. However, in contrast to UMA-based systems, thread and data placement have to be considered. In the future, NUMA system topology will grow in complexity and thus efficient thread and data placement will be more difficult. NUMA is the next big thing for highly parallel, memory-heavy applications.



(a) Layout of UMA systems: several sockets connected to the memory through a shared bus (Front Side Bus).



(b) Layout of NUMA systems: each socket has its own memory, connected to each other through an fast interconnect (QPI - Quick Path Interconnect).

Figure 1. Comparison of UMA and NUMA architectures

The remote memory accesses on NUMA-based systems introduce new kinds of performance problems: higher latency, congestions of the interconnect network and the memory controllers of the sockets. These problems can be solved on the operating sys-

tem, middleware or application layer of the system stack. NUMA-optimizations on each layer must consider data and/or thread placement. In this paper we are focusing on the optimization of applications on NUMA-based systems. To choose the best optimizations, it is necessary to know the runtime behaviour of the application, in particular the interaction between threads and data. With the knowledge about which thread on which socket is accessing which data on which socket, it is possible to detect certain patterns leading to the according optimizations. Additionally, it is important to know which source code line allocated the data in the first place to incorporate data placement optimizations. Profilers seem to be a good way to reach that goal. Unfortunately, most of them do not provide enough information about the allocation of the accessed data. That is because they have no knowledge about the dynamically created objects in the application. Some applications use the default allocation mechanisms of the underlying runtime libraries such as *malloc* and *new*. In this case it is possible to obtain information about dynamic data allocation by intercepting the calls to the standard runtime libraries. Other applications use their own sophisticated memory management. That requires specific knowledge about the application in order to optimize the data placement. We focused on applications with their own sophisticated memory management and present you a profiler to analyze the interaction between threads and data in the HANA database.
The paper is organised as follows. In Section 2 we will elaborate on other optimisation strategies on the different locations of the system stack as well as profilers. The architecture and design decisions of our approach will be covered in Section 3. In Section 4 the technical details to the approach are presented. Finally Section 5 will show possible extensions to our approach.

2 Related Work

The memory allocation policy of current operating systems like Linux is *first touch*. The data is placed on the node where the first reading or writing thread is executed [1]. This approach is rather static. Carrefour [5] presents a mechanism to dynamically migrate, interleave or replicate pages according to certain metrics gathered during runtime of the system. Another approach is presented in User-Level Dynamic Page Migration for Multiprogrammed Shared-Memory Multiprocessors [11]. As for the memory management there exist other approaches for NUMA-aware scheduling. The dino scheduler [3] classifies threads according to their memory consumption and considers on which socket the memory resides. Then it schedules the threads accordingly to minimize remote memory accesses.
Several approaches exists to analyze the runtime behaviour of an applicaton on a NUMA-based system:

- Memphis: Finding and fixing NUMA-related performance problems on multi-core platforms [10]
- Memory Access Behavior Analysis of NUMA-based Shared Memory Programs [14]
- Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures [13]
- A Tool Environment for Efficient Execution of Shared Memory Programs on NUMA Systems [12]
- MemProf: a Memory Proler for NUMA Multicore Systems [8]

Lachaize et al. [8] present an approach for a profiler on which we build on: MemProf. MemProf tracks the allocation of memory objects together with the accesses of threads on those objects. To obtain the memory access from a application MemProf uses instruction based sampling from AMD processors. Additionally, they build a patch for the Linux kernel to get further informations regarding thread lifecycles. After starting an application, MemProf overrides the *new* and *delete* C++-operators from the standard runtime library to track all dynamic allocations of data.

3 Approach

The NUMA4HANA profiler builds on the ideas of MemProf [8]. MemProf is presented in more detail in Section 2. The profiler has some limitation which we bypass by using other technologies to gather the required data. MemProf is only applicable with AMD processors and requires a certain Linux kernel. Additionally the profiler is not capable of detecting dynamically created data, since our test application uses its very own memory management. This means that the *new* and *delete* C++ operators are overridden and are not calling underlying *malloc* to obtain new memory since memory is preallocated in huge chunks. We extend the approach by adding the ability to handle applications with their own sophisticated memory management. We also extend the analysis of the outcome of the profiler by providing an easy, extendable access to the gathered data.
In the remaining sections we will talk about threads interacting with memory objects. We define a memory object as a region of memory which represents a programming language construct and serves a certain purpose. Usually a memory object gets allocated as a whole but can be extended during runtime. There are static and dynamic as well as global and local memory objects. Static memory objects are constructs of a programming language known at compile time. Dynamic memory objects are allocated on runtime. An instance

of a C++ class allocated on the stack is a local, dynamic memory object. A memory mapped file is an example of a global, dynamic memory object.

As mentioned in Section 1 optimisations for NUMA-based systems need to consider thread and data placement. Therefore it is necessary to know which function or source code line is responsible for the most remote memory accesses but also which memory objects are accessed and where they have been allocated. The NUMA4HANA profiler monitors the allocation of every memory object by intersecting memory allocation functions such as *malloc*, *realloc* and *new* and map them to the according part in the source code where the object is allocated. The memory accesses of threads are sampled and enriched with additional information about the core on which the thread was executed and if it was a read or written access. By post processing the data, we can investigate which function produces the most traffic. Furthermore we can also point at the source location where the responsible memory object is created. This gives performance engineers and developers a hint where they start to examine other data allocation strategies.

3.1 INTEL's PIN tool

We use the PIN framework from INTEL [7] to collect the required data. This framework performs runtime binary instrumentation of applications. It can be used on Linux or Windows and provides an instrumentation platform for building different program analysis tools which are called PIN tools [2]. Figure 2 shows the basic layout of PIN. On highest level, PIN consists of a virtual machine (VM), a code cache and an instrumentation API which is used by the PIN tools to control the behavior of PIN. The VM consists of a just-in-time compiler (jit), an emulation unit and a dispatcher. After PIN takes control of an application, the jit-compiler recompiles and instruments the application. The code is stored in the code cache for later use, so multiple recompiling of the source code is not necessary. Since PIN sits above the operating system it can only capture user level code. Therefore system calls require special handling. They are interpreted by the emulator.

While instrumenting an application, there are three different binaries in the same address space: PIN, the PIN tool and the application. In order to prevent undefined behavior, none of the binaries share any linkable library. That means that the C runtime library is executed three times.

PIN provides different ways to instrument an application, by instruction routines, by instruction blocks, or by whole binaries, called images. For the implementation, we used PIN to collect the data we need to analyze a program and its execution. In this case, we trace all parts where memory is allocated, reallocated or deleted. With this knowledge we can log all memory accesses and the location from where the memory is accessed.

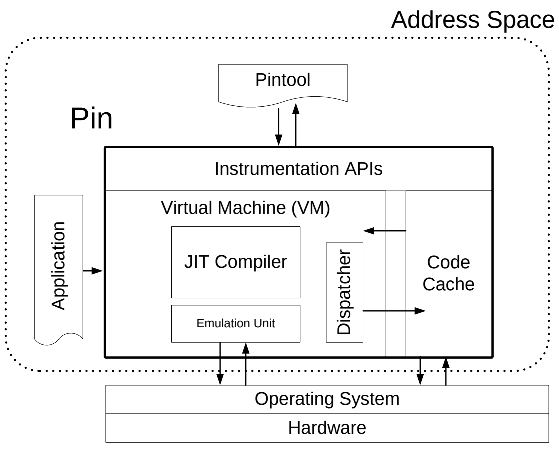


Figure 2. Pin's software architecture

3.2 Architecture

The profiler consists of several components: the pin-tool, postprocessing scripts and a SQL database as shown in Figure 3.

The pin-tool extracts all the needed data from the application and writes them into textfiles. Every memory operation is instrumented and later sampled. All memory object allocation functions with optional overrides are instrumented to track every memory object creation. The log writing is done lock free in parallel with a binary format to gain throughput. Every thread in the application allocates its own write buffer which is stored in a specific slot in its TLS (thread local storage). Every time the thread executes a memory operation or allocates a memory object the slot of the TLS is read and the reference to the output stream is obtained. Then the thread writes the output directly into its own file. No lock is needed since every thread writes in his own file and no shared memory is needed for the references to the output streams. In a postprocessing step the outcomes of the profiler are converted to delimiter separated files which are imported into the database.

4 NUMA4HANA Profiler

In order to identify memory objects the profiler must be able to intercept all kinds of memory object allocation. The allocation of memory in linux can happen with *malloc*, *realloc*, *calloc*, *mmap*, *mmap64* and the *new* operator. This can be a difficult task though, if the software is doing the memory management by itself and therefore overrides some of the functions/operators. The profiler needs to instrument all the allocation calls together with a callstack. The callstack is necessary for the performance engineer to know where the object was allocated in the first place. The allocation calls must intercept all the different types of object

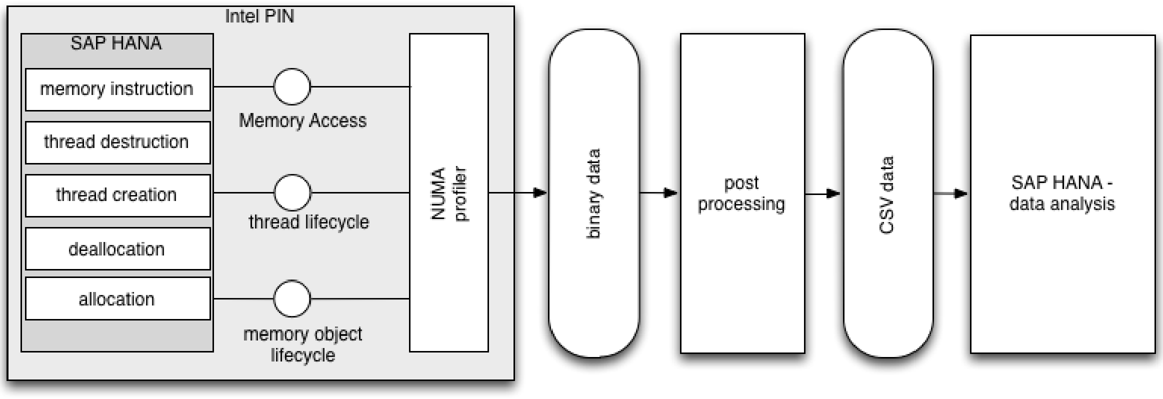


Figure 3. NUMA4HANA architecture

creation. To get the whole lifecycle it would be neccessary to also instrument the destruction calls. Since the profiler saves a timestamp together with the memory object allocation we can assume that if an object-adress is reused later, the original object got destroyed. This saves effort to not instrument the memory destruction calls. The log entry for a memory object allocation should contain which thread, running on which core has allocated a memory object on which numa-node at which position in the code at which time.
The memory object accesses from the threads are sampled for performance reasons. The rate is configurable. The log entry consists of a timestamp, the CPU core on which the thread runs on, the NUMA node on which the memory object resides currently and an indicator whether it is an read or a written memory operation. All the log entries are written in a binary format to get maximal throughput since, depending on the workload, many entries have to be written to disk. Currently no caching behavior is considered.

4.1 Layout of the collected data

In a postprocessing step the log files for the different threads for memory operations and memory object allocation are converted to csv files. This is necessary to import them directly into a SQL database. In the database the profiler use a specific table schema to help analyzing the data quickly. A typical analysis of the gathered data consists of the questions: Which memory object is accessed most by which threads? Which source lines cause the most remote memory object accesses? All the questions can be answered by SQL statements. Additionally it is possible to identify certain patterns in the data. Typically it is interesting to find memory object accessed which are read only by threads from different nodes. Another pattern is the accessing of memory objects of mixed read and written memory operations from different nodes. After finding patterns in the outcome of the profiler a performance engineer can choose to incorporate several strategies to gain more performance. One approach is to alter the allocation of the memory objects according to the following three strategies:

- Migration: memory pages get migrated from one NUMA node to another
- Interleaving: memory pages get equally distributed among the NUMA nodes
- Replication: memory pages get replicated among all or a subset of the NUMA nodes

Is a memory object mostly accessed by threads from one specific socket it is reasonible to use the migration strategie. With the knowledge about where the memory object was allocated it is possible to alter the mechanism and let the memory object be colocated at the according socket. Interleaving a memory object is recommendet if that object is read and written by threads from several sockets. If a memory object is accessed read-only by threads from several sockets a Replication of that object across the sockets is advisable.
The database layout as seen in Figure 4 are filled with the according data. The profiler gathers information about the topology of the machine running the application. This includes information which core is located on which NUMA node (table CoreToNode) and how high the communication costs between two nodes are (table Topology). For every instruction executed in the application information about the original source location are gathered (table SourceMapping). Data about thread accessing memory objects is stored in the table MemoryAccess. The data includes the thread id, the core on which the thread was executed, the instruction pointer, the memory address and the NUMA node on which the memory resides at the time of access. In the table MemoryObject and Callstack all informations about memory object creation are saved. For every allocation the thread id, the core on which the thread was executed, the size of the memory object,

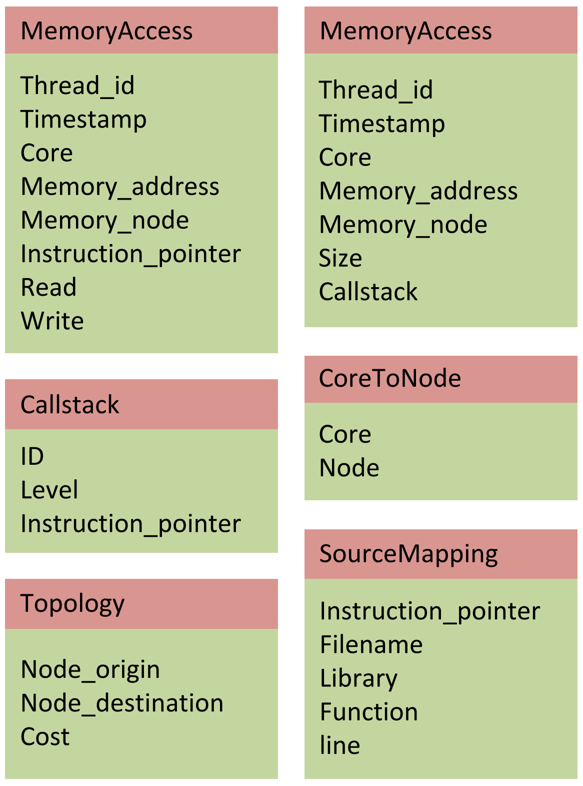


Figure 4. The database layout of the profiler

the memory object base address and a reference to the according callstack. A callstack includes the level together with the instruction pointer to give the performance engineer a hint where the memory object was allocated in the first place.

5 Future Work

The presented approach to develop a customized profiler for applications with their own memory management has several limitations. Currently there is no considerations of the cache behavior. This is rather challenging because the profiler relies on a debug build with debug symbols together with the invasive instrumentation it influences the execution of the application. The question arises if such a influenced execution is significant to study the cache behavior of the application. Nevertheless it is possible to use a cache emulator. This was not the scope of this research. Not all memory operations are relevant to the performance engineer. Therefore it is desirable to filter memory operations only from specific libraries inside the application. Currently the profiler starts sampling memory operations with the start of the application. The profiler should be able to be remotely controlled when the logging of the memory operations starts. Another option could be that the sampling of memory operations is triggered, depending on certain events, such as start of delta merge. Applications have memory intensive phases, so the sampling rate should be adjustable to the current memory operations throughput. This reduces the overhead introduced by the profiler. As with the relevant memory operations, the performance engineer is interested in certain memory objects from libraries or threads only. The profiler should be able to filter the allocations of only those memory objects to reduce the amount of captured data. The overall captured data should be automatically processed to find patterns and give suggestions for the performance engineer or developer for improvements in the code. The outcome of several runs with the profiler should be automatically compared to see if an improvement was achieved. For all the mentioned task there should be a user interface where the performance engineer or developer can easily interact with the outcomes of the profiler.

References

[1] J. Antony, P. P. Janes, and A. P. Rendell. Exploring thread and memory placement on numa architectures: Solaris and linux, ultrasparc/fireplane and opteron/hypertransport. In *High Performance Computing-HiPC 2006*, pages 338–352. Springer, 2006.

[2] M. M. Bach, M. Charney, R. Cohn, E. Demikhovsky, T. Devor, K. Hazelwood, A. Jaleel, C.-K. Luk, G. Lyons, H. Patil, and A. Tal. Analyzing parallel programs with pin. *Computer*, 43(3):34–41, Mar. 2010.

[3] S. Blagodurov, S. Zhuravlev, M. Dashti, and A. Fedorova. A case for numa-aware contention management on multicore systems. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, pages 1–1, Berkeley, CA, USA, 2011. USENIX Association.

[4] J. Casazza. First the tick, now the tock: Intel microarchitecture (nehalem). 2009.

[5] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quema, and M. Roth. Traffic management: A holistic approach to memory placement on numa systems. *SIGARCH Comput. Archit. News*, 41(1):381–394, Mar. 2013.

[6] C. N. Keltcher, K. J. McGrath, A. Ahmed, and P. Conway. The amd opteron processor for multiprocessor servers. *IEEE Micro*, 23(2):66–76, Mar. 2003.

[7] C. keung Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Janapa, and R. K. Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *In Programming Language Design and Implementation*, pages 190–200. ACM Press, 2005.

[8] R. Lachaize, B. Lepers, and V. Quéma. Memprof: a memory profiler for numa multicore systems. In *USENIX ATC*, 2012.

[9] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden. Ibm power6 microarchitecture. *IBM J. Res. Dev.*, 51(6):639–662, Nov. 2007.

[10] C. McCurdy and J. Vetter. Memphis: Finding and fixing NUMA-related performance problems on multicore platforms. *2010 IEEE International Symposium*

on Performance Analysis of Systems & Software (ISPASS), pages 87–96, Mar. 2010.

[11] D. S. Nikolopoulos, T. S. Papatheodorou, C. D. Polychronopoulos, J. Labarta, and E. Ayguad. User-level dynamic page migration for multiprogrammed shared-memory multiprocessors. In *ICPP*, pages 95–104, 2000.

[12] J. Tao and W. Karl. A tool environment for efficient execution of shared memory programs on numa systems. In *In Proceedings of the Fourth International Workshop on Advanced Parallel Processing Technologies (APPT'01*, pages 156–165, 2001.

[13] J. Tao, W. Karl, and M. Schulz. Visualizing the memory access behavior of shared memory applications on numa architectures. In V. N. Alexandrov, J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, editors, *International Conference on Computational Science (2)*, volume 2074 of *Lecture Notes in Computer Science*, pages 861–870. Springer, 2001.

[14] J. Tao, W. Karl, and M. Schulz. Memory access behavior analysis of numa-based shared memory programs. *Sci. Program.*, 10(1):45–53, Jan. 2002.

Visualization of Bag-of-Visual-Words Classifiers

Christian Hentschel, Peter Retzlaff, Harald Sack
Hasso - Plattner - Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
christian.hentschel@hpi.uni-potsdam.de
peter.retzlaff@student.hpi.uni-potsdam.de
harald.sack@hpi.uni-potsdam.de

Abstract

Visual Concept Detection aims to classify images based on their content. Bag-of-Visual-Words features that quantize and count local gradient distributions in images have proven to be meaningful image representations. In combination with supervised machine learning approaches, models for nearly every visual content can be learned by sufficient labeled training data. Similar to text classification the classification of an image depends on a weighted combination of the visual words from the vocabulary and the classifier is expected to learn these weights in order to obtain best results. Hence, the learned weights or importances of each visual word are a strong indicator for the robustness and reasonableness of the overall concept model. This work visualizes learned concept models by colorizing the image pixels using the importance value of the corresponding local visual by means of a heat map-like representation. Thereby, we explicitly show sources of misclassification and thus help to understand and improve varying results for different concept classes.

1. Introduction

Given a set of images and a set of concepts, the task of visual concept detection is to automatically assign one or more of these concepts to each of the images solely based on the visual content [5]. An approach that is commonly used to solve this task is the Bag-of-Visual-Words (BoVW) model, which extends an idea from text retrieval to visual classification [8]. In text classification systems, each text document is usually represented by a histogram of the frequencies of a set of vocabulary words. Similarly, an image can be described as a frequency distribution of *visual words*.

While the notion of a word in natural languages is clear, visual words are more difficult to describe. Typically, local image characteristics are used to represent visual words. Histograms of gradients, such as Scale Invariant Feature Transform (SIFT, [7]) features have been successfully used in BoVW classification. In order to provide some invariance to small changes within the appearance of objects and to reduce the computational complexity, the number of SIFT features is reduced by vector quantization approaches such as k-means and Gaussian Mixtures [6]. The derived cluster centers represent the visual vocabulary used to describe all images in the same way: By assigning the local SIFT features of each image to the most similar vocabulary vector, a histogram of visual word vector frequencies is generated per image. This frequency distribution is referred to as *Bag-of-Visual-Words* and represents a global image descriptor that can be used in subsequent machine learning steps. Typically, the learning stage optimizes a weight vector that emphasizes different features (i.e. visual words) depending on the classification task. Support Vector Machines (SVM) have become the default choice in most visual concept detection approaches [9]. While this approach often provides highly accurate classification results, analysis of reasons for misclassification (as well as for correct classification) tends to be difficult.

The BoVW approach, however, usually works as a black box and the classification rules learned are not transparent to the user. This is due to the fact that visual vocabulary words are hard to interpret for a human. Each vocabulary word is a prototype for a number of local SIFT features and each SIFT feature represents local gradient distributions at a specific image region. Hence, in contrast to text classification, the meaning of each vocabulary word is only implicitly available, which leads to considerable uncertainty when interpreting the classification result.

In this work we present an approach to visualize the impact of local image regions on the overall classification results by superposing the images with a heat map-like graphical representation of the learned visual word weights. This can be done by retrieving those local SIFT features in an image that were quantized into the same visual word and assigning the respective visual word's weight to the pixels of the image region contributing to these features. The obtained visualization provides an intuitive way of interpreting trained visual concept models by simply analyzing the image regions that contribute most (and least) to the overall result. Sources of misclassification are made explicit such as ill-chosen training examples that exhibit specific characteristics not specific for the actual concept to be classified. In order to avoid that a specific learning algorithm has an impact on the chosen visualization we have trained models for three different classifiers, namely linear SVMs, AdaBoost and Random Forests.

2 Implementation and Future SOC Lab resources

This section describes the implementation aspects of our approach focusing on the exploitation of the Future SOC SMP multicore architecture. We used a machine equipped with 24 processor cores and 64 GB of main memory. Development and testing was done on a local machine and the Future SOC was used to perform computational expensive and long-running tasks on the complete dataset. This includes feature extraction, training and grid search, and visualization. All of these tasks were parallelized and benefit largely from a multi-core architecture.

2.1 Feature Extraction

In our experiments we compute BoVW concept models for the 102 (101 plus background) classes of the Caltech-101 benchmark dataset [2]. We extract SIFT features at a dense grid of $s = 6$ pixels and at a fixed scale of $\sigma = 1.0$ and use k-means clustering to quantize the SIFT features to $k = 100$ vocabulary vectors. Thus, each image is described by a 100-dimensional histogram of visual words (see [4] for implementation details). These histograms will be used in the following as labeled samples for model training.
In order to increase the performance of Bag-of-Visual-Words extraction, each step of the process runs concurrently, making use of the available Future SOC multicore architecture. Extraction of SIFT features on the 9.247 training and testing images took approximately 30 minutes compared to 12 hours when computed on a single core (linear scaling is assumed here, since images are scaled to equal size and thus the number of SIFT features extracted per image is the same). Generating the prototypes of visual words using k-means clustering can also be parallelized since it consists of mainly independent distance computations between training vectors. Finally, nearest-neighbor search for computing the Bag-of-Visual-Words histograms can be parallelized over all images and therefore again provides a linear speed-up of factor 24.

2.2 Concept Training and Grid Search

Linear SVMs compute a linear hyperplane to best separate positive from negative samples of a given class in the original feature space. The trained model consists of a bias and a weight vector and an unknown sample is classified by simply computing the dot product between the weight vector and the sample's feature vector (plus the bias). While the classification results usually tend to be inferior to the results of non-linear SVMs, the linear model allows for an intuitive interpretation of the (normalized) weight vector as importance scores for the respective feature.
Ensemble methods, such Random Forests [1] and AdaBoost [3], are often based on decision trees where each node directly maps to a specific feature in the training set. These methods typically select features based on their capability of solving the classification problem beginning with the most perfect split, e.g. by computing the decrease in entropy of the obtained class separation. We use the *mean decrease in impurity* over all decision trees in an ensemble as direct indicator for feature importance.
Our implementation is based on the scikit-learn[1] library that provides Support Vector Machine Solvers and Ensemble methods. We train binary classifiers for each of the 101 concept classes in the Caltech-101 dataset using the 'background' data class as negative samples. Training is performed on 50% of the dataset images while the remainder is used for validation purposes of the obtained models. Different hyper parameters for each classifier have been optimized in a nested cross-validation by splitting the training set into four folds and performing an inner grid search with three-fold cross-validation for each of these folds. We optimize the number of decision trees for both, AdaBoost and Random Forests, the maximum depth of each tree for AdaBoost and tune the the regularization parameter C for linear SVM. We have selected reasonable parameter ranges for each parameter and create a parameter grid by generating all possible combinations of the respective values.
While the respective training algorithms are hard to parallelize, we utilize the Future SOC's parallel architecture to evaluate as many parameter combinations in parallel as possible. As the models for each parameter combination can be learned independently of each other, this greatly reduces the time needed for grid search and thus enabled us to explore a much larger space of combinations in the given time. Hence, grid search was parallelized by delegating the training for each parameter combination to a dedicated core.

2.3 Visualization

As discussed in Section 1 we have decided to apply a *heat map* like representation to colorize each pixel

[1]scikit-learn – `http://scikit-learn.org`

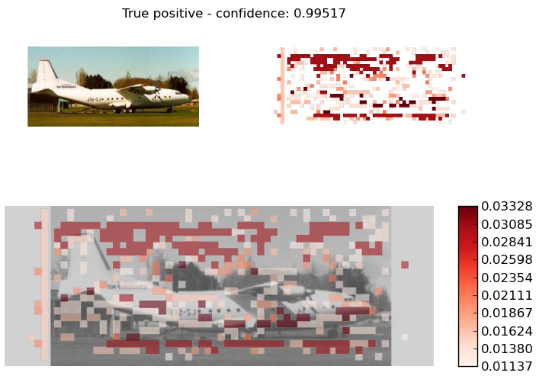


Figure 1: Visualization of feature importances of the AdaBoost classifier trained for the visual concept “airplanes”. Top left: original image. Top right: heat map of the upper quartile of the learned feature importances. Bottom: Desaturated original image with the superposed heat map.[2]

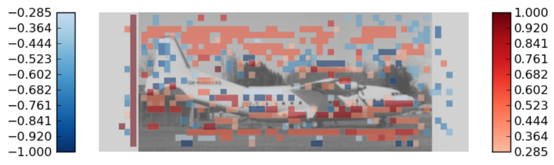
in the image, based on its importance score. This requires to map the learned importance scores of visual words to pixels in the image plane. Again, we rely on the Future SOC parallel architecture to reduce the time needed for this process, as multiple images can be processed in parallel. For the computation of the heat maps a near-linear speed-up can be achieved. However, disk I/O represents a bottleneck for the parallelization of this stage, as the resulting images need to be written to disk.

Since each feature of a BoVW-vector corresponds to a visual word in the vocabulary and the value of each feature is generated by binning local SIFT descriptors to the most similar visual words we can extend the learned importance scores to the respective SIFT descriptors. During the descriptor extraction process, we simply save the pixel coordinates of the support region of each SIFT descriptor and therefore are likewise able to correlate feature importances to image regions. As the support regions of neighboring descriptors overlap (by default SIFT uses a support region of 16×16 pixels and our dense sampling step size is set to $s = 6$ pixels) the importance score of each pixel is therefore set to be the maximum of all corresponding visual words. The rationale behind using the maximum is the intention to visualize the *most important* visual words.

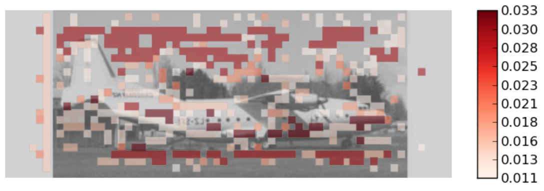
Figure 1 shows the resulting visualization for a sample of the category “airplanes”, correctly classified by our AdaBoost model with a confidence score of $c = 0.99$. For reasons of clarity we limit the visualized pixel contributions to the most important visual words, i.e. only the upper quartile of the importance scores obtained per visual word are shown. Darker areas mark more important regions and white pixels have least impact on the classification result. The visualization shows the original image, the heat map alone as well as both superimposed.

[2]All figures are best viewed in color and magnification.

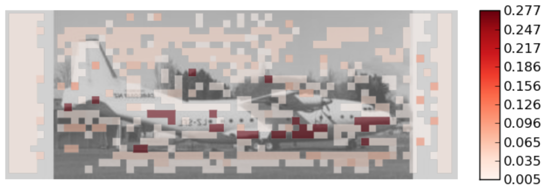
(a) Original



(b) linear SVM



(c) AdaBoost



(d) Random Forests

Figure 2: Visualization of feature importances of three different classification models.[2]

Finally, we compare the visualizations of the three classifications models for linear SVM, Random Forests and AdaBoost (see Fig. 2). Again, we restrict the highlighted regions to the upper quartile of the most important visual words for reasons of clarity. The visualization of the SVM model differs in that since SVMs produce negative as well as positive weights we visualize them using different colors (blue for negative weights, red for positive). Similar to AdaBoost and Random Forests the upper quartile of the most important positive weights and the most important negative weights are selected.

The visualizations confirm the similarity between AdaBoost and Random Forests. Both ensemble models produce almost identical heat maps that differ only in the absolute values of the respective importance scores. Surprisingly, the visualization of the trained SVM model is also very similar to those of AdaBoost and Random Forests.Please note that regions which have been assigned a high *negative* importance weight (color coded in blue) have likewise been selected by the ensemble methods as important.

When analyzing the visualizations with regard to the aimed ability to explain the classification results the figures immediately convey that considerably few important regions actually coincide with the pixels that belong to the airplane object itself. All three classification models assess the sky above the airplane as important features for classification of airplanes. While this seems to some extent reasonable ("airplanes are objects surrounded by sky") it likewise means, that other images with large sky-like areas will have a high chance of being falsely classified as "airplanes" as well (e.g. "birds"). When comparing the three different models, the Random Forest model stresses features of the airplane object more than the other models and can therefore be expected to generalize better w.r.t. the aforementioned restriction.

A second aspect that also becomes immediately apparent due to the visualization is that many photos of airplanes exhibit a white frame that surrounds the actual photo (in fact only 108 out of 800 images in the Caltech-101 dataset annotated as "airplanes" are not surrounded by a more or less prominent white border). While all three classification models have correctly learned this specificity of an airplane shot typically featuring a white border by selecting border pixels among the top 25% most important features it represents presumably an unwanted characteristic and classifying photos of airplanes that do not exhibit a white border will most likely show inferior classification results.

3 Summary

In this paper we have presented an approach for an intuitive visualization of different Bag-of-Visual-Words models. We have trained three different classifiers, linear SVM, Random Forests and AdaBoost, to prove the universality of our visualization scheme and compared the performance of these classifiers based on the Caltech-101 benchmark dataset. Our results indicate a comparable performance of all three classifiers. The use of Future SOC Lab resources helped to significantly reduce time required for feature extraction and model training.

The visualization we propose is based on a heat map like representation of the importance scores of all visual words as assigned by the respective classifier. We have shown that this representation enables the user of a BoVW-based visual concept classifier to understand *how* a decision is made rather than being restricted to simply accept the decision made. The provided examples show the efficiency of our visualization. Deficits in the model's ability to generalize from the the training examples as well as peculiarities within the Caltech-101 training material became immediately apparent by analyzing a single testing instance.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] R. Fergus and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, Aug. 1997.

[4] C. Hentschel, S. Gerke, and E. Mbanya. Classifying images at scene level: comparing global and local descriptors. 7836:72–82, 2013.

[5] M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection. In *Proceedings of the international conference on Multimedia information retrieval - MIR '10*, page 527, New York, New York, USA, 2010. ACM Press.

[6] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–16, Sept. 2012.

[7] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2. IEEE, 1999.

[8] J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, number Iccv, pages 1470–1477. IEEE, 2003.

[9] C. G. M. Snoek and M. Worring. Concept-Based Video Retrieval. *Foundations and Trends in Information Retrieval*, 2(4):215–322, 2009.

Regional climate simulations for West Africa: comparison of input bias correction methods

Dominikus Heinzeller[1]
[1]Institute of Meteorology and Climate Research, Karlsruhe Institute of Technology
Kreuzeckbahnstr. 19, 82467 Garmisch-P.
heinzeller@kit.edu

Harald Kunstmann[1,2]
[2]Department of Geography
Augsburg University
86135 Augsburg
harald.kunstmann@kit.edu

Abstract

Regional climate simulations are valuable tools to study climate change on local scales, yet do they often carry large biases. These stem from the bias of the regional model itself and from the bias of the driving global model. In this project, we developed a program to correct the global model data prior to ingesting it into the regional climate model. Two different algorithms favored by the climate modeling community were implemented in a fully parallelized Python program. Using regional climate simulations over West Africa, we compared the effects of both methods to the uncorrected input data and found that can both improve the accuracy of the regional climate model.

1. Introduction

West Africa is mostly covered by semi-arid regions with a strong variability in rainfall on intra-seasonal, inter-annual and inter-decadal time scales. This makes it a region highly vulnerable to climate change due to a very low adaptive capacity. Conversely, the West African monsoon precipitation response to future anthropogenic climate change is highly uncertain due to a large spread among the climate projections [6].

In West Africa, climate change projections have often been derived using global circulation models (GCMs). These are limited by their coarse grid spacing and often have problems in representing accurately the main West African Summer Monsoon (WASM) features [10]. Regional climate models (RCMs) are limited area models applied at higher resolution than GCMs and driven by GCM data at the lateral boundaries. The increase in resolution allows for a better representation of fine-scale forcing and land surface heterogeneities, important aspects of the physical response governing local and regional climate change signals [8].

Yet, a common problem of regional climate simulations are biases in physical quantities that limits their accuracy. These biases are of two origins, namely the bias introduced by the regional climate model itself, and the bias inherent in the driving GCM data. The bias of the RCM can be reduced by a suitable model configuration, derived from control runs using re-analysis data as lateral boundary conditions. The GCM bias on the other hand needs to be dealt with prior to ingesting it into the regional model.

In this project, we implemented two different bias correction algorithms and conducted regional climate simulations at 18km resolution over West Africa using the so-obtained data as lateral boundary conditions. The large amount of GCM data and the complexity of the algorithms required an efficient implementation, which we accomplished using Python and the in-memory NoSQL database Redis.

2. Bias correction methods

Two concurring approaches are currently favored among the climate modeling community and a clear concensus has not been found yet. Both methods rely on re-analysis data as reference (“truth field”) to correct the global model. Here, we used the ERA-Interim re-analysis [5] as reference (REF) data, and the MPI-ESM [9] as GCM data (see Fig. 1).

Pseudo-global warming method (pgw) [7] In this approach, model differences are calculated between a ten-year period at present and a ten-year period in the future from a GCM for each month for temperature, humidity, geopotential height and wind. These differences are then added to a current climate re-analysis to obtain a warming signal. This approach allows one to see how “current weather” would look like in a future climate, rather than to detect large degrees of changes to the atmospheric circulation patterns. It is therefore suitable for process studies as it preserves key climate features like patterns of inter-annual variability.

Perturbed average climate approach (pac) [2] Here, 6-hourly GCM and re-analysis data for a ten-year reference and a ten-year application period are

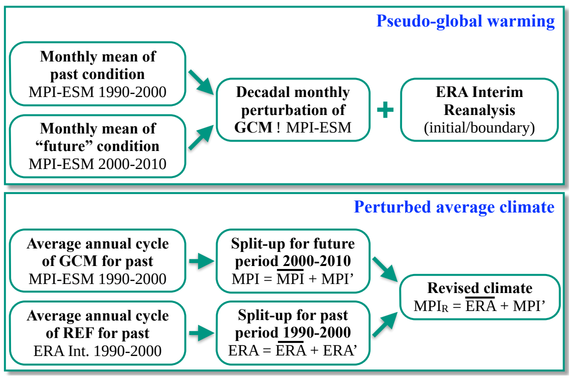

Pseudo-global warming
Monthly mean of past condition MPI-ESM 1990-2000
Monthly mean of "future" condition MPI-ESM 2000-2010
Decadal monthly perturbation of GCM ! MPI-ESM
+
ERA Interim Reanalysis (initial/boundary)
Perturbed average climate
Average annual cycle of GCM for past MPI-ESM 1990-2000
Average annual cycle of REF for past ERA Int. 1990-2000
Split-up for future period 2000-2010 MPI = MPI + MPI'
Split-up for past period 1990-2000 ERA = ERA + ERA'
Revised climate MPIR = ERA + MPI'

Figure 1. Bias correction methods.

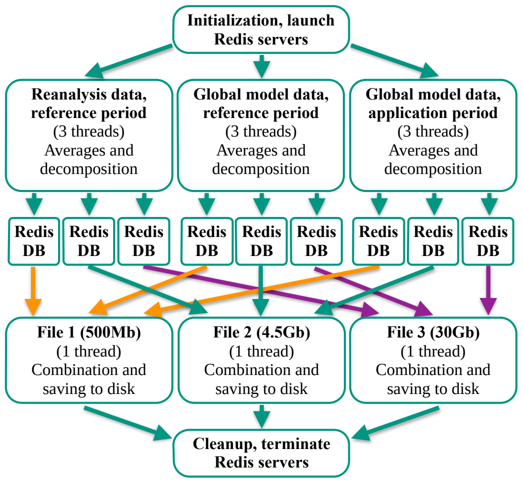

Initialization, launch Redis servers
Reanalysis data, reference period (3 threads) Averages and decomposition
Global model data, reference period (3 threads) Averages and decomposition
Global model data, application period (3 threads) Averages and decomposition
Redis DB
Redis DB
Redis DB
Redis DB
Redis DB
Redis DB
Redis DB
Redis DB
Redis DB
File 1 (500Mb) (1 thread) Combination and saving to disk
File 2 (4.5Gb) (1 thread) Combination and saving to disk
File 3 (30Gb) (1 thread) Combination and saving to disk
Cleanup, terminate Redis servers

Figure 2. Flowchart of parallelization.

broken down into an average annual cycle plus a perturbation term. The revised GCM data for the application period are then constructed from the average annual cycle of the re-analysis data and the perturbation term of the GCM data. This method attempts to allow one to look at changes in circulation and storm frequency patterns *and* changes in thermodynamic variables by imposing a mean bias correction.

3. Numerical approach

In our original proposal, we were planning to implement both bias correction methods using a hybrid Python/Fortran code, where Python as the main programming language makes parallel calls to Fortran routines. It turned out that the Python threading module, which offers the advantage of shared memory parallelization, is not suitable due to the global interpreter lock problem in Python [4]. The parallelization using Python multiprocessing on the other hand is limited to private memory use. To overcome the problem of communication between the individual processes, a parallelized Redis database (`http://redis.io`) was introduced in the code. Embedding Fortran code (using f2py) in the code did not improve the performance, since large data arrays need to be passed (copied) into and out of the Fortran routines and a direct communication with the Redis database is not possible due to the lack of a Fortran API.
For practical reasons, the global climate simulation data and re-analysis data is split into several files for each of the two decades: two files containing slowly changing 2D data (500Mb and 4.5Gb), and one file containing 6-hourly 3D data (30Gb). In the original pilot code (A), data was read consecutively from each of the files and stored in Python dictionaries. A parallelization by files allowed to run the code with nine threads in parallel and reduced the runtime to about 35% (B). Since two of the files are negligible in size compared to the third one, it is the parallel handling of the latter one which governs the runtime.
Next, we replaced (1) the data handling using Python dictionaries by numpy arrays and (2) the communication between processes using shared variables by the in-memory NoSQL database Redis (C). Since Redis is a serial application, we extended the code to dynamically launch and use as many Redis servers as parallel processes are created. To reduce the memory usage, adequately-sized blocks of the numpy arrays are encoded as binary strings for storage in Redis. These steps led to a reduction in runtime to about 2.5% of the initial pilot code, leaving the disk I/O as the primary bottleneck in the current implementation. Figure 2 illustrates the program in a flowchart, and Table 1 summarizes the increase in performance from code version A to C on a Fujitsu RX600 S5 2 blade.

Table 1. Runtime performance.

Code	Period length	Peak mem.	Runtime
A	2×30 days	3.7Gb	73min
B	2×30 days	7.6Gb	25min
C	2×30 days	2.7Gb	1.45min
C	2×10 years	320Gb	185min

4. Scientific evaluation

To compare the effects of the two bias correction methods on the regional climate projections, we set up a limited area domain at 18km resolution over West Africa, depicted in Figure 3. In addition to the bias-corrected data, we used ERA-Interim data and raw Echam6 data from 2000 to 2010 to drive the RCM.
These simulations are still ongoing and a full assessment of their skills yet remains to be done. Here we report on one particular example where the influence of the bias correction is clearly visible and beneficial. Over the West African continent, in particular the drier Sahel zone (∼ 12–18°N), the annual rainfall is dominated by contributions from a few events during the

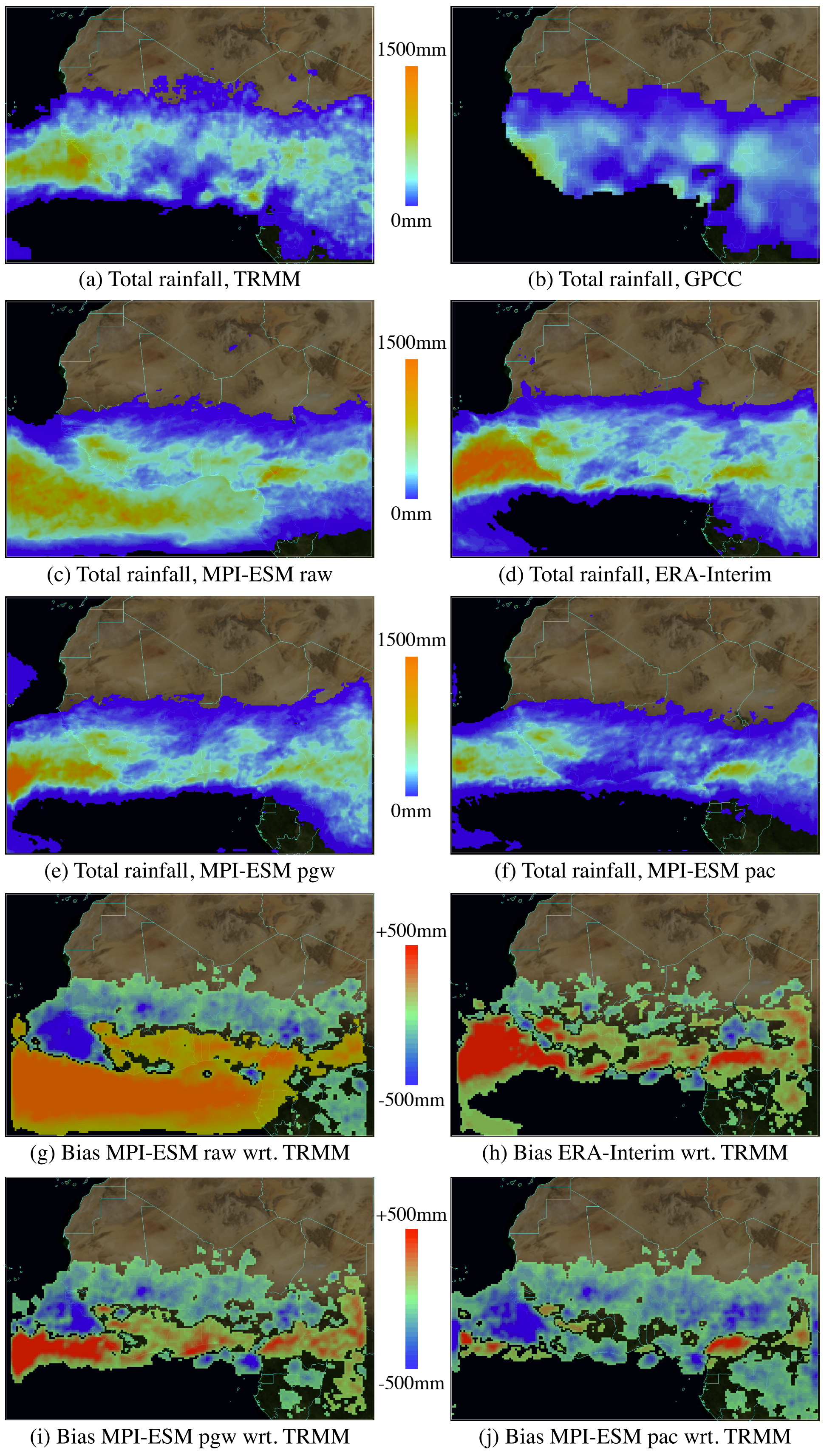

1500mm
0mm
(a) Total rainfall, TRMM
(b) Total rainfall, GPCC
1500mm
0mm
(c) Total rainfall, MPI-ESM raw
(d) Total rainfall, ERA-Interim
1500mm
0mm
(e) Total rainfall, MPI-ESM pgw
(f) Total rainfall, MPI-ESM pac
+500mm
-500mm
(g) Bias MPI-ESM raw wrt. TRMM
(h) Bias ERA-Interim wrt. TRMM
+500mm
-500mm
(i) Bias MPI-ESM pgw wrt. TRMM
(j) Bias MPI-ESM pac wrt. TRMM

Figure 4. Model evaluation: total rainfall in July 2001 over West Africa.

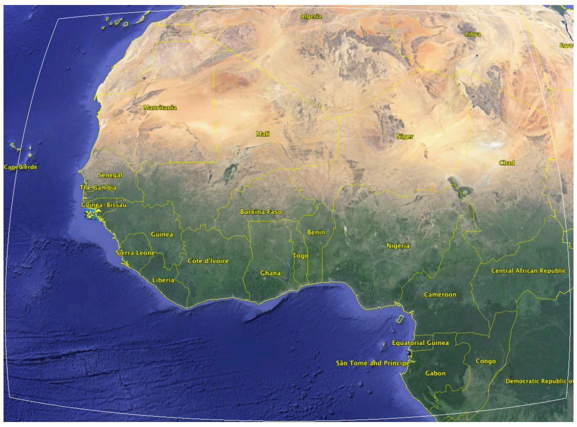


Figure 3. West Africa model domain.

Table 2. Evaluation of modeled rainfall.

Model	ME	MAE	STD	PCC
ERA-Interim	42.41	73.14	171.98	0.83
(land only)	10.54	36.03	85.22	0.80
(sea only)	31.87	37.11	151.61	0.88
MPI-ESM raw	68.41	150.63	267.32	0.27
(land only)	-10.00	45.75	97.21	0.68
(sea only)	78.41	104.88	245.85	0.11
MPI-ESM pgw	6.96	72.35	172.80	0.65
(land only)	-5.93	39.79	84.05	0.75
(sea only)	12.89	32.56	150.47	0.60
MPI-ESM pac	-43.95	62.36	111.26	0.76
(land only)	-33.18	43.99	84.93	0.70
(sea only)	-10.77	18.37	76.69	0.82

ME: mean error/bias in mm; MAE: mean absolute error in mm; STD: standard deviation in mm; PCC: Pearson correlation coefficient.

West African summer monsoon from July to September. Figure 4 displays the total monthly precipitation in July 2001, i. e., 18 months after model initialization, of the WRF regional climate model driven by the respective lateral boundary conditions. Observation data, based on station data and satellite-gauge merged precipitation products, are displayed in the top panels: the 0.25×0.25 degree TRMM data [3] and the 1×1 degree GPCP 1DD data [1]. A basic statistical evaluation of the model results against the TRMM data is summarized in Table 2. First detail to note is that the TRMM and GPCC observations show similar large-scale patterns but differ on smaller scale, illustrating a latent uncertainty in the observation data. With respect to the model performance, striking features are (1) the massive rainband south of the coast in the run driven by MPI-ESM raw data and (2) the area of intensive rainfall southwest of the coast in the ERA-Interim-driven run. In this particular case, the two bias correction methods remove both of these unwanted features to a certain extent. The model driven with MPI-ESM pgw data shows rainfall patterns closer to the ERA-Interim data, but an overall reduction in precipitation. The model driven with MPI-ESM pac data is notably dryer, in fact to dry, but shows least of the unwanted intensive rainfall patterns discussed above. The Pearson Correlation Coefficient is best for the ERA-Interim run and worst for the uncorrected MPI-ESM raw run. This indicates that both bias correction methods have the potential to improve the regional climate projections.
An important point to mention is that both bias correction methods considered here rely on an averaging of reference and model data over a certain period. A comparison of model results for a particular year and month is therefore of limited relevance in the framework of long-term climate simulations.

5. Conclusion and Outlook

We implemented and optimized two bias correction methods for GCM data, input to regional climate projections. In several steps, the initial pilot code was improved, leading to a 97.5% reduction in runtime. Further improvements need to be made to the disk I/O routines. Currently, compressed NetCDF4 data containing the initial and boundary data from the global climate simulations is read and written in serial using the PyNIO library. A promising strategy therefore is to implement parallel disk I/O using the Unidata C/C++ NetCDF4 library. Since Redis provides interfaces to C, C++ and Python, this can be incorporated in the existing codebase.
A first model comparison shows that the bias correction can have a positive impact on the models. However, a statistical analysis of the full ten-year period 2000–2010 is required for sound conclusions. Further, a ten-year reference and application period for the bias correction algorithm may not be enough to smooth out out inter-annual variability and patterns such as the El Niño-Southern Oscillation [2].

References

[1] Adler, R.F., Huffman, G.J., Chang, A., and 11 co-authors: *The Version 2.1 Global Precipitation Climatology Project (GPCP) Monthly Precipitation Analysis (1979-Present).* J Hydrometeor., 4(6): 1147–1167, 2003

[2] Done, J.M., Holland, G.J., Bruyere, C.L., Leung, L.R., Suzuki-Parker, A.: *Modeling high-impact weather and climate: Lessons from a tropical cyclone perspective.* NCAR Technical Note NCAR/TN-490+STR, 2012

[3] Huffman, G.J., Adler, R.F., Bolvin, D.T., and 6 co-authors: *The TRMM Multi-satellite Precipitation Analysis: Quasi-Global, Multi-Year, Combined-Sensor Precipitation Estimates at Fine Scale.* J. Hydrometeor., 8: 38–55, 2007

[4] Beazley, D.: *Inside the Python GIL.* Python Concurrency Workshop, Chicago, May 14–15, 2009

[5] Dee, D.P., Uppala, S.M., Simmons, A.J., and 33 co-authors: *The ERA-Interim reanalysis: configuration and performance of the data assimilation system.* Quarterly Journal of the Royal Meteorological Society 137(656): 553-597, 2011

[6] Giannini, A., Biasutti, M., Held, I.M., Sobel, A.H.: *A global perspective on African climate.* Climate Change, 90: 359-383, 2008

[7] Rasmussen, R., Liu, C., Ikeda, K., and 12 co-authors: *High resolution coupled climate-runoff simulations of seasonal snowfall over Colorado: a process study of current and warmer climate.* J. Climate, 24: 3015–3048, 2010

[8] Rummukainen, M.: *State-of-the-art with regional climate Models.* Climate Change, 1: 82–86, 2010

[9] Stevens, B., Giorgetta, M.A., and 15 co-authors: *Atmospheric component of the MPI-M Earth System Model: ECHAM6.* Journal of Advances in Modeling Earth Systems, 5: 146–172, 2013

[10] Sylla, M.B., Gaye, A.T., Jenkins, G.S., Pal, J.S., Giorgi, F.: *Consistency of projected drought over the Sahel with changes in the monsoon circulation and extremes in a regional climate model projections.* Journal of Geophysical Research, 115: D16108, 2010

Investigation of the integration of Apache Hadoop in SAP HANA

Prof. Dr. Ali Reza Samanpour
Fachhochschule Südwestfalen
Lindenstrasse 53
59872 Meschede
samanpour.ali-reza@fh-swf.de

André Ruegenberg, B.Eng.
Fachhochschule Südwestfalen
Lindenstrasse 53
59872 Meschede
ruegenberg.andre@fh-swf.de

Abstract

This report describes how Hadoop can be used alongside SAP technologies such as HANA. There are some major differences between these technologies. Apache Hadoop uses a network of servers to handle big sizes of data in ranges of Petabyte or potentially Exabyte, which is much higher than the range that SAP HANA or other conventional relational database management systems typically handle. But on the other hand, the Hadoop cluster is significantly slower than SAP HANA. The Hadoop cluster can take some minutes or hours to provide analytic results. This means that Hadoop will not enable you to understand your business at the speed of thought. However, by allowing you to store and access more voluminous and detailed data at lower cost, it lets you drill deeper and in different ways into the data underlying your business.

The result is that by putting SAP HANA and Hadoop together you have the potential to handle really big data really fast.

1 Introduction

Apache Hadoop is an open-source project governed by the Apache Software Foundation that allows the distributed processing of large data sets across clusters of computers. It is well suited for storing unstructured or semi-structured data, is good for manipulating very large files and is tolerant to hardware and software failures. The Hadoop family of products includes the Hadoop Distributed File System (HDFS), MapReduce, Pig, Hive, HBase, and much more. All of them are available as open source from Apache. In this report, the term "Hadoop" usually means the entire Hadoop family of products, regardless of their open source or vendor origins.

It is possible to use Hadoop as a flexible data store by storing data from various sources like social data streaming data, transaction data, etc.

2 Hadoop software architecture

A key feature of the Hadoop software architecture is the way it separates how data is stored from the way it is processed. Hadoop runs on hundreds or thousands of "low-cost" servers. Most of the servers are, in Hadoop terminology, "Data-Nodes", each of which contains just a part of the data.

Hadoop has a single level of data storage called the Hadoop Distributed File System (HDFS). It stores data using native operating system (i.e. Linux) files. This means Hadoop can support any type of data and data can be dumped in HDFS without directly using Hadoop software. This architecture allows multiple computation engines to run on top of Hadoop and leverage both HDFS and the MapReduce programming model.

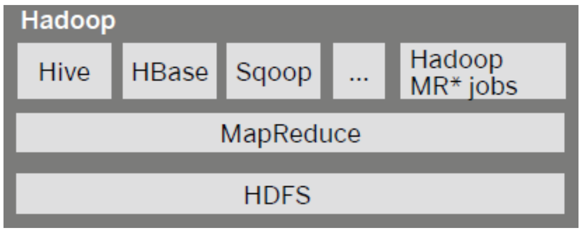


Figure 1: Hadoop software architecture

2.1 Programming Model

The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: *Map* and *Reduce*.

Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function.

The *Reduce* function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just zero or one output value is produced per *Reduce* invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory.

2.2 Example

Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code:

```
    map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
    EmitIntermediate(w, "1");
    reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
    result += ParseInt(v);
Emit(AsString(result));
```

The *map* function emits each word plus an associated count of occurrences. The *reduce* function sums together all counts emitted for a particular word.

In addition, the user writes code to fill in a *mapreduce specification* object with the names of the input and output files, and optional tuning parameters. The user then invokes the *MapReduce* function, passing it the specification object. The user’s code is linked together with the MapReduce library (implemented in java).

2.3 Types (definition MR-Function)

Even though the previous pseudo-code is written in terms of string inputs and outputs, conceptually the map and reduce functions supplied by the user have associated types:

```
map (k1,v1) ! list(k2,v2)
reduce (k2,list(v2)) ! list(v2)
```

I.e., the input keys and values are drawn from a different domain than the output keys and values. Furthermore, the intermediate keys and values are from the same domain as the output keys and values.

An C++ implementation passes strings to and from the user-defined functions and leaves it to the user code to convert between strings and appropriate types.

2.4 Hadoop and HANA

Hadoop can be used in various ways as shown below:

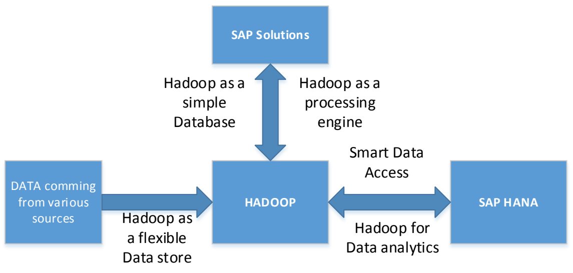

SAP Solutions
Hadoop as a simple Database
Hadoop as a processing engine
Smart Data Access
DATA comming from various sources
HADOOP
SAP HANA
Hadoop as a flexible Data store
Hadoop for Data analytics

Figure 2: Hadoop Use-Case

- Hadoop as a flexible data store by storing data from various sources including SAP and Non-SAP sources like social data, streaming data etc. By keeping all the data in Hadoop, we can get any information we want and can do any type of analysis.
- Hadoop as a simple database for storing and retrieving data in very large data sets. Data can retrieve from Hadoop using Hive or HBase.
- Hadoop as a processing engine by using the MapReduce programming model for many purposes.
- Hadoop for data analytics. Hadoop can be used for mining data held in Hadoop for business intelligence and analytics. Not all data stored in Hadoop is useful, so only useful data will be loaded in HANA

2.5 How to combine Hadoop and HANA

There are several ways to combine Apache Hadoop and SAP HANA:

- SAP Data Services provides a versatile set of data integration tools that can access any type of data (structured or unstructured), load data into any target and work in real time and batch mode. So it is possible to extract, load, parse, integrate, cleanse and match data in Hadoop and MapReduce to be processed by SAP Data Services via Pig scripts automatically.
- SAP has entered reseller agreements with Intel and Hortonworks to resell and support the Intel Distribution Apache Hadoop and the Hortonworks Data Platform with SAP HANA to customers. By reselling the Hortonworks Data Platform, SAP can assure their customers they are deploying an SAP HANA and Hadoop architecture fully supported by SAP. With performance optimizations for Intel hardware as well as encryption and decryption improvements for better security, the integration of the Intel Distribution for Apache Hadoop with the SAP HANA platform provides

enterprises with security and scalability without having to sacrifice performance.

- Since SAP Sybase IQ 15.4, SAP has include a native MapReduce application programming interface (API).

2.6 Differences between MapReduce in Hadoop and MapReduce in Sybase IQ

Sybase IQ allows the user to execute MapReduce-like processing within the database, to support big data applications with the added reliability and real-time capabilities of a database system.

Sybase IQ has an extensibility framework called User-Defined-Functions. UDF's allow the user to write functions in either C++ Java and execute the functions from SQL. UDF functions written in C++ execute within the same process space as the Sybase IQ server. UDF functions written in Java run in a separate process, but still close to the data for performance. There are several different APIs that UDFs can conform to, depending on the structure of data being input to and output from the function: single value, aggregates (multiple values), or tables.

- MapReduce functions can be written in popular programming languages
- MapReduce functions consume and produce data sets in bulk
- MapReduce functions execute as parallel job working on disjoint data sets
- Several levels of nested MapReduce function calls are possible, resulting in multi-level tree execution
- MapReduce processing is fault tolerant, with participating worker units taking over for failed worker units

Along with the strong similarities between MapReduce in Hadoop and MapReduce in Sybase IQ, there are also some important differences:

MapReduce in Hadoop	**Sybase IQ Native MapReduce**
MapReduce functions are invoked within a completely procedural framework.	MapReduce functions are invoked from declarative SQL.
Data store is a distributed file system, is batch oriented, has little to no security protection on data access, and complex joins are cumbersome.	Data store is a column store DBMS which allows ad-hoc queries, complex joins, and enterprise security protection.
Data is schema-less and requires no ETL.	Data requires a schema and at least some ETL.
Fault tolerant, however there are two single points of failure: NameNode and JobTracker	Sybase IQ has high fault tolerance, and any node can take over for another one to complete processing.
Requires a lot of hardware for performance – shared nothing MPP is mandatory.	Requires less hardware footprint for good performance – SMP or shared everything MPP.
Wider variety of programming language support: C++, Java, PHP, etc.	C++ only.

3 Conclusions

SAP HANA is particularly efficient at making real-time decisions and provides support systems for decision-making. It is also very good at managing large amounts of data, although not yet at the same level as Hadoop.

On the other hand, Hadoop enables large amounts of data to be stored arbitrarily in an efficient way. One of its primary strong points is that it allows to find a needle in a (huge and unstructured) haystack, by breaking down a processing job into hundreds or thousands of smaller jobs running in parallel on individual machines.

The MapReduce programming model has been successfully used at Google for many different purposes. We attribute this success to several reasons. Firstly, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing.

Secondly, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning, and many other systems.

In conclusion, linking these both complementary technologies can leverage their individual strengths to build a comprehensive big data solution.

References

[1] David Burdett, Rohit Tripathi: CIO Guide. *SAP Group*, Feb. 2013

[2] Jeffrey Dean, Sanjay Ghemawat: MapReduce - Simplified Data Processing on Large Clusters. *Google Inc.*, 2004

[3] Intel, Installation Guide for Intel® Distribution for Apache Hadoop* software, *Intel Corp.*, Version 2.4.1, June 2013

[4] SAP: Combining SAP Real-Time Data Platform with Hortonworks Data Platform, *SAP Group*, 2013

[5] Courtney Claussen: Sybase IQ In-database MapReduce. *Sybase Inc.*, July 2011

Next Generation Sequencing: From Computational Challenges to Biological Insight

Cornelius Fischer
cfischer@molgen.mpg.de
Maria Metsger
metsger@molgen.mpg.de

Annabell Witzke
witzke@molgen.mpg.de
Michael Boettcher
boettche@molgen.mpg.de

Sascha Sauer
Max Planck Institute for Molecular Genetics
Ihnestr. 63-73, 14195 Berlin, Germany
sauer@molgen.mpg.de

Abstract

Advances in high throughput sequencing technologies allowed studying different biological processes and systems on unprecedented level. However, storage and management of big data, generated with today's sequencing machines as well as subsequent data analysis and biological interpretation remain major challenges in the Next Generation Sequencing (NGS) field. Therefore, access to sufficient computer resources is essential for performing efficient analysis of sequencing data. We implemented diverse applications of NGS technology, including RNA-seq and single-cell RNA-seq methods to analyze cellular processes under metabolic and inflammatory stress conditions. Using the Future SOC Lab infrastructure, we established computational workflows and accomplished primary analysis of recently obtained data. Provided resources enabled us to perform parallel computation and as a result significantly accelerated our research.

1. Project idea

Transcriptome analysis by Next Generation Sequencing, or RNA-sequencing (RNA-seq), has become a routinely used method in many areas of biological research. However, traditional approaches of sample preparation for RNA-sequencing utilize RNA, isolated from thousands of cells, covering up heterogeneity of used tissue or cell population [1]. Rapidly developing technology of single-cell RNA-sequencing allows to overcome these limitations and to study regulation of gene expression in a more accurate way [2]. In our study we analyzed basic gene regulation processes underlying inflammatory and metabolic stress response in macrophages. Heterogeneous nature of macrophages has been recognized for a long time [3]. Therefore the macrophage model system allows to investigate general mechanisms of cellular response to specific environmental stimuli and to reveal heterogeneity in macrophage cell lineage using single-cell sequencing.

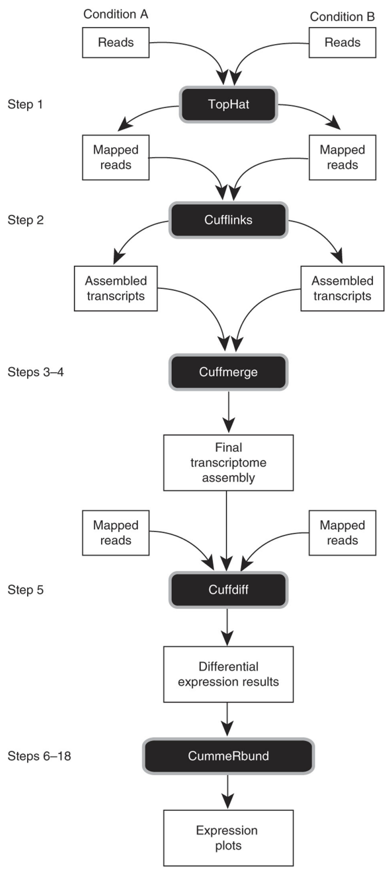


Figure 1. Overview of computational workflow for sequencing data analysis.

2. Used Future SOC Lab Resources

We used Hewlett Packard DL980 G7 server that was equipped with eight 8-core Intel Xeon X 6550 processors and 128 GB of RAM running Ubuntu Server 12.04 LTS. This system was perfectly suitable for our experiments.

3. Methods and tools

Raw sequencing reads were pre-processed using Trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic), and consequently mapped to human genome reference (Feb. 2009 ; hg19, GRCh37) using Tuxedo suite [4] (Bowtie 2, Tophat) and STAR [5]. Then FPKM values were obtained using Cufflinks, and differential expression analysis was performed using Cuffdiff. We applied the same analysis workflow (Figure 1) for each of 288 single-cell libraries and 9 bulk RNA libraries.

4. Findings

We used the Future SOC Lab resources for analysis of RNA-sequencing data, obtained from bulk RNA samples and single cells (Figure 2a). Application of parallel computation using multi-core architecture initially enabled us to test various analysis strategies and subsequently calculation time for time consuming processes such as read alignment and differential expression analysis.

Primary validation of our data showed a uniform distribution of sequencing reads across samples (Figure 2b) and good correlation between single-cell and bulk data (Figure 2c). To estimate sequencing depth sufficient for our single-cell samples, we generated saturation plots by random subsampling of sequencing reads from each sample library for each treatment condition (Figure 2d). Initial control of our data showed that the quality is sufficient for further investigations including analysis of gene regulatory networks as well as assessment of gene expression noise and heterogeneity of cell population.

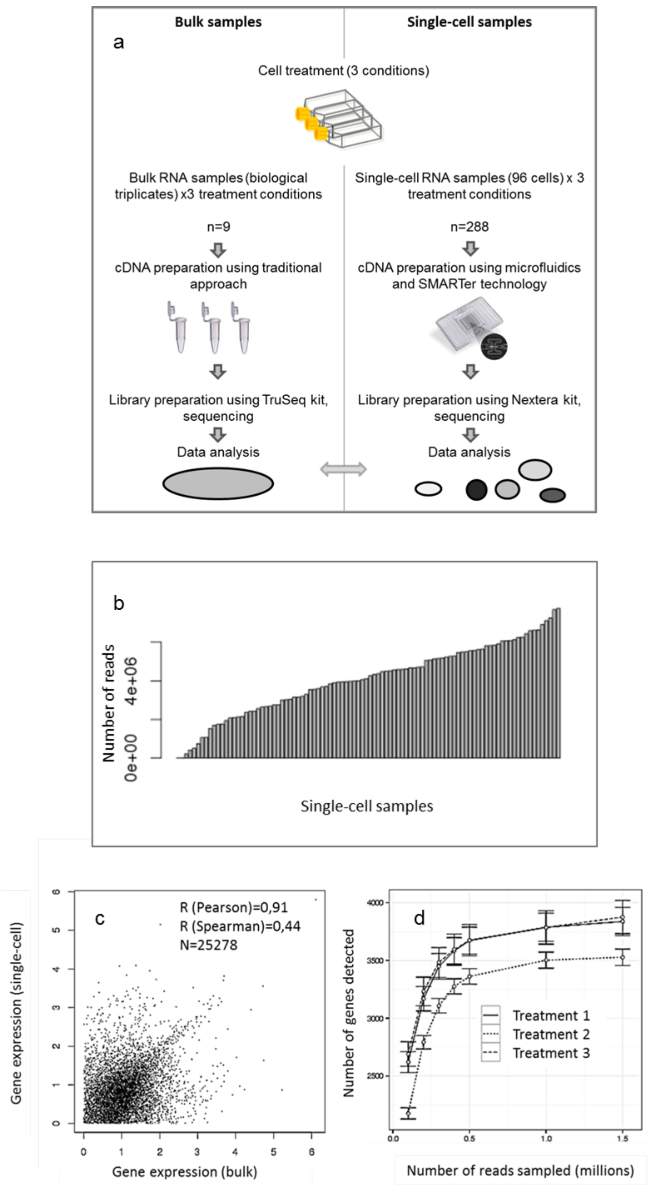


Figure 2. (a) Schematic of the experimental strategy for RNA-seq and single-cell RNA-seq sample processing. (b) Number of reads detected in single-cell samples for a representative treatment condition (c) Correlation between single-cell RNA-seq and bulk RNA-seq samples for a representative treatment condition. (d) Saturation plot, generated by randomly selecting a subset of reads from each sample library and then using the same alignment pipeline to call genes.

5. Next steps

Additional exploration of our sequencing data will include systematic regulatory network analysis and testing different statistical models of transcriptional noise. Additionally, recently developed technological strategies will allow us to analyze much more single-cell samples in the nearest future, which provides a great potential in terms of accuracy and sensitivity of our studies. For this purpose further participation in the HPI Future SOC Lab project would be essential to apply established analysis pipelines for newly generated sequencing data.

References

[1] Shalek, Alex K. et.al: Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature*, 498: 236 - 240, June 2013

[2] Wu, Angela R. et.al: Quantitative assessment of single-cell RNA-sequencing methods. *Nature Methods*, 11: 41 - 46, 2014

[3] Gordon, S. et. al: Monocyte and macrophage heterogeneity. *Nature Reviews Immunology*, 5 : 953 - 964 , December 2005

[4] Trapnell, C. et. al: Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols*, 7 : 562 - 578, 2012

[5] Dobin, A. et. al: STAR: ultrafast universal RNA-seq aligner.*Bioinformatics*, 5 : 953 - 964 , October 2012

Project Report: Statistical Analysis of Cloud Storage

Josef Spillner and Johannes Müller
Technische Universität Dresden
Fakultät Informatik
Nöthnitzer Str. 46
josef.spillner@tu-dresden.de, s9186231@mail.zih.tu-dresden.de

Abstract

Within this report we summarise our experiments on multiplexed access to distributed storage services which were conducted between January and March 2014. The work has contributed to a refined design of storage controllers and connector modules. Our open source implementations NubiSave (controller and splitter) and CloudFusion (connector) benefit greatly from the confirmed and refuted assumptions we previously had about small file uploads, multithreading and overall data availability determination.

1 Background

Distributed file storage, formerly considered a high-effort enterprise software feature, is increasingly propagating to consumer-oriented desktops and devices. Cloud storage services in combination with multi-cloud integration frameworks are the main enablers of this trend. Among the distribution models, dispersed file storage is a particularly interesting one due to its properties effectivity (minimal redundancy) and security (assuming the non-cooperation of providers) [2].

The Cloud Storage Lab at Technische Universität Dresden[1] has been set up in late 2013 to perform research with practical relevance on this subject. Emerging from a young investigator group FlexCloud[2] since 2011, the Cloud Storage Lab therefore offers a number of useful software, especially storage controllers and filesystems, and tools for experiments and visualisation of dispersed data.

A strong focus of the work is on user-controlled storage systems which juggle the best storage service combinations, where best is defined as weighted sum over all non-functional properties [3]. In practice, this is not a trivial technique due to highly heterogeneous storage services with varying capacity, price, availability, reputation and throughput. In addition, there are certain client-side constraints including storage policies [5] and complex software-defined storage flows with encryption and further modifications applied to the data in transit [6].

In mid-2013, peaCS – a testsuite for the performance and efficiency analysis of Cloud Storage – was created to gain more insight about the influence of certain non-functional properties for any given storage controller configuration [4]. While some insight was gained, we immediately intended to perform more analysis tasks at a larger scale. This project, carried out on FutureSOC Lab infrastructure provided by the Hasso-Plattner Institute (HPI), has realised this intention and uncovered a more precise handling in future storage controllers.

2 System Model

We consider a storage controller C which multiplexes file reads and writes to n storage targets $T_i (0 \leq i \leq n)$. Depending on the file splitting and distribution scheme, $n = k+m$ with k significant and m redundant fragments. Targets can be services with a well-defined interface (e.g. WebDAV, CIFS), indirect synchronised interfaces (e.g. Dropbox folders) or just local directories. For our experiments, we concentrated on services with multiple non-functional property vectors P_i. Important properties which are evaluated at runtime are (1) assumed availability, (2) price per amount of data, and (3) capacity. Fig. 1 explains our setup.

Within the system, services are represented as directories through transport modules with mount and unmount semantics. Often, these are file systems in userspace. Service properties are described in INI files which are evaluated at runtime. On top of those, the storage controller NubiSave - implemented in Java with a wrapped splitter core written in C - is also acting as a file system, with a corresponding INI file for its configuration. NubiSave and our reference transport module, CloudFusion, can be reconfigured at runtime, whereas most conventional transport modules require a remount in order to let a reconfiguration take effect.

[1]Cloud Storage Lab: `http://lab.nubisave.org/`
[2]FlexCloud: `http://flexcloud.eu/`

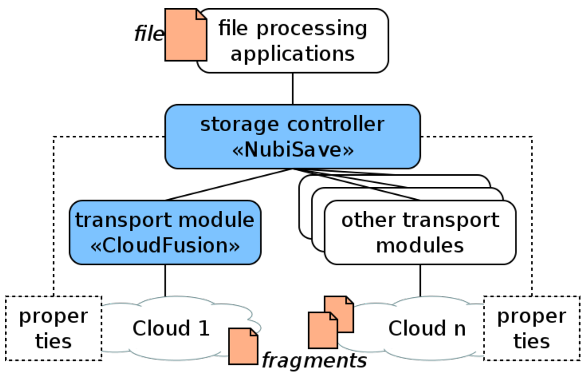

file
file processing applications
storage controller «NubiSave»
transport module «CloudFusion»
other transport modules
properties
Cloud 1
fragments
Cloud n
properties

Figure 1. Simplified architecture of a cloud storage controller

3 Procedure Description

The potential for improvements in how today's storage controllers split and distribute data was divided into the following key ideas:

1. As pointed out by a recent bachelor thesis on storage at Technische Universität Darmstadt [1], several providers including Ubuntu One and Box incorporate a rate-limited upload service. Details about the limitation are not documented but can be found out statistically. Together with size-limited services, this leads to a controller design which supports both chunking (1 file into x chunks) and bulking/batching (y files into 1 virtual file) orthogonal to the splitting (1 file into n fragments). This idea needed to be thoroughly tested with larger amounts of data.

2. Performance metrics also depend on the degree of parallelisation. Our idea was to include multithreading support into the transport/connector modules for parallel file uploads and downloads, and into the controller for parallel splitting and distribution. For fine-grained testing of the effects with many threads, we needed a multi-core machine.

3. The best combination of storage providers becomes a multi-dimensional or multi-objective optimisation problem. First of all, if the number of targets (n) is low, then just adding a few percent of redundancy only wastes space without contributing to higher availability of data. If the number of targets is high, then calculating the average availability takes a long time due to $O(2^n)$ complexity especially with large k. And if targets have interdependencies, e.g. Dropbox which actually uses Amazon S3, the perceived availability is higher than the actual one. Therefore, we proposed an improvement in the algorithm used to calculate the overall availability for a set of heterogeneous storage nodes and needed to confirm its correctness with combinatorial completeness which required a lot of compute resources.

4 Experimental Setting and Results

The specifications of the FutureSOC Lab machine which hosted the experiment clients in Potsdam are as follows:

1. Machine: Hewlett Packard DL980 G7 - 1
2. CPU: 8 x Xeon (Nehalem EX) X7560; 128 processor cores @ 2.27 GHz
3. RAM: 2048 GB
4. HDD: 2 x 146 GB; RAID-1; 75 GB available
5. HBA: Emulex 4Gb Fibre Channel
6. OS: Linux dl980 3.5.0-45-generic #68 precise1-Ubuntu SMP Wed Dec 4 16:18:46 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
7. Software: NubiSave storage controller for splitting and distributing the data, S3FS and CloudFusion FUSE filesystems as transports to the actual storage targets

As storage backends, we have used the commercial consumer-focused services offered by Deutsche Telekom, Dropbox, Google, Sugarsync and Amazon S3 as well as a WebDAV server under our control at Technische Universität Dresden. Both servers are connected through the DFN XWiN backbone. The WebDAV server's specifications are as follows:

1. Machine: VM with vSphere 5.5 on Fujitsu Primergy RX300S6
2. CPU: 1 x Xeon E5620 (out of 2 x); 4 processor cores @ 2.40 GHz
3. RAM: 24 GB (out of 48 GB)
4. HDD: 95 GB; provided by NAS via Fibre Channel; 88 GB available
5. OS: Linux cloudstorage-exp 3.2.0-4-amd64 #1 SMP Debian 3.2.51-1 x86_64 GNU/Linux
6. Software: Apache httpd 2.2.22 with WebDAV extension

After preparing the experiments locally, we reserved a slot on the FutureSOC machine initially from February 5 to February 14. While running them, we were spotting some errors which were quickly corrected. While they did not gravely affect the results, we decided to go for a second reservation which we got from March 5 to March 19.

4.1 Transport Analysis

First, we optimised the CloudFusion transport module to include multithreading and batching. In addition, there is a caching layer with a user-configurable hard limit, set by us to 7 GB. In a test with Dropbox, the benchmark tool ql-fstest locally wrote 237531 MiB at 10.9524 MiBps and read 728116 MiB at 30.3617 MiBps with artifically generated directories and files.
For the real throughput test, we backed up a complete home directory which is a more realistic scenario. First, small files were manually combined in a larger tar archives and then split by NubiSave, resulting in 33 files of 66.37 MB and 1 trailing file of 33.96 MB, totalling at 2224 MB. For the subsequent uploads, we achieved the following write performance values: Google Storage (with s3fuse) = 0.65 Mbps, Amazon S3 (with s3fuse) = 0.59 Mbps, T-Online (with davfs2) = 0.59 Mbps, Dropbox (with Cloudfusion, 3 threads) = 2.06 Mbps, and Sugarsync (with Cloudfusion) = 0.51 Mbps. We did not measure download rates due to the high seek times with tar archives, which would be unproblematic with zip-encoded files.
Instead, we then automated the batching of small files into larger ones for the CloudFusion transport. For a series of 100 kB files, we measured an improved upload throughput of 1.68 Mbps vs. 0.137 Mbps without batching. This indicates that transport modules should enable batching in general. To confirm the claim, we sequentially wrote 10 byte files with random content. Interestingly, s3fs takes 3 seconds per file, davfs2 needs 2 seconds, and s3fuse 1.6 seconds. CloudFusion, with its highly asynchronous implementation, surpasses all of these with only 0.02 seconds.
To compare the provider performance when using CloudFusion, we uploaded small samples and measured the time. Fig. 2 compares the file upload performance of the commercial providers. The x-axis specifies the file size in kB whereas the y-axis specifies the throughput in Mbps.

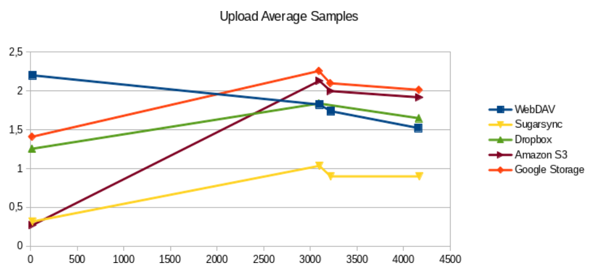


Figure 2. Upload statistics for five different provider and transport module pairs

4.2 Controller Analysis

The availability calculation was also successfully performed. Our proposal is an improvement in the algorithm used to calculate availability for a set of heterogeneous storage nodes. It decreases the worst case and average complexity by two. This is sufficient to calculate availabilities with high speed for configurations with less than 40 nodes, which enables the GUI of Nubisave to respond fluently to changes in the configuration.

Multithreading was also explored on the controller level to parallelise splitting and distribution of data. We invoked multiple throughput measurements on a $n = 100, m = 39$ configuration, which means a storage overhead of $o = m/k = 0.64$. 300 files of 8 MiB each were written, with 8 MiB being the size of the internal cache. All files were read and written in memory (RAM disk) to prevent deviation associated with disk seek times.

The mean throughput of parallel fragment encoding and splitting is summarised in the Table 1 for the jErasure library and in Table 2 for the JigDFS library. One can see that JigDFS performs much worse in a single thread but scales well with multiple threads. Therefore, while currently the splitter library is hardcoded into the controller, the performance characteristics on modern multi-core machines which host the controller (i.e. notebooks and PCs for single users and gateway servers for multi-user installations) can benefit from the Splitter-NG abstraction framework developed in the Cloud Storage Lab. Please note that the measurement with 60 threads was performed with a much larger cache size in order to show that there are more parameters than just the thread count which influence the throughput.

Table 1. Splitter parallelisation: Jerasure

# threads	Mbps
1	5.11154370702
2	8.45531546231
4	8.37964079935
8	8.43373030491

Table 2. Splitter parallelisation: JigDFS

# threads	Mbps
1	1.38163120315
2	2.76042033263
3	4.11243626312
4	5.4495440924
5	6.75469117301
6	8.03208464845
7	8.30822736515
8	8.3207848927
9	8.28686515957
60*	35.7836853506

5 Conclusions

The experiments helped us better understand the three topics of availability calculation, multithreading for file uploads, and batched fragment uploads. Detailed results will be documented in combination with more user-friendly storage controller configuration in the upcoming diploma thesis of Johannes Müller with the title »Autonomic Calibration of Cloud Storage Controllers«.

In parallel to the experiments, we have investigated the promising path towards calculation over split, encrypted and dispersed data. It becomes obvious that more application-level control is needed over the splitting algorithms to use in order to preserve valuable structures in the data formats. For the future, we envision more experiments which compare established algorithms for storage (erasure coding, secret sharing) with custom ones for both storage and processing. Therefore, we will reapply for another period of FutureSOC Lab projects.

References

[1] L. Diedrich. Combining Cloud Storage Systems. Bachelor thesis at Technische Universität Darmstadt, September 2013.

[2] D. Slamanig and C. Hanser. On cloud storage and the cloud of clouds approach. In *The 7th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 649–655, December 2012. London, United Kingdom.

[3] J. Spillner, J. Müller, and A. Schill. Creating Optimal Cloud Storage Systems. *Future Generation Computer Systems*, 29(4):1062–1072, June 2013. DOI: http://dx.doi.org/10.1016/j.future.2012.06.004.

[4] J. Spillner, M. Quellmalz, M. Friedrich, and A. Schill. peaCS - Performance and Efficiency Analysis for Cloud Storage. In *ESOCC Workshop on Cloud Storage Optimization (CLOUSO)*, volume 393 of *CCIS*, pages 47–58, September 2013. Málaga, Spain.

[5] J. Spillner and A. Schill. Flexible Data Distribution Policy Language and Gateway Architecture. In *1st Latin American Conference on Cloud Computing and Communications (LatinCloud)*, pages 1–6, November 2012. Porto Alegre, Brazil.

[6] J. Spillner and A. Schill. Orchestration of Distributed Storage Targets through Storage Flows. In *5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, December 2013. Bristol, United Kingdom.

Study of Appropriate Algorithm Classes for State-Of-The-Art Hybrid Hardware Architectures

Peter Tröger
Operating Systems and Middleware
Hasso Plattner Institute
Prof.-Dr.-Helmert.-Str. 2-3,
14482 Potsdam
Peter.Troeger@hpi.uni-potsdam.de

Frank Feinbube
Operating Systems and Middleware
Hasso Plattner Institute
Prof.-Dr.-Helmert.-Str. 2-3,
14482 Potsdam
Frank.Feinbube@hpi.uni-potsdam.de

Abstract

Trends in hardware developments emphasize the every-increasing importance of hybrid computing for future computer architectures and software systems. Competitive applications need to leverage the performance opportunities provided by emerging generations of accelerator technology. With the introduction of its K20 architecture, NVIDIA takes a big step towards a wider applicability of GPU computing by supporting new concepts like dynamic programming, on-device grid management, and direct data exchange. Intel's novel Xeon Phi accelerator, being a stand-alone PCI-Express card like GPUs, but still being x86 compatible, is approaching the field of hybrid computing from the general purpose side.

The purpose of this study is to survey suitable algorithms for the K20 architecture and the Xeon Phi accelerators.

1 Project Overview

1.1 Hypothesis

There is a number of algorithms / classes of algorithms / algorithmic problems that:

a. Did not benefit from GPU-based Hybrid Computing so far.
b. Benefits from NVIDIAs K20 architecture. (Runs faster or scales better.)
c. Or/and benefits from Xeon Phi architecture. (Runs faster or scales better.)

1.2 Progression

- 1 Month Research, Survey (Investigating the State of the Art in the field)
- 1-2 Month Planning, Designing, Prototyping (of Experiments / Survey Sheet)
- 2-4 Month Elaborated Survey
  - Categorization, Description
  - Prototypic Implementation, Evaluation.

1.3 Results

- Detailed description and reasoning of Survey and the algorithm characteristics looked into
- Identification and description of all considered algorithm classes
- Identification, description and reasoning for algorithm classes that are considered (very) suitable for K20 / Xeon Phi
- Discussion of the hypotheses
  - Literature Survey / Experiments indicating that 1.a hold
  - Survey, Reasoning, Experiments (Prototypes and Measurements) demonstrating that 1.b / 1.c holds

1.4 Publications

M. Linkhorst, F. Feinbube, and A. Polze; “Concurrent Tasks with Dynamic Parallelism on NVIDIA’s GK110 Architecture”; Master Thesis; Operating Systems and Middleware Group; Hasso Plattner Institute; University of Potsdam; 01/2014 [1]

M. Plauth, F. Feinbube, P. Tröger, and A. Polze; “Audio Signal Processing on GPU Compute Devices” ; Master Thesis; Operating Systems and Middleware Group; Hasso Plattner Institute; University of Potsdam; Potsdam; 03/2014 [2]

2 Concurrent Tasks with Dynamic Parallelism on NVIDIA's GK110 Architecture

Project Publication; M. Linkhorst, F. Feinbube, and A. Polze; Master Thesis at the Hasso Plattner Institute, Potsdam, 01/2014

One of the results of this project in the HPI Future SOC Lab is the master thesis of M. Linkhorst which was supervised by the HPI Operating Systems and Middleware group. The thesis explores the applicability of the Dynamic Parallelism Concept of OpenCL 2.0 using the example of NVIDIAs K20 GPU Compute Device.

A list of criteria for algorithms is presented that allows to assess opportunities and potential problems of this novel technology. Based on these criteria, a survey of the Berkeley dwarves and the algorithms of the Parboil hybrid benchmark was created. We found that due to its high rating, the class of Divide-And-Conquer algorithms is especially suited for an elaborate study of the applicability of Dynamic Parallelism. The Breadth-first Search (BFS) algorithm and the All-pairs Shortest Path (APSP) algorithm are selected for further investigations.

The state-of-the-art BFS algorithms are assessed and compared; a number of benchmarks are executed on the Future SOC hardware. That allowed us to create and optimize a new BFS algorithm that uses Dynamic Parallelism to allow for concurrent usage of the resources of GPU Compute Devices. Employing the Stream concept to the BFS algorithm enabled us to create an APSP implementation with significantly better execution performance than APSP implementations based on the best state-of-the-art BFS algorithms. This demonstrates the great advantages that Dynamic Parallelism provides to task-based algorithms. In contrast to existing solutions, the scaling behavior of our approach makes it very attractive for future generations of accelerators.

Furthermore, we show that Dynamic Parallelism can be used to provide a concurrent graph library directly on the GPU.

This work demonstrates the applicability of Dynamic Parallelism for demanding algorithms and illustrates ways in which programs can be restructured in order to benefit from it.

3 Audio Signal Processing on GPU Compute Devices

Project Publication; M. Plauth, F. Feinbube, P. Tröger, and A. Polze; Master Thesis at the Hasso Plattner Institute, Potsdam, 03/2014

One of the results of this project in the HPI Future SOC Lab is the master thesis of M. Plauth which was supervised by the HPI Operating Systems and Middleware group. The thesis explores the applicability of state-of-the-art accelerator technology for real-time audio signal processing. Studies were conducted with NVIDIAs K20 GPU Compute Devices and Intels Xeon Phi Accelerator Board.

The thesis evaluated the feasibility of using GK110-based GPU compute devices for the application of the FastICA algorithm in a live audio signal processing scenario. Furthermore, the benefits of leveraging GPU hardware for a batch processing implementation of the FastICA algorithm were investigated as well.

For the tested range between 2 and 8 signals, the batch processing mode achieved a median speedup of factor 18.63 and 13.66 using single precision and double precision, respectively. The speedup was determined in comparison to a parallel CPU-based implementation using the MKL and IPP libraries. For the live processing mode of operation, maximum execution times between 19.33 and 130.32 milliseconds were determined for the same number of signals. With one chunk representing 1365 milliseconds of audio, all tested chunks were processed within the deadline of 170 milliseconds.

Even though previous attempts of using GPU compute devices for the acceleration of FastICA (and other ICA algorithms) have been published, this work covered two major aspects which have not been covered by preexisting work. First of all, the capabilities of the compute-centered Kepler GK110 architecture were evaluated. From the point of view at the time of writing this thesis, even the latest of the preexistent publications uses a comparatively outdated GT200 GPU. Secondly, prior work mostly dealt with processing Electroencephalogram (EEG) data, which differs a lot from the characteristics of audio data.

We demonstrated the feasibility of GPU Compute Devices for live processing of complex audio signal processing tasks such as Blind Signal Separation using FastICA.

References

[1] M. Linkhorst, F. Feinbube, and A. Polze; "Concurrent Tasks with Dynamic Parallelism on NVIDIA's GK110 Architecture"; Master Thesis; Operating Systems and Middleware Group; Hasso Plattner Institute; University of Potsdam; Potsdam; 01/2014

[2] M. Plauth, F. Feinbube, P. Tröger, and A. Polze; "Audio Signal Processing on GPU Compute Devices" ; Master Thesis; Operating Systems and Middleware Group; Hasso Plattner Institute; University of Potsdam; Potsdam; 03/2014.

[3] Our Pattern Language (OPL), http://parlab.eecs.berkeley.edu/wiki/patterns/patterns; 03/2014

[4] The Landscape of Parallel Computing Research: A View from Berkeley, http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html; 03/2014

[5] Dwarf Mine, http://view.eecs.berkeley.edu/wiki/Dwarf_Mine; 03/2014

[6] Intel Xeon Phi Coprocessor x100 Product Family: Specification Update

[7] Whitepaper: NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110

[8] CUDA Documentation, http://docs.nvidia.com/cuda/index.html; 03/2014

A Case Study of Image Processing Algorithm Optimization on Accelerators

Ahmad Kiswani
Technion
Israel
sahmad@mail.technion.ac.il

Uri Verner
Technion
Israel
uriv@cs.technion.ac.il

Abstract

This project investigates design and optimization techniques for image processing algorithms that execute on compute accelerators. As a case study, we focused on the Image Filtering (convolution) algorithm on NVIDIA GPUs; this algorithm represents a family of image processing algorithms and can be effectively analyzed.

Our implementation was carefully designed and optimized to achieve maximum hardware performance. Using profiling tools and theoretical analysis, we identified a set of bottlenecks in the compute and memory systems, and designed implementations that stress out each bottleneck.

1 Introduction

The amount of collected data that needs to be processed is increasing rapidly across many fields. An important part of applications that process such data apply image processing algorithms on streams of images that arrive from external sources. Image processing algorithms are used in a variety of fields, including computer vision, production control, medical equipment, gaming, security, and more. In many applications, such algorithms need to operate at real-time speeds, processing data at speeds of up to gigabytes-per-second on a server. For example, in a wafer-production inspection system (wafer metrology), silicon wafers are inspected at fabrication time by using high-resolution images. The computational process needs to analyze these images and locate defects in real-time. Due to the high volumes of data, the information cannot be stored and processed offline, so the inspection must keep pace with the production line.

This project investigates code design and optimization techniques for the implementation of image processing algorithms on GPUs. We achieved predictable performance that matches the maximum performance based on device specification. This report presents several implementation approaches, provides a profound performance analysis for each approach, describes optimization techniques that remove unnecessary computations and stalls, and shows that the achieved performance matches the theoretical one.

2 Image Filtering (Convolution)

The image filtering algorithm, illustrated in Figure 1, receives a 2D image and computes a modified image that has the same size. Each output pixel is computed as the sum-product between a predefined 2D filter matrix and a patch of the input image that surrounds the corresponding input pixel. In images where pixel values are represented by a vector of values, such as in the RGB (red/green/blue) format, the sum-product is computed for each attribute independently. For simplicity, in this report we assume that the image is formatted in grayscale 1-byte per pixel encoding.

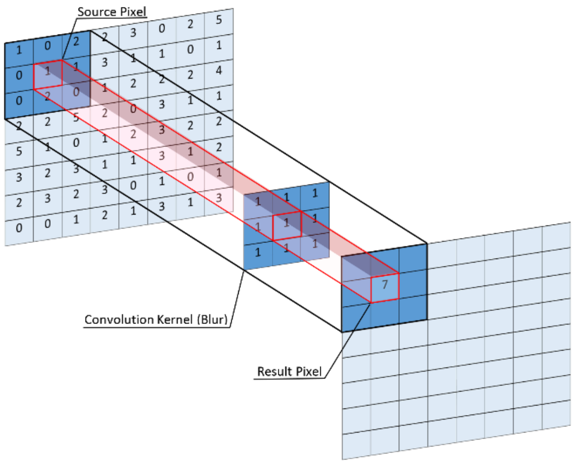


Figure 1 Image filtering

3 GPU Architecture

NVIDIA GPUs are composed of a number of streaming multiprocessors (SMs) that execute on independent work units. Commonly, the GPU is used as an accelerator for parts of an application that runs on the CPU. To execute a task on the GPU, the user specifies a kernel function and defines a grid of work units. Each work unit is executed by a collection of threads, named thread block. Each thread executes the kernel code in a separate context, possibly using different input data. The threads inside a thread block can cooperate to execute the task by exchanging data and synchronizing execution. Upon kernel launch, the GPU scheduler starts to distribute the thread blocks between SMs.

Each SM internally schedules the threads in a thread block at *warp* granularity, which causes these threads to execute in lock-step, and also allows the system to execute more efficiently, for example by coalescing data transfers issued by the threads. So far, the warp size on NVIDIA architectures was 32 threads. Code divergence inside a warp is handled by executing all the execution paths and masking out the threads that do not take the execution path.

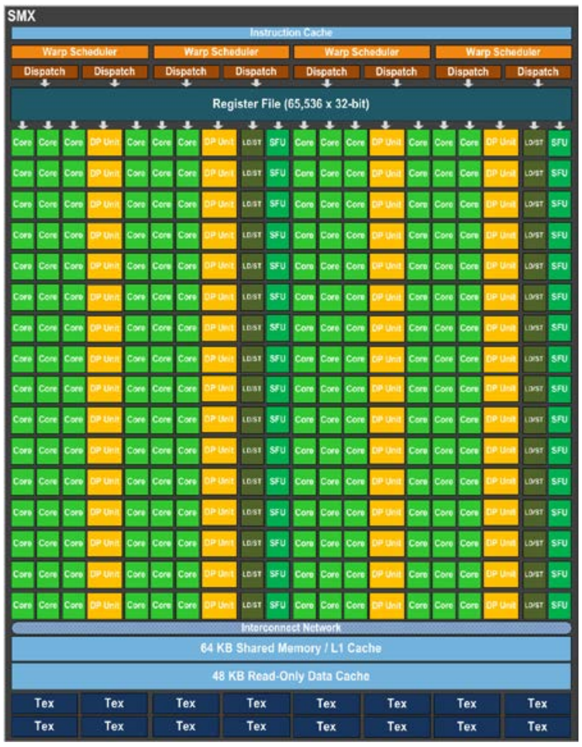

SMX
Instruction Cache
Warp Scheduler
Dispatch
Register File (65,536 x 32-bit)
Core
DP Unit
LD/ST
SFU
Interconnect Network
64 KB Shared Memory / L1 Cache
48 KB Read-Only Data Cache
Tex

Figure 2 GK110 SMX

The Kepler GK110 SMX (enhanced SM) illustrated in Figure 2 features four warp schedulers, each capable of dispatching two independent instructions per warp. The SMX instructions throughput is listed in Table 1. As seen in the table, the throughput of float multiply-add operations is six times higher than integer multiply-add operations. We will use this observation when choosing the implementation.

The GK110 SMX includes two types of memory:

1) 64KB on-chip memory that can be split between L1 cache and shared memory in one of 3 configurations, 16 KB/48 KB, 32 KB/32 KB and 48 KB/16 KB.
2) 48 KB read-only memory that is used as cache for texture memory and read-only arrays.

Unlike the read-only memory, which is managed automatically, the shared memory is managed by the programmer. In Kepler, it has 32 banks with 8-byte wide ports. This gives the shared memory a maximum bandwidth of $256[\frac{Bytes}{Cycle}]$ per SMX.

The K20 card used in our tests includes 13 SMXs that share a 1536 KB L2 cache.

Operation Type	Throughput
32-bit floating point multiply, multiply-add	192
32-bit integer multiply, multiply-add	32
32-bit integer shift	64
32-bit bitwise and, or	32
Type conversions from 8/16-bit integer to 32-bit types	128
All other type conversions	32
Warp shuffle	32

Table 1 Throughput of native arithmetic instructions per clock cycle per multiprocessor for GK110 SMX

4 Code Design and Optimizations

The serial code implementation of the image filtering algorithm is shown in Figure 3. The code iterates over the pixels in the input image, and for each pixel iteratively computes the sum-product of the surrounding patch and the filter. In the basic GPU implementation of this algorithm, the outer loops are replaced by an array of threads that compute a single output value each. Since there is no data dependency between the computations of different pixels, the threads can be executed in parallel.

```
for(int y = 0; y < h; y++)
for(int x = 0; x < w; x++)
{
    float v = 0.0;
    for(int fY = 0; fY < fH; fY++)
    for(int fX = 0; fX < fW; fX++)
    {
        int imgX = x-fW/2+fX;
        int imgY = y–fH/2+fY;
        v += img[imgY][imgX];
        res[y][x] = min(max(int(v),0),255);
    }
}
```

Figure 3 Serial code for image filtering

Kernel	Execution Time [msec]	Run Time without Overhead [msec]	Achieved Throughput [Gpix/sec]	Theoretical Throughput [Gpix/sec]
Texture cache	0.0224	0.0152	40.935	83.794[1]
Read-only cache	0.0221	0.0149	41.094	46.644
Shared memory	0.0220	0.0146	41.909	39.41
Warp Shuffle	0.0283	0.0213	32.276	34.732
Float	0.0400	0.0327	18.784	18.664
Texture cache (100 images)	0.02001 (per image)	0.01994 (per image)	34.476	79.981[1]
Read-only cache (100 images)	0.0248 (per image)	0.0247 (per image)	31.550	38.566
Shared memory (100 images)	0.0159 (per image)	0.0158 (per image)	38.563	39.213

Table 2 performance comparison of several kernels using a 3x3 filter

Although functionally correct, this implementation does not achieve good performance on the GPU, as its design does not utilize the GPU resources efficiently. In subsection 4.1 we describe approaches for achieving maximum performance and code optimizations that apply to all or several approaches. Then, we evaluate these approaches in subsection 4.2 and describe their performance limiting factors.

4.1 Design approaches

The kernel execution time depends on the time it takes to bring the data to the computational units, and the computation time. As described in Section 3, the GPU has several memory types that have different operation modes and throughput. For example, the input image can be read from the global or texture memory. The computation time depends on the data type used for computations (byte or float), stalls due to data dependency, control code overhead, etc.

We implemented our programs using CUDA C, which is an extension to the C language for programming GPU kernels. To better understand the performance results, we examined the low-level PTX code that is generated by the compiler after applying the optimizations. From examining the PTX code, we were able to remove redundant commands by using loop-unrolling and computing address offsets at compile time. We also inserted inline PTX code in the CUDA code, where we found that the compiler inserts redundant commands. Other optimizations we applied include assigning independent work to each thread, tuning the thread block size, using memory coalescing and more. The Kepler Tuning Guide [1] provides more information about optimization techniques.

One of the implementation decisions is whether to perform the computations in byte or float units. The input and output images use the byte formatting, so performing the computations in bytes seems natural and does not require type conversions. However, current GPUs are optimized for working with floats, and can perform 6 times more multiply-add operations on float than on integer values. In this decision, there is a tradeoff between computing the sum-product and the type conversions. The texture memory has a mode of operation that implicitly converts byte image values to floats, and may provide better performance than performing the type conversions explicitly.

Another important decision is how to bring image data to the compute units efficiently. The challenge here is not only to bring the pixel values for the first time, but also to store and reuse them by multiple threads. The input image is stored in the off-chip memory and can be read using global-memory accesses or texture fetches. The options for data reuse are to use the automatic read-only and texture caches, manually manage a cache in shared memory, and use special intra-warp communication operations.

To examine the performance of the various implementation options, we wrote a code generator that creates optimized kernels given a set of implementation parameters. In addition to the data read and compute options specified above, several other parameters were examined, including filter size, thread-block size, and number of pixels computed per thread.

4.2 Evaluation

We performed our tests on a $2000 \times 300[pixels]$ PGM images (a grayscale image with 1 byte per pixel), and a floating point convolution filter ranging from $3x3[pixels]$ to $13x13[pixels]$, these data sets are taken from a work environment and represent a real world input set. With each test, we calculated the kernel's theoretical throughput based on the number of arithmetic operations in the PTX code (ignoring memory bandwidth limitations). To calculate the

[1] Theoretical throughput for texture cache can be misleading due the implicit conversions it performs.

achieved throughput, we measured the kernel’s execution time and subtracted the kernel launch overhead (approximately $7[\mu sec]$) to get the actual kernel runtime. Table 2 summarizes our experiment results.

Since we needed to convert the source image to float, texture cache seemed the logical way to access the image pixels, texture cache is optimized for 2D access patterns and can return the image pixels as floats without explicit conversions. However, with this kernel were able to achieve only half the theoretical throughput. NVIDIA visual profiler showed that texture stalls are the main performance limiter (Figure 4).

An alternative method is to use the same read-only cache directly with the *__restrict__* qualifier. However, since the data now does not go through the texture unit, we needed explicit byte to float convert operations in the code. Unfortunately, the overhead of the convert operations significantly reduces the theoretical performance. In this implementation, the theoretical throughput matched the measured throughput. From this observation we conclude that the implementation is compute bound.

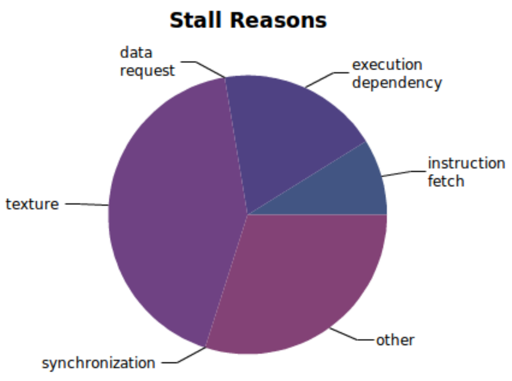


Figure 4 Stall Reasons for Texture kernel

The next step after achieving maximum hardware utilization is to try improving performance by identifying bottlenecks, Figure 5 shows that only 20% of the kernel’s computational time goes to floating point multiply-add instructions, and more than 60% goes to byte extraction (shared memory is accessed using 8-bytes word to utilize the full bandwidth) and conversion.

Since each pixel is accessed by multiple threads, we tried to convert each pixel to float once, and store the result for later use in shared memory. This approach did not prove to be effective as it quadrupled memory traffic and turned the kernel to memory bound.

Another approach was to use warp shuffle instructions to share data between threads in a warp, but the low warp instruction throughput resulted in a lower performance.

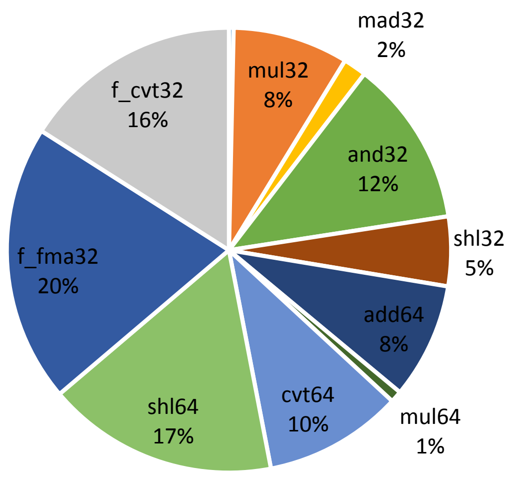


Figure 5 percentage of time spent on each instruction type (Shared Memory Kernel)

Another major bottleneck to performance is kernel launch overhead, the $7[\mu sec]$ overhead is about 30% of the total execution time, since this is out of our control, we resorted to performing convolution on multiple images within a kernel to reduce the overhead per image.

While the read-only cache kernel saw a performance decrease when we introduced the changes, both texture cache and shared memory kernels had a performance increase (Although only the shared memory version was able to mask the $7[\mu sec]$ overhead). Profiling tools showed a higher cache miss rate per image than before, this suggests an inefficient caching mechanism when accessing large 2D data sets.

Overall, the implementation that resulted in shortest execution time in our experiments was the one that used shared memory to cache pixel values read from the global memory.

5 Conclusions

We investigated design and optimization techniques for implementing image processing algorithms, taking image filtering as a case study. This algorithm was chosen because it has characteristic that also appear in other image processing algorithms, such as its memory access pattern and working on 2D byte array input.

We have shown that the convolution algorithm is computationally bound as were able to reach the card’s theoretical arithmetic throughput. The three kernels that we implemented (texture, cache and shared memory) yielded similar results, which further supports our conclusion.

As GPUs are built primarily with computer gaming in mind, they have a high floating point instruction throughput. Therefore, even with multiply-add instructions being the primarily operation needed to calculate convolution, it’s not the bottleneck of the operation as it only mount to about 20% of the computational time.

33% of the total execution time is spent on kernel launch, calculating multiple images per kernel launch can significantly reduce execution time when shared memory is used. The same increase cannot be expected when using read-only memory (whether in cache or texture mode) due to inefficient caching of large data sets.

60% of computation time is spent on byte extraction and conversion, the low throughput of instructions needed for those operations is very low and stands as the main performance limiter.

This work can be extended to other compute accelerators, such as the Xeon Phi, and low-power accelerators. The design and optimization considerations for each device type may differ, and it would be interesting to draw juxtapose them for the different devices.

6 Bibliography

[1] "Kepler Tuning Guide :: CUDA Toolkit Documentation v5.5," [Online]. Available: http://docs.nvidia.com/cuda/kepler-tuning-guide.

[2] "NVIDIA’s Next Generation CUDA Compute Architecture :: Kepler GK110 Whitepaper" [Online]. Available: http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf.

Energy-Efficient Analysis of Cloudlet-based Mobile Offloading Systems

Huaming Wu
Department of Mathematics and Computer Science
Free University of Berlin
Berlin, Germany
huaming.wu@fu-berlin.de

Abstract

Offloading is a promising method for sending heavy computation tasks from mobile devices to cloud or closely located computing resources known as cloudlet. This report studies offloading decision criteria to decide when to perform the application locally, when to forward it directly for remote execution on a cloud infrastructure and when to delegate it via a nearby cloudlet to the cloud. We derive a control algorithm using Lyapunov optimization which determines when and how to offload such that energy consumption is minimized with low delay penalty. Performance evaluation shows that the proposed algorithm saves more energy and has less computational complexity than the commonly used LARAC algorithm.

1. Introduction

Mobile devices usually have multiple wireless interfaces, such as 3G/EDGE and WiFi for data transfer. However, direct cloud offloading suffers from high network access latency and low network bandwidth. To save battery power, mobile devices can offload part of their computational workload via a nearby cloudlet to a remote cloud service considering changes of the wireless environment. The design objective of our energy-efficient offloading decision algorithm is to identify under which circumstances would offloading be beneficial. We aim to minimize the energy consumed by the mobile device, while meeting a deadline.

2 Offloading Decisions

Since wireless LAN bandwidth is considerably higher than the bandwidth provided by radio access to a mobile device, different wireless technologies offer a competitive choice to connect to a nearby cloudlet and then to the cloud. As depicted in Fig.1, the bandwidth between the mobile device and cloudlet is B_1, which generally uses a high-bandwidth wireless LAN. The bandwidth between the cloudlet and cloud is B_2, which is usually based on broadband technology like internet. The bandwidth between the mobile device and cloud is B. Mostly, we have $B \leq B_1$ and $B \leq B_2$.

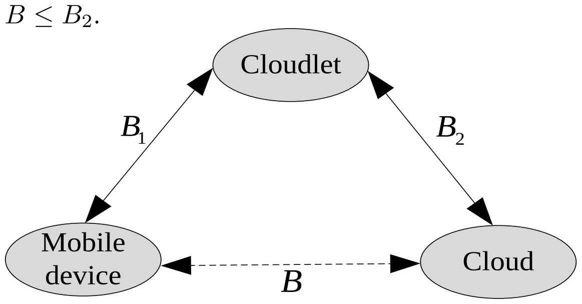


Figure 1. Model of cloudlet-based offloading systems [1]

A graphical model of adaptive offloading partition is depicted in Fig.2. Suppose there are $N+1$ application components that can be classified into two classes [2]:

1) **Unoffloadable**: In general, not all application components can be offloaded, we assume there are m components that should be unconditionally executed locally on the mobile device, either because transferring relevant information would take tremendous time and energy or because these tasks must access local components (e.g., sensors, user interfaces, etc.) [3]. Fortunately, there are no communication costs or delays.

2) **Offloadable**: $N+1-m$ application components in a mobile device are flexible tasks that can be processed either on the mobile device, or remotely on a cloud infrastructure, offloaded directly or via a cloudlet to the cloud. Many tasks fall into this category, and the offloading decision depends on whether the communication costs outweigh the local processing costs [4].

The problem of taking offloading decision correctly does not exist for unoffloadable components. However, as for offloadable ones, we need consider when they should be executed locally, when they should be offloaded directly onto the remote cloud for execution

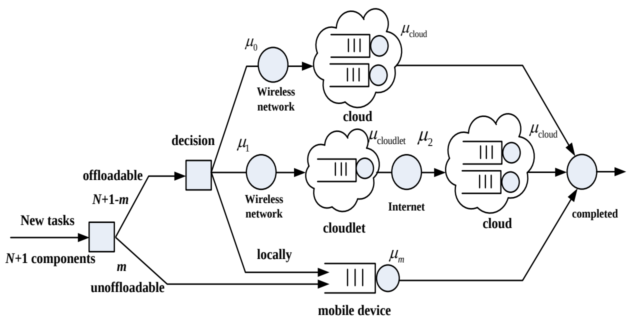


Figure 2. Mathematical model of adaptive offloading partition

and when they should be offloaded through a nearby cloudlet to the remote cloud based on available networks, response time or energy consumption. The mobile device has to take an offloading decision based on the result of a dynamic optimization problem.

3 Partition Problem Formulation

We use a graph $G = (R, S)$ with $|R| = N + 1$ to represent the relationship among the $N + 1$ application components. Each vertex $v \in R$ denotes a component and D_{uv} along the undirected edge (u, v) represents the size of data migrating from vertex u to v. When there is a request for execution, a controller in the mobile device determines which components to execute locally and which ones to execute remotely [5].
At the t^{th} execution, let the offloading decision vector be defined as

$$\boldsymbol{\omega}(t) = \big\{\omega_n(t)|n \in \{0, 1, \cdots, N\}, \omega_n(t) \in \{0, 1, 2\}\big\}_{1\times(N+1)} \tag{1}$$

where $\omega_n(t) = 1$ denotes that the n^{th} component is executed locally, $\omega_n(t) = 0$ denotes that it is directly offloaded to the remote cloud, and $\omega_n(t) = 2$ denotes that it is via a nearby cloudlet to the cloud. We can dynamically determine the total response time and energy consumption as follows.

3.1 Total Response Time

$$\begin{aligned} T(\boldsymbol{\omega}(t)) &= \sum_{v\in R} \omega_v(t) \cdot T_v^m(t) + \sum_{v\in R} |1 - \omega_v(t)| \cdot T_v^r(t) \\ &+ \sum_{(u,v)\in S} \big(2 - |\omega_u(t) - \omega_v(t)|\big) \cdot T_{uv}(t) \end{aligned} \tag{2}$$

where $\omega_v(t)$ and $\omega_u(t)$ are the elements from Eq. (1), the local execution time is

$$T_v^m(t) = \begin{cases} > 0 & \text{if } \omega_v(t) = 1 \\ 0 & \text{otherwise} \end{cases},$$

the remote cloud execution time is

$$T_v^r(t) = \begin{cases} > 0 & \text{if } \omega_v = 0 \text{ or } 2 \\ 0 & \text{otherwise} \end{cases},$$

and the transfer time from component u to v is

$$T_{uv}(t) = \begin{cases} \frac{D_{uv}}{B(t)} & \text{if } \omega_u(t) \oplus \omega_v(t) = 1 \\ \frac{D_{uv}}{B_1(t)} + \frac{D_{uv}}{B_2(t)} + T_c(t) & \text{if } \omega_u(t) \odot \omega_v(t) = 0 \\ 0 & \text{otherwise} \end{cases},$$

$\oplus$ is XOR computation and $\odot$ is NOR computation for binary variables. The corresponding parameters are listed in Table 1.

Table 1. Parameters for offloading

Symbol	Meaning
$T_v^m(t)$	Time taken when v is executed locally
$T_v^r(t)$	Time taken when v is executed remotely
$T_{uv}(t)$	Time to transfer data from u to v
$E_v^m(t)$	Energy consumed when v is executed locally
$E_v^i(t)$	Energy consumed in idle due to offloading
$E_{uv}(t)$	Energy consumed to transfer data from u to v
D_{uv}	Communication data to transfer from u to v
p_m	Power for computing
p_i	Power while being idle
p_{tr}	Power for sending and receiving data

The total execution time when all the components are executed locally is calculated as $T_{\text{local}}(t) = \sum_{v\in R} T_v^m(t)$.

3.2 Total Energy Consumption

$$\begin{aligned} E(\boldsymbol{\omega}(t)) &= \sum_{v\in R} \omega_v(t) \cdot E_v^m(t) + \sum_{v\in R} |1 - \omega_v(t)| \cdot E_v^i(t) \\ &+ \sum_{(u,v)\in S} \big(2 - |\omega_u(t) - \omega_v(t)|\big) \cdot E_{uv}(t) \end{aligned} \tag{3}$$

where $E_v^m(t) = p_m \cdot T_v^m(t)$, $E_v^i(t) = p_i \cdot T_v^r(t)$ and

$$E_{uv}(t) = \begin{cases} p_{tr} \cdot \frac{D_{uv}}{B(t)} & \text{if } \omega_u(t) \oplus \omega_v(t) = 1 \\ p_{tr} \cdot \frac{D_{uv}}{B_1(t)} + p_i \cdot \left[\frac{D_{uv}}{B_2(t)} + T_c(t)\right] & \text{if } \omega_u(t) \odot \omega_v(t) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Similarly, the total local energy consumption is $E_{\text{local}}(t) = \sum_{v\in R} E_v^m(t)$.
Three cases after making offloading decisions are listed in Fig.3. Suppose components 1 is unoffloadable, while the others are offloadable components. In case 1, components 3 is executed on the mobile device, component 4 is offloaded directly to the cloud while component 2 is offloaded via the cloudlet to the cloud, thus the decision combination vector is $\boldsymbol{\omega}_1(t) = \{1, 2, 1, 0\}$. Similarly, we have $\boldsymbol{\omega}_2(t) = \{1, 2, 0, 2\}$ and $\boldsymbol{\omega}_3(t) = \{1, 0, 0, 0\}$.
Challenges: Let $\boldsymbol{\Phi}$ be the set of all possible decision combinations. When the application has N offloadable components, we can obtain $|\boldsymbol{\Phi}| = 3^N$. For each execution, the steps to search for the optimal solution (i.e., to determine whether $\omega_n(t)$ should be 0, 1 or 2 for $\forall n = 1, 2, \cdots, N$) grow exponentially with the number of vertices [6]. Therefore, it is difficult to obtain the optimal solution directly.

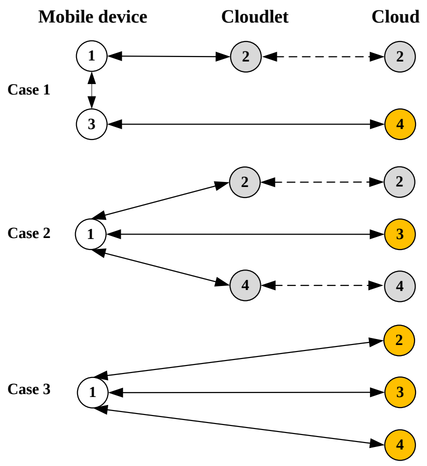


Figure 3. A partitioning example

4 Energy-Efficient Offloading Algorithm

4.1 Lyapunov Optimization

The constraint is that the total response time of that partition should be less than or equal to a deadline named T_d. Let the execution indicator variable be defined as

$$\sigma(\boldsymbol{\omega}(t)) = \begin{cases} 0 & \text{if } T(\boldsymbol{\omega}(t)) \le T_d \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

A decision combination vector $\boldsymbol{\omega}(t)$ is feasible if the total response time satisfies the delay constraint, which is denoted as $\sigma(\boldsymbol{\omega}(t)) = 0$, otherwise, we have $\sigma(\boldsymbol{\omega}(t)) = 1$. A feasible $\boldsymbol{\omega}(t)$ with minimum energy consumption is the optimal solution among all the feasible decision vectors. Formally, we have

$$\min_{\boldsymbol{\omega}(t)} \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \tag{5}$$

$$\text{subject to } \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\sigma(\boldsymbol{\omega}(\tau))\} \le \rho \tag{6}$$

where ρ is the violation ratio of the number of executions which violates the deadline to the total number of executions. Eq. (6) ensures that the system is stable.
We define the dynamic offloading system as

$$Q(t+1) = \max[Q(t) - \rho, 0] + \sigma(\boldsymbol{\omega}(t)) \tag{7}$$

where $Q(t)$ is defined as the system state at the t^{th} execution, the larger $Q(t)$ is, the longer the system's response time is.
And the conditional Lyapunov drift $\Delta(Q(t))$ is the expected change in the continuous execution of the Lyapunov function. We have

$$\Delta(Q(t)) \triangleq \mathbb{E}\{L(Q(t+1)) - L(Q(t)) | Q(t)\} \tag{8}$$

where $L(Q(t)) = \frac{1}{2}Q^2(t)$ is the Lyapunov function. To stabilize the queue state while minimizing the average energy consumption, we incorporate the expected energy consumption over one execution [7]

$$\Delta(Q(t)) + V\mathbb{E}\{E\big(\boldsymbol{\omega}(t)\big)|Q(t)\} \tag{9}$$

where $V \ge 0$ denotes an "importance weight" on how much we emphasize the energy minimization compared to the violation ratio of deadline.
Note that our objective is to minimize the average energy consumption. This is accomplished by searching for a feasible $\boldsymbol{\omega}(t)$ that greedily minimizes the decision criterion as follows

$$\arg\min_{\boldsymbol{\omega}(t)} \Big[VE\big(\boldsymbol{\omega}(t)\big) + Q(t)\sigma\big(\boldsymbol{\omega}(t)\big)\Big] \tag{10}$$

For any control parameter $V > 0$, we achieve average energy consumption and queue backlog satisfying the following two constraints [7]

$$\bar{E} = \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le \frac{C}{V} + E^* \tag{11}$$

$$\bar{Q} = \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Q(\tau)\} \le \frac{C + V(E^* - \bar{E})}{\varepsilon} \tag{12}$$

Discussion: Since the system state is closely related with response time, it follows a $[O(1/V), O(V)]$ tradeoff between the energy consumption and response time. We can achieve an average energy consumption $\bar{E}$ arbitrarily close to the optimum E^* with a diminishing gap $(1/V)$ while maintaining queue stability. However, this reduction is achieved at the expense of a larger delay because the average system state $\bar{Q}$ increases linearly with V. Choosing a large value of V can thus push the average energy arbitrarily close to optimal. However, this comes at a cost in average system state and delay that is $O(V)$ [8].

4.2 LARAC Algorithm

For comparison with the proposed dynamic offloading decision algorithm, we improve the LARAC algorithm, which uses the concept of aggregated costs and provides an efficient method to find the optimal multiplier based on Lagrange relaxation [9].
Our objective is still the same, i.e. to find the minimum energy consumption subject to the constraint that the total response time should be less than or equal to the deadline T_d. A decision combination vector $\boldsymbol{\omega}(t)$ is feasible if the total response time meets the deadline. A feasible $\boldsymbol{\omega}^*(t)$ with the minimum energy consumption is the optimal solution among all the feasible decision combination vectors. Mathematically, we have

$$\min_{\boldsymbol{\omega}(t)} \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \tag{13}$$

$$\text{subject to } \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{T(\boldsymbol{\omega}(\tau))\} \le T_d \tag{14}$$

To solve this optimization problem efficiently, we define the aggregated cost function as [23]

$$f(\lambda) = \mathbb{E}\{E(\boldsymbol{\omega}(t)) + \lambda T(\boldsymbol{\omega}(t))\} - \lambda T_d \quad (15)$$

where λ is a Lagrange multiplier.
Using Lagrange duality principle, we obtain

$$f(\lambda) \leq \mathbb{E}\big\{E\big(\boldsymbol{\omega}^*(t)\big)\big\} \quad (16)$$

which gives a lower bound for the optimal solution of the offloading policy.
Next we apply the improved LARAC algorithm as shown in Table 2, to find an optimal combination vector $\boldsymbol{\omega}^*(t)$ among all the possible offloading decision combinations. If we can find a minimum-energy combination vector that satisfies the deadline, this combination is the solution. However, if the minimum-time combination vector violates the deadline, there is no solution; otherwise we repeatedly update $\boldsymbol{\omega}^{\boldsymbol{E}}(t)$ and $\boldsymbol{\omega}^{\boldsymbol{T}}(t)$ to search for the optimal $\boldsymbol{\omega}^*(t)$.

Table 2. An Improved LARAC Algorithm

```
Finding the optimal solution with $\boldsymbol{\omega}^*(t)$
1: **Input**: $\mathbb{E}\{E(\boldsymbol{\omega}(t))\}, \mathbb{E}\{T(\boldsymbol{\omega}(t))\}$ and $T_d$
2: **Output**: $\boldsymbol{\omega}^*(t)$
3:     $\boldsymbol{\omega}^{\boldsymbol{E}}(t) = \arg\min_{\boldsymbol{\omega}(t)} \mathbb{E}\{E(\boldsymbol{\omega}(t))\}$
4: **if** $\mathbb{E}\{E(\boldsymbol{\omega}^{\boldsymbol{E}}(t))\} \leq T_d$ **then**
5:         return $\boldsymbol{\omega}^{\boldsymbol{E}}(t)$
6: **end if**
7: $\boldsymbol{\omega}^{\boldsymbol{T}}(t) = \arg\min_{\boldsymbol{\omega}(t)} \mathbb{E}\{T(\boldsymbol{\omega}(t))\}$
8: **if** $\mathbb{E}\{T(\boldsymbol{\omega}^{\boldsymbol{T}}(t))\} > T_d$ **then**
9:         return "There is no feasible solution"
10: **end if**
11: **while** true **do**
12:         $\lambda = \frac{\mathbb{E}\{E(\boldsymbol{\omega}^{\boldsymbol{E}}(t))\} - \mathbb{E}\{E(\boldsymbol{\omega}^{\boldsymbol{T}}(t))\}}{\mathbb{E}\{T(\boldsymbol{\omega}^{\boldsymbol{T}}(t))\} - \mathbb{E}\{T(\boldsymbol{\omega}^{\boldsymbol{E}}(t))\}}$
13:         $\boldsymbol{\omega}^*(t) = \arg\min_{\boldsymbol{\omega}(t)} \mathbb{E}\{E(\boldsymbol{\omega}(t)) + \lambda T(\boldsymbol{\omega}(t))\}$
14:         **if** $\mathbb{E}\{E(\boldsymbol{\omega}^*(t)) + \lambda T(\boldsymbol{\omega}^*(t))\} ==$
15:             $\mathbb{E}\{E(\boldsymbol{\omega}^{\boldsymbol{E}}(t)) + \lambda T(\boldsymbol{\omega}^{\boldsymbol{E}}(t))\}$ **then**
16:                 return $\boldsymbol{\omega}^{\boldsymbol{T}}(t)$
17:         **else**
18:                 **if** $\mathbb{E}\{T(\boldsymbol{\omega}^*(t))\} \leq T_d$ **then**
19:                     $\boldsymbol{\omega}^{\boldsymbol{T}}(t) = \boldsymbol{\omega}^*(t)$
20:                 **else**
21:                     $\boldsymbol{\omega}^{\boldsymbol{E}}(t) = \boldsymbol{\omega}^*(t)$
22:                 **end if**
23:         **end if**
24: **end while**
```

5 Simulation and Results

As depicted in Fig.4, the energy consumption falls quickly at the beginning and then tends to descend slowly while the response time grows linearly with V first and then tends to increase slowly. This finding confirms that there is a $O(1/V), O(V)$ tradeoff between average energy consumption and average response time. A good operating point would be to pick

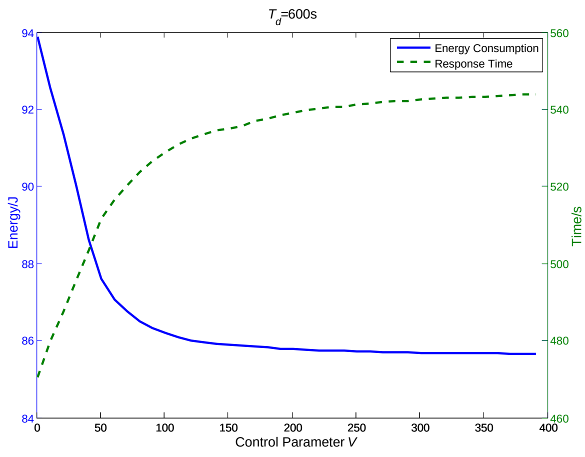


Figure 4. Energy consumption and response time

a V value (e.g., $V = 100$) where a unit increase in V yields a very small reduction in $\bar{Q}$. At such point, the energy gains may not be worth the response time rise from increasing V.
We compare the average response time and energy consumption with the following methods:

- *Local scheme*: All application components are executed locally on the mobile device.
- *Cloud scheme*: All offloadable application components are directly offloaded to the cloud for further processing.
- *Cloudlet scheme*: All offloadable application components are offloaded via the cloudlet to the cloud for further processing.
- *Lyapunov scheme*: The dynamic offloading decision algorithm using Lyapunov optimization.
- *LARAC scheme*: Optimal scheduling using the improved LARAC algorithm.

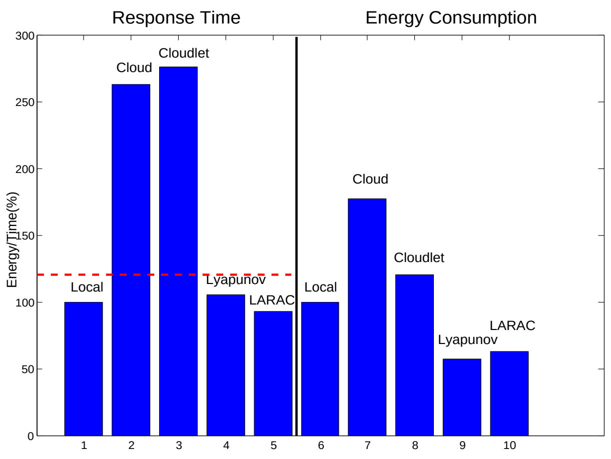


Figure 5. Comparison of average response time and energy consumption under different schemes

Fig.5 shows the average response time and energy consumption, normalized to the local scheme. The red dotted line denotes the deadline. It is found that the Lyapunov scheme can help to save around 50% of the energy consumption compared to the local scheme while only sacrificing a small portion of response time. This is because the Lyapunov scheme offloads components dynamically according to network bandwidth and transmit power, while both the cloud scheme and the cloudlet scheme do not take the network bandwidth into consideration. Especially, when the network bandwidth is so low that offloading components to the cloud or via the cloudlet to the cloud may not be beneficial. Besides, when compared it with the optimal scheduling using the LARAC algorithm, our proposed scheme also saves more energy while only sacrificing a small portion of response time.

6. Conclusions

We present an approach for dynamical offloading decisions and consider all factors such as application responsiveness, energy characteristics and particularly the changing landscape of network connectivity (cellular network vs. WiFi to cloud vs. cloudlet). We formulate the offloading decision problem as an optimization formulation which minimizes the total energy expenditure while satisfying the deadline. Numerical results show that our proposed dynamic algorithm can save around 50% of the energy consumption as compared with local execution while only slightly sacrificing response time. Validation based on real workloads and more realistic application examples will be provided in the future to demonstrate insights about efficiency of the proposed algorithm.

References

[1] H. Wu, K. Wolter, and A. Grazioli.: Cloudlet-based Mobile Offloading Systems: a Performance Analysis. In: 31st International Symposium on Computer Performance, Modeling, Measurements and Evaluation 2013 Student Poster Abstracts, Vienna, Austria (2013)

[2] E. Hyytiä, T. Spyropoulos, and J. Ott.: Optimizing Offloading Strategies in Mobile Cloud Computing. Submitted (2013)

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl.: Maui: Making Smartphones Last Longer with Code Offload. In: 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62. ACM, New York (2010)

[4] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava.: A Survey of Computation Offloading for Mobile Systems. In: Mobile Networks and Applications, vol. 18, no. 1, pp. 129–140. Springer, Heidelberg (2012)

[5] D. Huang, P. Wang, and D. Niyato.: A Dynamic Offloading Algorithm for Mobile Computing. In: IEEE Transactions on Wireless Communications, vol. 11, no. 6, pp. 1991–1995. IEEE Press, New York (2012)

[6] B. G. Chun and P. Maniatis.: Dynamically Partitioning Applications between Weak Devices and Clouds. In: 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond (MCS), no. 7. ACM, New York (2010)

[7] M. J. Neely.: Stochastic Network Optimization with Application to Communication and Queueing Systems. In: Synthesis Lectures on Communication Networks, vol. 3, no. 1, pp. 1–211. Morgan & Claypool Publishers (2010)

[8] M. Ra, J. Paek, A. Sharma, R. Govindan, M. Krieger, and M. Neely.: Energy-Delay Tradeoffs in Smartphone Applications. In: 8th International Conference on Mobile Systems, Applications, and Services, pp. 255–270. ACM, New York (2010)

[9] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó.: Lagrange Relaxation Based Method for the QoS Routing Problem. In: IEEE Infocom 2001, vol. 2, pp. 959–868. IEEE Press, New York (2001)

Smart Data Analysis for the Support of Rational Decision Making in the Energy Sector
– Project Report –

Witold Abramowicz Wioletta Sokolowska Tymoteusz Hossa Jakub Opalka
Karol Fabisz Agata Filipowska Mateusz Kubaczyk

Department of Information Systems
Faculty of Informatics and Electronic Economy
Poznan University of Economics
Al. Niepodleglosci 10, 61-875 Poznan, Poland
firstname.lastname @kie.ue.poznan.pl

Abstract

This report gives an insight into activities performed in the field of prototyping of an in-memory Business Intelligence solution for the support of decision making in the field of energy sector by combining structured and unstructured energy data sources and employing the analytical and computational power of SAP HANA (using PAL, R, Python SAP HANA Text Analysis) and Tableu Software, so to expand the possibilities of previously created working prototype of Business Intelligence solution and equip the business analysts with a simple, but an effective tool.
The report provides information on the project main objectives, used HPI Future SOC Lab resources, findings as well as next steps envisioned.

1. Introduction

The energy sector is currently undergoing major changes in terms of technology, security, market operations and business models etc [2], [3], [4], [5]. The energy operators focus on planning of energy generation (from own sources) and energy acquisition (on the market) in order to satisfy the predicted energy demand. They are trying to address one of the biggest challenges, namely how to accurately predict a short- and long-term value of energy demand, as well as the level of energy production from different sources. However, due to the ongoing market liberalization, the emergence of new participants (e.g. prosumers, new competitors), there are new issues emerging, such as monitoring of customer satisfaction regarding operators offer, service quality etc. that are becoming crucial. Thus, to make rational decisions and react quickly to changes in the business environment and to build a competitive advantage, organizations need to constantly analyze numerous information sources. However, taking into account the sheer amount of available data, in order to identify the relevant one to be analyzed and then, e.g., prepare various forecasts and simulations or (and) evaluate the emotional load of the published texts, there is a need to automate the assessment process and employ various IT techniques (i.e. natural language processing and sentiment analysis techniques), as well as an adequate analytical tool.

The current project is a continuation of three projects *Quasi Real-Time Individual Customer Based Forecasting of Energy Load Demand Using In Memory Computing*, *Forecasting of Energy Load Demand and Energy Production from Renewable Sources using In-Memory Computing*and *Prototype of an In-Memory Business Intelligence Solution for the Support of Forecasting of Energy Load Demand* ran previously under HPI Future SOC Lab.

Within this one, the goal was to broaden the scope of analysis for the needs of rational decision-making in the energy sector by focusing not only on the structured, but also on unstructured data, in order to expand the possibilities of the previously created working prototype of Business Intelligence solution. The research hypothesis that we focused on, was that by combining information from various data sources, internal as well as external ones, and employing the analytical and computational power of SAP HANA both for the structured and unstructured data, it will be possible to equip business analysts with a tool, which by providing relevant information, will allow to decrease the uncertainty connected with the decision making process.

The document presents the attemps undertaken within this project and it is organized as follows. First, the project aims are shortly presented. Then, the used Future SOC Lab Resources are pointed and few technical details are given. Next, the obtained results are briefly summarized. The document concludes with final remarks and an outlook on the future work.

2. Project Aims

As already mentioned, the project reported in this document is a part of a cycle of undertakings aiming at building an analytical solution using SAP HANA for the support of business analysts in the energy sector.
The short synopsis of the projects main scenario is depicted in the Figure 1.

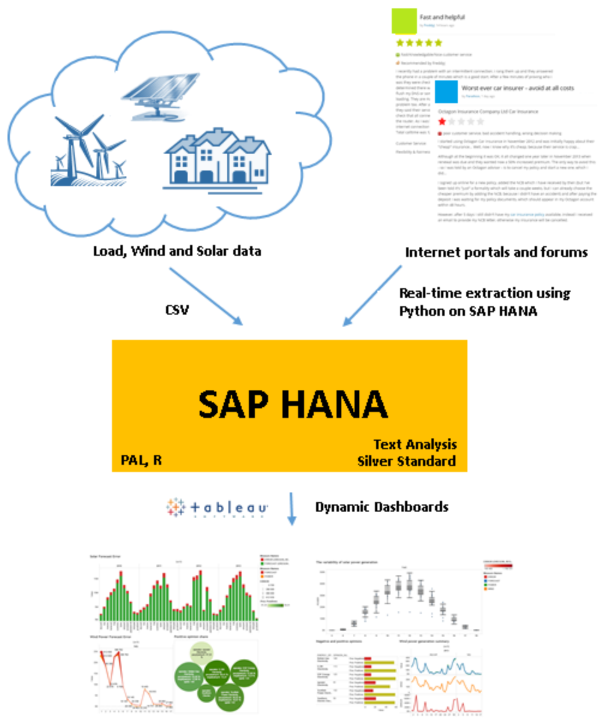

Load, Wind and Solar data
Internet portals and forums
CSV
Real-time extraction using Python on SAP HANA
SAP HANA
PAL, R
Text Analysis Silver Standard
tableau
Dynamic Dashboards

Figure 1. Project scenario

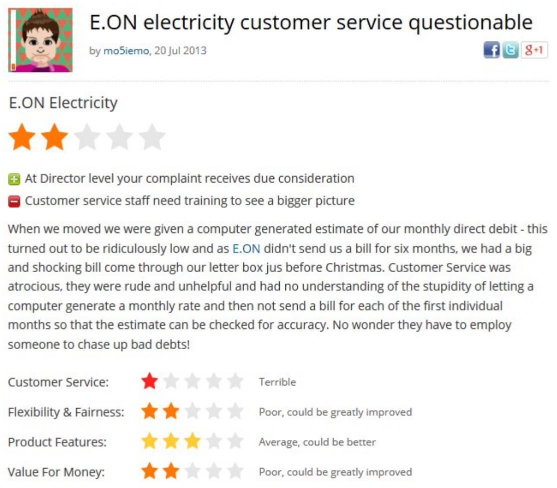
As shown in the Figure 1, research activities envisioned within this project focused on analyzing a large volume of the unstructured data and improving the accuracy of so far proposed forecasting models (both energy demand and energy generation). Therefore, the emphasis was put on:

- identifying interesting data sources where the opinions on the energy provider and the provider's offer were published;
- connecting Python with SAP HANA and building code enabling for automated data acquisition from sources identified in the previous step;
- employing SAP HANA text analysis and performing a sentiment analysis on the automatically acquired data (with the use of standard HANA dictionary);
- designing and implementing new forecasting methods (both energy demand and energy generation)to prepare more reliable predictions while using both PAL and R;
- buliding methods that would allow for combining the results of both analyses;
- expanding our dashboard-like solution with new functionalities and creating more interactive visualizations.

The additional goal was to evaluate efficiency and performance of Text Analysis of SAP HANA. Moreover, to improve the reporting capabilities and to create dynamic visualitations we connected SAP HANA with Tableau Software 8.2.

3. Future SOC Lab Resources Used

During the project, we accessed a standard physical machine with SAP HANA instance (12) together with SAP HANA Predictive Analysis Library (PAL) and combined it with R for more advanced predictive analyses. Moreover, as mentioned before, the work on the unstructured data required to connect Python with SAP HANA and to use SAP HANA Text Analysis. To make our findings clearer for business analysts we decided to connect SAP HANA with Tableu Software 8.2.
Due to the very diligently carried out state-of-the-art literature research, we decided to use the opinions of current and future clients on the energy provider and on the provider's offer posted on the Internet fora, instead of tweets that many researchers have already tried to analyze with more or less satisfying results [1]. We found our approach even more challenging as in general there were no research activities with the use of sentiment analysis in the field of energy sector. Therefore, we weren't able to directly compare achieved results.
Using the fact that Python is integrated with SAP HANA, we managed to scrap automatically and to upload (to stream them in almost real-time) the data directly from directly to the SAP HANA. Our Python code visited each of sub-webpages and gathered not only the opinions on energy providers but also star-ratings provided in four sub-categories: customer service, flexibility and fairness, features, value for money and one overall rating, all represented by 1 to 5 value. 893 opinions were automatically gathered from the period of over two years between February 2012 and July 2014. In average the length of each opinion was greater than 160 characters. Figure 2 presents one of many opinions of energy providers' clients available on the aforementioned website.
With the use of SAP HANA text analysis tool (in particular the EXTRACTION CORE VOICEOFCUSTOMER configuration) we were able to analyze the



E.ON electricity customer service questionable

by mo5iemo, 20 Jul 2013

E.ON Electricity

★★☆☆☆

🟩 At Director level your complaint receives due consideration

🟥 Customer service staff need training to see a bigger picture

When we moved we were given a computer generated estimate of our monthly direct debit - this turned out to be ridiculously low and as E.ON didn't send us a bill for six months, we had a big and shocking bill come through our letter box jus before Christmas. Customer Service was atrocious, they were rude and unhelpful and had no understanding of the stupidity of letting a computer generate a monthly rate and then not send a bill for each of the first individual months so that the estimate can be checked for accuracy. No wonder they have to employ someone to chase up bad debts!

Customer Service:	★☆☆☆☆	Terrible
Flexibility & Fairness:	★★☆☆☆	Poor, could be greatly improved
Product Features:	★★★☆☆	Average, could be better
Value For Money:	★★☆☆☆	Poor, could be greatly improved

Figure 2. Opinion of energy provider's client - example

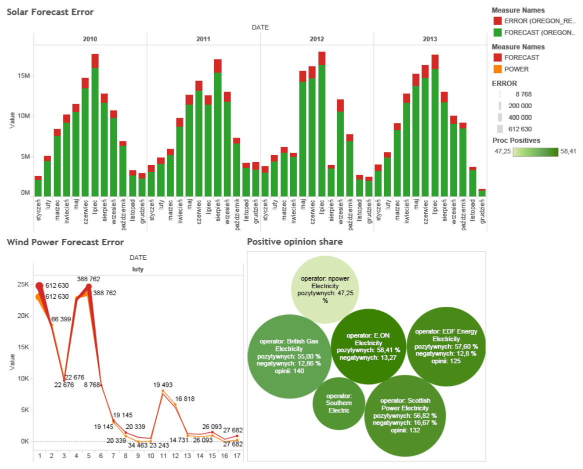


Figure 3. Energy sector business analyst dashboard - example 1

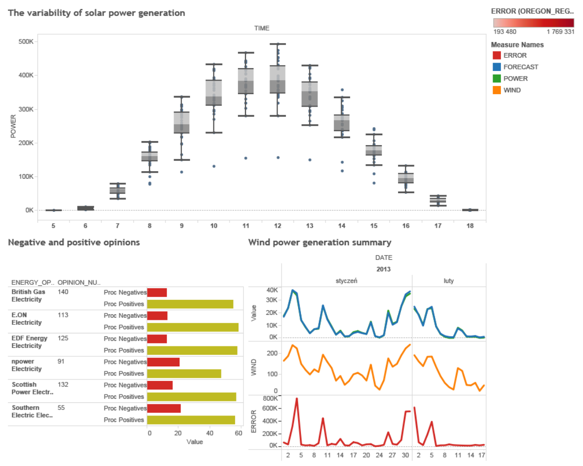


Figure 4. Energy sector business analyst dashboard - example 2

acquired opinions and thus to extract relevant information and finally to transform it into more structured form so it could be leveraged in different ways.
To fulfil our goals, we also used the previously acquired data on the energy load demand obtained from a major Polish energy distributor and on electricity generation from renewable sources (namely solar data and wind data) complemented with additional meteorological data. The continuation of using this data was reasonable due to the fact that current project was an extension of the experiments carried out earlier.
The outcomes of both experiments were combined with the use of Tableau Software.

4. Findings

All visualizations presented in this section were made in Tableau Software 8.2, which allows for creating interactive dashboards powered by the computational and analytical power of SAP HANA. Figure 3 depitcs one of many possible data and charts combination that might be useful for any energy sector analyst. This particular example represent an information on three different subjects: solar energy generation forecast, wind energy generation forecast and a result of sentiment analysis on customer opinions about market energy operators.
The chart at the top of the dashboard 3 presents solar forecast error. The histogram depicts solar power forecasts (green color), with a share of forecast error (red color). Data is aggregated as sum of forecasts (and errors) within specific months from the period of 2010 to 2013. The wind energy generation forecast error is presented at the bottom left side. The chart contains of a real value of produced power from one of the examined turbines (orange color), with a comparison to its forecast (red line). Moreover, the thickens of each line represents the level of the forecast error. The thicker the line the bigger the forecast error. These two simple chart can provide an important information for energy sector analyst in a quick and convenient way. This part of the dashboard allows him to assess globally how accurate were his forecasting methods.
The last diagram that was included in the dashboard 3 (bottom right side) presents the graphical results of the conducted sentiments analysis. The size of a bubble is determined by the number of acquired opinions on specific energy operator, while the color indicates the percentage of the volume of the amount of positive opinions in relation to the total number of opinions for the same energy operator. Moreover, the dark green represents the greatest number of positive opinions among the examined companies. The color becomes paler, the lower was the participation of the positive opinions.
The second example of generated dashboard is pre-

sented in the Figure 4. Here, the comparison between chart generated from different data types (structured and unstructured) can be seen. At the top of dashboard, there is a chart that describes the variability of the solar power generation for the selected time interval. It is easy to notice that the power generation in this course is very volatile, especially in the hours around the noon. The chart on the bottom right side of the dashboard presents the wind power generation summary from two last months.
This the top chart compares wind power real value and its forecast, in the middle chart are presented values of wind strength and at the bottom chart there is forecast error. It is easy to notice that wind and generated power variables are highly correlated. May also be observed that the biggest forecast error occurs during strong wind speed changes. Last figure at the bottom left of presented dashboard presents another chart about sentiment analysis, where all the results of this research was gathered in one place. In this figure is presented the percentage of negative and positive feedback for six different energy operators. The values presented at the chart are the share of comments with the same sentiment orientation to the all examined comments. This analyses was conducted basing on raw comments on the forums which are created by energy customers. For example energy sector business analyst can get know that the company EDF Energy Electricity positive opinion share is about almost 58% percent. At the same time this company has slightly less than 13% of negative reviews share.
To perform evaluation of the gained results (measures used) and in particular the performance of the SAP HANA text analysis component we used so-called silver standard. The precision, recall and F-measure parameters were calculated with the reference to the star-ratings provided by the users in the four abovementioned sub-categories. The obtained results are presented in Table 1. The average F-measure value was at the level of 0.51. The experiment showed that the identification of the positive sentiment among comment is quite good, as it is indicated by relatively high values of the precision and recall measure. However, SAP HANA is unfortunately underperforming in field of identifying negative and neutral sentiments.
At this stage we are finalizing the process of evaluating our method by comparing the results with a human annotated opinions (so called gold standard).

	Comment		
	Negative	Neutral	Positive
Precision	0.36	0.41	0.73
Recall	0.56	0.36	0.71
F-Measure	0.44	0.38	0.72

Table 1. Silver Standard values of precision, recall, F-measure parameters for positive, neutral and negative comments.

5. Conclusions and Next Steps

The conducted research using the Future SOC Lab resources allowed us to gain deep insight into SAP HANA, especially into its sentiment analysis capabilities. Therefore, we were able to expand the possibilities of previously created working prototype of Business Intelligence solution and to equip the business analysts with a simple, but an effective tool that combines information coming from different sources (structured and unstructured).
The research proved that business analyst may take an advantage of using our application that employs the analytical and computational power of SAP HANA, PAL, R, Python, SAP HANA Text Analysis and Tableau Software.
The main conclusions from our experiments are as follows:

- making a right decision concerning the use of methods, programs, analytical environment, etc. is crucial in the context of preparing a right tool for the specific group of recipients such as business analysts in the energy sector;
- working with code and queries is too technical for any practical use, therefore a lot of work has to be done before the data may be visualized in an easy to read and interpret manner:
  1. finding the reliable (complete, accurate, free from errors and mistakes) data source is the crucial step at the beginning;
  2. the process of data extraction, preparation and in-HANA implementation takes time;
  3. a great emphasis should be put on choosing, preparing and implementing the right methods in SAP HANA environment that are responsible for calculation of the sentiment values, moreover the default HANA dictionary reguires reconfiguration towards i.e. the domain specific terms;
- the better (more user-friendly) software or program - the faster results - the more satisfied customer.

To conclude, within our following project we aim at extending the scope of the sentiment analysis for the needs of rational decision-making by focusing only on unstructured data and expanding the possibilities of previously created working prototype of Business Intelligence solution.

We hope that by combining information from various unstructured data sources and employing the analytical and computational power of SAP HANA, we will be able to develop automated and precise sentiment analysis tool dedicated for business analysts from the energy sector.

References

[1] M. Arias, A. Arratia, and R. Xuriguera. Forecasting with twitter data. *ACM Trans. Intell. Syst. Technol.*, 5(1):8:1–8:24, Jan. 2014.

[2] K. Fabisz, A. Filipowska, and T. H. R. Hofman. Profiling of prosumers for the needs of energy demand estimation in microgrids. In *Proccedings of the 5th International Renewable Energy Congress*, 2014.

[3] A. Filipowska, K. Fabisz, T. Hossa, M. Mucha, and R. Hofman. Towards forecasting demand and production of electric energy in smart grids. In *Perspectives in Business Informatics Research, 12th International Conference BIR2013*, 2013.

[4] T. Hossa, A. Filipowska, and K. Fabisz. The comparison of medium-term energy demand forecasting methods for the needs of microgrid management. In *Proceedings of SmartGridComm, IEEE International Conference on Smart Grid Communications*, 2014.

[5] W. Sokolowska, J. Opalka, T. Hossa, and W. Abramowicz. The quality of weather information for forecasting of intermittent renewable generation. In J. Marx Gmez, M. Sonnenschein, U. Vogel, A. Winter, B. Rapp, and N. Giesen, editors, *INFORMATION AND COMMUNICATION TECHNOLOGY FOR ENERGY EFFICIENY, Proceedings of the 28th International Conference on Informatics for Environmental Protection (EnviroInfo 2014)*. Oldenburg: BIS-Verlag, Carl von Ossietzky University Oldenburg, Germany, 2014.

Logical SDNs: Reaping Software-Defined Networking Benefits Through Incremental Deployment

Stefan Schmid[†,•]
stefan@inet.tu-berlin.in

Fabian Schaffert[†]
fabian@badpacket.in

Dan Levin[†]
dan@inet.tu-berlin.de

Marco Canini[‡]
marco.canini@uclouvain.be

[†]TU Berlin
Straße des 17. Juni 135
D-10623 Berlin

[•]T-Labs
Ernst-Reuter-Platz 7
D-10587 Berlin

[‡]Université catholique de Louvain
Place de l'Université 1
1348 Louvain-La-Neuve, België

Abstract

Although SDN promises to address long-standing network operations problems, with the exception of a few notable deployments, e.g., Google's B4, it remains largely an experimental technology. As the transition of existing networks to SDN will not be instantaneous, we consider hybrid networks that combine SDN and legacy networking devices an important avenue of research; yet research focusing on these environments has so far been modest. Hybrid networks posses practical importance, are likely to be a problem that will span years, and present a host of interesting challenges: Namely, the beneficial, co-existence of radically different networking paradigms.

In this work, We argue for a hybrid networking approach that introduces SDN devices into existing networks to abstract the legacy network devices away as "expensive wires" and expose a programmatic "logical SDN" interface — conceptually, a representation of the network limited to just the SDN devices. To better understand the potential and limits for the logical SDN abstraction for hybrid networks, we showcase the power and utility of the logical SDN by reasoning through and implementing use-case control applications built on this abstraction. This abstraction comes at the cost however, of re-directing traffic through SDN devices. We thus explore the effects on network traffic flow performance through experiments in a high-performance emulation environment.

1. Introduction

The term "single pane of glass" [1, 3] has been coined in the systems and network operations community, to describe the ideal, operational "holy grail" where every input to a complex systems (e.g. a computer network) is expressed through a single, unified, common interface. Software-defined networking (SDN) is an attractive paradigm that potentially pushes network management closer to this ideal state.

SDN entails a logically-centralized control plane running different control applications, managing the forwarding behavior of a collection of switches via a standardized interface. The centralized perspective and simple interface have the potential to make the network more programmable, thereby reducing the complexity of network management (today an often cumbersome and manual process), and facilitate better-optimized and -automated network operation and troubleshooting.

Despite the need for principled approaches to long-standing network operational problems, with the exception of a few notable deployments in the wild (*e.g.*, Google's B4 [5]), SDN remains largely an experimental technology for most organizations.

One major reason for this mismatch is the *SDN deployment problem*: on the one hand, potential SDN adopters must first be able to establish confidence in SDN, but on the other hand SDN is not merely a "new feature" that can be "switched on" to provide value to existing networks. Moreover, as budgets are constrained, it is often not possible to replace all existing legacy hardware by SDN in one shot, but rather, *only a part of the network can be upgraded* at a time. An upgrade to SDN hence, does not begin with a green field, but with the existing deployment, and is typically a staged process. Even Goggle's B4 system required a significant multi-year deployment undertaking before its benefits could be realized. Smaller organizations will not typically have the resources to roll out their own SDN in a similar fashion. As such, we envision that transition to SDN will first occur in the form of partial deployments that co-exist with legacy hardware—that is, hybrid networks. Crucially, however, these partial deployments must provide value from the very beginning.

One frequently encountered hybrid deployment model occurs in the datacenter where SDN can be deployed at the edge (*i.e.*, on the server's hypervisor) [7]. In other settings, *e.g.*, in many enterprise networks, an upgrade of the edge is prohibitively expensive and out-of-the-question: the edge constitutes a significant fraction of the entire network; moreover, unlike in the datacenter, the edge is typically a legacy hardware switch not a software switch.

Our Contributions. In this paper, we consider the problem of how to operate an arbitrary hybrid network with the goal of enabling a partial SDN deployment to provide substantial benefits of the SDN programming interface. We propose a very general approach that abstracts the hybrid network as a *logical SDN*. Such a logical abstraction is attractive as it directly supports existing control applications which have been designed for full SDN deployments: the application can simply run on the provided logical SDN abstraction, which appears to the application as a "full deployment" of *just* the SDN switches.

The abstraction of a hybrid network as a logical SDN can be achieved by *SDN waypoint enforcement* [9]: the requirement that every packet between a source and a destination traverses at least one SDN switch, where the network policies are applied to the traffic using *e.g.*, the match-action paradigm.

This paper investigates the opportunities and limitations of such logical SDN abstractions, and we showcase the power of the logical SDN by reasoning through and implementing use-cases. Via these use cases, we demonstrate the utility of the programming interface offered by the logical SDN. The logical SDN abstraction comes at the cost however, of diverting traffic from legacy switches through SDN devices. We thus explore the impact of the logical SDN on the network traffic flow performance through experiments in a large-scale emulation environment. We find encouraging evidence that the traffic performance costs of waypoint enforcement may in many cases be moderate, and in some cases, performance can even improve.

2. Logical SDN

We proceed now to introduce Panopticon, a hybrid SDN architecture that enables incremental deployment to realize the logical SDN abstraction. Subsequently, we discuss alternative, useful network control abstractions that can be realized with the logical SDN, e.g., the big switch abstraction or the middlebox view.

2.1 Panopticon: Hybrid SDN

Panopticon [9] is an architecture to enable incrementally deployable, hybrid software-defined networks. Given an *arbitrary* deployment of SDN switches into an existing network, Panopticon allows the network operator to abstract away the legacy devices in the network and operate the network as an SDN comprised of just the SDN-capable switches. Using this approach, with careful planning of the hybrid deployment [8], SDN capabilities can be extended to potentially every switchport in the network, not just the ports of SDN switches. Alternately, not every port need be included in the logical SDN, and in practice, resource constraints in the network may prevent a full SDN abstraction.

Panopticon works on the principle that that every packet in the network that traverses at least one single SDN switch can have the end-to-end network policy (e.g., access control) applied to it via the SDN programming interface. Panopticon extends SDN capabilities to legacy switches by ensuring that all traffic to or from any SDN-controlled (SDNc) port is always restricted to a safe end-to-end path, that is, a path that traverses at least one SDN switch. We call this property *Waypoint Enforcement.* Panopticon uses mechanisms available on legacy switches (i.e., VLANs) to restrict forwarding on legacy switches to guarantee Waypoint Enforcement.

Example. Consider the example hybrid network depicted in Figure 1a. In this example, the Solitary Confinement Tree of A, SCT (A) is the tree that consists of the paths $5 \to 1 \to 2$ and $5 \to 3 \to 4$. Note that SCT (B), which corresponds to the path $6 \to 2$, includes a single SDN switch because switch 2 is the only SDN switch adjacent to cell block $c(B)$. Figure 1b shows the logical view of the physical hybrid network enabled by SCTs. In the logical view, every SDNc port is connected to at least one frontier SDN switch via a pseudo-wire (VLAN). Via this approach, Panopticon can realize a broad spectrum of logical SDNs.

2.2 Big Switch Abstraction

Perhaps the simplest and most elegant network control abstraction achievable by a logical SDN is the Big Switch abstraction: only the ingress and egress ports of the network are exposed through the SDN programming interface and the network itself is considered a black box. Such an abstraction is ideal for defining policies whose implementation (i.e., whose flow-table rules) can be defined on any arbitrary switch located along the path between two ports. Examples of such policies include access control, mobility management

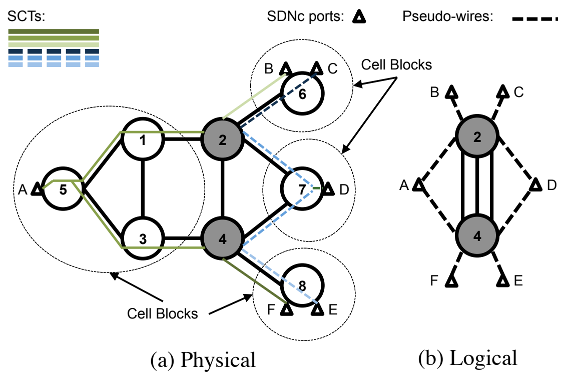


Figure 1. Example hybrid network of 8 switches (SDN switches are shaded). (a) Shows the SCTs (Solitary Confinement Trees) of every SDNc port overlaid on the physical topology. (b) The corresponding logical view where SDNc ports connect to SDN switches via pseudo-wires.

(*e.g.*, via address-locator separation), or application server load balancing. In general, this view is appealing for specifying policies that do not require visibility into the internal ports and connectivity between the devices in the network.

2.3 The Middlebox View

Many networks rely on middleboxes to increase security, improve performance and ensure policy compliance. Network planners and operators however, face challenges to carefully plan the network topology to ensure that traffic traverses the desired sequence of middleboxes in the right order, raising the overall network complexity. As noted in previous work [11], SDN offers a solution to realize middlebox traffic steering.

Through the logical SDN abstraction, we reason that it becomes possible within hybrid networks to benefit from the use of centralized management to orchestrate middlebox policy enforcement. To do so, the fidelity of the logical SDN must include enough detail (SDN switches) to permit the creation of a forwarding policy that steers traffic through the desired sequence of middleboxes. For example, the logical SDN could consist of a virtual "chain" where switches and middleboxes are interleaved. To support this kind of logical SDN, one would define a mapping to embed the logical-layer forwarding policy onto the underlying hybrid network, by using SDN waypoint enforcement.

3. Traffic Emulation Study

The abstraction of the "logical SDN" does not come for free, as the waypoint-enforcement of traffic through SDN switches can lead to increased path lengths in some cases. Panopticon however, also introduces new opportunities to improve traffic control within the network, e.g., enabling multipath forwarding for load-balancing when sufficient path diversity exists.

To investigate the consequences of Panopticon on traffic, we conduct a series of emulation-based experiments on portions of a real enterprise network topology. These experiments (*i*) provide insights into the consequences of the Panopticon architecture waypoint enforcement on TCP flow performance and (*ii*) let us explore the extent to which the deployment size impacts TCP flow performance when every access point is operated as an SDNc port. To emulate traffic in a Panopticon deployment, we make use of mininet [4] as well as topology metadata from real enterprise networks with associated traffic workloads and network resource constraints.

Topologies: Detailed topological information, including device-level configurations, link capacities, and end-host placements is difficult to obtain for sizeable networks: operators are reluctant to share these details due to privacy concerns. Hence, we leverage publicly available enterprise network topologies [12,14] to provide the input to our emulation experiments. In our topology dataset, every link in the topology is annotated with its respective capacity. When port-channels (bundled links) are present, we represent them as a single link of their aggregate capacity. Summary information on the topology is given in Table 1.

In order to overcome the system resource bottlenecks when emulating such a large network, we necessarily scale down key aspects of the network topology: We (*i*) reduce the network size to a subgraph of the full topology by pruning the graph along subnet boundaries, (*ii*) scale down the link capacities by 2 orders of magnitude, and (*iii*) correspondingly reduce the TCP MSS to 536 bytes to reduce packet sizes in concert with the reduced link capacities. These measures allow us to avoid system resource bottlenecks which would otherwise interfere with traffic generation and forwarding, thus influencing TCP throughput.

We run our experiments on a 64-core system Ubuntu Linux using OpenVSwitch version 2.1.90. Our baseline experiments indicate that our system is capable of sustaining 489 simultaneous TCP connections in excess of 34Gb/s, sufficiently saturating the emulated aggregate link capacity of every traffic sender in our experiments.

Thus, our emulation experiments involve 489 SDNc ports located at "access switches" at which traffic is sent into and received from the network. The distribution network consists of 77 switches and routers, comprising a L2/L3 network in which

Topology	Access/Dist/Core	max/avg/min degree
Full	1296 / 412 / 3	53 / 2.58 / 1
Emulated	489 / 77 / 1	30 / 6.3 / 1

Table 1. Emulated Network Topology Characteristics

28 devices are identified as IP router gateways, bridging Ethernet broadcast domains over the remainder of the switches. Within each Ethernet broadcast domain, we introduce a single spanning tree to break forwarding loops.

Workload: The workload we apply to our experiments is defined both in terms of the traffic matrix defined over the 489 SDNc ports as well as a synthetically generated flow size distribution. We use a methodology similar to that applied in SEATTLE [6] to generate a traffic matrix based on packet-level traces from an enterprise campus network, the *Lawrence Berkeley National Laboratory* (LBNL) [10]. The LBNL dataset contains more than 100 hours of anonymized packet level traces of activity of several thousand internal hosts. The traces were collected by sampling all internal switchports periodically. We aggregate the recorded traffic according to source-destination pairs and for each sample, we estimate the load imposed on the network. We note that the data contains sources from 22 subnets.

To map the traffic matrix onto our topology, we use the subnet information from the traces to partition the topology into subnets as well. Each of these subnets contains at least one distribution switch. In addition, we pick one node as the Internet gateway. We associate traffic from each subnet of the LBNL network in random round-robin fashion to candidate SDNc ports. All traffic within the LBNL network is aggregated to produce the intra-network traffic matrix. All destinations outside of the LBNL network are assumed to be reachable via the Internet gateway and, thus mapped to our designated gateway node in the topology. By running 10 different random port assignments for every set of parameters, we generate different traffic matrices which we use in our simulations.

We use a Weibull distribution with shape and scaling factor of 1 to define the object sizes that define our TCP flow size distribution, given in Table 2. Using our flow size distribution with the traffic matrix, we deterministically initiate TCP connections with the same request patterns over 10 repeated experiments, each using a differently seeded traffic matrix. Using iperf, every SDNc port (traffic source) in a sequential iterative fashion initiates TCP connections with the partners defined in the traffic matrix, transferring data defined from the flow size distribution until a maximum limit of 100MB has been sent into the network. Once every SDNc port has reached this limit, the experiment stops and the flow completion times are collected.

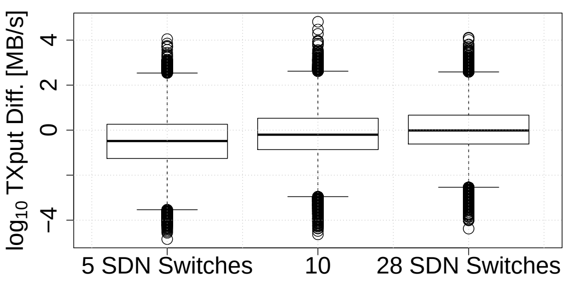


Figure 2. $\log_{10}$ **difference in TCP throughput between Panopticon vs. original legacy deployment. In both scenarios** B **and** C **(10 and 28 SDN Switches), the median throughput over all experiments remains very close to the performance of the original legacy network.**

Scenarios: We consider three different deployment scenarios in which we evaluate the effects of Panopticon on TCP traffic: Scenario A in which 5 of the 77 switches and routers, selected according to the VOL algorithm outlined in [8] are operated as SDN switches, and scenarios B and C in which 10 and respectively all 28 L3 IP routers of the topology are operated as SDN switches. In each scenario, we compare TCP flow throughput in the Panopticon deployment to the conventional L2/L3 shortest path IP routed network with minimum cost spanning trees. Table 2 gives the relevant path stretch statistics for each topology, namely, the ratio of SDN (waypoint-enforced) to legacy path length for every src-dst path in the network. In Figure 2 we illustrate the impact of Panopticon waypoint enforcement on TCP performance in the three scenarios. The first observation we make is that in scenario C, when just the 28 IP routers are replaced with SDN switches, the impact on median TCP throughput is negligible. This is perhaps expected, as all traffic across subnets must traverse some IP router in the legacy network, regardless. In extreme cases, 5% of the TCP flows experience significant deviations from the legacy network case: Some flows experience congestion due to the waypoint enforcement. Other flows actually experience a speed-increase due to the availability of multiple alternate paths in Panopticon. As the SDN deployment shrinks to more conservative sizes in scenarios B and A, the effects of waypoint-enforcement clearly become more apparent, although in all scenarios the median TCP connection throughput, never decreases by more than a factor of 3 (keep in mind the $\log_{10}$ y axis) when compared to the legacy network. These results

Parameter or Metric	min	25 %ile	50 %ile	avg	75 %ile	max
Flow Size Distribution (in MB)	0.00005	2.88	6.91	9.94	13.72	101.70
Path Stretch (5 SDN Switches)	1.0	1.0	1.33	1.25	1.33	3.0
Path Stretch (10 SDN Switches)	1.0	1.0	1.0	1.16	1.33	3.0
Path Stretch (28 SDN Switches)	1.0	1.0	1.0	1.002	1.0	1.67

Table 2. Traffic Parameter and Path Stretch Metric Statistics

are encouraging, as they demonstrate that a network of fewer than 10% SDN switches can operate as an SDN while accommodating 25% of its traffic with performance better than or equal to the original network and median throughput no worse than 1/3 of its original rate.

4. Related Work

The concept of Waypoint Enforcement used by Panopticon is grounded on previous experience.
Hybrid and transitional networking. Recently the Open Networking Foundation Migration Working Group [2] has shed some light on high-levels guidelines and methods to incrementally deploy SDN in existing networks. For example, Google's transition to a software-defined WAN involved an overhaul of their *entire* switching hardware to improve network performance [5]. Vissicchio *et al.* [13] explore hybrid SDN models that combine SDN-style control with traditional L3 networking protocols. They show a number of use cases in which hybrid models can mitigate the respective limitations of traditional and SDN approaches, providing incentives to (partially) transition to SDN. Complementary to these works, Panopticon represents an additional point in the space of hybrid networks that takes an explicit stance at transitioning to an SDN control plane without the need for a complete hardware upgrade.

5. Conclusion

This paper initiates the study of the logical SDN abstraction for hybrid software-defined networks. We weigh the benefits and limitations of different logical SDN abstractions by implementing and reasoning about several use cases. We also investigate the impact along with the opportunities presented by the abstraction in terms of network traffic performance in emulation.
We understand our work as an initial step towards a better understanding of the implications and trade-offs of SDN abstractions. In particular, we believe that the introduced notion of "logical SDN" is relevant far beyond the specific scope of incremental SDN deployment, but for other instances of hybrid, system architectural transitions. We therefore believe that our work addresses an interesting and relevant field of research.

6. References

[1] Manage Devices Through a 'Single Pane of Glass'. `http://bit.ly/1ffFDau`, accessed: 15.03.2014.
[2] Open Networking Foundation Migration Working Group: Migration Use Cases and Methods. `http://bit.ly/19yj7Hk`, accessed: 15.03.2014.
[3] L. Bitincka, A. Ganapathi, and S. Zhang. Experiences with workload management in splunk. In *Proceedings of the 2012 Workshop on Management of Big Data Systems*, MBDS '12, pages 25–30, New York, NY, USA, 2012. ACM.
[4] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 253–264, New York, NY, USA, 2012. ACM.
[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hoelzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-Deployed Software Defined WAN. In *SIGCOMM*, 2013.
[6] C. Kim, M. Caesar, and J. Rexford. Floodless in Seattle: A scalable ethernet architecture for large enterprises. In *SIGCOMM*, 2008.
[7] T. Koponen *et al.*. Network virtualization in multi-tenant datacenters. In *NSDI*, 2014.
[8] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Panopticon: Reaping the benefits of partial sdn deployment in enterprise networks. Technical report, Technical Report TU Berlin `http://bit.ly/1n1U3LD`, accessed: 15.03.2014, 2013.
[9] D. Levin, M. Canini, S. Schmid, and A. Feldmann. Toward transitional sdn deployment in enterprise networks. *Proc. Open Networking Summit (ONS)*, 2013.
[10] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proc. ACM IMC*, 2005.
[11] Z. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *SIGCOMM*, 2013.
[12] Y.-W. E. Sung, S. G. Rao, G. G. Xie, and D. A. Maltz. Towards systematic design of enterprise networks. In *CoNEXT*, 2008.
[13] S. Vissicchio, L. Vanbever, and O. Bonaventure. Opportunities and research challenges of hybrid software defined networks. *ACM Computer Communication Review*, 44(2), April 2014.
[14] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown. Automatic test packet generation. In *CoNEXT*, 2012.

Performance Optimization of Data Mining Ensemble Algorithms on SAP HANA

David Müller, Sabrina Plöger, Christoph M. Friedrich and Christoph Engels
University of Applied Sciences and Arts Dortmund, Department of Computer Science
Emil-Figge-Str. 42, D-44227 Dortmund
david.mueller@fh-dortmund.de, sabrina.ploeger001@stud.fh-dortmund.de,
christoph.friedrich@fh-dortmund.de, christoph.engels@fh-dortmund.de

Abstract

Ensemble methods (like random forests, quantile forests, gradient boosting machines and variants) have demonstrated their outstanding behavior in the domain of data mining techniques.

This project focuses on an implementation of an ensemble method on SAP HANA to combine a powerful environment with a fully developed data mining algorithm.

1 Project Idea

In the first two FSOC Lab periods the University of Applied Sciences and Arts Dortmund successfully addressed the topic *Data Mining on SAP HANA* with their project *Raising the power of Ensemble Techniques* [12]. The project idea was to compare different opportunities, which enable the usage of predictive analytical techniques on SAP HANA.

SAP is offering the Predictive Analytical Library (PAL), which contains more than 40 well-known algorithms in the fields of classification analysis, association analysis, data preparation, outlier detection, cluster analysis, time series analysis, link prediction and others [23].

In the first project period very accurate predictions could be achieved by using PAL's decision tree implementation [14]. On the other hand performance problems for certain functions in combination with special datasets occurred as the PAL implementation is relatively new and programmers have not exploited the full potential of the HANA architecture [6]. Furthermore, no ensemble methods were part of the comprehensive selection of algorithms offered by PAL, yet [23].

In the previous period the project team focused on the implementation of an ensemble method on HANA by using the SAP internal language L [13][21].

As this implementation could not match the expected performance advantages, a new project has been initiated in order to write and implement the random forest algorithm in C++ by using the SAP HANA AFL SDK, to utilize HANA's powerful capabilities for CPU-intensive algorithms [20].

Why Ensemble Methods?

Predictive statistical data mining has evolved further over the recent years and remains a steady field of active research. The latest research results provide new data mining methods which lead to better results in model identification and behave more robustly especially in the domain of predictive analytics. Most analytic business applications lead to improved financial outcomes directly, for instance demand prediction, fraud detection and churn prediction [1][3][11][15][16][26]. Even small improvements in prediction quality lead to enhanced financial effects. Therefore the application of new sophisticated predictive data mining techniques enables business processes to leverage hidden potentials and should be considered seriously.

Especially for classification tasks ensemble methods (like random forests) show powerful behavior [7][8][24] which includes that

- they exhibit an excellent accuracy,
- they scale up and are parallel by design,
- they are able to handle
    - thousands of variables,
    - many valued categories,
    - extensive missing values,
    - badly unbalanced datasets,
- they give an internal unbiased estimate of test set error as primitives are added to ensemble,
- they can hardly overfit,
- they provide a variable importance and
- they enable an easy approach for outlier detection.

Why SAP HANA?

SAP HANA is a "flexible, data-source-agnostic toolset […] that allows you to hold and analyze massive volumes of data in real time" [4]. It enhances data processing by sophisticated technologies like Massive

Parallel Processing (MPP), in-memory computing, columnar data storage and others [4][17][19][22]. Through this project the powerful capabilities of SAP HANA shall be exploited to gain fast processing of CPU-intensive predictive calculations.

Project Goal and Strategy

The overall project idea is to implement a random forest on SAP HANA by using the language C++, as computations in L could not attain the expected performance advantages.

The project consists of the following milestones:

- Lessons Learned – examine L implementation to work out important insights for the upcoming project tasks.
- Construct a concept for random forest and its prediction in C++.
- Understand how to use the SAP HANA SDK und how to work with C++ on SAP HANA.
- Implement a plain library on SAP HANA.
- Implement a decision tree and its prediction in C++.
- Extend the decision tree to a random forest.
- Test all decision tree approaches available on SAP HANA, to get valid statements about accuracy and runtime results of the C++ implementation, comprising
    - PAL
    - C++ (own implementation)
    - L (own implementation)
    - Different R packages (C50, tree, rpart)

  The Tests should deliver performance and accuracy results for both, the C++ decision tree and the random forest approach.

2 Used Future SOC Lab Resources

For this project a HANA environment (HW and SW) with the latest PAL distribution and the access to use the HANA AFL SDK is needed. For testing purposes, an R server is needed which runs in an environment equal to the HANA environment. Therefore both systems should run in virtual machines with the same set up.

3 Project Findings and Impacts

Impacts on the project and its results are listed in this chapter, as well as the project findings.

3.1 Usage of PAL Functions

PAL functions are written in C++ and its algorithms are making use of special HANA database table types, which are not accessible if the SAP HANA AFL SDK is used [18]. Therefore the coding of the decision tree has to be done from scratch, inspired by the L implementation.

3.2 Parallelization

A random forest can be executed highly parallel and the HANA AFL SDK offers the possibility of parallelization for C++. To make the C++ decision tree implementation comparable to PAL and R algorithms, it has to be implemented sequential, as the parallelization approaches of PAL and self-provided libraries are different. Therefore, parallelization is implemented on random forest level (coarse grain parallelization) and not on tree level.

3.3 Datasets for Testing

Five datasets are picked for the testing, comprising

- KRKOPT: Chess endgame database for white king and rook against black king. The goal of this classification is to predict the number of moves for white to win. The dataset consist of 28.056 observations and 6 discrete attributes [1].
- Car-Purchase: The goal of this dataset is to predict if the car purchased at an auction is a good or bad buy. The dataset consist of 72.666 observations, 13 discrete and 13 numeric attributes [9].
- Connect4: Predicting if player one wins the game or not. The dataset consist of 67.557 observations and 42 discrete attributes [25].
- Covertype: Predicting the forest cover type from cartographic variables. The dataset consist of 581.012 observations, 44 discrete attributes and 10 numeric attributes [5].
- Pokerhand: Each record is an example of a hand consisting of five playing cards. The class describes the poker hand. The dataset consist of 1.025.010 observations and 10 discrete attributes [10].

Those datasets are heterogeneous in the number of observations, the number of attributes, the distribution of discrete und numeric attributes and the number of distinct values of both, discrete values and the class column. Thus they are providing a foundation for solid and applicable test results.

3.4 Performance

For running the tests, virtual machines with the same setting have to be used for HANA und the R environment.

The C++ implementation achieves the shortest runtimes for creating a decision tree in nearly all of the completed test cases. This covers all tests with KRKOPT, Car-Purchase, Connect4 and Covertype (see appendix 1).

Only for Pokerhand, the R C50 implementation is faster, independent from the sample size of the training data. Furthermore, the runtime for the PAL decision tree increases slower, the larger the sample size of the training data is. As the runtime of the C++ implementation is increasing in a relatively linear fashion, there is a breakeven point at a training dataset of about 824.500 observations. From this point, PAL performs the calculation faster than the C++ implementation if the tree is unpruned.

Tests are pending, if this coherence is applicable for all large datasets.

The test cases for the prediction are offering similar results. The C++ algorithm is the most efficient implementation for all datasets except Pokerhand, where C50 is the fastest implementation (see appendix 2).

The random forest implementation achieves good performance results as well. Despite the application of 50 unpruned trees, the training algorithm is only three to ten times slower than creating only one tree with C++ and the prediction is just two to seven times slower. This results can be attributed to the usage of bagging, randomization and parallelization.

3.5 Prediction Accuracy

The prediction results are satisfying and the implemented algorithm runs reliably. Depending on the dataset, parameters and the selection of test and training data, the prediction accuracy can either be better or worse compared to the PAL C4.5 decision tree. The difference in accuracy between the C++ approach and PAL is very small, as those methods work very similar (see appendix 3).

However, it is important to point out, that the random forest leads to different results. In most cases, this means a better prediction, compared to all decision trees, tested in this period (see appendix 3). For Covertype data, the random forest can increase the accuracy up to nine percent.

4 Final Results / Deliveries

The main contribution of this project is the random forest implementation in C++. For now, this implementation is one of the fastest prediction models available on SAP HANA, proven by a variety of tests. Furthermore, it could be verified, that the random forest leads to better accuracy results in many test cases.

5 Next steps

There are a lot of opportunities to use the project results for further improvements.

On the one hand, the implementation can be optimized. Especially the prediction can be processed much faster by using a linked list instead of holding the decision tree in a SQL table.

Beside the performance improvement, there are some opportunities to optimize the algorithm and its prediction quality, for example different approaches for identifying the best split for numeric attributes, handling of missing values or implementing a post pruning by regarding a validation dataset.

Furthermore, the algorithm should be adjusted to formal specifications of existing random forest concepts.

The delivered random forest can even be used to adapt the algorithm to other predictive models, as for example quantile forests or gradient boosting machines.

6 Conclusion

The ensemble technique is implemented successfully in this project period and all project goals are accomplished. The implementation runs reliably and offers a strong and fast predictive model. Nevertheless, there are still some opportunities to optimize the implementation with respect to performance and accuracy of prediction by application of other programming paradigms and further-developed prediction methods.

7 References

[1] M. Bain, A. van Hoff (University of New South Wales, Sydney, Australia): KRKOPT Database, (1994), UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml, accessed on 15.10.1014

[2] R. E. Banfield; R.E., et. al.: "A Comparison of Decision Tree Ensemble Creation Techniques", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 1 (2007).

[3] S. Benkner, A. Arbona, G. Berti, A. Chiarini, R. Dunlop, G. Engelbrecht, A. F. Frangi, C. M. Friedrich, S. Hanser, P. Hasselmeyer, R. D. Hose, J. Iavindrasana, M. Köhler, L. Lo Iacono, G. Lonsdale, R. Meyer, B. Moore, H. Rajasekaran, P. E. Summers, A. Wöhrer und S. Wood: „@neurIST Infrastructure for Advanced Disease Management through Integration of Heterogeneous Data, Computing, and Complex Processing Services", DOI:10.1109/TITB.2010.2049268, IEEE Transactions on Information Technology in BioMedicine, 14(6), Pages 1365 - 1377, (2010).

[4] B. Berg, P. Silvia: "SAP HANA An Introduction", 2nd edition, GalileoPress, Boston (2013).

[5] J. A. Blackard (Colorado State University): Covertype Database, (1998), UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml, accessed on 15.10.1014

[6] J.-H. Böse, SAP Innovation Center Potsdam, personal communication, Aug. 2013.

[7] L. Breiman: „RF / tools – A Class of Two-eyed Algorithms“, SIAM Workshop, (2003), URL: http://www.stat.berkeley.edu/~breiman/siamtalk2003.pdf , accessed on 11.03.2014.

[8] L. Breiman: “Random Forests”, (1999), URL: http://www.stat.berkeley.edu/~breiman/randomforests-rev.pdf, accessed on 11.03.2014.

[9] Car-Purchase Dataset, (2011), Kaggle, URL: https://www.kaggle.com/c/DontGetKicked, accessed on 15.10.1014

[10] R. Cattral (Carleton Univerity): Pokerhand Database, (2007), UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml, accessed on 15.10.1014

[11] C. Engels: „Basiswissen Business Intelligence.“, W3L Verlag, Witten (2009).

[12] C. Engels, C. Friedrich: „Proposal - Raising the power of Ensemble Techniques“, Proposal to summer 2013 period at the HPI Future Lab, (2013).

[13] C. Engels, C. Friedrich: „Proposal - Follow up & extension activities to *the Raising the power of Ensemble Techniques* project “, Proposal to winter 2013 period at the HPI Future Lab, (2013).

[14] C. Engels, C. Friedrich, D. Müller: „Report - Raising the power of Ensemble Techniques“, Report to summer 2013 period at the HPI Future Lab, (2013).

[15] C. Engels; W. Konen: „Adaptive Hierarchical Forecasting”. Proceedings of the IEEE-IDACCS 2007 Conference, Dortmund (2007).

[16] J. Friedman: „Computational Statistics & Data Analysis”, Volume 38, Issue 4, 28 February 2002, Pages 367–378, (2002), URL: http://dx.doi.org/10.1016/S0167-9473(01)00065-2, accessed on 11.03.2014.

[17] J. Haun, et al.: “Implementing SAP HANA”, 1st edition, Galileo Press, Boston (2013).

[18] D. Johannsen, SAP Innovation Center Potsdam, personal communication, April. 2014.

[19] R. Klopp: “Massively Parallel Processing on HANA”, (2013), URL: http://www.saphana.com/community/blogs/blog/2013/04/22/massively-parallel-processing-on-hana, accessed on 11.03.2014.

[20] D. Müller, C. Engels, C. Friedrich: „Proposal - Performance Optimization of Data Mining Ensemble Algorithms on SAP HANA“, Proposal to summer 2014 period at the HPI Future Lab, (2014).

[21] D. Müller, C. Engels, C. Friedrich: „Report - Follow up & extension activities to *the Raising the power of Ensemble Techniques* project“, Report to winter 2013 period at the HPI Future Lab, (2014).

[22] SAP AG: “SAP HANA Developer Guide (document version: 1.0 – 27.11.2013, SPS 07)”, (2013), URL: http://help.sap.com/hana/SAP_HANA_Developer_Guide_en.pdf, accessed on 11.03.2014.

[23] SAP AG: “What´s New? SAP HANA SPS 07 - SAP HANA Application Function Library (AFL)”, (2013), URL: http://www.saphana.com/servlet/JiveServlet/download/4267-1-12720/What%C2%B4s%20New%20SAP%20HANA%20SPS%2007%20-%20AFL%20Predictive.pdf, accessed on 11.3.2014

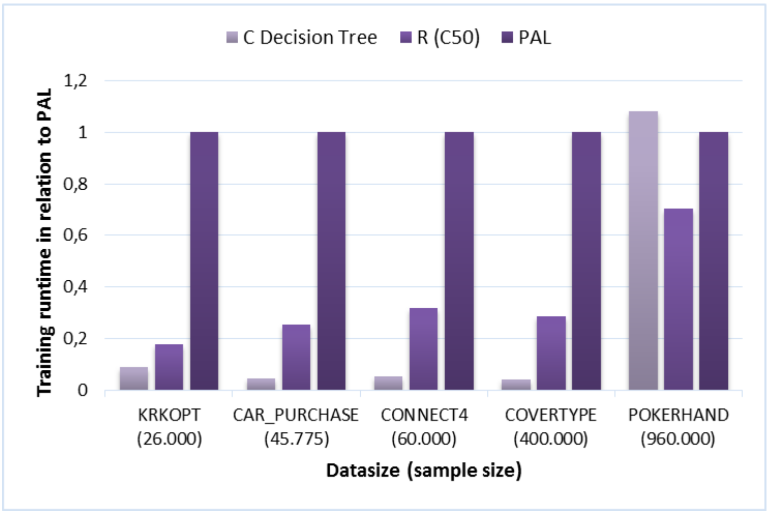
[24] G. Seni, J. Elder: “Ensemble Methods in Data Mining”, Morgan & Claypool, San Rafael, California (2010).

[25] J. Tromp: Connect4 Database, (1995), UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml, accessed on 15.10.1014

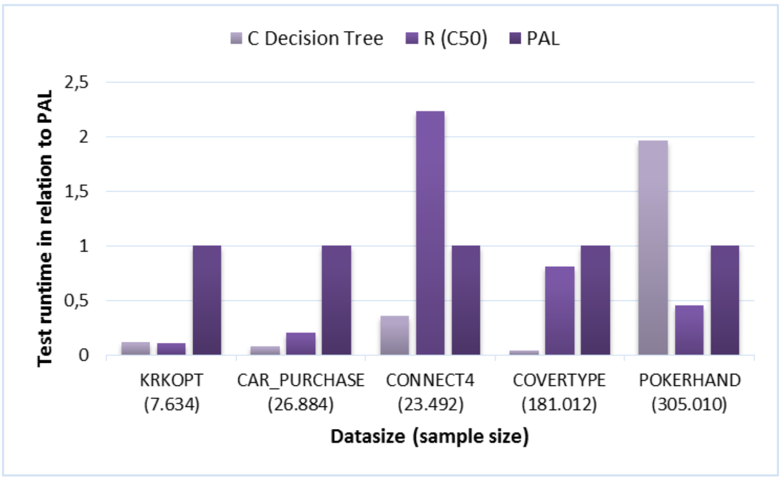
[26] G. Üstünkar; S. Özögür-Akyüz; G. W. Weber; C. M. Friedrich und Y. A. Son, „Selection of Representative SNP Sets for Genome-Wide Association Studies: A Metaheuristic Approach“, DOI:10.1007/s11590-011-0419-7, Optimization Letters, Volume 6(6), Seite 1207-1218, (2012)

Appendix:

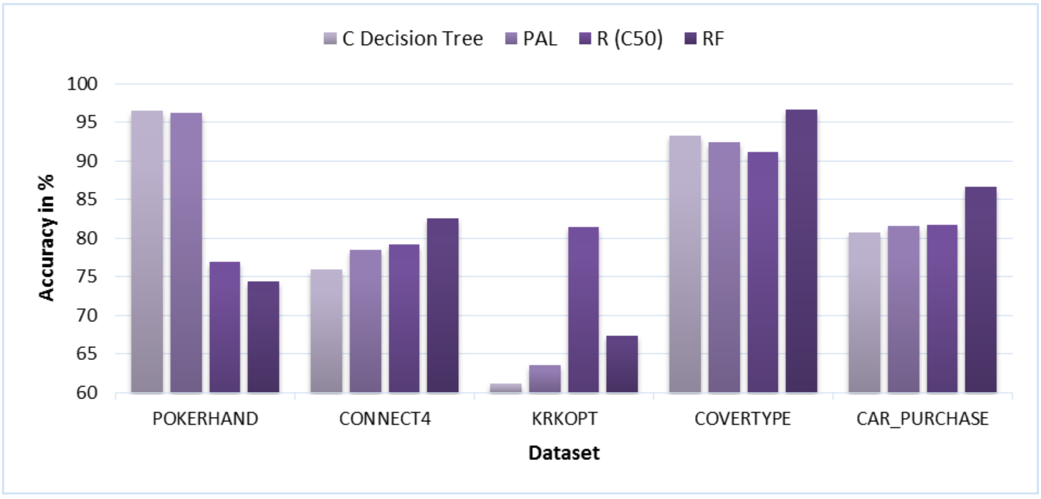
Appendix 1: Training Runtime of C Decision Trees, R (C50) and PAL in Relation to PAL



Appendix 2: Test Runtime of C Decision Trees, R (C50) and PAL in Relation to PAL



Appendix 3: Accuracy of C Decision Trees, PAL, R (C50) and Random Forest



Multi-Facet BPM: Identification, Analysis and Resolution of Resource-Intensive BPM Applications

Tom Thaler, Sharam Dadashnia, Peter Fettke, Peter Loos
Institute for Information Systems (IWi) at the
German Research Center for Artificial Intelligence (DFKI) and
Saarland University
Campus D3 2, 66123 Saarbrücken
{tom.thaler|sharam.dadashnia|peter.fettke|peter.loos}@iwi.dfki.de

Abstract

Within the last years, the information systems research discipline has been faced with more and more resource intensive application scenarios and an increasing amount of data. Taking this development into account, the paper at hand exemplarily addresses three concrete calculation scenarios in order to gain insights on how to utilize a high performance IT infrastructure to solve the corresponding problem statements. Therefore, the concept of architectural prototyping is used as a research approach. This "system under development" made it possible to develop an outstanding algorithm calculating process matches and to evaluate it with the IWi process model corpus. While the work on the two other scenarios (1) state explosion in practice and (2) process mining on Big Data is still in progress, several new interesting application scenarios could be identified.

1 Introduction

The project Multi-Facet BPM aims at addressing new challenges of resource-intensive BPM application scenarios, wherefore techniques of parallel and distributed computing as well as techniques for the handling of Big Data are necessary. In order to gain insights on how to utilize a high performance IT infrastructure, the following scenarios are arranged:

1. Study the behavior of different process model similarity measures by applying them on heterogeneous data sets. Explore the existence of similarities and node correspondences in process models from different domains.
2. Study the state-explosion problem in real applications and investigate the borders of extracting all possible traces of business process models.
3. Process Mining in terms of extracting business process models from log-files on large data foundations with several millions or even billions of records.

In order to handle the mentioned scenarios, a specific research approach is applied, which is described in section 2. Section 3 provides some further information on the different scenarios and presents the accumulated results structured by established approach, research in progress and further research directions. Section 4 provides information on the developed software, while section 5 concludes the preliminary work and gives an outlook on the follow-up project.

2 Research Approach

Within the project, the concept of architectural prototyping is used as a research approach. An architectural prototype is a learning and communication vehicle used to explore and experiment with alternative architectural styles, features and patterns in order to balance different architectural qualities [1]. The main objective is to enable the calculation of the mentioned application scenarios, which is not possible with existing tools. Thus, the architectural prototype is primarily used for getting insights that may otherwise be difficult to obtain before a system is built [1].

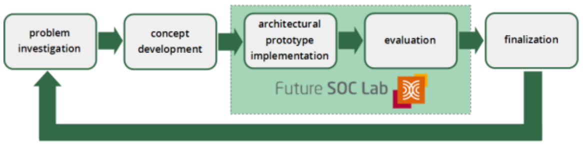


Figure 1: research approach

As shown in figure 1, the research approach is considered as a repeating cycle consisting of five phase. In the first phase, the research problem is investigated and explicated as a problem statement. Within the second phase, a concept for the solution of the problem is developed and, afterwards, implement in terms of an architectural prototype in the third phase. The implemented concept is then evaluated in phase four. At the end of an iteration, it is decided whether a further iteration is necessary or not.

The phases three and four are proceeded on the IT infrastructure provided by the HPI Future SOC Lab consisting of a dedicated blade with 24 cores, 64 GB main memory and Ubuntu as the operating system. The implementation is based on a multi-thread enabled php compilation in terms of a first architectural prototype and on java in terms of further stable implementations.

3 Calculation Scenarios and Accumulated Results

3.1 Established Approach

The first mentioned scenario aims at studying the behavior of different process model similarity measures by applying them on heterogeneous data sets. Similarity measures are necessary for the handling of large process repositories, for compliance analyses or in context of mergers and acquisitions. Calculating process similarities generally requires the availability of node matches, the assignment of node sets of one model to the corresponding node sets of another model [2]. Thereby, the investigated objects range from natural language over graphs to the execution semantics of process models. Since the generation of such matches is an optimization problem with a np-complete complexity, this can be seen as the bottleneck of the whole calculation.

However, the applied research approach made it possible to further develop a process matching algorithm, which outperforms the existing state-of-the-art algorithms in that research field and which was honored with the *Outstanding Matcher Award* at the Process Model Matching Contest 2013 [3]. Only that further development enabled the calculation of node matches between all models of the IWi process model corpus [4]. The concept of the multi-thread implementation of the algorithm (description of the algorithm in [3]) is visualized in figure 2. Two parts, namely the semantic data preparation and the binary mapping extraction could be parallelized. However, only the mapping of the SAP R/3 reference model (604 single models) on itself took about 3.5 days under maximum processor utilization of the blade and a permanent consumption of more than 32 GB main memory (see figure 3).

Within the application scenario, the matches between 2,290 single models with 63,354 nodes overall were calculated which led to more than 2 billion node comparisons and more than 2.6 million models pairs. Some interesting results are the identification of relatively high similarities between the SAP reference model and the Y-CIM reference model [5] with about 42% matched nodes and between the SAP reference model and ITIL [6] with about 36% matched nodes.

3.2 Research in Progress

3.2.1 State Explosion in Practice

The second scenario aims at investigating the theoretical state explosion problem of EPCs, which is primarily caused by the execution semantic of the OR connector. It is tried to answer the question of the relevance of that theoretical problem in real process models. Thus, also for this scenario, the IWi process model corpus is used as a data basis.

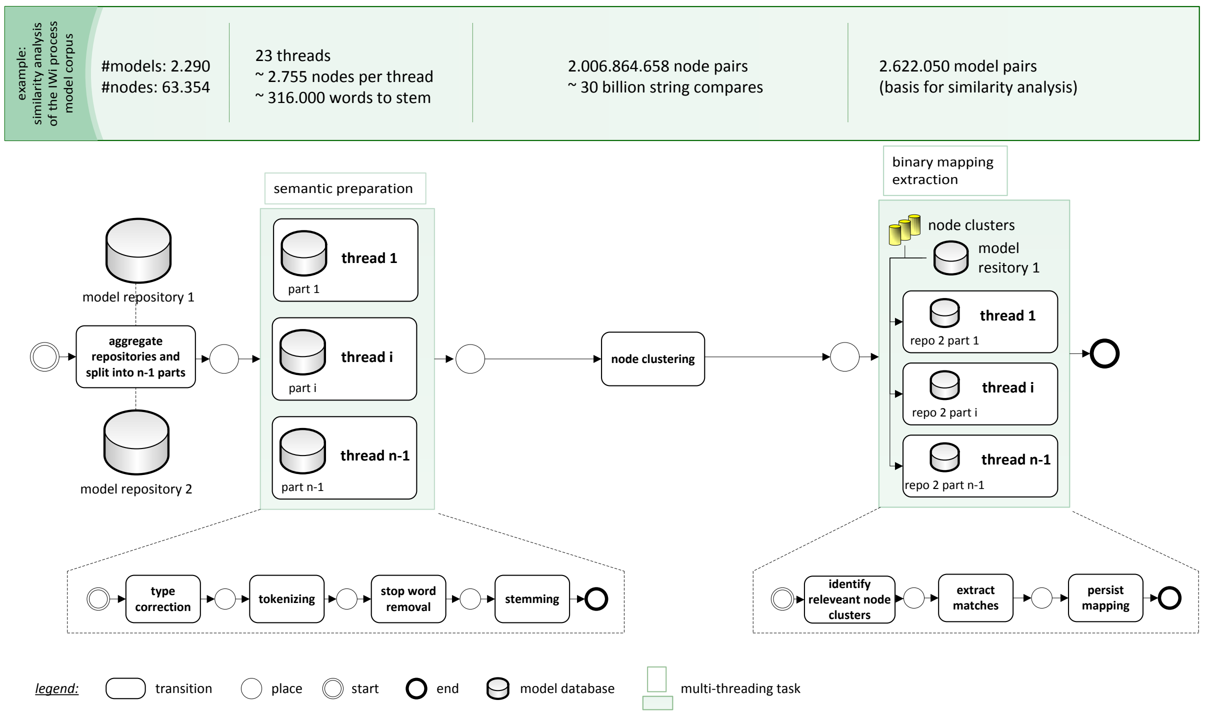
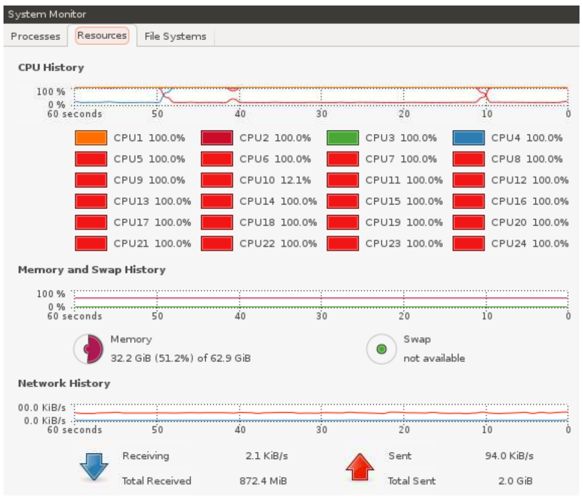


Figure 2: Multi-threaded n-ary semantic cluster matching algorithm (RefMod-Miner/NSCM)


```
System Monitor
Processes   Resources   File Systems

CPU History
CPU1  100.0%   CPU2  100.0%   CPU3  100.0%   CPU4  100.0%
CPU5  100.0%   CPU6  100.0%   CPU7  100.0%   CPU8  100.0%
CPU9  100.0%   CPU10 12.1%    CPU11 100.0%   CPU12 100.0%
CPU13 100.0%   CPU14 100.0%   CPU15 100.0%   CPU16 100.0%
CPU17 100.0%   CPU18 100.0%   CPU19 100.0%   CPU20 100.0%
CPU21 100.0%   CPU22 100.0%   CPU23 100.0%   CPU24 100.0%

Memory and Swap History
Memory                              Swap
32.2 GiB (51.2%) of 62.9 GiB        not available

Network History
Receiving        2.1 KiB/s     Sent         94.0 KiB/s
Total Received   872.4 MiB     Total Sent   2.0 GiB
```

Figure 3: Utilization while matching calculation

A concrete application scenario motivating that research is the implementation, respectively the automatic transformation, of process models to software code. The complexity of an implementation as well as the resulting software product depends (amongst others) on the number of possible execution paths.

Existing concepts addressing that topic are the graph theory in general and the token concept for EPCs and the refined process structure tree (RPST) in particular. In a first step, the token concept for EPCs [7] was implemented and extended using an architectural prototype in order to be able to handle not only models with a single entry and a single exit (SESE) but also multiple entries and multiple exits, which are very common in practice. A further algorithm calculating all possible execution paths based on reachability graphs was implemented.

Within the evaluation of the concept, 926 single models from the model corpus served as input data and it was tried to derive all possible execution paths for all models. As a result, it can be noted that for 86% of the models, all possible execution paths could be calculated within reasonable time (less than 5 minutes). The calculation aborted for 13% of the models because of time exceedance and for 1% of the models because of syntax errors within the models. It should further be noted, that the calculation nearly permanently reserved more than 10 GB of main memory, although it was processed on only one processor core.

In order to also enable the calculation of the missing 13% of the models, it is planned to (1) use the refined process structure tree for a sped up execution path derivation and to (2) identify tasks within the algorithm which can be parallelized, to develop a corresponding concept and to implement that concept. Thus, it is necessary to carry out further iterations of the mentioned lifecycle in section 2.

3.2.2 Process Mining on Big Data

The third scenario aims at developing new process mining algorithms, which are able to handle large amounts of instance data. The main objects of interest are instance logs (how to reduce the mass of data), instance cluster techniques (how to cluster the instance data in a meaningful manner – e.g. in order to generate manageable process models) and process mining techniques (how to design process mining algorithms being able to handle large log files). In that context, process mining algorithms are not only used for process discovery but also for checking the conformance of process executions to the planned processes and the enhancement of existing models with data from real execution.

This scenario is currently in the concept development phase of the presented lifecycle and focuses on the discovery of process models from large log files. It is investigated which possibilities of parallelizing the mining process do exist and which software infrastructure might be meaningful for that task. In a first step, some simple algorithms like the alpha algorithm will be implemented on the Hadoop Map-Reduce Framework to replicate recent research findings [8]. Furthermore, an expansion in terms of an analogue implementation of state-of-the-art algorithms is planned.

The evaluation scenario is covered by the log data of an android app on mobile devices. Thereby, more than 6,000 users generate more than 81 million records with more than 850,000 different tasks every month. Based on that data, it should be analyzed which usage scenarios do exist on mobile devices and whether it is possible to identify different user groups.

3.2.3 Further Research Directions

During the investigation of the mentioned scenarios, some further research directions were identified, which should briefly be introduced in the following:

- Process clustering: In context of mergers and acquisitions, it might be meaningful to cluster similar processes, e.g. in order to compare them.
- Process integration: As a follow-up step of process clustering, e.g. in order to standardize business processes, one possibility is the integration of process models which leads to a new process model aggregating all commonalities and differences of all input models.
- Inductive reference model development: Next to the traditional way of developing reference models in a deductive manner, another way is the inductive development of reference models based on existing models. The idea is to extract the best known practice and use that information to construct a new model.

- Model corpora and catalogs: As described in the previous sections, model collections like specific process model corpora or catalogues serve as adequate input data for several evaluation scenarios. Against that background, the development, analysis and the usage of such corpora are named as a further research direction as only they enable the replicability of research findings.
- Natural language processing: Natural language, e.g. in the form of node labels, process descriptions or meta-data is a very important artifact in business process management. Thus, NLP techniques are e.g. used for the identification of correspondences or for text-to-model / model-to-text transformations.

4 RefMod-Miner

As mentioned in section 2, there are two implementation stages. The first stage (php - solely command line) is primarily used for first drafts and is characterized by a trial and error implementation. This is based on that fact that, in php, types can be neglected in most cases, which leads to first results in very short time. The source code of the existing implementation is publicly available and can be downloaded at https://refmodmine.googlecode.com/svn.

The second stage is developed in Java and covers the more stable research prototype which is called RefMod-Miner. Generally approved approaches are implemented and implementations in an early state are explicitly marked as such. The RefMod-Miner as well as the corresponding documentation and exemplarily use cases are available at http://refmod-miner.dfki.de.

5 Conclusion and Outlook

The project Multi-Facet BPM made a first step towards addressing new requirement regarding the need for high performance computing in the field of business process management. A concept of architectural prototyping was used as a research approach and delivered insights in context of the focused application scenarios, which may otherwise be much more difficult to obtain before a system is built.

The adaption of an already professionally approved technique for the identification of correspondences between nodes of process models led to two important results. First, only that enabled the application of the technique on a large amount of data. Without that further development, it would not be possible to analyze a large model corpus with regard to the contained similarities. Second, it allowed the collection of experiences on how to design an adequate software in order to ideally utilize a high performance IT infrastructure.

However, the investigation of the two other scenarios is still in progress. Indeed, there exists a first results in the area of exploring the state explosion problem of EPCs in practice. Nevertheless, further iterations of the architectural prototyping lifecycle are necessary for a concluding statement.

The field of process mining on Big Data is currently in the phase of concept development and will be further arranged in a follow-up project.

Another result of the project is the identification of five additional scenarios, which will be focused in a later period of HPI Future SOC Lab.

Acknowledgement

The provided high performance IT infrastructure from the HPI allowed the investigation of concrete problem fields in information systems research. The authors thank the HPI Future SOC Lab for the chance of using these resources and appreciate a continuation of the project.

The basic concepts were developed in context of the project “Konzeptionelle, methodische und technische Grundlagen zur induktiven Erstellung von Referenzmodellen (Reference Model Mining)”, which is funded by the Deutsche Forschungsgemeinschaft DFG (GZ LO 752/5-1).

References

[1] Bardram, J. E., Christensen, H. B., Hansen, K. M.: Architectural prototyping: an approach for grounding architectural design and learning, In: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture, IEEE, pp. 15-24, 2004.

[2] Thaler, T., Hake, P., Fettke, P., Loos, P.: Evaluating the Evaluation of Process Matching Technique, In: Leena Suhl; Dennis Kundisch (ed.). Tagungsband der Multikonferenz Wirtschaftsinformatik (MKWI2014), February 26-28, Paderborn, Germany, Universität Paderborn, pp. 1600-1612, 2014.

[3] Cayoglu, U., Dijkman, R., Dumas, M., Fettke, P., Garcia-Banuelos, L., Hake, P., Klinkmüller, C., Leopold, H., Ludwig, A., Loos, P., Mendling, J., Oberweis, A., Schoknecht, A., Sheetritt, E., Thaler, T., Ullrich, M., Weber, I., Weidlich, M.: The Process Model Matching Contest 2013, In: Business Process Management Workshops – BPM 2013 International Workshops, Beijing, China, Springer International, pp. 442-463, 2013.

[4] Thaler, T., Walter, J., Ardalani, P., Fettke, P., Loos, P.: Development and Usage of A Process Model Corpus, In: Proceedings of the 24th International Conference on Information Modelling and Knowledge Bases EJC 2014. June 3-6, Kiel, Germany, 2014.

[5] Scheer, A.-W.: Business Process Engineering - Reference Models for Industrial Enterprises. 2. ed., Berlin, Springer, 1994.

[6] Office of Government Commerce, ITIL - Service Strategy, Service Design, Service Operation, Service

Transition, Continual Service Improvement, Norwich TSO Information & Publishing Solutions, 2010

[7] Mendling, J.: Detection and Prediction of Errors in EPC Business Process Models. Doctoral Thesis, Vienna University of Economics and Business Administration. Vienna, Austria, May 2007.

[8] Evermann, J., Assadipour, G.: Big Data meets Process Mining: Implementing the Alpha Algorithm with Map-Reduce. ACM Symposium on Applied Computing, Gyeongju, Korea, 2014.

Smart Meter Data Map – Conceptual Overview and Improvements

Robert Wehlitz, Robert Kunkel, Marcus Grieger, Bogdan Franczyk
Leipzig University
Information Systems Institute
Grimmaische Str. 12
04109 Leipzig
{wehlitz, kunkel, grieger, franczyk}@wifa.uni-leipzig.de

Abstract

The increasing use of intelligent metering devices will contribute to a steady rise of smart meter data traffic in the upcoming years. Energy consumption values, which are typically gathered by smart meters every 15 minutes, need to be stored and processed on behalf of both energy companies and consumers. Advanced information and communication technologies are the enablers for handling such large amounts of data. In April 2014, we introduced the SMDM concept and a related prototype that uses in-memory technologies for analysing and visualising smart meter data in real-time. In this paper, we summarise the project idea and give an overview of the improvements which have been achieved since then.

1 Introduction

In September 2012, the research project *10.000 Smart Meters in the Model Region Leipzig* was initiated by the research group Smart Energy IT Systems that is associated with the Information Systems Institute at Leipzig University. In the course of this project the researchers collaborate with a local meter operator which was charged with the roll-out of more than 1,000 smart meters in the city of Leipzig in order to establish a research platform (cf. [1][2]).

When the roll-out was finished in late 2013, the researchers started to develop potential use cases for applying real-time analyses of the incoming smart meter data. An early concept of a Smart Meter Data Map (SMDM) and a related software prototype were the results (cf. [3]).

The objective of this paper is to outline the project idea and to describe the improvements of the SMDM project since it was initially introduced in April 2014.

At first, we will give a brief overview of the upcoming challenges that cause the need for advanced information and communication technologies (ICT) within the smart metering domain. After that, the SMDM project, particularly the architecture and the implemented use cases, are going to be described. Finally, the prototype improvements will be summarised and an outlook of the future work will conclude this paper.

2 Challenges

Smart metering comprises all processes of automated capturing, transmission, administration and management of energy consumption and production data [4][5][6][7][8]. A smart meter is an electronic meter that is embedded in a communication network. It is primarily used for remote meter reading. Despite the European Union's directive that at least 80 % of the European consumers shall be equipped with intelligent metering devices by 2020 the current mandatories for smart meter installations in Germany only apply to:[1]

- New buildings,
- Existing buildings undergoing major renovations,
- Consumers with an annual consumption of more than 6,000 kWh,
- Producers with new energy generators having a capacity of 7 kW or more.

The majority of the already conducted smart meter roll-out projects in Germany took place within pilot studies with a relatively small amount of devices [4]. However, the German government increasingly intends a massive roll-out. A cost-benefit analysis conducted by *Ernst&Young* also recommends an expansion of the previously mentioned mandatories for ensuring the economic efficiency of smart metering as a whole [9]. Therefore, the number of installed devices as well as the amount of transmitted energy consumption data will increase.

This data is typically captured every 15 minutes, which means that for each tariff register, 96 data records are delivered by a single smart meter per day.

[1]The mandatories depend on the respective technical feasibility.

Considering the fact that intelligent metering devices enable the use of multiple tariffs per smart meter, the number of captured and transmitted consumption values, as shown in Table 1, might be far larger.

Number of smart meters	Number of data records			
	Daily	Weekly	Monthly	Yearly
100	96×10^2	627×10^2	$2,880\times10^2$	$35,040\times10^2$
1,000	96×10^3	627×10^3	$2,880\times10^3$	$35,040\times10^3$
10,000	96×10^4	627×10^4	$2,880\times10^4$	$35,040\times10^4$
100,000	96×10^5	627×10^5	$2,880\times10^5$	$35,040\times10^5$
1,000,000	96×10^6	627×10^6	$2,880\times10^6$	$35,040\times10^6$

Table 1: Number of data records over time

As a result, the energy industry stakeholders need to be capable of handling such large amounts of data. Energy companies should furthermore use advanced ICT in order to transform the gathered smart meter data into knowledge. This knowledge can be used to develop and offer value-added services in the end.

3 Concept

The SMDM concept deals with potential use cases for real-time analyses of smart meter data to support meter operators in running their smart meter infrastructure. The development of value-added services based on this data shall be supported as well. The SMDM is a web mapping service that analyses and visualises smart meter data depending on spatial data. The relevant information can be accessed via an easy-to-use graphical user interface (GUI).

3.1 Architecture

The SMDM prototype was built on top of the SAP HANA platform. Therefore, we use the SAP HANA Studio as the development environment.

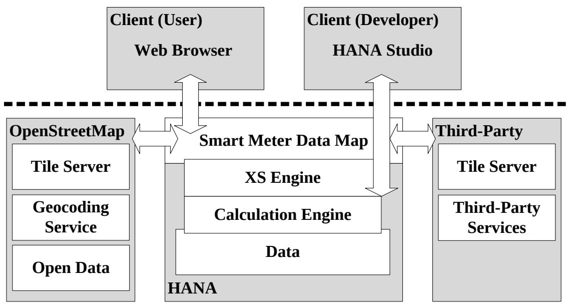


Figure 1: Architecture overview

The gathered energy consumption values and the address data about smart meter locations are stored in the SAP HANA in-memory database. Considering the data processing, graphical calculation views are being used. These views are invoked by web services using *latitude*, *longitude* and *date* as parameters. The web services, based on server-side JavaScript that is interpreted by the XS engine, return the calculated results as JavaScript Object Notation (JSON) objects. These can be easily processed by the SMDM front-end. In order to integrate map data into our web application, we decided to use resources from the *OpenStreetMap* project. This data source seemed to have fewer restrictions regarding the terms of use than *Google Maps*. It provides us with so called tiles. Tiles are graphical map sections that are dynamically loaded into a web page object through JavaScript calls. Web services that allowed us to transform addresses into spatial data and to add further information to the map are available as well, e.g. street names or shop locations. Furthermore, because of the open architecture, the prototype might be extended by third-party services. The SMDM front-end itself is developed by means of a lightweight JavaScript library called *Leaflet*. The *Leaflet* library is used very often for integrating map data from the *OpenStreetMap* project into web pages. It allows the dynamic loading of map tiles and provides a lot of interactive features such as zooming or setting markers. We also used the *jQuery* library for adding our own GUI elements to the map and the *Bootstrap* framework to implement a responsive web design.

3.2 Use Cases

One use case we are considering deals with the visualisation of daily, weekly and monthly electricity consumption with regard to smart meter locations on a web map. Thus, meter operators or energy suppliers are able to analyse the consumption behaviour of their customers in order to offer them energy consulting services.
The identification of weak points within the smart meter infrastructure concerning the data transmission via GPRS represents another use case. This kind of remote meter reading is susceptible to various influencing factors. In the course of our research project, we noticed the daily incoming smart meter data is incomplete because not all smart meters were reachable at any time. We consider it to be useful for meter operators to find out which locations occasionally cause problems, so that the decision-making for fault-clearing actions can be accelerated.
The third use case taken into account refers to the classification of consumption. The respective consumption behaviour of households or companies is classified using specific boundary values for each customer segment. It might be another part of advisory services to show the customer how his consumption is ranked compared to others of the same segment.

3.3 Improvements

In the last couple of months, we optimised the SMDM front-end for different devices and resolutions by implementing a responsive web design and revising the GUI. The smart meter locations are currently being clustered depending on the chosen zoom level to achieve a better clarity. In addition to the listing of daily cumulated consumption values for each smart meter, we added weekly as well as monthly views, and, as shown in Figure 2, implemented graphical charts.

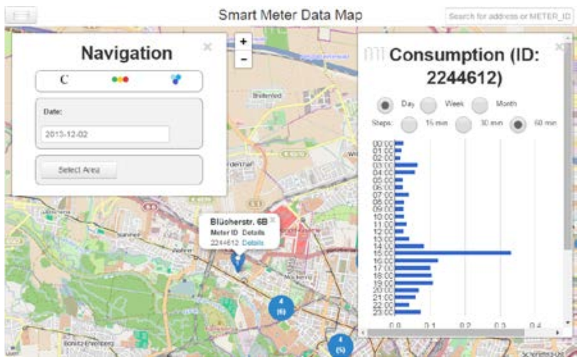


Figure 2: Consumption visualisation via graphical charts

In order to compare the energy consumption of different customers or the statuses of several smart meters with one another we provide the user with the ability to do this by selecting a specific map area (Figure 3).

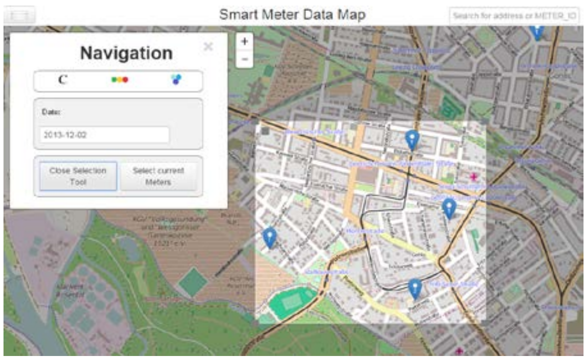


Figure 3: Comparison of smart meter data through map area selection

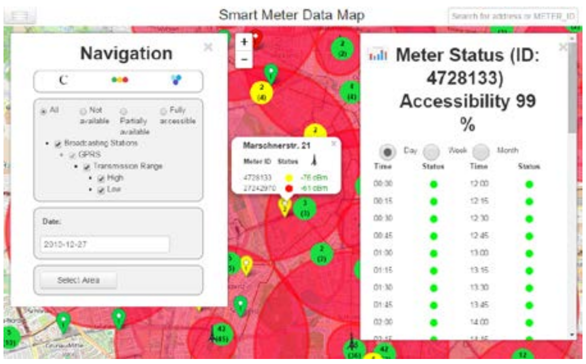


Figure 4: Displaying the GPRS network coverage

We also added display filters to allow the user to look into one or several data aspects, like whether or not gaps in the local GPRS network coverage could cause communication problems (Figure 4), at a time. Beside the new implemented features we fixed major and minor bugs that partially led to contradictions between different consumption views and we achieved an Increase of the overall performance.

4 Conclusion and Outlook

The number of installed smart meters as well as the smart meter data traffic will hugely increase in the upcoming years. Energy industry stakeholders need advanced ICT in order to be capable of handling such large amounts of data and to utilise the contained information. In order to achieve real customer value it is, for some kinds of services, necessary to analyse smart meter data in real-time.

The researchers, therefore, developed and prototypically implemented the extensible concept of a SMDM. This web application supports meter operators in running a smart meter infrastructure and developing value-added services for their customers.

The experiences gained by the researchers indicate that SAP HANA provides the necessary technologies and tools for developing suitable applications that enable large-scale smart metering services.

In the future, we want to consider and implement further use cases that have a significant need for smart meter data processing in real-time. We would also like to demonstrate the prototype to energy companies at our research lab and to implement their ideas and feedback for further improvement.

Acknowledgement

The research project *10.000 Smart Meters in the Model Region Leipzig* is funded by the European Regional Development Fund (ERDF) and the Free State of Saxony (Sächsische Aufbaubank – SAB). During the development of the SMDM, the researchers were provided with a SAP HANA instance by the HPI Future SOC Lab.

References

[1] R. Wehlitz, A. Werner, B. Franczyk: SMIM – A Cloud-based Approach for the Digitisation of Smart Meter Installation Processes. In: Journal of Industrial and Intelligent Information Vol. 2 (3), pp.169-174, 2014.

[2] M. Grieger, A. Werner, R. Wehlitz, J. Pfeifer, B. Franczyk, S. Sprick, T., Ryll: How ICT Could Overcome the Smart Meter Installation Delay - An Assessment of Rollout Experiences. In: Proceedings of the Energy EcoSystems Conference 2013, pp. 71-82, Leipzig, 2013.

[3] R. Wehlitz, A. Werner: In-Memory Computing in Context of Smart Metering. Accepted paper for the HPI Future SOC Lab - Proceedings 2014.

[4] BEAMA Limited: European Smart Metering Alliance – Final Report, http://ec.europa.eu/energy/intelligent/projects/sites/iee-projects/files/projects/documents/esma_publishable_report_en.pdf, accessed: 15/10/2014.

[5] U.C.C. Jagstaidt, J. Kossahl, L.M. Kolbe: Smart Metering Information Management. Business & Information Systems Engineering, vol. 3, no. 5, pp. 323-326. Springer Gabler (2011).

[6] B. Neenan, R.C. Hemphill: Societal Benefits of Smart Metering Investments. The Electricity Journal, vol. 21, no. 8, pp. 32-45. Elsevier, 2008.

[7] J.C.P. Kester, M.J.G. Burgos, J. Parsons: Smart Metering Guide - Energy Saving and the Customer, http://www.ecn.nl/docs/library/report/2011/o11004.pdf, accessed: 15/10/2014.

[8] K.S.K. Weranga, S. Kumarawadu, D.P. Chandima: Smart Metering Design and Applications. Springer, Singapore (2014).

[9] Ernst&Young GmbH: Kosten-Nutzen-Analyse für einen flächendeckenden Einsatz intelligenter Zähler. (Cost-Benefit Analysis for the Comprehensive Use of Smart Metering Systems), http://www.bmwi.de/DE/Mediathek/publikationen,did=586064.html, accessed: 15/10/2014.

Project OliMP: In-Memory Planning with SAP HANA

Benjamin Hemken, Mariska Janz, Jonas Schlemminger and Daniel Stratmann
Department of Very Large Business Applications (VLBA)
Carl von Ossietzky University of Oldenburg

Abstract

While dealing with planning and optimization it is important not only to perform strategic decisions, but also to adequately react to environmental changes. On operational planning and optimization level, the feedback must be provided in real-time and with high certainty. To do it in current situation, the in-memory solution should be used to accelerate the speed of the analysis and provide near realtime response. In details, beside collecting and performing ETL, the project is splitted into 3 major parts: (a) Postum Analysis; (b) Planning and Simulation; (c) Predictive Analytics.

1. Introduction

In-memory computing allows the processing of very large amounts of real-time data in the main memory of the server, so that results from analyzes and transactions are immediately available.

How can the use of in-memory planning and forecasting tools provide a better simulation of future effects of today taking decisions in business questions? How can responsible actors be supported to take transparent decisions rather than subjective or feeling decisions? Can the technological advancement generate economic/business added value?

The goal of our project group "In-Memory Planning with SAP HANA"[1] is to answer the questions above and to obtain the following working skills:

- Software development
- Use of software development tools
- Writing reports and project documentation
- Teamwork and personal soft skills
- Use of SAP HANA to obtain fast in-memory computing
- Use of predictive and analytical tools like SAP Predictive Analysis

[1]For more information about our project group, see Appendix A.

In the following, we will describe our findings and results from the seminar phase, our project idea, the next steps we plan to undertake and the Future SOC Lab resources we are using.

2. Findings from Fundamental Research

The following section presents results and interesting findings from the seminar papers of the project members.

Processes of planning in an enterprise from technical and organizational point of view and their goals

Rapidly changing trends are challenging today's businesses. Therefore business should apply huge effort in order to act and operate on long term prospective and be more sustainable on market. In this sense, "innovation" is seen as key to successful company development and it's one of the core stones on which company's future should rely. Innovations within a company are the main drivers, which lead a company to greater success. Nowadays, one of the innovative factors that companies should focus on is mobility. For example, mobility in the company can extremely enhance employees' flexibility, or company also can merge together data from traffic and data from goods flow to create new insights for its employees. However, the advantages that are brought by the innovations such as mobility are also bringing a lot of new challenges. One of such big challenges is planning. Planning itself always was one of the most difficult and crucial processes in the company, and with new challenges brought by innovations planning is getting more and more complex.

The paper explains basics of the planning process in addition to the implementation within the organizational and operational structure of a company. Examples show how information technologies could be used for enterprise planning.

Opportunities and challenges in the classic corporate planning

Within any corporation there is a huge amount of processes running continuously. All these processes need to be coordinated to fulfill the main goal of the company. Therefore, every corporation needs to plan its goals. Sometimes decisions made by the management are not ideal for the sake of reaching these goals. Hence there are many challenges that need to be considered in economical science. However, corporate planning supports reaching more ideal decision, when applied correctly.
There are many interdependences between the process of planning an economical predictions. Economical predictions can be considered as a part of the planning process. However the process of prediction is highly influencing the process of planning and the other way around. It is not possible to do any predictions without planning and it is not possible to plan without having any idea about possible predictions. Therefore prediction is a chance that improves the company's future, which comes with corporate planning.
One of the most important challenge corporate planning is facing, is the dynamic environment the company is confronted with on a daily basis. The company's decisions need to observe many changes within the market and its supply chain.
Another challenge are the principles of decisions that do influence each other. It is very hard to get to know all the dependencies that a decision is influencing as it is impossible to know every atomic change that got influenced by any rule made by the management. Especially the unknown side effects of decisions lead to a blind spot that need to be considered. However, it is a paradox to consider a blind spot, because it is unknown. It is a challenge to work with this blind sport, keeping in mind it is there.

Integrated business planning

Due to increasing the international competition and globalization, companies must be able to respond quickly so that the integrated business planning is gaining more and more importance in business.
The paper shows the importance of planning in the company and also representing the necessity of BI (Business Intelligence) in an integrated corporate planning. The integrated business planning represents the companys relationship between three levels of hierarchy, namely, the strategic, tactical and operative planning. Strategic planning deals with the fields of activity of the company mainly according to the criteria of the best earnings profile or the total maximum profit point (breakeven point). The tactical planning specifies the frameworks of strategic planning. In operational planning, it may be possible to work with the criterion of the maximum typical profit per year, but it is sensible to execute a multi period optimization through medium-term optimization in a business.

Pros and Cons of InMemory-Computing

With the development of data storage technology, it is possible to keep the data in the main memory of a computer, which had to be stored previously in a separate and slower storage known as disks. In-memory therefore stores all data and information in the main memory of the server and processes them.
Nowadays the volume of data increases very fast. Business companies have to analyse the huge volume of data from different sources, for example, structured data as well as unstructured data from web, Twitter, blogs, etc., in a real time. It should be highlighted that even a few seconds can make a huge difference in the evaluation of profit or loss at the end. To overcome these challenges in memory computing is the best solution, because the data from different sources are stored on the column oriented memory. It is noted that the access times are possible in 100 nanoseconds and an access to hard drives requires about five milliseconds. As the data can be accessed very quickly, it helps to accelerate the decision-making and planning. Therefore managers can grab the market opportunities quickly and also identify the market threats earlier. Finally it helps to increase the performance of the companies. In memory computing is beneficial in the area of telecommunication, media, software development, research, etc., where huge volume of data has to be analysed every day and computing is required in a real time.

Analytical capabilities of SAP HANA: integration with R & Excel

These days, corporations have to work with a continuously growing amount of data. An in-depth analysis under certain criteria on the usual database systems can often lead to a stronger increase in processing time. This is partly attributed to the reading as well as writing speeds of hard disks, as hard disks are generally slower in processing data than the main memory or similar technologies, and partly attributed to the way the data is stored, as storing greater amounts of data in tables can often lead to unnecessary joins that consume a lot of time and resources.
In-memory computing is a possible solution to this problem, because it optimizes both processing speed as well as the way the data is stored in the first place. SAP HANA is the practical implementation of an in-memory solution. Unfortunately the core functions of SAP HANA only allow basic analytical possibilities, but in return allows the external integration of other tools.
In this case the tools R and Excel are used as an example for such an integration. For this SAP HANA uses

the RLANG Parameter, which allows SAP HANA the execution of R-commands. This also allows a wider range as well as more specific commands, that minimize unnecessary action in the analytical process. Moreover it enables the usage of a wider range of visualization and reporting capabilities. In case of Excel SAP HANA uses the MDX-interface to interact with Excel, as both SAP HANA and Microsoft Excel have native support for this interface. This allows the user to combine the familiar functionality of Microsoft Excel, like visualizations and reporting, with the speed and scalability of SAP HANA.

Statistical methods for updating historical data

Corporate planning and decision-making processes are often based on subjective estimations and experience of the responsible actors. There is often a missing background of objective information that could provide a more transparent planning process and thus make decisions comprehensible. To solve this problem, the idea is to provide the responsible actors with objective information based on simulation and prediction of future impact on today taken decisions using large amounts of historical data. This could lead to more transparency in the planning process and provide participants with objective information for taking the right decisions.
This work deals in detail with the prediction algorithms Elman networks and the M5' model tree method. This machine learning methods are used to calculate the desired simulations and predictions based on large sets of historical data with only small knowledge about the relationships of the data. To classify the overall problem, first the Knowledge Discovery in Data Process (KDD) will be presented. The next step is a more detailed description of the sub-process data mining where the different methods like prediction, association-analysis or cluster-analysis are presented. An introduction to time-series analysis follows as well. With this basis, the operation of the prediction algorithms mentioned above can be explained in detail. The paper ends with a short assessment of the two algorithms regarding business issues and an approach to increase the prediction accuracy of both algorithms.

Estimation of the use of predictive methods and tools for SAP (SAP Predictive Analysis

SAP Predictive Analysis is the latest addition to the SAP BusinessObjects BI suite and introduces new functionality to the existing BusinessObjects toolset. Predictive Analysis extends the visual intelligence offerings of SAP Lumira (formerly known as SAP Visual Intelligence) to include new predictive functionality powered by both open source R and SAP-written algorithms.
Predictive models are important because they allow businesses to forecast the outcomes of alternative strategies prior to implementation and determine how to most effectively allocate scarce resources, such as marketing dollars or labor hours.
An overview of the generic predictive modeling process is provided in this paper before going into details about SAP Predictive Analysis modeling engines and the software's features and functionality. It also takes a look at how Predictive Analysis integrates with SAP HANA.

Planning and forecasting tools with their strengths and weaknesses using the example of the systems SEM-BPS, BW-IP and BPC by SAP AG

The paper explains what planning and forecasting tools are, how they are classified and which properties and functions they have.
The basic characteristics of planning systems are listed and an overview of the market segments based on scope and price is presented. The different aspects of planning systems covering their structure and functionality are illustrated with two examples: SAP SEM-BPS and Applix Interactive Planning. The two examples show major similarities and some differences which helps understanding what EPM systems generally cover and how they might differ.
The second part of the paper gives a detailed overview of the different planning systems developed by SAP: from SEM Business Planning and Simulation (SEM-BPS) over BW Integrated Planning (BW-IP) to SAP BusinessObjects Planning and Consolidation (BPC).
The concluding outlook describes the long planned merging of BW-IP and BPC and points out how the new Planning Application Kit (PAK) makes the benefits of SAP HANA available for Business Planning.

Design Thinking - Using Design Thinking in the project group

How do we come up with radically better solutions? This is the core question of innovation and the issue that has been constantly bothering companies. The search for an answer is the way to creating and maintaining a competitive edge. This is precisely the goal of design thinking. It is a method that focuses on the satisfaction of user needs as well as the production of innovations. Many see it as a new form of management philosophy or as an effective remedy for disruptive innovations.
In this regard, design thinking represents the source of a new idea of a moderated iteration of openness of ideas on the one hand and the need to focus on results, based on interdisciplinarity on the other, linked together. In this way it does not only support innovative thinking, but it also helps the user, it makes processes more efficient and products more innovative. It

helps companies become unique, hence competitive. Furthermore, the ideas that design thinking provides turn out to be highly accepted among the target group, because the target group is involved in finding the solution on their own.
The main objective of the paper is first of all to explain what is hidden behind the term of design thinking. Next the insights on the key features and requirements are outlined, as well as their importance for the development of both creative products and innovative ideas in the context of a team. In addition, experts experiences, applicable in the practical use of design thinking in large companies, will be shared.

Classical vs. Agile Software Development – Using Scrum in the project group

The paper is about finding a process model for our project. The first important finding is that the project contains risks related to the complexity of the potential solution. It also has to be recognized that the requirements can be changed during the development process. After that insight, the common characteristics of process models are analyzed and evaluated. Lightweight process models are better usable for projects where requirements change because less documentation has to be adjusted. Incremental and iterative process models help to control complexity and they are more flexible. Agile process models help to get valuable results in shorter time. In this project creativity and flexibility are more important than strict planning. Classical process models are not flexible enough when requirements change. Typically they're bureaucratic. To be able to react to those changes during development process and to focus on results we use a lightweight, incremental and iterative and agile process model. There are two possible process models to choose: software Kanban and SCRUM. In comparison software Kanban is less structured but more flexible. We have decided to use SCRUM because more structured agile process models help new founded teams to gain development productivity in shorter time.
It is planned to work in teams with 4 to 5 people. Each team has one Scrum Master and one Product Owner. Team members are self-dependent and they get the needed authority to solve their tasks by themselves. They're coordinated by arranging SCRUM meetings like Sprint Planning Meetings, Estimation Meetings, Daily Scrums, Sprint Retrospectives and Scrum of Scrums. The Scrum Board helps team members to get an overview of tasks that are open, in progress or done.

3. Project Idea

Data is intended to be processed and analyzed in real time from selected data sources. Center of this approach is to find and analyse so-called events. Events influence the energy capacity utilization of power networks. Picture 1 shows the main aspects. Events can be different in how often they appear, in the kind of influence and in the type of the event (for sample a mass rally or a season change).

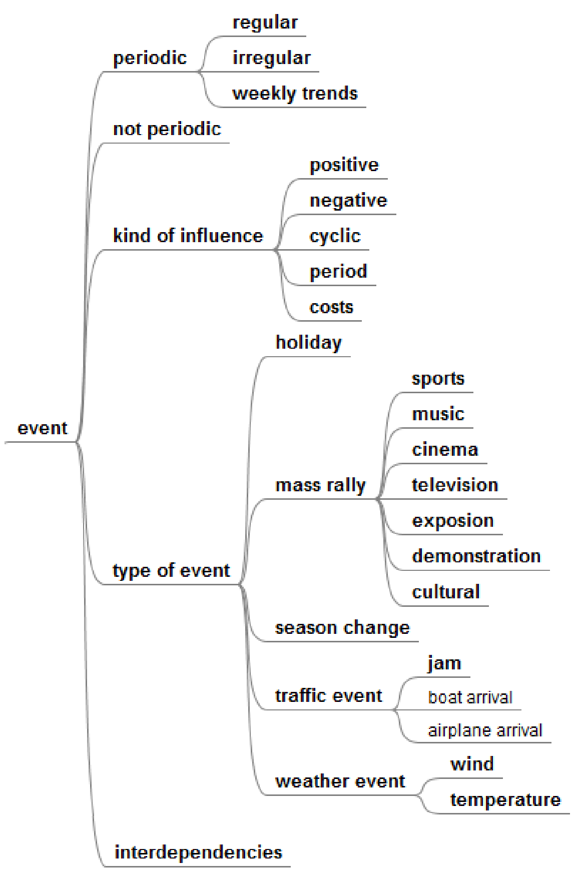


Figure 1. Mind map event

After identifying events, predictions can be created considering previous values. Additional external data sources help forecasting the occurrence of events in future. By the use of in-memory technology, data can be processed faster and decisions can thus be derived from the data faster.

4. Next steps

One of the next important steps is to clarify which data sources are best to be used in order to implement our project idea as described in the previous chapter. In addition to conventional sources, the social networking service Twitter could potentially be a useful data source for gathering information about events and should therefore be analysed. Trends for the possible development of sales data in the energy industry are to be drawn from the amount of tweets. An example is the World Cup: The large amount of tweets for the Cup could allow the conclusion that more electric current is needed at that time. Another example: In summer, when the weather is warm, many tweets on the issue ”go swimming” could allow the inference that less power is required in the houses.

There are two possible strategies to implement the event approach. The first possibility is to analyse and forecast the influence of one specific event type (for example ship arrival). The second possibility is to detect so far unknown events and the possible influence (general approach).

5. Used Future SOC Lab resources

The Future SOC Lab provided us our own SAP HANA instance. We have used it for evaluating and testing first approaches to earn initial practical experience in how to handle the appliance. With the usage of the SAP HANA appliance we have designed a prototypical Twitter interface for loading mass data (tweets) from Twitter into SAP HANA for testing purposes.

The delivered resources are significantly involved in the development process and needed for further tasks. Future SOC Lab delivers the indispensable basis for our project group work.

References

[1] R. Dillerup and R. Stoi. *Unternehmensführung*. Vahlen, 2013.
[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–50, 1996.
[3] R. Fischer. *Unternehmensplanung mit SAP SEM/SAP BW: Operative und strategische Planung mit SEM-/BW-BPS ; [integrierte Unternehmensplanung: Theorie und Praxis, projektorientierte Anwendung von SAP SEM/SAP BW]*. SAP PRESS. Galileo Press, Bonn, 2., aktualisierte edition, 2005.
[4] A. S. Friederike Orths. Dfb: Big data zur fuball-wm. *SAP News center*, 2014.
[5] B. Gloger. *Scrum - Produkte zuverlssig und schnell entwickeln*. Hanser, Wien, 2013.
[6] J. M. Gomez, C. Rautenstrauch, P. Cissek, and B. Grahlher. Einführung in sap business information warehouse.
[7] U. P. HPI. Was ist design thinking?, 2014.
[8] J. Marx Gómez, P. Cissek, and C. Rautenstrauch. *Einführung in Business Intelligence mit SAP NetWeaver 7.0*. Springer Berlin Heidelberg, Berlin und Heidelberg, 2009.
[9] D. Maximini. *Scrum - Einfhrung in der Unternehmenspraxis*. Springer Gabler, Berlin, 2013.
[10] H. Petersohn. *Data Mining:*. Bod Third Party Titles, 2005.
[11] G. Pieroth. *Systematische Prognosefehler in der Unternehmensplanung: Eine ökonomisch-psychologische Analyse*. Schriften des Center for Controlling & Management. Springer Gabler, 2013.
[12] T. Rudny, M. Kaczmarek, and W. Abramowicz. *Analytical Possibilities of SAP HANA: On the Example of Energy Consumption Forecasting*. Advances in Intelligent Systems and Computing. Springer International Publishing, 2014.
[13] SAP. Sap predictive analysis, 2014.
[14] SAP AG. SAP HANA Master Guide, 2014.

Appendix A. The Team and Work Organization

We are 11 students working in a project group at the Business Informatics faculty with five supervisors from the department of Very Large Business Applications (VLBA) at the Carl von Ossietzky University of Oldenburg (Germany). The project group is a compulsory activity for receiving a master's degree at our university.

Our project group called "OliMP[2] - In-Memory Planning with SAP HANA" started at the beginning of April 2014. The project group has a duration of one year and stops in the end of March 2015. Firstly, we did Massive Open Online Courses provided by openSAP and the HPI. The main objective of doing these online courses was to improve our knowledge skills in the SAP HANA technology. Every project member did the following online-courses:

- An Introduction to SAP HANA by Dr. Vishal Sikka
- Introduction to Software Development on SAP HANA by Thomas Jung
- In-Memory Data Management 2013 by Prof. Hasso Plattner

The first phase of our project consisted of specific course works: Every project member wrote a seminar paper about 10-15 pages about a specific topic. These topics are connected to different project perspectives in order to get an all-overview of the working package. Moreover every project member gave a presentation about his topic in order to share his knowledge with the other project members. So, all members of the project group have an equal level of knowledge. The topics of the seminar papers and the project members working on them are as follows:

- **Business-related processes**
  - Processes of planning in an enterprise from technical and organizational point of view and their goals (Igor Perelman)
  - Opportunities and challenges in the classic enterprise planning (Johannes Steffen Scheer)
  - Integrated enterprise planning systems (Farhad El-Yazdin)
- **Basics of SAP HANA and inMemory Computing**
  - Pros and Cons of InMemory-Computing (Rima Adhikari K.C.)
  - Analytical capabilities of SAP HANA: integration with R & Excel? (Eduard Rajski)
- **Planning processes and tools**
  - Statistical methods for updating historical data (Daniel Stratmann)
  - Estimation of the use of predictive methods and tools for SAP (SAP Predictive Analysis) (Jonas Schlemminger)
  - Planning and forecasting tools with their strengths and weaknesses using the example of the systems SEM-BPS, BW-IP and BPC by SAP AG (Abdulmasih Hadaya and Mariska Janz)
- **Development tools and organisation methods of projects**
  - Design Thinking (Ivaylo Ivanov)
  - Classical vs. Agile Software Development: Using Scrum in the project group (Benjamin Hemken)

An important topic in connection with team work and organization was to find an appropriate process model for the project. This project contains risks related to the complexity of the potential solution. Another important fact is that the requirements can be changed during the development process. To be able to react to such changes we use the agile process model SCRUM.

[2]OliMP is a acronym for **Ol**denburger **inM**emory **P**lanung (http://www.ol-imp.de/)

Next Generation Operational Business Intelligence
exploring the example of the bake-off process

Alexander Gossmann
Research Group Information Systems University of Mannheim
agossman@mail.uni-mannheim.de

Abstract

Large retail organizations have to plan customer demands accurately, to achieve customer satisfaction and loyalty. The primary objective is to avoid out-of-shelf situations. On the other hand, losses of perished goods, especially in case of fresh food, have to be minimized. The handling of the trade-off between availability and loss can be dramatically improved by a real-time analytic system. The challenge is to analyze large amounts of data (big data), typically derived from the transactions in the retail process, enhanced by external data, like weather and holidays. Different management groups require specific information with short response times at reasonable costs. Transferred to the retail domain, local store managers are focused on operational decision making, while top management requires a view on the business at a glance.
Both requirements rely on transactional data, whereas the analytic views on this data differ completely. Thus different data mining capabilities in the underlying software system are targeted, especially related to processing masses of transactional data.
The examined software system is a SAP HANA in-memory appliance, which satisfies the aforementioned divergent analytic capabilities, as will be shown in this work.

Introduction (Project Idea)

Operational Business Intelligence is becoming an increasingly important in the field of Business Intelligence, which traditionally was targeting primarily strategic and tactical decision making [1]. The main idea of this project is to show that reporting requirements of all organizational levels (operational and strategic) can be fulfilled by an agile, highly effective data layer, by processing directly operative data. The reason for such architecture is a dramatically decreased complexity in the domain of data warehousing, caused by the traditional ETL process [2]. This requires a powerful and flexible abstraction level of the data layer itself, as well as the appropriate processability of huge amounts of transactional data.
The SAP HANA appliance software is currently released in SPS 07. Important peripheral technologies have been integrated, such as the SAP UI5 Presentation Layer and the SAP Extended Application Services, a lightweight Application Layer. This project proves the tremendous possibilities offered by this architecture which allows a user centric development focus.
This report is organized in the following chapters. The first chapter provides a general overview of the explored use case. In the second chapter the used resources will be explained. The third and fourth chapters contain the current project status and the findings. This Document concludes with an outlook on the future work in the field.

1 Use Case

This project is observing a use case in the field of fast moving goods of a large discount food retail organization. Specifically, the so called bake-off environment is taken into account. Bake-off units reside in each store and are charged with pre-baked pastries based on the expected demand. The trade-off between product availability and loss hereby is extremely high.
From the management point of view, the following user group driven requirements exist: On the one hand, placing orders in the day to day business requires accurate and automated data processing, to increase the quality of the demand forecast. On the other hand, strategic decision makers need a flexible way to drill through the data on different aggregation levels, to achieve a fast reaction time to changing market conditions.
The observation period of two years is considered. The basic population consists of fine grained, minute wise data for thousands of bake-off units, providing all facts related to the bakery process.

1.1 Store Level Requirements

On the store level, the store manager will be supported with matters regarding daily operational demands. Primarily for order recommendations, a certain amount of historical data is taken into account to satisfy the appropriate statistical calculation on time series. Additionally location related and environmental information increases the accuracy of the forecasting model. Environmental variables, like historical weather and holidays, are considered in correlation with historical process data to improve the forecast model. Furthermore, forecasted weather data and upcoming holidays are taken into account for ex-ante data in order to improve the prediction. Model fitting and operational data analysis are being processed ad hoc and on demand by the appropriate store manager.

1.2 Corporate Level Requirements

On the corporate level a 'bird's eye view' is the starting point, where highly aggregated key figures indicate business success or problems. These measures deliver information on a very high level, whereas the reasons for the appearance of these indicators can vary strongly. For accurate decisions it is tremendously important to drill down to the line level, to indicate the reasons for certain business patterns. As the strategic reporting is based on one common data foundation of operational data, navigation to the line level is implicated. It is important that the system is having user satisfying response times, allowing the exploration of a huge amount of data. The application provides the detection of certain patterns and correlations for a more complex classification. For example, the daily availability is analyzed based on certain thresholds, provided by minute wise real time data. To sum up, real-time enabled reporting on strategic level allows reactions on market changes to reach an unprecedented level of effectiveness.

2 Project set up

This chapter illustrates the used technology. After a listing of the architectural resources the appropriate implementation domains will be described in more detail.

2.1 Used Resources

As stated in the introduction, the used architecture is based on the SAP HANA Appliance Software SPS 07 [3].
The presentation layer is built upon the HTML5 based framework SAP UI5. The communication with the SAP HANA In-Memory database and user handling is established through SAP Extended Application Services (XS Engine). Data intensive calculations and data querying are handled by the appropriate APIs in the database, such as the calculation engine (CE), the SQL engine, the Application Function Library (AFL), and particularly the Predictive Analytics Library (PAL) [4]. Additionally, the newly introduced development language River will be used to create an abstraction from the HANA artifacts and enable a higher flexibility of the implementation process.
For time series analysis the Rserve based R integration is used. The data load of CSV formatted transactional data, as well as the data replication and 3rd party are implemented in Java and imported through the JDBC API. The considered 3rd party data consists of weather data, as well as school and public holidays.

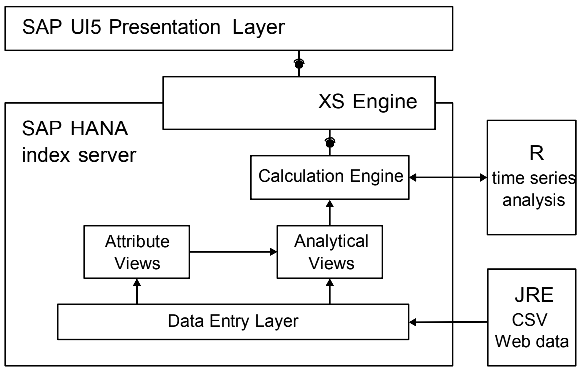


Figure 1 – Architecture

The used architecture, as described in the following chapters, is summarized in Figure 1.

2.2 SAP Front End

As stated above the SAP UI5 constitutes the presentation layer. The Model View Controller pattern is being conducted for front end implementation. For web and mobile versions of the application two different view variants are implemented.
The entry point for a specific user group is the login screen, whereas the different management roles are distinguished by specific HANA user roles. The user groups are differentiated into the strategic, tactical, and operational management role. The strategic and tactical roles are showing the same reports, restricted by the related aggregation level. On the operational level, completely different reports are provided and are mainly focused on daily analysis. Additionally, order recommendations for the next three days are visualized. Each report relies on *one* associated calculation view, described later in the back end section. The selection parameter invoked by a user are handled by OData services, with the belonging data binding, or manually by SQL Script calls.

2.3 SAP HANA Back End

The HANA in-memory database is the core technology of this investigation. In the following section, the data model will be shortly discussed.
The data entry layer consists of two main fact tables. One fact table contains daily aggregated sales related key figures. The second fact table consists of minutely wise measures, derived from the bakery process. This fact table has an expected cardinality of approximately two billion records. In the current implementation this table has rounded 500 million records for the first testing runs. An appropriate partitioning policy, based on time related range partitioning is conducted here, in regards to the expected limit of 2 billion records per table. Several master data tables contain information about stores, regions, products, and holidays. Historical weather data is stored in an appropriate table, whereas weather forecasts will be stored separately and merged daily into the historical data table. All tables are implemented as column tables.

Upon this data entry layer several attribute views are implemented, building up the product, store and regional dimensions. The time dimension is based on the generated time table with minutely level of granularity (M_TIME_DIMENSION), provided by HANA standardly.
The two analytical views contain the fact tables, whereas the daily based fact table is additionally enhanced by the weather and holiday dimensions.
Based on this multidimensional data model eight calculation views are implemented, to satisfy user reporting scenarios about availability, loss, and sales on tactical and strategic level. Additionally one calculation view provides reporting needs on operational level, showing the relevant process information of the current and previous days.
For more sophisticated data mining on the strategic level, as well as data preprocessing of time series data, PAL is used [5]. Specifically the linear regression model function is used to draw trends of dynamically aggregated sales data over time. Further the anomaly detection function is used for outlier detection in daily sales data.

2.4 Peripheral technology

The load of historic and transactional data is handled by a proprietary Java import module, using the JDBC API. The reason for this implementation mainly relies on huge amount of heterogeneous CSV formatted files. Approximately two hundred thousand different types of CSV files have been imported into the HANA database. Therefore, a special bulk load strategy has been used, especially in spite of the insert properties of column oriented tables in the entry layer. Furthermore, historic weather data as well as weather forecast and holiday data is loaded via the JDBC interface of the import module.

Holidays

Both school and public holidays have been downloaded for the past two years, and until the year 2015 from the online portal 'Schulferien.org'. The data is available in the iCal format, and covers all dates for the different states of Germany. These files were loaded into the HANA index server, after conversion into CSV format, using the appropriate build in wizard.

Weather

The historical weather data has been imported from the web weather API ‘wonderground.com’. For model training of the forecast module, the corresponding time interval values of daily, city wise consolidated store data was called from the API. This results in approximately one million JSON files (one file corresponds to one data record), generated by the REST interface, afterwards converted into CSV format and loaded via JDBC of the import module.

Forecast

The demand forecast requirements are primarily developed using the R environment. The appropriate time series are generated on demand, and invoked by a store manager who is responsible for one’s store. As stated in the previous section, the time series are being preprocessed in advance by the PAL framework primarily for performance reasons.
The important outlier detection and handling have been additionally implemented in the R environment, as here more advanced algorithms are available in the R community. Furthermore, two different forecast models have been utilized for comparison reasons. The ARIMA (Auto Regressive Integrated Moving Average) model as well as the ANN (Artificial Neuronal Network) based model have been observed.

2.5 Development environment

The eclipse based HANA Studio is used as the main IDE for the development. In addition to the newly introduced SPS05 features, regarding the ‘HANA development’ perspective, the Java import module is implemented as well.
For usability reasons the following implementation strategy of the R environment has been utilized: Each developer uses a local R runtime for coding R script and model testing. The appropriate time series data is supplied through the ODBC interface. After finalizing a model in R, it is transformed into the HANA environment using the RLANG extension in SQL Script [5].
All artifacts, including java classes, java script, UI5 artifacts and R script have been set under version control with git [6].
The prototype has been completely redesigned regarding the SAP HANA components. The SAP HANA repository has been used for this purpose, to store all relevant design time artifacts like:

- hdb tables
- hdb roles
- procedures

3 Findings

This chapter contains findings on technological as well as on the process level. The findings will be explained analogous to the outline of the previous chapter. In conclusion the outcome of this project will be summarized.

3.1 SAP Front End

Through the tight integration of the controller and model layer, the presentation layer profits of the advantage of a high abstraction level. The data binding feature of the OData services is especially beneficial for strategic and tactical reporting. Hereby flexible data navigation for the top management user is provided, by selecting free time intervals and break-

ing down into different products, regions, or stores. Never the less, the store management invokes an ad hoc data mining and forecasting capability by calling a SQL Script procedure through a Java Script DB connection call.
For parameterization of the calculation views the following limitations exist:

- exclusively input parameters are used, instead of variables for performance reasons
- for input parameters, no ranges are supported and graphical calculation views require additional filter expressions
- character based date parameters work with the OData interface (thus no type safety is provided, implicit cast)

3.2 SAP HANA Back End

In the previous chapter (2.3) the data model has been explained. The biggest column based table contains two years of minutely based transactional data. It has been partitioned by regions. The response times of the appropriate calculation view calls are absolutely satisfying . Nevertheless, the following main restrictions have been experienced which are listed by the appropriate domain:

Predictive Analytics Library

- usability of PAL functions is inconvenient and non-transparent
- restrictive parameterization policy
- very limited exception handling

The restriction in the design time usability especially in the case of PAL, compromises the performance experience of the data analysis.
The AFL framework is in a relatively early stage of maturity and in this project context, only few functions could be utilized. The major functionality in the area of time series analysis has been conducted in the R environment, as stated in the next section.

3.3 Forecast

The demand forecast for each store is calculated on demand. The appropriate time series is generated and sent, together with the belonging weather and holiday information to the R runtime. Hence the sent data frame to R contains daily related time series derivates of additional environmental data to the historic sales data for a certain pastry and store.

Time Series Preprocessing (Outlier Adjustment)

Parameter	Value	Comment
THREAD_NUMBER	4	
GROUP_NUMBER	3	number of clusters k
OUTLIER_DEFINE	1	max distance to cluster center
INIT_TYPE	2	
DISTANCE_LEVEL	2	
MAX_ITERATION	100	

Table 1 - PAL parameterization

As depicted in Table 1, the used PAL function uses a k-means cluster algorithm, whereas GROUP-_NUMBER corresponds to the number of associated clusters (k). Please note that this function detects always one tenth of the underlying number of lags in each time series as outliers. This could not be controlled by the parameter OUTLIER_PERCENTAGE, as expected and thus, limits this function enormously. In the R environment the k-means clustering for outlier detection is used as well. A straightforward approach of outlier handling is used. The majority of given outliers belongs to the class of additive outliers due to public holiday related store closing. The effect is even more significant, the longer a closing period is. Here the precedent open business date shows an abnormal high characteristic. Other outlier classes are by far less significant and cannot be assigned directly to events. Different outlier handling strategies have been tested and implemented, and will be investigated in further proceedings.

ARIMA based forecast

An automated ARIMA model has been implemented in R. The used package is mainly the package 'forecast' [7] available at CRAN (Comprehensive R Archive Network [8]). The automated ARIMA fitting algorithm 'auto.arima()'[9] has been utilized for this project purposes, which is based on the Hyndman et al algorithm [10]. Specifically seasonality, non-stationarity, and time series preprocessing (see outlier handling) required manually coded model adjustment. All additional predictor variables like holidays and weather information could be processed automatically, passed by the 'xreg' matrix parameter.

ANN based forecast

Alternatively to the ARIMA approach, an Artificial Neuronal Network model has been implemented and is especially for capturing automatically nonlinear time series shapes. As expected in the retail context, ANN is supposed to deliver more accurate forecast results [11]. In this use case the 'RSNNS' [12] (Stuttgarter Neural Nets Simulator [13]) package has been utilized. Similarly to the ARIMA model (see above), the independent variables, primarily the daily sales an all additional related variables are used for model fitting.

Summary Forecast with R

One major design change has been made; the R runtime has been transferred to another server. As the previous solutions has been running together with the HANA instance on a virtual machine with 64 cores the parallelized ARIMA based forecast used all available resources on the Linux server. This is not recommended by SAP, as it could harm the processes of the HANA instance itself. Thus, an R runtime on a separated server is obligatory. It can be stated that the performance behaves nearly inversely proportional to the number of cores for the ARIMA algorithm, proposed above. The performance of retrieving, serialization and deserialization of the data frames is nearly neglectable (in the area of ms). Nevertheless, different loading and presentation strategies are required, to provide user acceptance in response times. For instance, asynchronous XSJS calls could be performed, to avoid persisting trained models. This is especially true for ANN algorithms which are only tenuous parallelizable.

3.4 Conclusion

The built prototype was expected to satisfy the reporting requirements of the different stakeholders of information consumption. Although the data analysis capabilities differ throughout the organizational roles of managers, all human recipients expect short response times of a system. With the usage of the SAP HANA appliance software this challenging task could be achieved.

From the development perspective, previously not known effectiveness could be achieved. As all reporting and predictive analytics requirements rely on only a few physical tables, the main effort consists in providing different views on this data. Even more complex measure calculations, like availability and some regression analysis, are processed on the fly. This is a completely new way of designing a reporting system. Compared to traditional ETL based data warehousing tools this saves a lot of manual effort in the loading process. However, this does not imply that the effort for implementing the business logic disappears, merely that the programming paradigm is straightforward. SAP constantly improves the appropriate API functionality (e.g. by introducing the 'HANA Development' perspective.

The capability of providing demand forecasts based on long time series intervals for thousands of stores and different products particularly supports operational decision makers on the day to day business. This could not, or only very difficultly, be achieved with traditional disk based data warehouse approaches focused on aggregated measures. In this prototype, forecast algorithms are performed on demand. This makes sense, as the underlying models require readjustments with each new transaction.

4 Outlook

The upcoming work will focus on four key areas. The first area focuses on enhancing the existing outbound connectivity by implementing routines which automatically load the external data. For this purpose the newly introduced XS Job Scheduling feature of HANA will be used. As the performance of the analysis of data is highly influenced by the quality of the implemented model, the second area focuses on the testing and improvement of the existing analysis models. Having a growing project with growing complexity, it is inevitable to standardize and abstract different objects of the project. For this purpose the third area focuses on the utilizing of River language, as part of SPS07, which has recently been introduced. By using River, the amount of time needed to implement new artefacts and business logic in general can be reduced significantly, which enables test cases that are more flexible. For example new user inputs can be considered very fast in the business logic, when building upon River enabling a more in-time application development. In order to be able to use the major benefits of the in memory technology and the analysis models a high-class graphical interface is needed. This requirement is met by the fourth area which focuses on the implementation of a powerful user interface. In the previous project period, a desktop and mobile version of the application has been implemented by using the UI5 framework. Due to the high abstraction of the data interfaces a new framework has come into focus, which enables the research on additional areas like for example the adaption of the user interface, depending on the type of mobile device, or the adaption, depending on the operating system. This research will be done by using the Sencha Touch framework. To provide further information on the applicability of the developed models in the fresh food industry, an additional use case will be observed. Fresh vegetables and fruits have a sell-by date of only a few days

References

[1] C. White: The Next Generation of Business Intelligence: *Operational BI. DM Review Magazine.* Sybase 2005

[2] H. Plattner: A common database approach for OLTP and OLAP using an in-memory column database. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.* ACM, 2009.

[3] SAP HANA Developer Guide. *http://help.sap.com/hana/SAP_HANA_Developer_Guide_en.pdf*, 19th of March 2014

[4] SAP HANA Predictive Analysis Library (PAL) Reference. *help.sap.com/hana/hana_dev_pal_en.pdf,* 23 of January 2013

[5] SAP HANA R Integration Guide. *help.sap.com/hana/hana_dev_r_emb_en.pdf,* 29th of November 2012

[6] *http://git-scm.com/*

[7] *http://cran.r-project.org/web/packages/forecast/forecast.pdf*

[8] *http://cran.r-project.org/*

[9] *http://otexts.com/fpp/8/7/*

[10] Hyndman, Rob J., and Yeasmin Khandakar. Automatic Time Series for Forecasting: The Forecast Package for R. No. 6/07. Monash University, Department of Econometrics and Business Statistics, 2007.

[11] Doganis, P., Alexandridis, A., Patrinos, P., & Sarimveis, H. (2006). Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *Journal of Food Engineering, 75*(2), 196-204.

[12] *http://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf*

[13] *http://www.ra.cs.uni-tuebingen.de/SNNS/*

Using SAP ERP and SAP BW on SAP HANA: A mixed workload approach

- Final Report -

Galina Baader
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
galina.baader@in.tum.de

Robert Meyer
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
robert.meyer@in.tum.de

Sonja Hecht
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
sonja.hecht@in.tum.de

Helmut Krcmar
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3, 85748 Garching, Germany
krcmar@in.tum.de

Abstract

The purpose of our research project was to evaluate the mixed workload of SAP HANA as it was stated as the vision of SAP [6],[9]. In a first step we have performed performance comparisons of SAP ERP running on SAP HANA vs. IBM DB2 with the help of the Rational Performance Tester from IBM. In a second step, we used the Star Schema Benchmark (SSB) and TPC-DS benchmark to measure the performance of BW on HANA vs. BW on DB2. This examination revealed a performance boost for OLAP queries with SAP HANA, but no significant improvement for OLTP queries. As a last step we used virtual InfoProviders to simulate a mixed workload approach on BW. Although SAP HANA here also rendered better results than IBM DB2, we were not able to implement a true mixed workload with the current BW release provided by SAP, which still requires performing an ETL process. So even when sharing the same database, the workload is not truly mixed as different tables are accessed.

1 Introduction

Most relational database systems rely on disk storage, whereas in-memory databases (IMDB) store data within the main memory. This is not a completely new concept as IMDBs have been around since the 1980s (e.g. TimesTen) [2]. With today's growing capacities, diminishing latencies and sinking costs of DRAM it is possible to run large enterprise applications with data residing solely in main-memory. Along with this comes an increase in computing power through multi-core architectures, which can improve performance by parallelizing computations. Besides leveraging multi-core parallelism with multi-processor systems, technologies like quick path interconnect (QPI) or hyper threading further support the performance gains and real-time data processing that in-memory database systems promise [7].

Common ERP systems rely heavily on Online Transaction Processing (OLTP). OLTP is characterized by departments saving data tuples in row-stores and performing small transactional database updates or data retrieval operations. Analytical and financial planning workloads (referred to as Online Analytical Processing (OLAP)) were separated into own systems to prevent the OLTP systems from being throttled back by time consuming and complex queries [8]. The discussion (as in [6];[9]) of unifying the processing of both OLAP and OLTP workloads in one system supported by a column-oriented organization of data led to questions, whether in-memory databases were capable of handling ad-hoc OLAP queries on transactional data in real-time. To analyze such a mixed workload approach the Hasso Plattner Institute's Future SOC Lab provided us access to four systems: A SAP BW running on SAP HANA; SAP BW running on IBM DB2; SAP ERP running on SAP HANA and SAP ERP running on IBM DB2. The respective database machines had 67 CPUs, 1 TB of RAM, 600 GB of hard disk space and ran SLES 11.2. The SAP HANA DB version provided was 1.00.73.00.389160 (SPS7) and the IBM DB2 version employed was 10.1.0000.

2 Project Goal

The project goal was to test the feasibility (in terms of performance) of SAP's vision to process OLAP and OLTP queries in the same database [6];[9]. To get an idea how to handle the different benchmarks and get an idea of the response times of SAP ERP and SAP BW running on different not-mixed workloads in our setup, the first step was to benchmark SAP ERP running on SAP HANA with SAP ERP running on IBM DB2. Correspondingly, the same was done for SAP BW running on SAP HANA with SAP BW running on IBM DB2. Finally, we then benchmarked SAP BW running on SAP HANA as wellas on IBM DB2 with a mixed workload.

3 Results of comparing SAP ERP on SAP HANA versus SAP ERP on IBM DB2

The first project phase aimed at benchmarking SAP ERP on SAP HANA vs. SAP ERP on IBM DB2. In order to generate controlled load on the database, we employed the "IBM Rational Performance Tester" (RPT). The RPT simulates virtual users performing pre-defined use cases, which have to be recorded first. For this purpose, we used existing teaching cases provided by the SAP UCCs as exemplary workload. In a first step, these use cases have been recorded with the RPT. Afterwards, we ran the recorded use cases with 20 simultaneously active users. These users were represented by 4 so called distributed RPT agents controlled by central RPT host. Each agent was able to run 5 parallel users.

The results of the first project phase indicated that in 3 out of 9 test cases IBM DB2 performed better than SAP HANA within the context of the given workload, while SAP HANA performed slightly better for the others. Taking all test cases into account, SAP HANA needed 283.58ms (SAP HANA) on average while IBM DB2 took 335.29ms of processing time. For further details please refer to the previous project report[1] and to the thesis of Jonas Hueber[2].

4 Results of comparing SAP BW on SAP HANA versus SAP BW on IBM DB2

The results of the comparison BW on HANA and BW on DB2 are presented in the following.

4.1 Selection of a Performance Benchmark

The aim of a performance benchmark is to help system architects in their design decisions. On the other hand they help users to compare different systems within a controlled environment regarding certain system characteristics [1]; [3]. In order to decide which benchmark to choose, we took the criteria established by Gray [4] into account. The four criteria are: (1) relevance, (2) transferability, (3) scalability and (4) simplicity. Furthermore, we had to take the following criteria into account: (5) availability of the benchmark for research purposes and (6) compatibility to SAP BW.
Based on this criteria, we have compared the APB-1, TPC-DS, SSB and SAP BW-EML benchmark. The results are summarized in table 1:

	APB-1	TPC-DS	SSB	SAP BW/EML
Database model	1 fact table, 5 dim. table, snowflake	7 fact table 17 dim.table snowflake	1 fact table 4 dim.table star schema	3 InfoCubes 7 DSOs 16 dimensions extended star schema
Workload model	10 queries	99 queries	13 queries	8 reports
(1)Relevance	Yes	Yes	Yes	Yes
(2)transferability	No	Yes	Yes	No
(3)Scalability	Yes	Yes	Yes	Yes
(4)simplicity	Yes	Yes	Yes	Yes
(5) availability	Yes	Yes	Yes	No
(6)compatibility	No	No	No	Yes

Table 1: Comparison of different benchmarks (Source: Authors' design)

Based on the results shown in table 1, we were not able to use the SAP BW-EML, as it was and still is not publicly available. Furthermore SAP declined our request use the benchmark in our research endeavor. The other benchmarks were not directly devised for SAP BW. However, due to their openness we were able to implement the data model and the queries to the SAP BW system. We chose two benchmarks – the SSB due to its simplicity as well as the TPC-DS due to its actuality paired with its extensive database and workload models.

[1] Project Report handed in on March 2014: "Using SAP ERP and SAP BW on SAP HANA: A mixed workload approach" – Research in Progress" Report.

[2] Jonas Hueber: SAP ERP: Load Testing of In-Memory and Relational Database Systems Using the Example of SAP HANA and IBM DB2; Bachelor Thesis

4.2 TPC-DS benchmark implementation

Implementing the TPC-DS benchmark requires creating a database model. It consists of a logical and physical data model. The logical one is given by the benchmark itself, while the physical has to be adapted. The TPC-DS benchmark represents a retail business with three distribution channels: branch, catalogue and Internet. The underlying schema contains data for business customers, orders and products. Each distribution channel has a corresponding fact table. Each fact table accesses the dimensions customer data, contact data time components and so on. We modeled this by using InfoCubes, Multi-Providers and InfoSets in SAP BW. The TPC-DS benchmark generates 99 different queries in SQL format. As SAP BW to our knowledge has no possibility to run SQL queries directly, they had to be modeled by using the SAP BEx Query Designer, which faces some shortcomings compared to SQL. Therefore, we were not able to implement all 99 queries. As the TPC-DS benchmark proposes queries from four different categories (ad-hoc, reporting, iterative and data mining) two exemplary queries were implemented for each category.

4.3 TPC-DS benchmark results

The TBC-DS benchmark proposes three measurement categories: initial loading, workload and ETL-process, which are discussed in detail below:

(1) Initial Loading
Initial loading denotes the load of the data from flat files on the application server into the data source. The results show that loading data from tables holding 50.000 records or less renders no performance differences between SAP HANA and IBM DB2. The second set of test flat files includes tables holding between 50.000 and 4.000.000 records. Concerning this setup, SAP HANA performed on average 9 times better than IBM DB2. The last test setup included flat files holding 7.000.000 and more records. The measured performance for SAP HANA was much higher than it was for IBM DB2. Regarding the biggest table, "Inventory", a performance gain of 196 times compared to IBM DB2 was realized.
The results for loading data from the DataSource to the InfoProvider were slightly different. Up to a size of roundabout 300.000 records both databases performed with similar speed. However, with an increasing number of records of records SAP HANA continued to show better performance than IBM DB2. Performance increases from 11 up to 57 times faster could be observed. Detailed information can be found in the thesis compiled by Florian Acker[3].

(2) Workload Model
The workload model analyzes the performance of a database under load. Therefore, the queries were run subsequently with the help of process chains. The queries were run in debug-mode in order to be able to deselect the caching option, as the results should not be falsified by cached data. The transaction STAD delivered with SAP NetWeaver was used for gaining statistical insights on database, CPU and total response times.
The results show that SAP HANA performed better than the IBM DB2 in all queries. The in-memory database performed up to 3 to 56 times faster than IBM DB2. However, we could not identify any patterns for which queries the SAP HANA database performed better. Taking the total response time into account, we were even able to observe up to 95% faster query processing. On average the total response time of the tested 8 queries was 57% faster on SAP HANA than on IBM DB2.
As we have conducted each query multiple times, we also analyzed the response times' predictability. To do this we calculated the variation coefficient for both databases. It is calculated by dividing the standard deviation of all measures by the arithmetic mean. The smaller the variation coefficient, the better the query execution time can be predicted. SAP HANA had lower numbers in 6 out of 8 queries. Therefore, we assume that the query execution time is more predictable for SAP HANA.

(3) ETL Process
The ETL-Process is the last part of the performance test. Therefore, we generated three expert routines with update tables. Records from the fact table should be deleted and new records should be updated. The load process revealed minor improvements for SAP HANA. One reason for the small differences might be the small data volume which was used.

4.4 SSB implementation

In a second step we took a different benchmark into account. The SSB is easier to implement and therefore more suitable for the mixed workload comparison. As a first step we have compared SAP BW on SAP HANA vs. SAP BW on IBM DB2.

To implement the Star Schema Benchmark (SSB), we first had to adapt the benchmark's database schema to SAP BW, load the benchmark data into SAP BW (while performing the load test), build the queries as proposed by the SSB and lastly, execute the performance test. An overview of the star schema used is given in appendix figure 1. All implemented queries are available in appendix table 1 and table 2.

[3] Florian Acker: "Performance comparison of relational databases and in-memory databases using the example of SAP NetWeaver BW on IBM DB2 and SAP HANA" Master Thesis

4.5 SSB results

The SSB benchmark aims at analyzing the initial loading and workload test (named load and power test in SSB benchmark).

(1) Load test

Again, the time taken to load data into SAP HANA was faster compared to IBM DB2. On average it showed, that uploading data into SAP HANA was four times faster than on IBM DB2. However, one interesting result was observed: the total space required to store the InfoCube on SAP HANA was 3,5 times higher than on IBM DB2. We assume that this might be due to more advanced compression algorithms employed by IBM DB2.

(2) Performance Tests

We received very similar results to the TPC-DS benchmark. The results are summarized accordingly in Appendix table 2 and table 3. In total 13 queries were executed. SAP HANA was roundabout 16 times faster than the IBM DB2 database. For detailed information please refer to the thesis of Jegan John Brito.[4]

5 The mixed workload approach

This project's goal was to analyze the SAP vision stated in [6];[9] to run a mixed workload consisting of OLAP and OLTP queries on one database. While designing the experiment, we realized that to our knowledge SAP BW and SAP ERP still do not offer any possibilities to run queries on one dataset. Our initial thought process of interweaving the processing of OLAP queries on content constantly streamed from an ERP system into SAP BW still had the caveat, that SAP BW still requires an ETL process. Therefore, we came up with the idea of Virtual Providers. SAP BW features Virtual Provider since release 7.0. Whenever a query is run against a Virtual provider, it connects to the SAP ERP system (or any other connected source) and fetches the latest data. We chose this approach to incorporate real-time data provisioning in our experimental design.

5.1 Designing the experiment

The idea of the mixed workload approach is therefore to use the "virtual" InfoProvider (here Virtual InfoCube) to examine the mixed workload approach. Figure 1 describes the experimental setup. Just as in the SSB, data is first loaded into the Persistent Staging Area (PSA) of the SAP BW system and then transferred to the Star Schema (InfoCube). The benchmark data is physically stored in the first InfoCube. The virtual provider (Virtual InfoCube) on the other hand connects to the SAP ERP system when the query is triggered (1). As queries can always run on one InfoProvider, a MultiProvider is set on top of both InfoCubes. A MultiProvider operates like a union of all InfoCubes contained within it.

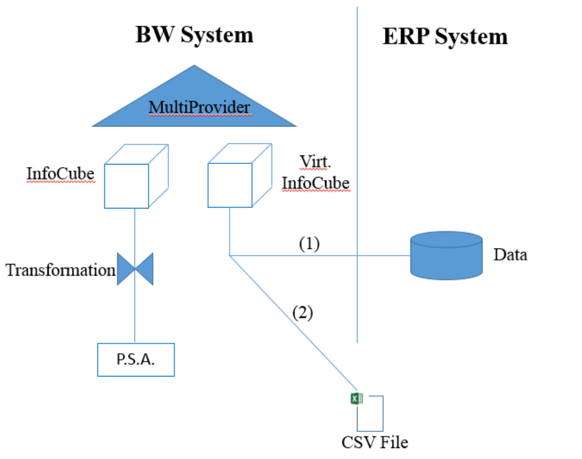


Figure 1: Mixed Workload Approach (Source: Authors' design)

Figure 1 describes the design of the experiment. As in the SSB data is first loaded into the Persistent Staging Area (PSA) in the BW system and then transferred to the Star Schema (InfoCube). The data is physically stored in the first InfoCube. The virtual provider (Virtual InfoCube) on the other hand connects to the ERP system when the query is triggered (1). A query can always run on one InforProvider. Therefore, we set a MultiProvider on top of both InfoCubes. A MultiProvider operates as a union of both InfoCubes. Our first approach was to fetch the data from the SAP ERP system. However, our project partners had some issues hampering the availability of our test environment's systems, so that the SAP ERP systems were not available for a prolonged amount of time. Therefore, we changed the experiment slightly as described in figure 1 part (2). Instead of connecting to a SAP ERP system and fetching the data from there, the Virtual InfoCube connects to a CSV flat file and collects the data from there. In our opinion, this setup can still be considered a mixed workload as the data is fetched from different sources (transactional as well as analytical) while at the same time being uniformly queried on the database level via the used MultiProvider.

[4] Jegan John Brito: „Performance Comparison of a Business Warehouse running on Disk based Relational Database and In-memory Database" Master Thesis

5.2 Experiment implementation

For this experiment we changed the SSB approach to fit our needs. We uploaded the generated data from the CSV flat file into the InfoCube. On the other hand we created exactly the same InfoCube (with the same dimensions and fact table) but made it a virtual provider. To union both InfoCubes, we put a MultiProvider on top. Whenever a Query is executed the data from the real InfoCube is calculated exactly the same way as in our previous benchmark. Simultaneously the virtual InfoCube connects to a CSV flat file and fetches the data. The result of both is shown in the output of the query.

5.3 Experiment results

The experiment results can be reviewed in appendix figure 4 and 5. Comparing the results to the SSB-based performance tests without mixed workloads, we noticed that especially CPU time increased a lot. We assume that establishing a connection and fetching the flat file data is (still) done on the application server side and therefore cannot be enhanced by employing an in-memory database. A further observation was that the maximum size of the CSV flat file could not exceed 250 MB or otherwise the SAP BW system would exit the testing procedure with a shortdump. The detailed results can be reviewed in appendix figure 4 and 5. Overall, we were again able to see a better performance running the SAP BW system on SAP HANA than on IBM DB2.[5]

6 Conclusion

The project goal was to analyse the mixed workload approach of OLAP and OLTP. To reach this aim, we first compared both types of queries isolated by performing a performance comparison test of SAP ERP on SAP HANA vs. SAP ERP on IBM DB2, as well as SAP BW on SAP HANA and SAP BW on IBM DB2. Our results show, that significant performance gains could be realized for the SAP BW system running on SAP HANA. The comparison of SAP ERP on SAP HANA vs. IBM DB2 only showed minor improvements in favour of SAP HANA in terms of performance. On the other hand, we could determine that the storage size of SAP HANA in our setup increased almost fourfold compared to IBM DB2.

Our first approach was to activate the SAP BW module in the standard SAP ERP system to test the mixed workload approach. However, we still had to perform the ETL process, which lead us to the conclusion that a real “mixed workload” on one dataset was not feasible. Hence, we made use of SAP BW’s virtual provider feature. Virtual providers are triggered by the execution of a query against them and connect to the underlying backend systems fetching the latest record delta from them. As we did not have access to our SAP ERP system for several weeks due to technical issues, we changed the experimental setup: Instead of connecting to a SAP ERP system, we fetched the data from a CSV flat file. If the CSV flat file’s size was increased to volumes higher than 250 MB, the system would quit any performance test execution run with a shortdump. As of now, we do not see any possibility to combine SAP ERP and SAP BW systems into one database, as the ETL process still remains. Maybe, this will change with the future adoption of SAP Business Objects tools on SAP HANA. Our results showed a significant performance boost for OLAP queries, but no real improvements regarding OLTP queries. Therefore, we think that it is still a long way to go until the vision mentioned in [6];[9] can become a reality.

[5]Further information: Cagla Sahini: Performance Benchmark of a Mixed Workload on an In-Memory Database compared to the relational Database

References

[1] Darmont, J., Bentayeb, F., & Boussaid, O. (2007). Benchmarking data warehouses. *International Journal of Business Intelligence and Data Mining, 2*(1), 79-104.

[2] DeWitt, D.J.; Katz, R.H.; Olken, F.; Shapiro, L.D.; Stonebraker, M.R.; Wood, D.A. (1984): Implementation techniques for main memory database systems. *Presented at: 1984 ACM SIGMOD international conference on Management of data*, Boston, Massachusetts, p. 1-8.

[3] Frank, M., Poess, M., & Rabl, T. (2012). Efficient update data generation for DBMS benchmarks. *Paper presented at the Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*.

[4] Gray, J. (1993). The Benchmark Handbook: For Database and Transaction Processing Systems: Morgan Kaufmann.

[5] Krueger, J.; Grund, M.; Boissier, M.; Zeier, A.; Plattner, H. (2010): Data structures for mixed workloads in in-memory databases. *Presented at: 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT) 2010*, p. 394-399

[6] Plattner, H.; Zeier, A. (2012): In-Memory Data Management: Technology and Applications, Springer-Verlag, New York 2012.

[7] Plattner, H. (2009): A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database. *Presented at: 2009 ACM SIGMOD International Conference on Management of data*, Providence, Rhode Island, USA, p. 1-7.

[8] Helfen, M.; Trauthwein, H.M. (2011): Testing SAP Solutions. (2 Ed.), Galileo Press, Boston 2011.

[9] Sikka, V. Re-thinking the Performance of Information Processing Systems, Paper presented at the 29th International Conference on Data Engineering (ICDE), p. 8-12, 2013

Appendix

1) Part One: Star Schema benchmark

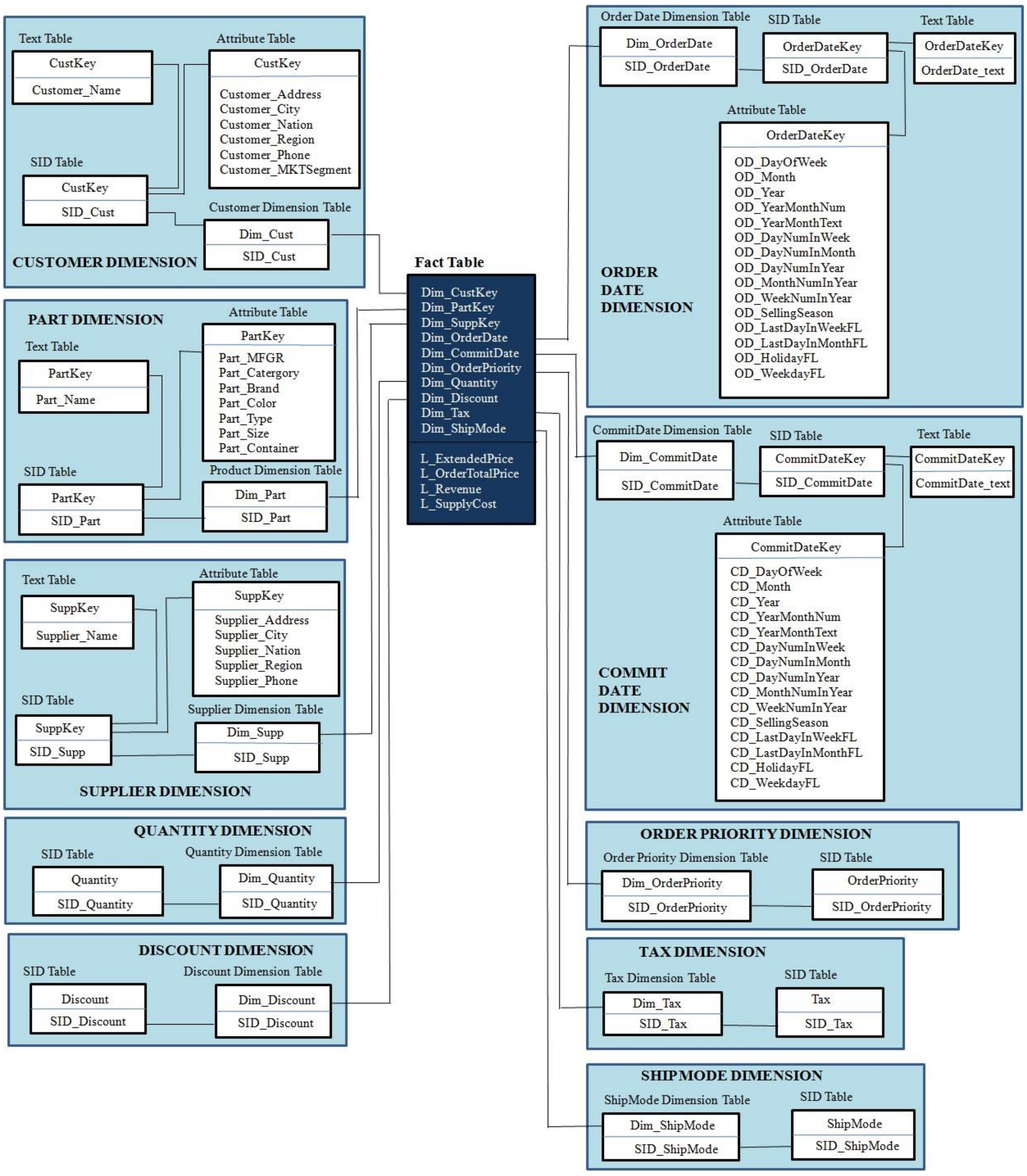


Figure 1: Extended Star Schema for SSB
Source: Own illustration

	Query structure	Where clause costraints			

SSB Query Flight 1	SELECT sum(lo_extendedprice*lo_discount) as rev-enue
	FROM lineorder, date
	WHERE lo_orderdate = d_datekey
	AND d_year = [D_YEAR]
	AND lo_discount BETWEEN [LO_DISCOUNT] - 1 AND [LO_DISCOUNT] + 1
	AND lo_quantity < [LO_QUANTITY];
		Q1.1
		
		D_Year = 1993
		LO_Discount between 1 and 3
		LO_Quantity < 25
			Q1.2
			
			D_YearMonthNum = 199401
			LO_Discount be-tween 4 and 6
			LO_Quantity between 26 and 35
				Q1.3
				
				D_WeekNumInYear = 6
				D_Year = 1994
				LO_Discount between 5 and 7
				LO_Quantity between 26 and 35	

SSB Query Flight 2	SELECT sum(lo_revenue), d_year, p_brand FROM lineorder, date, part, supplier WHERE lo_orderdate = d_datekey AND lo_partkey = p_partkey AND lo_suppkey = s_suppkey AND p_category = [P_CATEGORY] AND s_region = [S_REGION] GROUP BY d_year, p_brand ORDER BY d_year, p_brand;	**Q2.1** P_Category = “MFGR#12” S_Region = “AMERICA”	**Q2.2** P_Brand between “MFGR#2221” and “MFGR#2228” S_Region = “ASIA”	**Q2.3** P_Brand = “MFGR#2339” S_Region = “EUROPE”	
SSB Query Flight 3	SELECT c_nation, s_nation, d_year, sum(lo_revenue) as revenue FROM customer, lineorder, supplier, date WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_orderdate = d_datekey AND c_region = [C_REGION] AND s_region = [S_REGION] AND d_year = [D_YEAR] GROUP BY c_nation, s_nation, d_year ORDER BY d_year ASC, revenue DESC;	**Q3.1** C_Region = “ASIA” S_Region = “ASIA” D_Year >= 1992 and <= 1997	**Q3.2** C_Nation = “UNITED STATES” S_Nation = “UNITED STATES” D_Year >= 1992 and <= 1997	**Q3.3** C_City = “UNITED K11” or “UNITED K15” S_City = “UNITED K11” or “UNITED K15” D_Year >= 1992 and <= 1997	**Q3.4** C_City = “UNITED K11” or “UNITED K15” S_City = “UNITED K11” or “UNITED K15” D_YearMonth = “DEC1997”

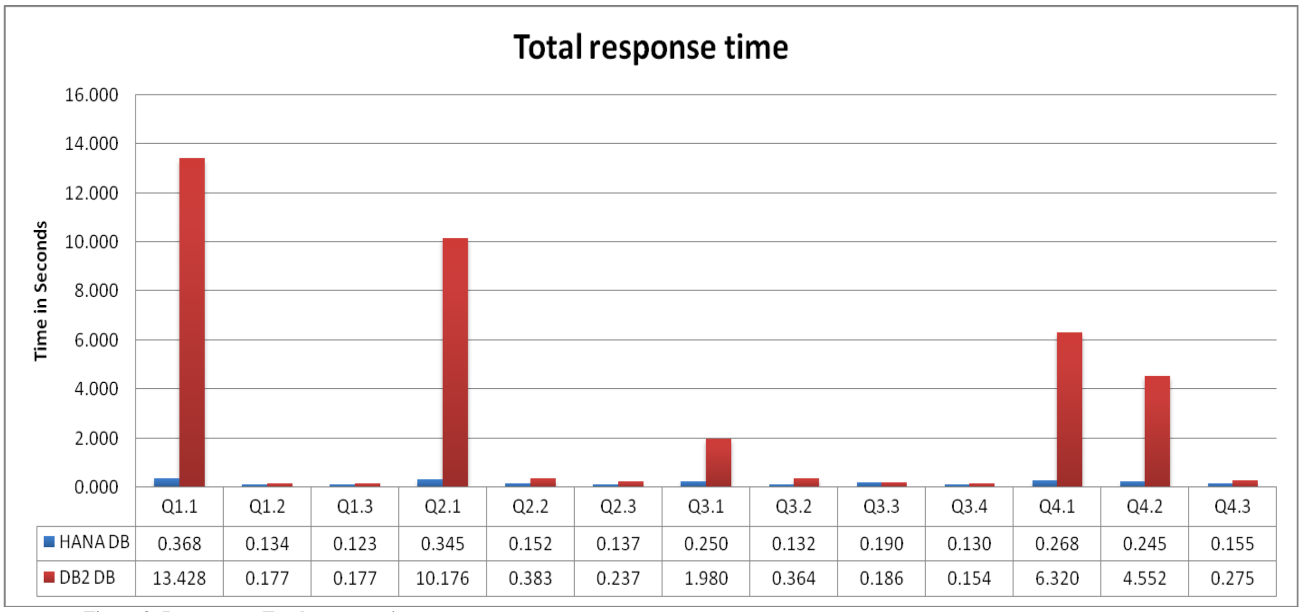
Table 1: SSB implemented Queries
Source: (O'Neil/O'Neil/Chen 2007)

Query Flight	Queries		
SSB Query Flight 4	**Q4.1** SELECT d_year, c_nation, SUM(lo_revenue - lo_supplycost) as profit FROM date, customer, supplier, part, lineorder WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_region = 'AMERICA' AND (p_mfgr = 'MFGR#1' OR p_mfgr = 'MFGR#2') GROUP BY d_year, c_nation ORDER BY d_year, c_nation	**Q4.2** SELECT d_year, s_nation, p_category, SUM(lo_revenue - lo_supplycost) as prof-it FROM date, customer, supplier, part, lineorder WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_region = 'AMERICA' AND (d_year = 1997 or d_year = 1998) AND (p_mfgr = 'MFGR#1' OR p_mfgr = 'MFGR#2') GROUP BY d_year, s_nation, p_category ORDER BY d_year, s_nation, p_category	**Q4.3** SELECT d_year, s_city, p_brand, SUM(lo_revenue - lo_supplycost) as profit FROM date, customer, supplier, part, lineorder WHERE lo_custkey = c_custkey AND lo_suppkey = s_suppkey AND lo_partkey = p_partkey AND lo_orderdate = d_datekey AND c_region = 'AMERICA' AND s_nation = 'UNITED STATES' AND (d_year = 1997 OR d_year = 1998) AND p_category = 'MFGR#14' group by d_year, s_city, p_brand order by d_year, s_city, p_brand

Custom Query Flight 2	**Q5.1 All Regional report**
	
	SELECT s_region, s_nation, s_city, SUM(lo_revenue), SUM(lo_supplycost), SUM(lo_revenue - lo_supplycost) as profit
	FROM date, supplier, lineorder
	WHERE lo_suppkey = s_suppkey
	AND lo_orderdate = d_datekey
	AND d_year =1998
	GROUP BY s_region, s_nation, s_city
	ORDER BY s_region, s_nation, s_city
		Q5.2 Forecast
		
		SELECT s_region, s_nation, s_city, SUM(lo_revenue) *110/100 as FC_1999_revenue
		FROM date, supplier, lineorder
		WHERE lo_suppkey = s_suppkey
		AND lo_orderdate = d_datekey
		AND d_year =1998
		GROUP BY s_region, s_nation, s_city
		ORDER BY s_region, s_nation, s_city	

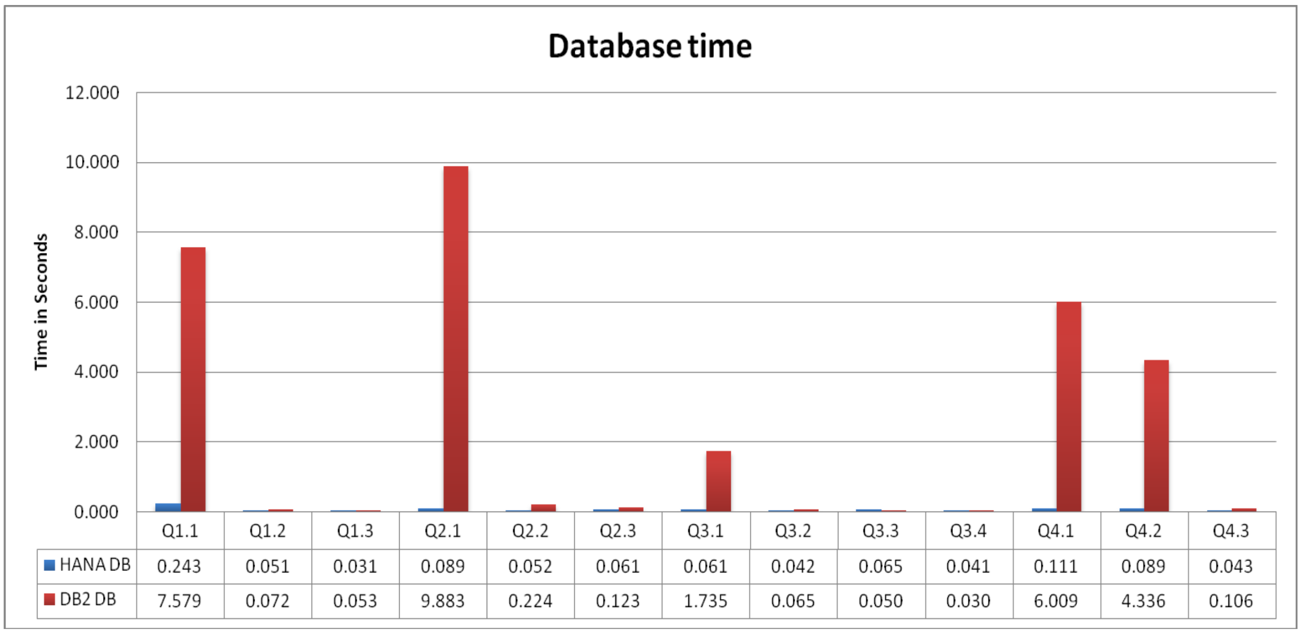
Table 2: SSB implemented Queries
Source: (O'Neil/O'Neil/Chen 2007)

Results – Star Scheme Benchmark



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
🟦 HANA DB	0.368	0.134	0.123	0.345	0.152	0.137	0.250	0.132	0.190	0.130	0.268	0.245	0.155
🟥 DB2 DB	13.428	0.177	0.177	10.176	0.383	0.237	1.980	0.364	0.186	0.154	6.320	4.552	0.275

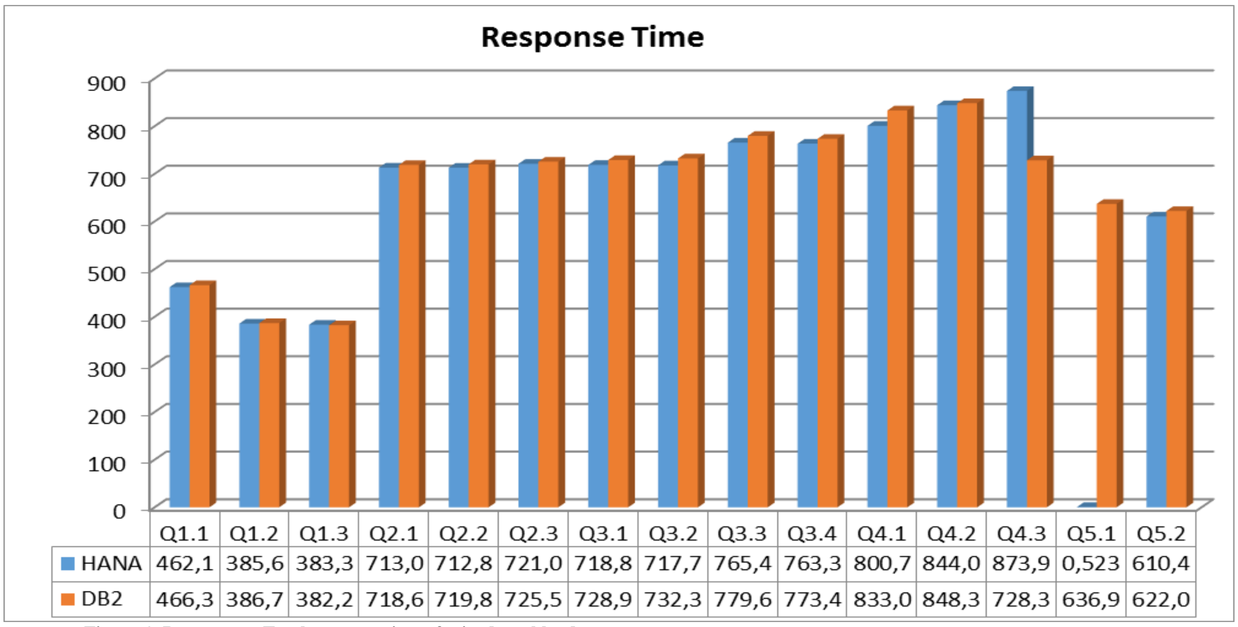
Figure 2: Power test - Total response time
Source: Own illustration



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
🟦 HANA DB	0.243	0.051	0.031	0.089	0.052	0.061	0.061	0.042	0.065	0.041	0.111	0.089	0.043
🟥 DB2 DB	7.579	0.072	0.053	9.883	0.224	0.123	1.735	0.065	0.050	0.030	6.009	4.336	0.106

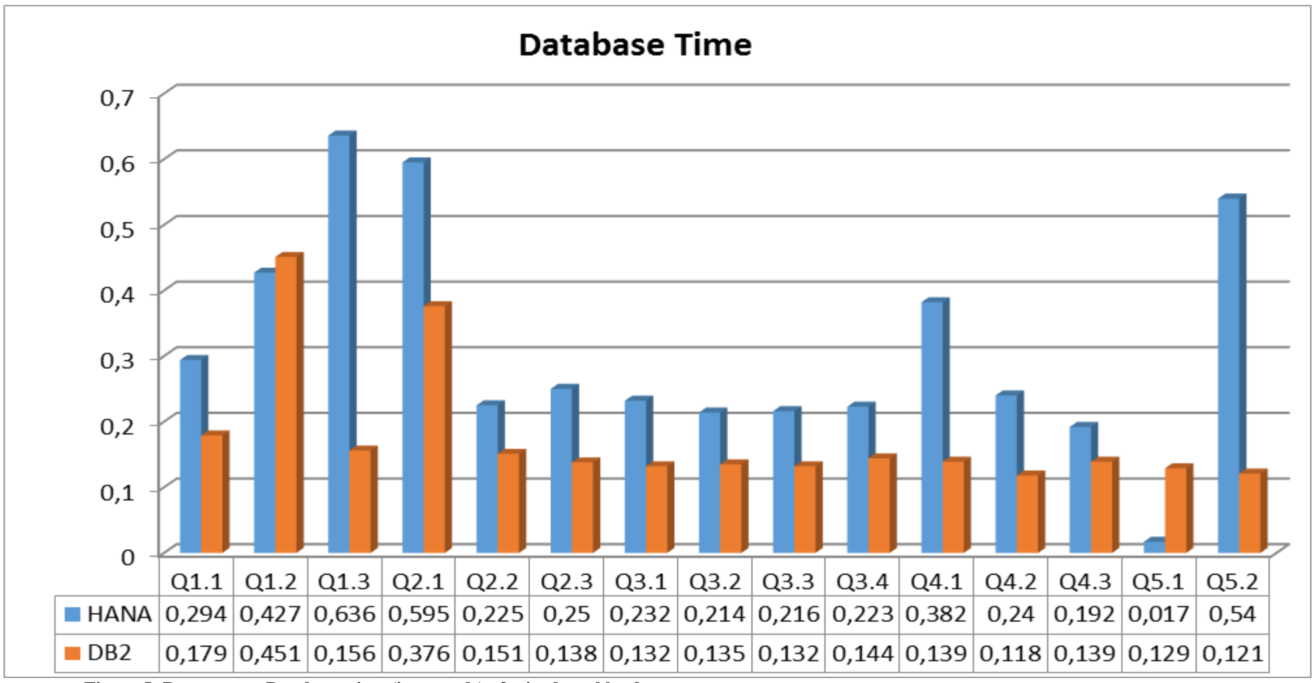
Figure 3: Power test - Database time
Source: Own illustration

2) Part 2: Results of the mixed Workload Benchmark – same Queries implemented



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Q5.1	Q5.2
■ HANA	462,1	385,6	383,3	713,0	712,8	721,0	718,8	717,7	765,4	763,3	800,7	844,0	873,9	0,523	610,4
■ DB2	466,3	386,7	382,2	718,6	719,8	725,5	728,9	732,3	779,6	773,4	833,0	848,3	728,3	636,9	622,0

Figure 4: Power test - Total response time of mixed workload
Source: Own illustration



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Q5.1	Q5.2
🟦 HANA	0,294	0,427	0,636	0,595	0,225	0,25	0,232	0,214	0,216	0,223	0,382	0,24	0,192	0,017	0,54
🟧 DB2	0,179	0,451	0,156	0,376	0,151	0,138	0,132	0,135	0,132	0,144	0,139	0,118	0,139	0,129	0,121

Figure 5: Power test – Database time (in seconds) of mixed workload
Source: Own illustration

Regional climate simulations for West Africa: optimization of input bias correction methods

Dominikus Heinzeller[1]
[1]Institute of Meteorology and Climate Research, Karlsruhe Institute of Technology
Kreuzeckbahnstr. 19, 82467 Garmisch-P.
heinzeller@kit.edu

Harald Kunstmann[1,2]
[2]Department of Geography
Augsburg University
86135 Augsburg
harald.kunstmann@kit.edu

Abstract

Regional climate simulations are valuable tools to study climate change on local scales, yet do they often carry large biases. These stem from the bias of the regional model itself and from the bias of the driving global model. In this project, we developed a program to correct the global model data prior to ingesting it into the regional climate model. In a first step, two different algorithms favored by the climate modeling community were implemented in a fully parallelized Python program. In a second step, we added a complete suite of unit tests to allow for easy and safe further development of the code. We also changed the internal structure of the code from a simple, massively parallel to a queuing-system based program. This allows us to run the bias correction code on machines with less random access memory than the rx600s5-2 blade on which we developed the code initially. Using regional climate simulations over West Africa, we compare the effects of both methods to the uncorrected input data. We found that both methods have advantages and disadvantages, depending on the question one tries to answer.

1. Introduction

West Africa is mostly covered by semi-arid regions with a strong variability in rainfall on intra-seasonal, inter-annual and inter-decadal time scales. This makes it a region highly vulnerable to climate change due to a very low adaptive capacity. Conversely, the West African monsoon precipitation response to future anthropogenic climate change is highly uncertain due to a large spread among the climate projections [5].
In West Africa, climate change projections have often been derived using global circulation models (GCMs). These are limited by their coarse grid spacing and often have problems in representing accurately the main West African Summer Monsoon (WASM) features [11]. Regional climate models (RCMs) are limited area models applied at higher resolution than GCMs and driven by GCM data at the lateral boundaries. The increase in resolution allows for a better representation of fine-scale forcing and land surface heterogeneities, important aspects of the physical response governing local and regional climate change signals [8].

Yet, a common problem of regional climate simulations are biases in physical quantities that limits their accuracy. These biases are of two origins, namely the bias introduced by the regional climate model itself, and the bias inherent in the driving GCM data. The bias of the RCM can be reduced by a suitable model configuration, derived from control runs using re-analysis data as lateral boundary conditions. The GCM bias on the other hand needs to be dealt with prior to ingesting it into the regional model.

In this project, we implemented two different bias correction algorithms and conducted regional climate simulations at 18km resolution over West Africa using the so-obtained data as lateral boundary conditions. The large amount of GCM data and the complexity of the algorithms required an efficient implementation, which we accomplished using Python and the in-memory NoSQL database Redis.

In a first step (application period October 2013 to March 2014), we implemented a simple, massively parallel version of the bias correction code. Thereby we took advantage of the huge amount of random access memory available on the rx600s5-2 blade. As a consequence, the minimum requirement to run the bias correction code were significant and forbid an application on our own machines. In a second step (period April 2014 to September 2014), we added a suite of unit tests to the code, which allows for an easy and safe further development of the code. We modified the internal structure and implemented a queuing to system reduce the requirements on random access memory while maintaining a comparable performance.

2. Bias correction methods

Two concurring approaches are currently favored among the climate modeling community and a clear consensus has not been found yet. Both methods rely on re-analysis data as reference ("truth field") to correct the global model. Here, we used the ERA-Interim re-analysis [4] as reference (REA) data, and the MPI-ESM Echam6 [9] as GCM data (see Fig. 1).

Pseudo-global warming method (PGW) [7] In this approach, model differences are calculated between a ten-year period at present and a ten-year period in the future from a GCM for each month for temperature, humidity, geopotential height and wind. These differences are then added to a current climate re-analysis to obtain a warming signal. This approach allows one to see how "current weather" would look like in a future climate, rather than to detect large degrees of changes to the atmospheric circulation patterns.

Perturbed average climate approach (PAC) [1] Here, 6-hourly GCM and re-analysis data for a ten-year reference and a ten-year application period are broken down into an average annual cycle plus a perturbation term. The revised GCM data for the application period are then constructed from the average annual cycle of the re-analysis data and the perturbation term of the GCM data. This method attempts to allow one to look at changes in circulation *and* in thermodynamic variables.

3. Numerical approach

In a first step (application period October 2013 to March 2014), we implemented a straightforward, massively parallel version of both algorithms in Python. Initially we planned to realize the parallelization using shared-memory threads. We found that the Python threading module is not suitable for this purpose due to a global interpreter lock problem in Python [3]. The parallelization using Python multiprocessing on the other hand is limited to private memory use. To overcome the problem of communication between the individual processes, a parallelized Redis database (`http://redis.io`) was introduced in the code.
For practical reasons, the global climate simulation data and re-analysis data is split into several files for each of the two decades: two files containing slowly changing 2D data (500Mb and 4.5Gb), and one file containing 6-hourly 3D data (30Gb). In the original code, data was read in parallel from all files and for all variables at the same time. This corresponds to nine parallel threads with very different load, since two of the files are negligible in size compared to the third one. The major bottleneck turned out to be writing the bias-corrected data to disk, which implied a huge memory consumption (more than 200Gb for reference/application periods of 10 years). It was therefore impossible to run the code on our own machines. Our plan for the second application period was to first implement a unit testing suite for the entire code base to facilitate further development. We took advantage of the Python unit testing framework, which greatly facilitates the implementation. In total, 22 unit tests and 2 end-to-end tests were added to the code.
Next, we modified the internal structure of the code to reduce the memory requirements. We added an additional Redis database server, administered by a separate thread, which contains a control database. This control database acts as a queuing system for the Python multiprocessing threads.
In this realization, one thread is started for each variable contained in either of the three files. While all write threads (currently only three in total, one for each file) is checking the queue for data ready to combine and write to disk, only a certain number of read threads is admitted at the same time (Fig. 2). This is controlled at runtime by a parameter (maxdbsize, maximum database size) in a configuration file. This modification greatly reduces the memory requirements while maintaining nearly the same performance as the original code (see Table 1 and Fig. 3). However, in particular from Fig. 3 it is clear that the main bottleneck still lies in writing the bias-corrects data to disk. This will be addressed in the future by replacing the 3 polling Python threads by a larger number of polling C/C++ threads. These C/C++ threads will be able to write to the output NetCDF files in parallel through the NetCDF C API.

Table 1. Runtime performance

Maxdbsize	Period	Peak mem	Runtime
no limit	2×30 days	2519Mb	75s
10	2×30 days	1229Mb	76s
3	2×30 days	833Mb	82s
1	2×30 days	781Mb	119s
no limit	2×1 year	29.2Gb	15m 22s
1000	2×1 year	26.8Gb	14m 50s
10	2×1 year	9.42Gb	14m 56s
5	2×1 year	6.79Gb	15m 05s
no limit	2×10 years	230Gb	231m 21s
1000	2×10 years	180Gb	229m 02s
20	2×10 years	110Gb	221m 38s
10	2×10 years	75Gb	216m 12s

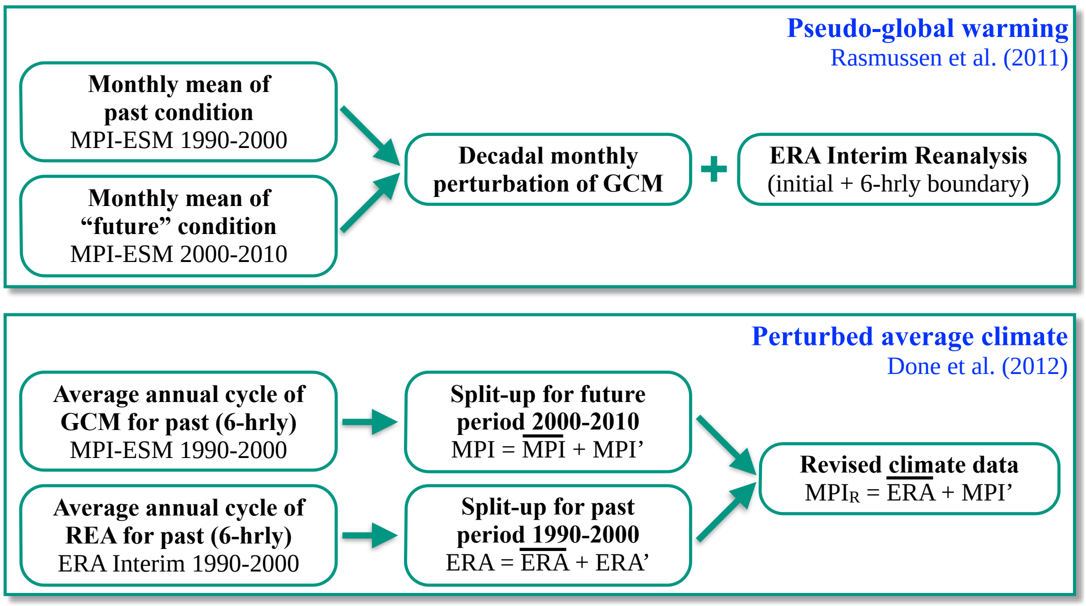

Pseudo-global warming
Rasmussen et al. (2011)
Monthly mean of past condition MPI-ESM 1990-2000
Monthly mean of "future" condition MPI-ESM 2000-2010
Decadal monthly perturbation of GCM
+
ERA Interim Reanalysis (initial + 6-hrly boundary)
Perturbed average climate
Done et al. (2012)
Average annual cycle of GCM for past (6-hrly) MPI-ESM 1990-2000
Average annual cycle of REA for past (6-hrly) ERA Interim 1990-2000
Split-up for future period 2000-2010 MPI = $\overline{MPI}$ + MPI'
Split-up for past period 1990-2000 ERA = $\overline{ERA}$ + ERA'
Revised climate data MPI_R = $\overline{ERA}$ + MPI'

Figure 1. Bias correction methods

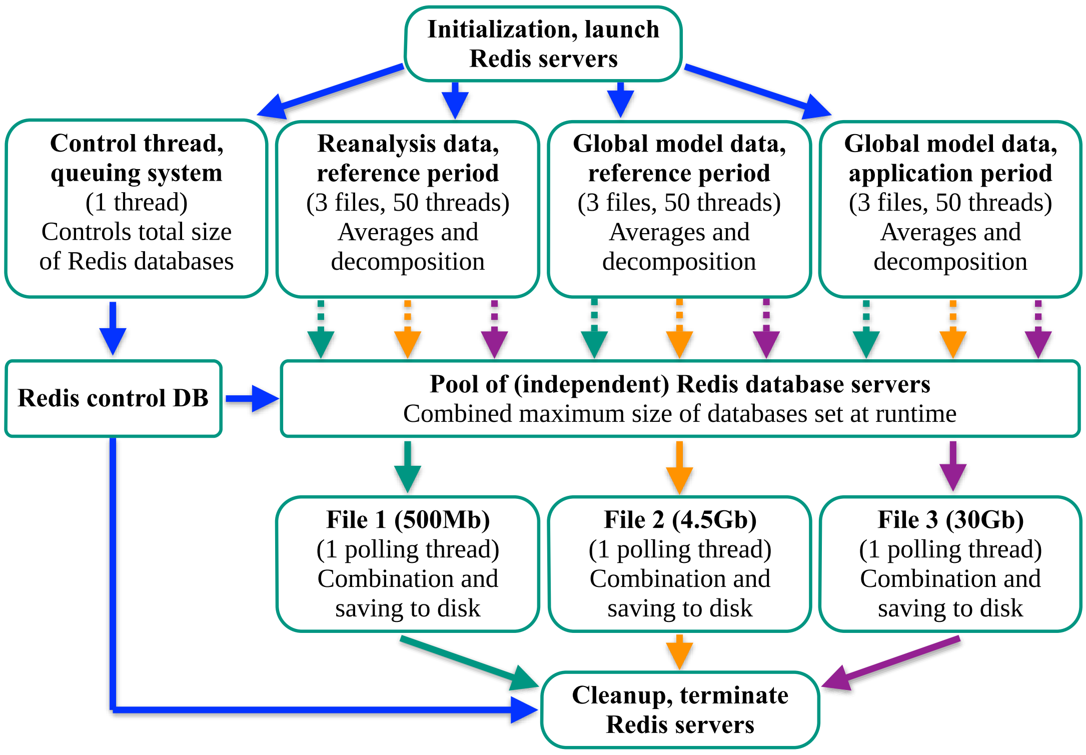

Initialization, launch Redis servers
Control thread, queuing system (1 thread) Controls total size of Redis databases
Reanalysis data, reference period (3 files, 50 threads) Averages and decomposition
Global model data, reference period (3 files, 50 threads) Averages and decomposition
Global model data, application period (3 files, 50 threads) Averages and decomposition
Redis control DB
Pool of (independent) Redis database servers Combined maximum size of databases set at runtime
File 1 (500Mb) (1 polling thread) Combination and saving to disk
File 2 (4.5Gb) (1 polling thread) Combination and saving to disk
File 3 (30Gb) (1 polling thread) Combination and saving to disk
Cleanup, terminate Redis servers

Figure 2. Flowchart of parallelization

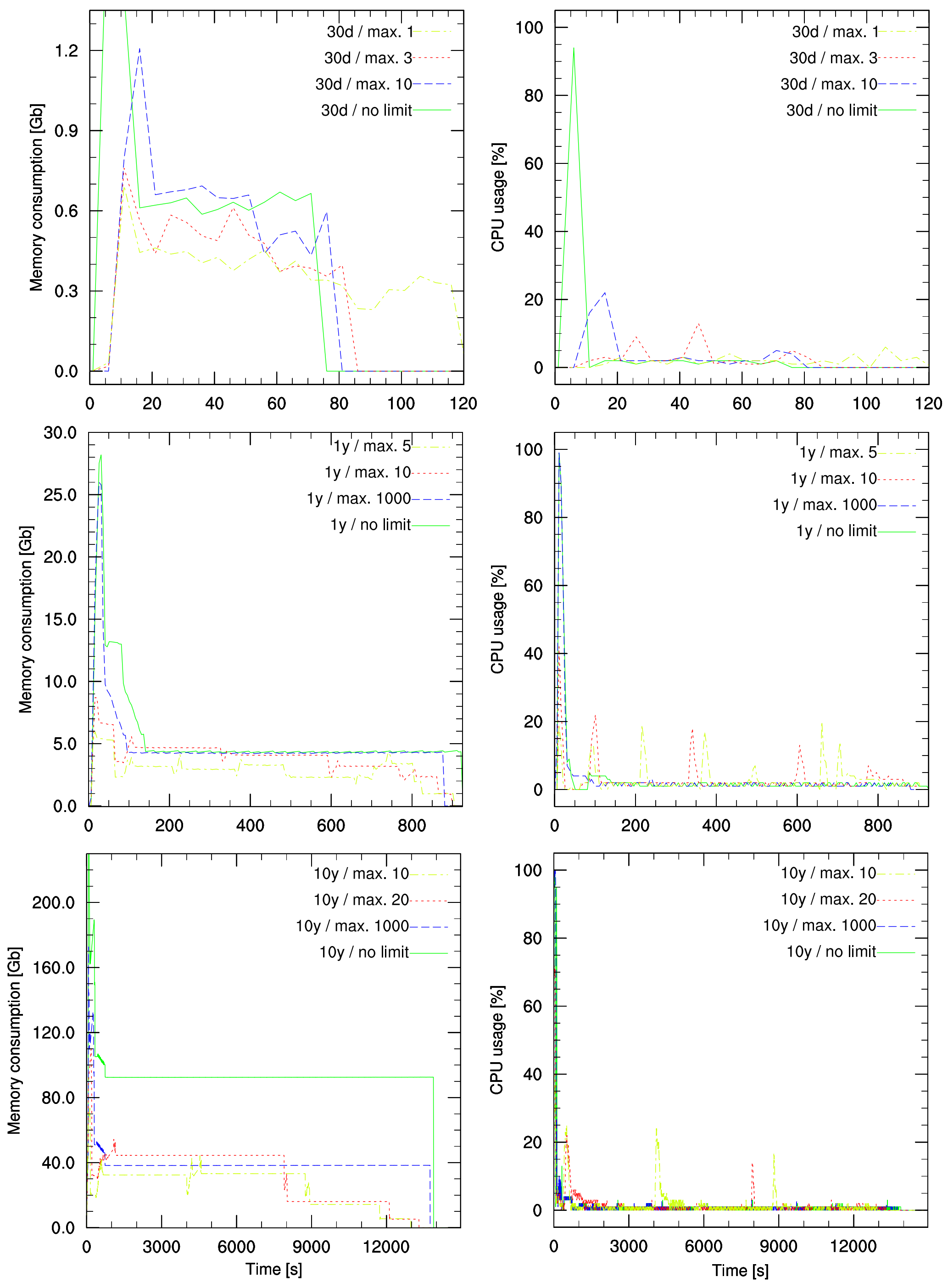


Figure 3. Memory and CPU usage for different periods and maximum database sizes

4. Scientific evaluation

To compare the effects of the two bias correction methods on the regional climate projections, we set up a limited area domain at 18km resolution over West Africa. In addition to the bias-corrected data PAC and PGW, we used ERA-Interim re-analysis data and MPI-ESM Echam6 GCM data to drive the regional climate model WRF [10]. These simulations were initiated in the first application period and finished mid 2014. Here, we report briefly on some of our results, details will be presented in a forthcoming publication.

The climate and in particular the annual rainfall cycle in West Africa are dominated by the monsoon. This seasonal movement of the rain band from South to North peaks in August over the Sahel zone (12–18°N), which usually receives more than 90% of its annual precipitation between July and September. In Fig. 4, we show the 9-year average (2001-2009) of the August precipitation for two observational data sets, the 0.5×0.5 degree CRU TS v3.21 data [6] (only available over land area) and the 0.25×0.25 degree TRMM data [2], as well as for the "truth field" (ERA-Interim re-analysis), the uncorrected GCM (MPI-ESM Echam6) and the two bias corrected models (PAC and PGW). First detail to note is that the two observational data sets agree in their spatial extent of the rain band, but that the total amount of precipitation is quite different in some areas. Among the model runs, ERA and PGW agree best with the observations. While this is expected for ERA, which is a re-analysis product for the same period 2000–2009, it is not *per se* true for PGW, which is constructed from ERA data for the previous decade 1990–1999 and differences in MPI data between the two periods 1990–1999 and 2000–2009. The uncorrected GMC data set MPI shows relatively good agreement over land, but hugely over-predicts rainfall over the ocean, in particular over the Golf of Guinea. This is largely caused by too warm sea surface temperatures (not shown). The PAC method, like the PGW method, succeeds in removing this bias in sea surface temperature, but leads to a too narrow rain band with little to no precipitation in the Sahel. Hence, with respect to the August monsoon precipitation over land, the PGW method outperforms the uncorrected GCM, while the PAC method has adverse effects.

Figure 5 sheds light on the reasons for these differences between the model runs: Here, we show results of the model surface winds for ERA-Interim re-analysis and MPI-ESM Echam6 GCM for both the reference and the application period, in addition to the two bias correction methods. Again, the data is averaged over 9 years for August only.

A key feature of the West African summer monsoon is the formation of the Saharan Heat Low (SHL), which can be seen clearly as a region of low winds, circulating counter-clockwise around its center at about 5°W 22°N in both ERA model runs. Conversely, while the SHL exists in the MPI 1990–1999 model run, it is no longer visible in the MPI 2000–2009 run. Apparently, the driving GCM suggests a drastic change in wind patterns for the first decade of the 21st century, which was not the case in reality. Such a large change in pressure and wind patterns between two adjacent decades leaves is imprints on both bias-corrected data sets. As expected, the PGW bias correction method stays closer to ERA and establishes a weak SHL, slightly shifted south-west. The PAC method, on the other hand, follows the change in circulation imposed by the GCM. At the same time, however, the above-mentioned correction of the sea surface temperature implies weaker and drier monsoon winds from the South-West, which in the end allows the dry Harmattan winds from the North-East to penetrate even further into the Sahel than in the MPI model run. This explains the narrow rain band of PAC in Fig. 4.

5. Conclusion and Outlook

We implemented and optimized two bias correction methods for GCM data, input to regional climate projections. In this application period, we modified the parallel code developed previously to reduce the required random access memory in order to fit on our hardware. This was realized with a queuing system, implemented as an additional Redis database table, and led to significantly smaller memory consumption with at the same time nearly identical runtimes. We also added a suite of unit tests and end-to-end tests to facilitate further development. We identified the writing of the bias-corrected data to disk as the major bottleneck in the current implementation. This will be improved in the future through parallel I/O, which will require to rewrite about 20% of the Python code in C/C++.

To study the effect of the two bias correction methods, we conducted regional climate simulations for a reference and application period of 10 years each. Here, we briefly investigated the results for the August (monsoon) rainfall and winds over West Africa. We detected a strong change in the large-scale circulation in the driving GCM data, not supported by the observations or the re-analysis data. This has a negative impact on the performance of the PAC model run, while the PGW model run stays closer to the re-analysis data and outperforms the raw GCM model run. From this one might think that PGW is superior and should be the preferred method. However, it is important to note that such a change in circulation might be a plausible scenario in the future, in which case the PGW method will remove valuable information. A detailed investigation of these regional climate simulations will be presented in a forthcoming paper.

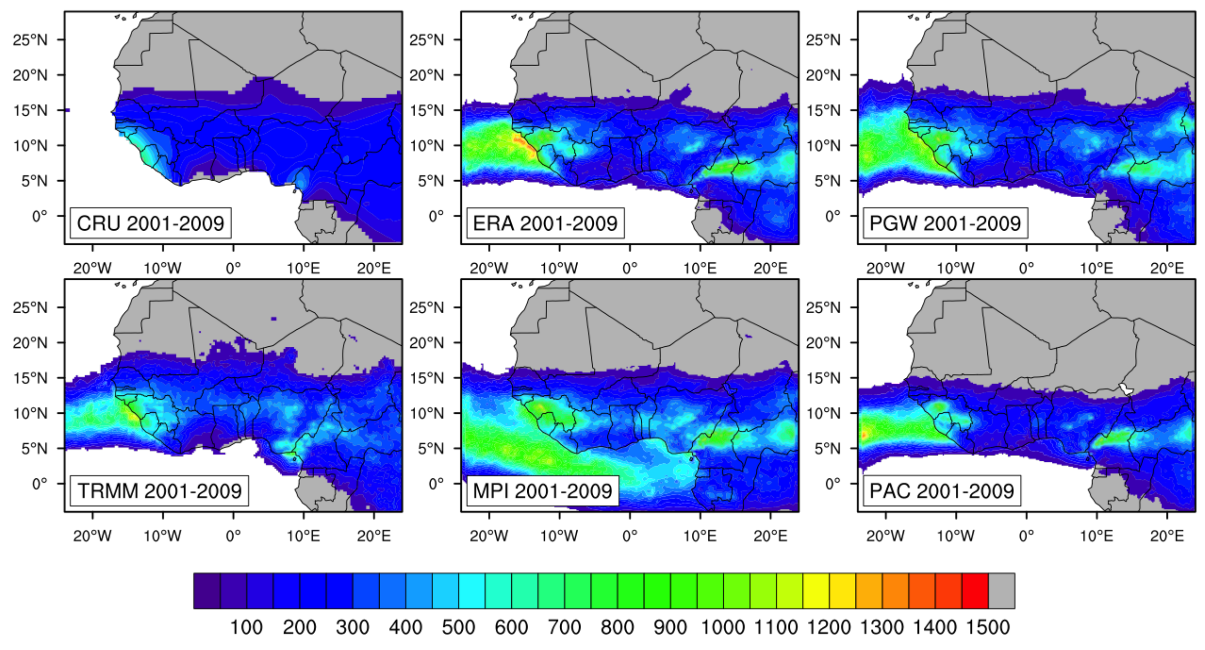


Figure 4. Model evaluation: average August precipitation [mm] for 2001–2009

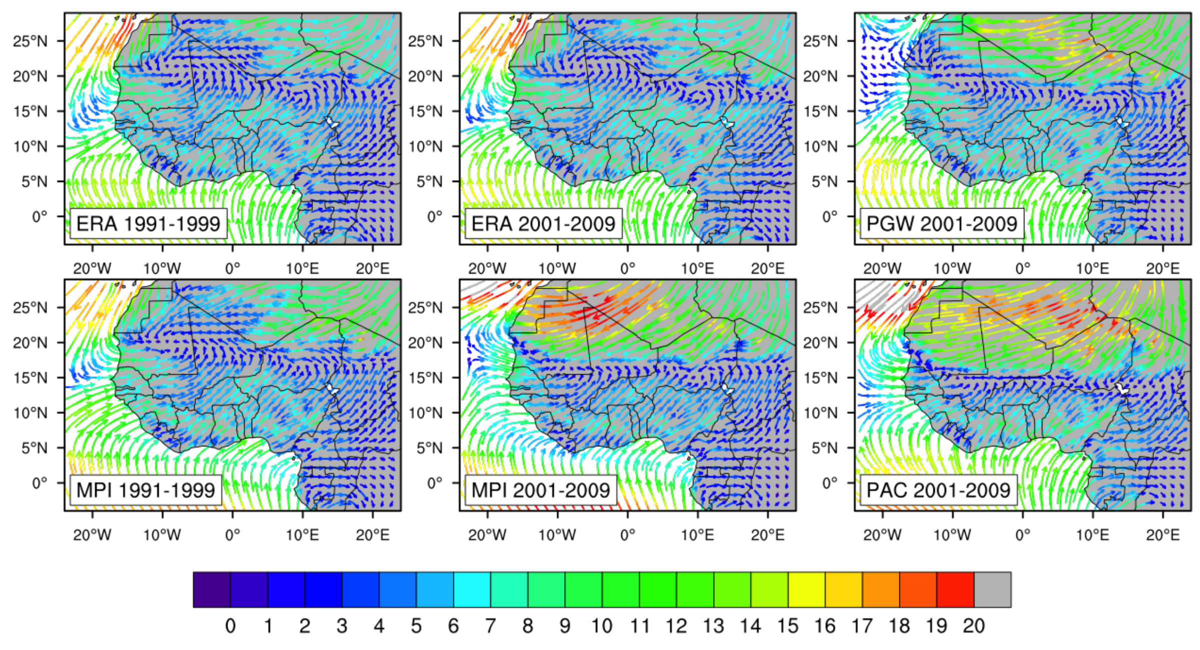


Figure 5. Model evaluation: average August wind fields [m/s] for 1991–1999/2001–2009

References

[1] Done, J.M., Holland, G.J., Bruyere, C.L., Leung, L.R., Suzuki-Parker, A.: *Modeling high-impact weather and climate: Lessons from a tropical cyclone perspective.* NCAR Technical Note NCAR/TN-490+STR, 2012

[2] Huffman, G.J., Adler, R.F., Bolvin, D.T., and 6 co-authors: *The TRMM Multi-satellite Precipitation Analysis: Quasi-Global, Multi-Year, Combined-Sensor Precipitation Estimates at Fine Scale.* J. Hydrometeor., 8: 38–55, 2007

[3] Beazley, D.: *Inside the Python GIL.* Python Concurrency Workshop, Chicago, May 14–15, 2009

[4] Dee, D.P., Uppala, S.M., Simmons, A.J., and 33 co-authors: *The ERA-Interim reanalysis: configuration and performance of the data assimilation system.* Quarterly Journal of the Royal Meteorological Society 137(656): 553-597, 2011

[5] Giannini, A., Biasutti, M., Held, I.M., Sobel, A.H.: *A global perspective on African climate.* Climate Change, 90: 359-383, 2008

[6] Harris, I., Jones, P.D., Osborn, T.J., Lister, D.H.: Updated high-resolution grids of monthly climatic observations ? the CRU TS3.10 Dataset. International Journal of Climatology, 34(3): 623-?642, 2014

[7] Rasmussen, R., Liu, C., Ikeda, K., and 12 co-authors: *High resolution coupled climate-runoff simulations of seasonal snowfall over Colorado: a process study of current and warmer climate.* J. Climate, 24: 3015–3048, 2010

[8] Rummukainen, M.: *State-of-the-art with regional climate Models.* Climate Change, 1: 82–86, 2010

[9] Stevens, B., Giorgetta, M.A., and 15 co-authors: *Atmospheric component of the MPI-M Earth System Model: ECHAM6.* Journal of Advances in Modeling Earth Systems, 5: 146–172, 2013

[10] Skamarock, W.C., Klemp, J.B., Dudhia, J., and 6 co-authors: A description of the Advanced Research WRF version 3. NCAR/TN-475+STR, 2008

[11] Sylla, M.B., Gaye, A.T., Jenkins, G.S., Pal, J.S., Giorgi, F.: *Consistency of projected drought over the Sahel with changes in the monsoon circulation and extremes in a regional climate model projections.* Journal of Geophysical Research, 115: D16108, 2010

Machine Learning for Security Analytics powered by SAP HANA

Andrey Sapegin, Feng Cheng, David Jaeger, Amir Azodi, Marian Gawron,
Daniel Stelter-Gliese, Christoph Meinel
Hasso Plattner Institute
{firstname.lastname}@hpi.de

Abstract

The overall extent of log data which is created by modern networks grows continuously. The current problem is to benefit from the data and gain insights from the logged events. Since this problem requires powerful methods to perform analytics on big data sets, we faced this problem with SAP's new In-Memory database HANA. In the presented case we consider log data from the Active Directory Domain Controller, which gives information about special events, such as login, logout, and ticket requests. Therefore we had to preprocess the data accordingly to use them in the desired machine learning algorithms. We developed a method to convert textual data into numeric formats, since the investigated machine learning algorithms require numeric data. We evaluated different approaches, such as a naive implementation in OCTAVE, the built-in Predictive Analysis Library (PAL), or the integration of an additional R server. The SAP Predictive Analysis, which contains numerous machine learning algorithms, provides a simple user interface and could directly work on the data stored in HANA. This front-end was created by SAP, which limits it to use algorithms that are implemented already. In contrast, the R integration, which allows to create machine learning algorithms in R and use them in HANA, requires more configuration and could be used in SQL queries directly. Finally we compared the approaches concerning the differences in performance, usability, and extensibility.

1 Project concepts

Within this project we aim to implement and test the machine learning approach for security analytics based on SAP HANA. Under this approach we focus on the analysis of user events, particularly login and logout events. These goals imply the following objectives to be reached:

- Setup of SAP HANA system on the infrastructure provided by Future SOC Lab
- Creation of a testbed for data generation
- Enhancement of data normalization method
- Setup of an environment for performance tests
- Implementation of an analysis using machine learning techniques
- Estimation of the performance of machine learning analysis of Active Directory events on SAP HANA platform

To complete these objectives, we have extended the existing SIEM system — Real-Time Event Analysis and Monitoring System (REAMS) — as well as developed new modules for import, normalization and analysis of Active Directory events. The architecture of the resulting system is described in the next section.

2 Architecture

The Real-Time Event Analysis and Monitoring System (REAMS) is a combination of Intrusion Detection and Security Information and Event Monitoring Systems under ongoing development in the Hasso Plattner Institute. Under this project we aim to collect, normalize, process and correlate security events in real time. The current state of the system is presented on the Figure 1.
The gatherer component collects logs, as well as other information about network and computers and sends them to the core system, where they are normalized using the information from knowledge base. The knowledge base contains regular expressions for normalization of every supported log format [1]. Using regular expressions, we extract information and meta-data from every log line and put it into Object Log Format [1]. Therefore, all log messages of different types from different systems are stored in one place and in one format, which significantly simplifies further processing and correlation of log messages, which should be done in real-time. To achieve the goal of processing events in real-time, we use an In-Memory database - SAP HANA. All the normalized data is therefore stored and processed directly inside the main memory

[1]currently we support multiple log formats, including Windows Event Log, Cisco, iptables, snort, ssh, MySQL, Apache, etc.

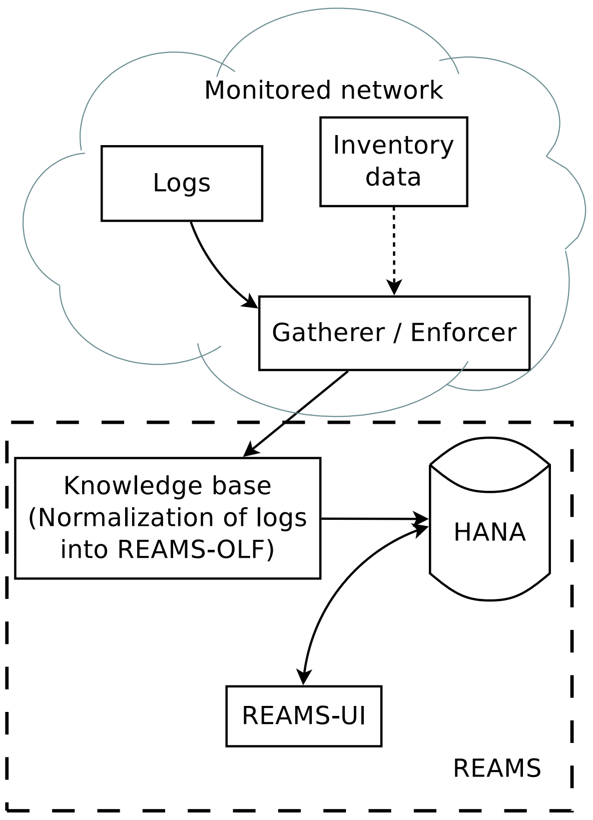


Figure 1. Original architecture of HPI REAMS system

of the database server. This technique allows us to considerably speed-up the processing of events, and also enables the capability of analysis using machine learning algorithms nearly in real-time. To use to analyze the machine learning capabilities of SAP HANA for analysis of user behavior, we supplemented our system as presented on the Figure 2.

The Figure 2 contains new REAMS modules, which are marked with green. The user behavior data is generated using the testbed with Active Directory service and then collected by gatherer using a new module capable to extract Windows Events from the Domain Controller[2].

The collected Windows Events are later normalized into Object Log Format [1] using regular expressions to extract different parts of the event into different fields of the Object Log Format. The data, converted into this format is stored in the SAP HANA database, which offers various integrated capabilities for high-speed analysis of log messages:

- **complex queries** First of all, as for any other SQL database, the SQL queries could be used to filter out the most obvious attacks. Since we focus on the user behavior, the simple filters such as highlighting of failed login events does not always point on the malicious behavior or other problems. So we decided to concentrate on other possibilities for data analysis, namely functions offered through Predictive Analysis Library (PAL) and integration with R language.
- **PAL** A Predictive Analysis Library (PAL) [2] offers various machine learning algorithms for data analysis. The major advantage of using PAL is that the data is processed directly inside the database[3], while for many other databases the data should be first retrieved from the database. Moreover, the PAL functions could be called directly from SQL, which simplifies the integration of the analysis functions into REAMS. Finally, SAP offers Predictive Analysis software [4], which provides a graphical interface for testing the available algorithms, as well as tools for visualization of initial data and results of the analysis.
- **R integration** SAP HANA also offers an integration with R language[4]. This approach allows to use any R functions or libraries — for example, “kernlab” R library [6] containing implementations for most popular machine learning algorithms — for the analysis. To use it, the data should be transferred to Rserve for analysis. However, since the data is stored in the memory of SAP HANA database server, the transfer to the memory of Rserve server could still be performed in a short time, especially if the Rserve is installed just in another virtual machine on the same hypervisor. The calls to R functions could be also performed directly from SQL and therefore be easily integrated into the system.

Thus, for the analysis of user behavior, we concentrated on the capabilities of PAL and Rserve. Both approaches provide capabilities for fast analysis and easy integration into existing system. In addition, SAP Predictive Analysis allows to test the machine learning algorithms with different parameters using the graphical interface, which simplifies the estimation of different algorithms. After testing of machine learning techniques for detection of attacks and user behavior anomalies, we plan to integrate the most efficient and successful approaches into user interface (REAMS-UI), which currently stays unchanged. To test the available algorithms, we have generated the data using the Active Directory testbed, as described in the following Section.

[2]This task could seem to be easy, however, the Windows Event Log stores events with different EventIDs in different XML schema. Moreover, the XML schemes could also differ for the events with the same EventID but different Provider. Also, the offered extraction tools, i.e. Microsoft LogParser, do not extract all the information available for the event. Therefore, we have implemented the extraction using Win32 libraries for Python (win32evtlogutil, win32evtlog, win32api, etc.) to gather the complete information about every security event.

[3]similar approach is used in Oracle Data Mining module for Oracle RDBMS Enterprise Edition [3]

[4]requires Rserve [5] installed on the separate server

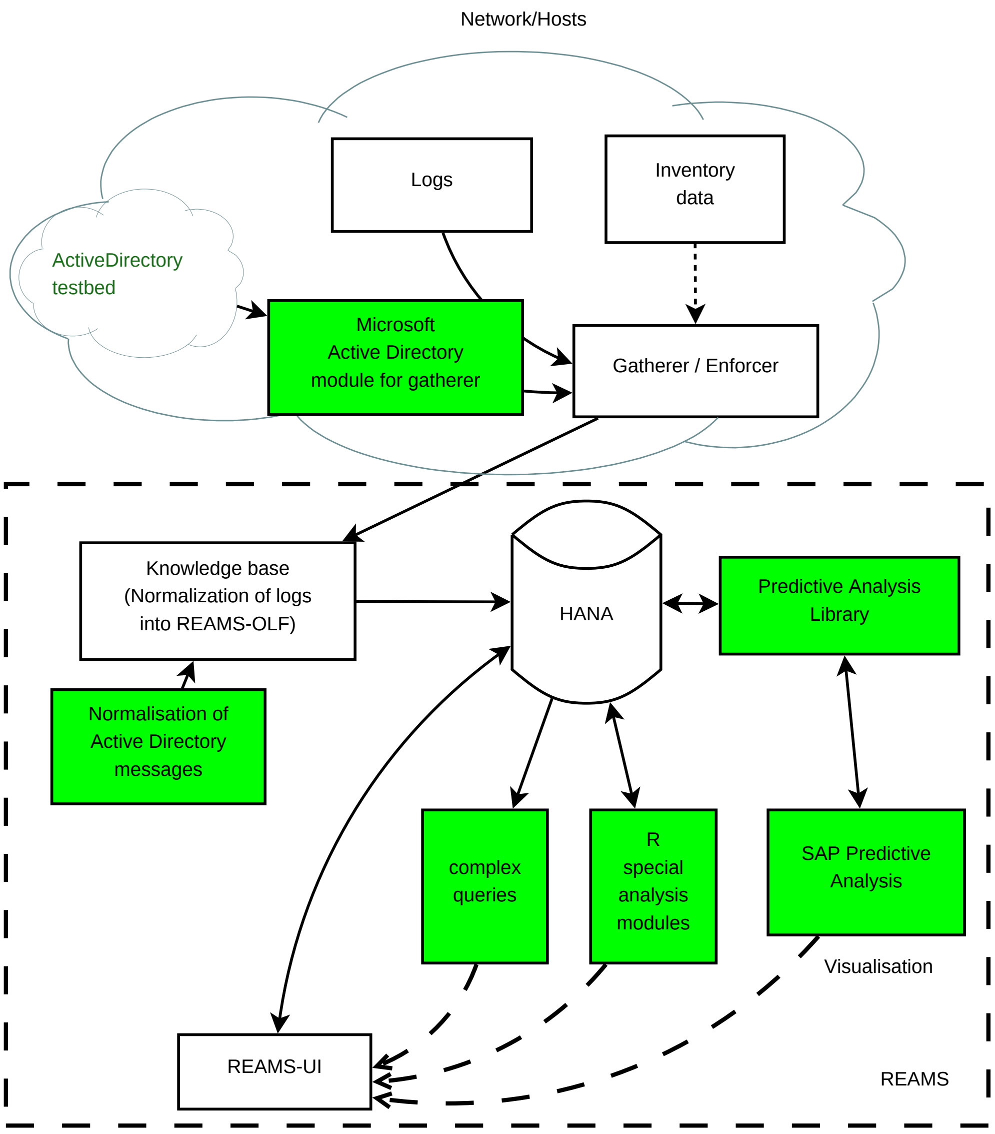

Network/Hosts
Logs
Inventory data
ActiveDirectory testbed
Microsoft Active Directory module for gatherer
Gatherer / Enforcer
Knowledge base (Normalization of logs into REAMS-OLF)
HANA
Predictive Analysis Library
Normalisation of Active Directory messages
complex queries
R special analysis modules
SAP Predictive Analysis
Visualisation
REAMS-UI
REAMS

Figure 2. Enhanced architecture of HPI REAMS system

3 Data

To analyze efficiency of machine learning approach for analysis of security events, we have created a testbed with a Windows network. The virtual network includes an Active Directory Domain Controller, several client machines and, optionally, additional Windows Server systems. The created testbed was used for the following purposes.

- generate initial dataset for testing of newly created system modules, including data gatherer, normalization of Windows Events and machine learning analytics. For testing of machine learning analytics module, we have generated several simple brute-force attacks.
- create a dataset for performance tests by replication of initial dataset
- simulate more sophisticated cases of malicious user behavior

The detailed description of the testbed, including software used on virtual machines and attack scenarios is provided in the subsections below.

3.1 Testbed

The virtual network consists of 3 virtual machines with the following operation systems:

- Windows 8 x64
- Windows 2000 Pro SP4
- Windows 7 Pro

For the Domain Controller, the Windows Server 2012 R2 Datacenter was used. The Domain Controller has

Audit policy with enabled logging of login and logout events.
In addition to the domain administrator account, we have created 4 users, that are allowed to login on the any of 3 virtual machines.
After the initial setup of the testbed, we have simulated several attacks on Domain Controller and one of regular virtual machines. Please see the Table 1 for details.

Date	Attack performed
28.01.2014, 13:11 GMT+1	Unsuccessful password brute-force with Hydra via RDP on Domain Controller
29.01.2014, 10:00-10:10 GMT+1	successful brute-force of LDAP using Hydra
29.01.2014, 12:30-13:10 GMT+1	manual password brute-force in console of virtual machine with Windows 2000 Pro SP4

Table 1. Attacks in the initial dataset

The collected logs, including the messages generated during attack simulation, cover two time periods - from 28.11.2013 till 13.12.2013 and from 14.01.2014 till 29.01.2014. Together, they contain information about 477,172 security events[5]. However, we are interested not in all Security events collected by the domain controller, but only in specific ones, as soon as our focus is the user behavior anomalies. Therefore we have selected only those events for analysis, that reflects the user behavior. Please see the Table 2 for details.

EventID	Description
4768	A Kerberos authentication ticket (TGT) was requested
4769	A Kerberos service ticket was requested
4771	Kerberos pre-authentication failed
4776	The domain controller attempted to validate the credentials for an account
4624	An account was successfully logged on
4625	An account failed to log on

Table 2. Windows security events selected for the analysis

After we filter out the events with selected IDs, the total number of events will be reduced to 188,457.

3.2 Dataset replication

Since the dataset created using a testbed has only 188,457 security events that are relevant for the analysis of user behavior, we needed more data for the performance tests. However, for a performance measurements, the structure and quality of the dataset is much less important, than the number of events that should be analyzed. Therefore, we created several datasets for the performance tests by replicating the original data created using the testbed. The number of events in the each dataset is presented in the Table 3.

Number of events	Number of events with selected IDs
477,172	188,457
954,344	376,914
1,908,688	753,828
3,817,376	1,507,656
7,634,752	3,015,312
15,269,504	6,030,624
30,539,008	12,061,248
61,078,016	24,122,496
122,156,032	48,244,992

Table 3. Size of the datasets for performance tests

In the Table 3 we show both number of all security events, as well as number of events with selected IDs in the replicated dataset. Although we replicated all available security events, we use only the events with selected IDs for performance testing.

3.3 Support for gathering of real-world data

Although using our testbed we were able to simulate realistic data for analysis, including not only simple brute-force attacks, but also more complicated cases of malicious user behaviour, we had an interest to analyze real data to prove our concepts in real world. However, to get a real-world data, it is not always possible to install a self-developed monitoring system or at least log gatherers, on the productive system. The reasons for this are increased requirements for privacy, fault tolerance and security which could be described as follows:

- **security**. Since all logs are initially collected by domain controllers, the gatherer should be installed there. However, the domain controller is always the most important part of an IT infrastructure, and therefore has increased security requirements. The installation of the third-party software, like the gatherer could be therefore very complicated to agree on.
- **fault tolerance**. The number of security logs on the domain controller in a big network could be high. Therefore, collecting and exporting high amounts of security logs could take the resources of the domain controller and affect the processing of other tasks on the DC.

[5]here we mean number of events that were collected by the Domain Controller and were visible in the “Security tab” of an Event Viewer on the Domain Controller

- **privacy**. Finally, the data such as Domain Controller logs often contain a personal information, e.g. user ID, time of login and logout events. This information is often an object of data privacy and should be anonymised before exporting for analysis.

To deal with such issues, we have created a standalone script for export of Windows security events, that anonymises the data related to user privacy and takes care about hardware resources of the Domain Controller. Moreover, the script is light-weight PowerShell executable, which could be easily checked for security issues by system administration staff.

3.4 Post-processing of events for machine learning analysis

We store the normalized events in Object Log Format in our database. However, the machine learning analysis could not be applied directly on this data for two reasons.

First of all, not all fields of the stored event are relevant for the analysis of user behavior. To perform such analysis, we have selected 11 features as presented in the Tables 4 and 5.

Tables 4 and 5 show how Windows Event fields for different EventIDs will be mapped into the Object Log Format. The Windows Event fields are listed in the columns under 6 pre-selected EventIDs (see Table 2). The names on the left side of each table shows fields of Object Log Format. Since Windows Events have different schemes for different events, we have manually selected for each EventID, which fields should be mapped into the same fields of Object Log Formats (e.g., 'TargetSid' for EventID 4768 and 'TargetUserSid' for EventID 4771).

The major requirement to normalize the data in an appropriate format to perform machine learning algorithms on it was the transformation into a numeric format. In general machine learning algorithms process data in numeric format only, so we had to transform data, such as TargetUserName, IpAddress, or Host in a numeric form. During this process we had to keep track of the correct mapping from textual data to numeric data.

4 Analytics and visualization

To analyze data described in the Section 3, we have used functions provided by Predictive Analysis Library in SAP HANA and by the R language which was integrated with SAP HANA using Rserve. We describe both libraries, as well as visualisation techniques and analysis results in the subsections below.

4.1 Predictive Analysis Library capabilities

Predictive Analysis Library is a part of Application Function Library in SAP HANA database. The Predictive Analysis Library provides functions for data mining, i.e. for data classification and clustering. The library is implemented as an extension for SQL in HANA database. It could be called from HANA SQL Script functions. Since the library processes the data directly in the in-memory database, it should have a higher performance in comparison with traditional solutions, which do not store the data in the main memory, or require extraction of data from a database before processing.

Apart from function calls from SQL Script, SAP also offers Predictive Analysis software. Predictive Analysis provides a Graphical User Interface and significantly simplifies evaluation of algorithms available in Predictive Analysis Library, since they be selected, ordered — using drag&drop —, configured and executed without writing any SQL statements. Figure 3 shows example analysis scenario for Anomaly Detection algorithm in SAP Predictive Analysis.

4.2 Integration with Rserve

SAP HANA supports integration with R language. To set it up, the separate server for Rserve is required. After the installation of Rserve according to the official documentation [7], we were unable to install "kernlab" library into Rserve. This would result in the inability to use other machine learning algorithms, although k-means could be used without "kernlab". To keep flexibility for future use we suggest to install "kernlab" library anyway. Therefore, we provide our own notes on the configuration of R integration for SAP HANA. Please follow the steps below for successful configuration of Rserve server:

- **OS install**. We have installed OpenSUSE 11.4 into the virtual machine
- **OS configuration**. We have installed 'xorg-x11-devel', 'gcc-fortran' and 'readline-devel' packages into the OpenSUSE 11.4.
- **Rserve installation**. To install Rserve, we have used a patched version from OpenSUSE repository. To install it, execute the following commands in your OpenSUSE 11.4 environment:

```
VERSION=$(grep VERSION /etc/
SuSE-release | \
sed -e 's/VERSION = //')

zypper addrepo -f \
http://download.opensuse.org/ \
repositories/devel\: \
```

		Windows Event ID		
		4768	4769	4771
Object Log Format fields	subjectUser.userId			
	subjectUser.username			
	targetUser.userId	TargetSid		TargetUserSid
	targetUser.username	TargetUserName	TargetUserName	TargetUserName
	additional[win.ad.login.type]			
	network.srcIpv4 / network.srcIpv6	IpAddress	IpAddress	IpAddress
	network.srcHost			
	eventTypeId	EventID	EventID	EventID
	time	TimeCreated	TimeCreated	TimeCreated
	producer.host	Computer	Computer	Computer
	application.statusCode	Status	Status	Status
	additional[win.ad.sub.status]			
	additional[win.ad.failure.reason]			

Table 4. list of selected features as stored in Object Log Format

		Windows Event ID		
		4776	4624	4625
Object Log Format fields	subjectUser.userId		SubjectUserSid	SubjectUserSid
	subjectUser.username		SubjectUserName	SubjectUserName
	targetUser.userId		TargetUserSid	TargetUserSid
	targetUser.username	TargetUserName	TargetUserName	TargetUserName
	additional[win.ad.login.type]		loginType	loginType
	network.srcIpv4 / network.srcIpv6		IpAddress	IpAddress
	network.srcHost	Workstation	WorkstationName	WorkstationName
	eventTypeId	EventID	EventID	EventID
	time	TimeCreated	TimeCreated	TimeCreated
	producer.host	Computer	Computer	Computer
	application.statusCode	Status		Status
	additional[win.ad.sub.status]			SubStatus
	additional[win.ad.failure.reason]			FailureReason

Table 5. list of selected features as stored in Object Log Format

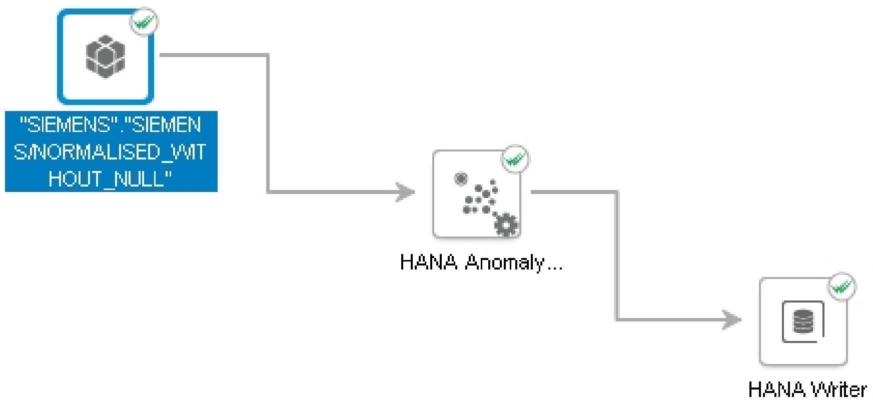


Figure 3. Anomaly Detection scenario in SAP Predictive Analysis

```
languages\:R\:patched/
openSUSE_$VERSION/R-patched

zypper install R-patched
zypper install R-patched-devel
```

- **installation of Rserve and "kernlab" library**. To perform it, open R console using "R" command and run the following:

```
install.packages("Rserve")

install.packages("kernlab")
```

After these steps, Rserve is installed with "kernlab" library containing implementations for most popular machine learning algorithms. The process of further configuration and connection to SAP HANA database does not differ from the official documentation [7]. The configuration parameters are HANA specific, so we do not describe them here.
If everything was configured correctly one could create and use R functions using SQL queries in SAP HANA like shown in listing 1:

Listing 1. Example for R integration

```
CREATE PROCEDURE someMethod(IN
    table1 TABLE1, OUT result
    TABLE_OUTPUT)
LANGUAGE RLANG AS
BEGIN
 ...
END;
CALL someMethod(INPUTTABLE,
    RESULTTABLE);
```

The visualization of results could be done in R directly. The only requirement is that the results of the method, called in HANA, are stored as a temporary variables, which then can be used for visualization.

Finally we would state that the R integration allows an easy extension and usage of self-designed machine learning algorithms. One could use any script or library function, which is available in R. Thus a user with the requirement to use a high variety of machine learning algorithms should definitely consider the advantage of extensibility of R over easy usage of PAL.

4.3 Security analytics

First of all, to try out our approach, we have used the initial dataset of 188,457 events described in the Section 3.1. The data is presented on the Figure 4.
Two of three (see Table 1) attacks are easily visible as two high spikes on the Figure 4 due to the high number of events generated by brute-force attacks.
To check if we could find these attacks in the data automatically, we needed to select algorithms, that fit for our dataset and analysis purposes. In particular, we prefer unsupervised algorithms as soon as they fit for unlabeled datasets and could recognise previously unknown attack types. This is different to supervised learning algorithms, that should be trained on the dataset containing all possible attacks. Also our testing dataset contains only a few attacks, which could be not enough for the training phase of supervised algorithms. Generally, we prefer algorithms that do not need high number of attacks or anomalies to be presented, since we do not expect, that attacks and malicious user behavior will generate major number of Windows Events. Taking into account these requirements and the limited number of algorithms presented in PAL, we have selected two algorithms: (1) Anomaly Detection and (2) k-means.
For both algorithms we have used the following parameters specified in the SAP Predictive Analysis:

- percentage of anomalies for Anomaly Detection: 10%
- number of clusters for k-means: 8

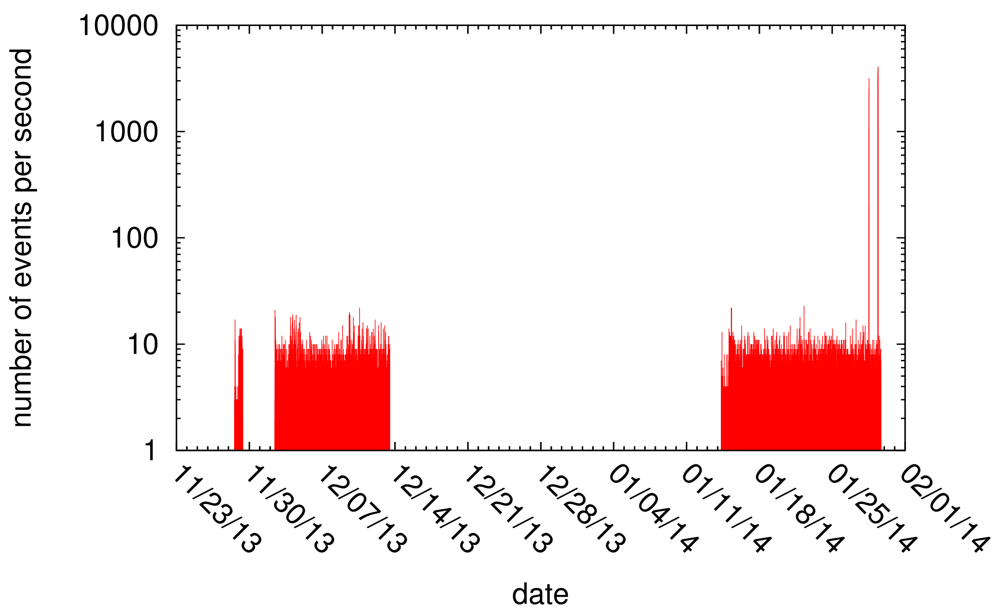


Figure 4. Performance of machine learning algorithms on the test dataset

- number of threads: 8
- anomaly detection: by sum of distances from all centers
- normalization type: based on row values
- distance measure: Euclidean Distance

In addition to Anomaly Detection and k-means executed via SAP Predictive Analysis, we also tried to analyze the data using k-means algorithm from the R integration.

The results of the analysis using Anomaly Detection and k-means in SAP predictive analysis are presented on the Figures 5 – 6. k-means on the Rserve produced results identical to k-means in SAP Predictive Analysis, so we do not provide a separate plot for it.

Using Anomaly Detection, we were able to detect both automated brute-force attacks (seen as two spikes in the end of January). Two small spikes on the left are false positive results received due to overstated predefined percentage of anomalies (10%). Finally, manual password brute-force attack was not detected because of small number of login failures.

Different to Anomaly Detection, results received using k-means algorithm are not so easy to visualize. The output contains 8 unlabeled clusters with unknown characteristics. However, since we were trying to detect brute-force attacks, we consider reasonable to visualize clusters using number of events. Figure 6(a) therefore shows all 8 clusters, where the size of dots represents number of events (in a log scale) logged per time unit. Clusters 3 and 4 have several enormously big circles, which could indicate brute-force attacks. Indeed, if we look at Figures 6(b) and 6(c), the brute-force attacks are divided between clusters 3 and 4. Cluster 4 has only events related to two automated brute-force attacks from the Table 1, while cluster 3 covers all brute-force attacks together with other events. The number of other events (false positives) is quite high, so we are not able to filter out the third attack (manual password brute-force), even if it is included into the cluster 3.

4.4 Performance of ML algorithms

To evaluate the performance of machine learning analysis of Windows Events, we repeat the analysis described in the Section above, but on the replicated dataset.

Our measurements were done on two virtual machines installed on the VMware ESXi Server, shared with other virtual machines. Please see the details in the table 6.

During our tests, we measured the following parameters:

- **execution time** was measured by SAP HANA Studio and SAP Predictive Analysis. SAP HANA Studio prints a query execution time to the output by default. SAP Predictive Analysis writes it to the log file.
- **CPU usage** was measured using ‘top’ and ‘sar’ (from ‘systat’ package) tools on both servers. The measurements were started and stopped via ‘ssh’ using ‘bash’ script.
- **I/O wait** was measured using ‘sar’ tool on both servers.

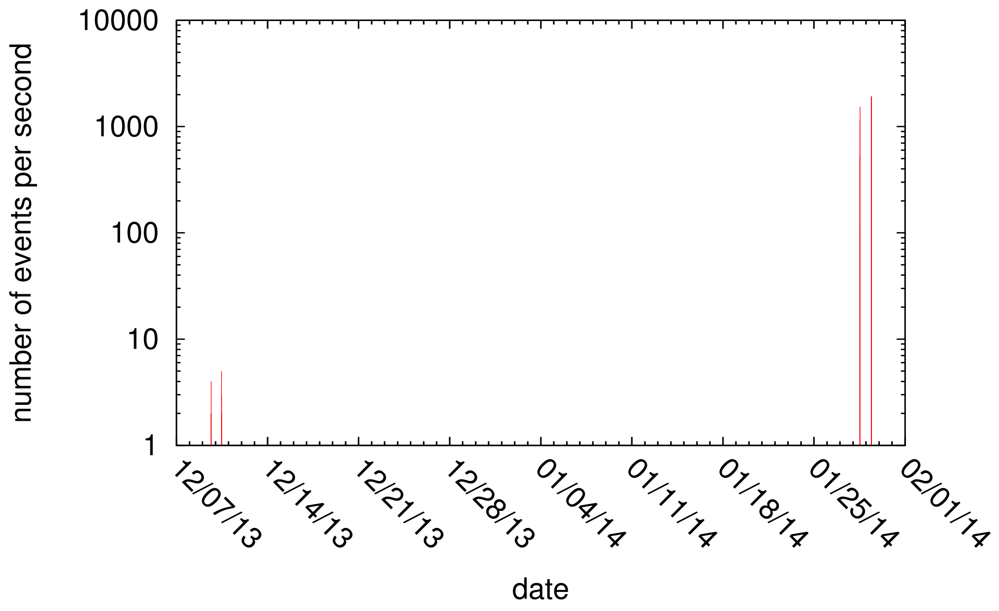
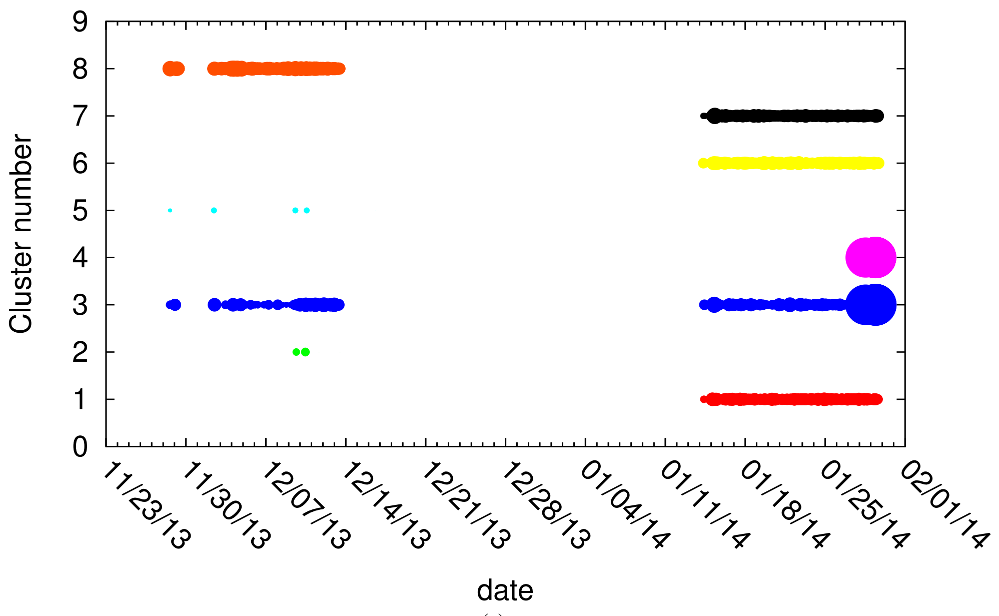
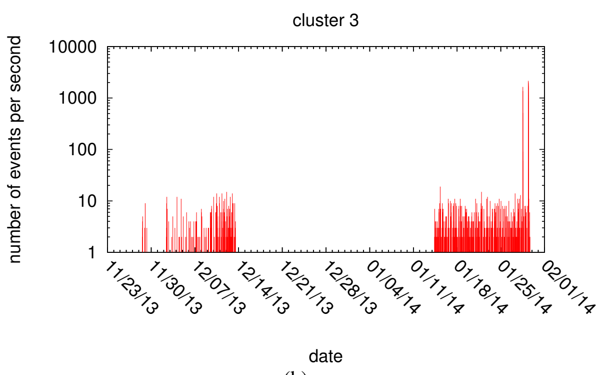


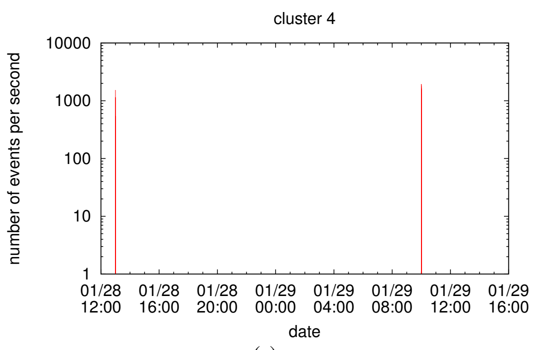
Figure 5. Results from Anomaly Detection algorithm



(a)



(b)



(c)

Figure 6. Results from k-means algorithm with 8 clusters

System	Operating System	CPU	HDD	RAM
SAP HANA SPS06 VM	SLES for SAP Applications 11.3 (x86_64)	16 vCPU	314 Gb	80 Gb
Rserve (R 3.0.2) VM	openSUSE 11.4 (x86_64)	8 vCPU	40Gb	16 Gb

Table 6. System configuration

- **memory usage** was measured using 'top' on both servers [6]

We present the performance of machine learning algorithms used on the Figure 7. The red line shows execution time for Anomaly Detection algorithm run using Predictive Analysis, green — for k-means algorithm run using Predictive Analysis. Finally, blue line shows execution time for k-means algorithm implementation from 'kernlab' library on the Rserve server integrated with SAP HANA. The last algorithm was executed using SQL query in SAP HANA Studio.
Of course, before processing using machine learning algorithms, the data was post-normalized (see Section 3.4). To be confident in our performance evaluation, we also include the performance measurements for post-normalization, please see Figure 8.
The time slump on the chart at 3M events is caused by the optimization of post-normalization script, that we did after it become relatively slow and had issues with memory limit of HANA Server.
Both Figure 7 and 8 provide processing time for maximum of 48 million events. This boundary comes from the limitations of our setup with 80 Gb main memory for SAP HANA database instance. The only algorithm, that had issues due to the implementation weaknesses on 24 million events was k-means using R integration.
Besides the speed measurements, we have also checked for limitation factors, that could affect a performance of SAP HANA and Rserve servers. To evaluate it, we recorded CPU usage and I/O Wait on both servers. Please see the results on the Figure 9.
The CPU usage of both HANA database and Rserve server shown on the Figures 9(a) and 9(c) allows us to conclude, that CPU becomes a limiting factor only on high data volumes. Indeed, during the analysis of 12 and more millions of events, the maximum CPU usage of HANA server always reaches at 100%. However, if Anomaly Detection algorithm or k-means algorithm with relatively small number of clusters is used for analysis, the results still could be received in a reasonable time (several minutes) even when all 16 vCPUs on HANA server are fully loaded (see Figure 7). The CPU of Rserve server was never fully loaded, since the default implementation of k-means algorithm does not support multi-threading.

[6] we do not provide memory usage statistics in this report

The disk drive speed monitored using I/O Wait measurements, could be a limiting factor, but only for Rserve server, where maximum I/O Wait value almost reaches 45% once (see Figure 9(d)), preventing CPU to be used more effective. The maximum I/O Wait values on HANA server stays relatively low, even while processing 48 Million events (see Figure 9(b)) and never reaches 35%. This is expectable, since SAP HANA is an in-memory database and it's performance should not be affected by a disk drive.

5 Further work

In the next project phase, we will concentrate on following aspects: (1)Optimization of the current implementation for getting better performance (2)more comprehensive simulation testing and performance measurement (3) improvement of the detection techniques, including usage and further development of other algorithms in R language, (4) integration of machine learning analysis approach into high-speed SIEM product prototype and (5) advanced visualization techniques
Within this project, we have created and evaluated several modules for our SIEM system, including gatherer for Windows Events, normalization and machine learning analytics modules. We will finalize an integration of these modules into our system (REAMS) and provide a module for the user interface (REAMS-UI), to create a standalone system capable of analyzing security events using machine learning algorithms in automated mode. The problem of result visualization becomes more important in this case, since the operator of such system should be able to distinguish attacks rapidly. Moreover, not only presence of attacks should be highlighted, but also events related to an attack and hosts, affected by an attack.
During the development of such standalone system, we also plan to implement extra features described below:

- support for other log formats
- visualization of attack path on the network graph
- auto-generation of attack signatures (complex queries) based on anomaly detection results
- vulnerability analysis using network inventory data and vulnerability database

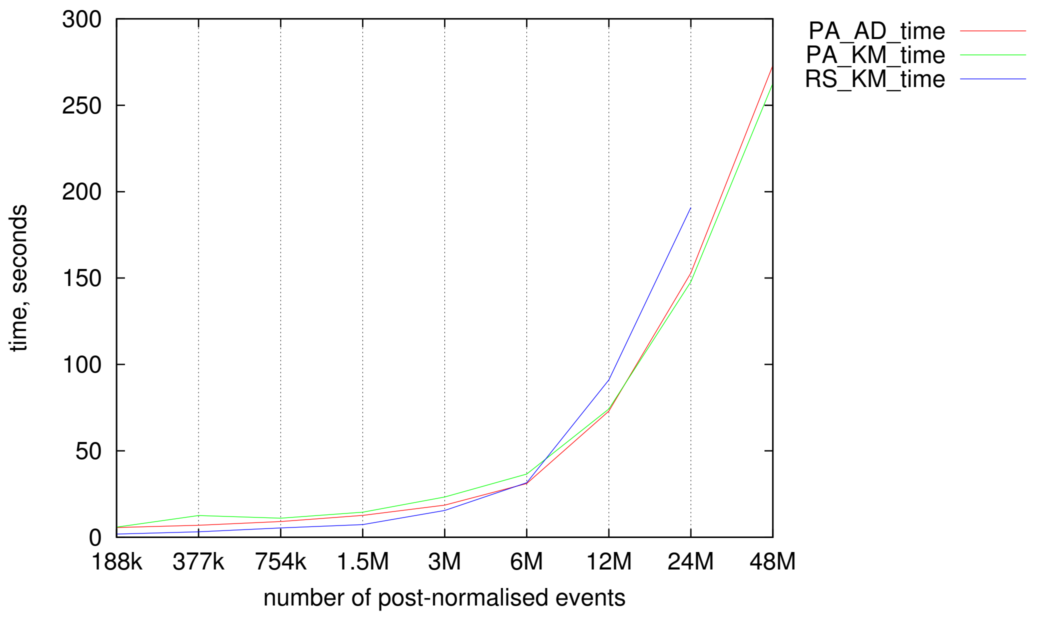


Figure 7. Performance of machine learning algorithms on the test dataset

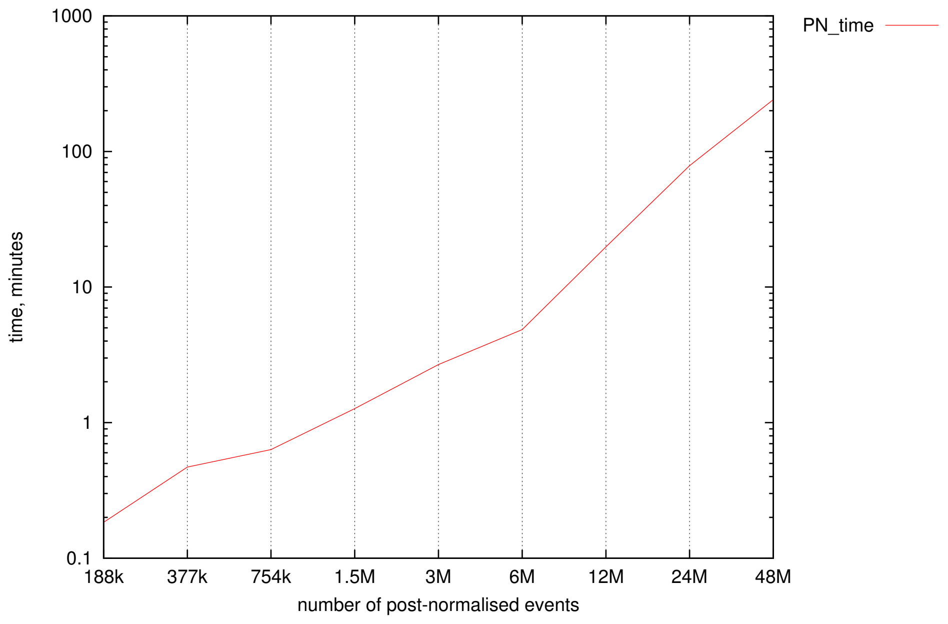
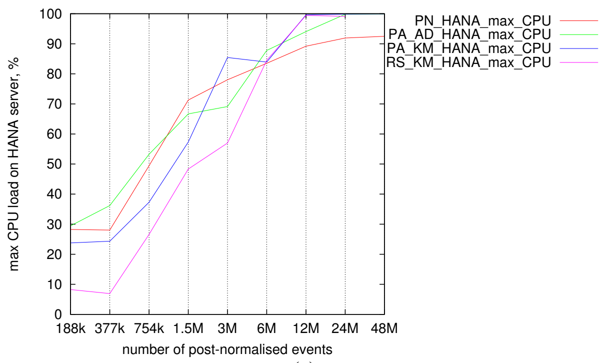


Figure 8. Performance of post-normalization process on the test dataset

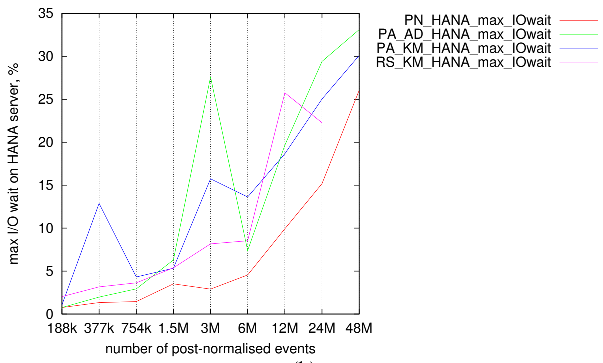
The resources offered by Future SOC Lab, including SAP HANA system with pre-installed Predictive Analysis Library and preconfigured Rserve server will allow us to concentrate on the planned features and easily expand our system to achieve higher analysis performance.

6 Conclusion

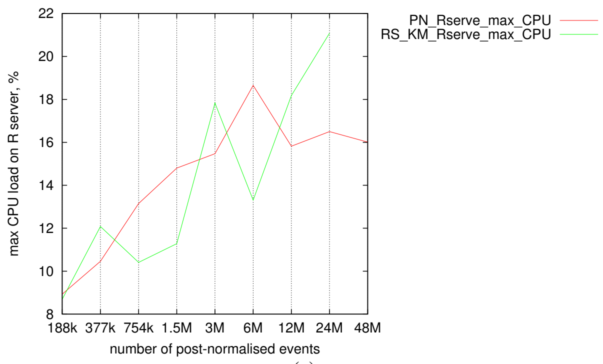
Security monitoring systems deal with large volumes of heterogeneous log messages. Using our existing SIEM (REAMS), we were able to extend it's capabilities using machine learning tools provided by SAP HANA database and libraries of R language (also integrated with HANA database). The main limitation of such approach — low analysis speed on big data volumes — was avoided due to high performance of the in-memory database itself, and also the fact, that the data was processed directly in the database. Of course, to be able to analyze heterogeneous data, one also need to normalize it into one format. For our system, we have used the Object Log Format [1], that we developed specially for security events. Thus, under this project, we built a novel high-speed prototype of security analytics system. For example, using a database server with only 80 Gb of RAM and 16 vCPUs, we



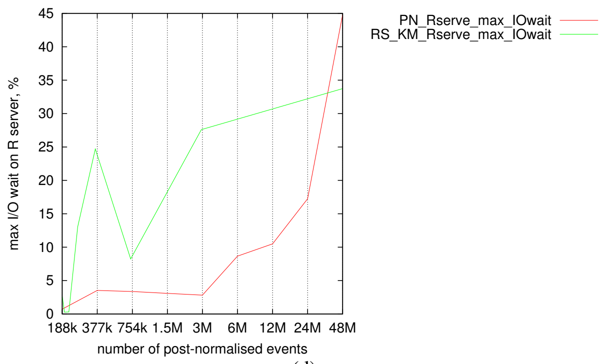
(a)



(b)



(c)



(d)

Figure 9. CPU usage and I/O wait during performance measurements

were able to process 48 million events within 5 minutes using Anomaly Detection or K-Means algorithms. In addition we compared the two approaches — SAP Predictive Analysis and R integration — in extensibility and usability. The result was that the SAP Predictive Analysis provides a simple user interface, where it is possible to create processing sequences with drag and drop of processing objects. Finally one only needs to connect the objects specify some parameters and the result is visualized as well. The disadvantage of the Predictive Analysis is the limitation of algorithms, since there is no possibility to extend the number of algorithms or integrate own solutions.
On the other hand the R integration requires more knowledge about configuration and the algorithms themselves before one could get a result. That means in terms of usability R has some disadvantages. But with R one could use every library, function, or machine learning algorithm which could be written in R. So with R one could benefit from a much wider variety of algorithms.
Therefore, we have tested and proved the efficiency of our approach and think that our results are promising. The resources of Future SOC Lab allow us to analyse billions of security events in a relatively short time interval and we hope to continue our project to create faster and reliable software for multivariate security analysis in its second phase.

References

[1] Andrey Sapegin, David Jaeger, Amir Azodi, Marian Gawron, Feng Cheng, and Christoph Meinel. Hierarchical Object Log Format for Normalisation of Security Events. In *Proceedings of the 9th International Conference on Information Assurance and Security*, IAS '13, 2013.

[2] Predictive Analysis Library. `http://help.sap.com/hana/SAP_HANA_Predictive_Analysis_Library_PAL_en.pdf`.

[3] Oracle Data Mining. `http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/index.html`.

[4] SAP Predictive Analysis. `https://help.sap.com/pa`.

[5] Binary R server. `http://rforge.net/Rserve/`.

[6] Kernel-based Machine Learning Lab. `http://cran.r-project.org/web/packages/kernlab/index.html`.

[7] SAP HANA R Integration Guide. `https://help.sap.com/hana/SAP_HANA_R_Integration_Guide_en.pdf`.

Multilevel Requirements Coverage for Discrete Manufacturing

Frank Morelli
Pforzheim University of Applied Sciences
Tiefenbronnerstr. 65
75175 Pforzheim
frank.morelli@hs-pforzheim.de

Jörg Hofmann
Pikon International Consulting Group
Kurt-Schumacher-Str. 28-30
66130 Saarbrücken
joerg.hofmann@pikon.com

Abstract

This research project aims to provide the liaison between operative sales manager and MRP controllers with an improved information base by using new opportunities within the area of multilevel requirements coverage based on SAP ERP on HANA. It focuses on the design and the implementation of a practice-oriented solution which easily allows understanding the current situation in adequate depth. As a concrete result, a report for managing multilevel requirements even for low-level materials is presented. Because the implemented result is based on a "classical" predecessor (which stands for an ABAP program with no optimization based on the HANA platform) performance questions in this context are discussed as well.

1 Introduction

Innovative approaches for business process optimization are a major concern of practice-oriented research. Several surveys show that increasing operational efficiency as well as value contribution using IT solutions are highly ranked in practice. [1] The (exception) handling of single cases reveals itself as a growing challenge for operative business process management. Corresponding decisions turn out to be core competencies for a company.

The focus of this project is to provide adequate information for sales representatives collaborating with MRP controllers of the operative management level. Therefore improved means within the area of multilevel requirements coverage based on SAP ERP on HANA are applied. The intention is to handle a realistic situation by using a generated generic prototype solution. This is the reason why the discrete manufacturing industry has been chosen as field of action.

2 Business Scenario

The following subchapters describe the research project from a business perspective. They cover a description of the chosen industry as application area and a characterization of the situation within a company intended to be improved by an SAP ERP solution.

2.1 Discrete Manufacturing

Companies within the discrete manufacturing industry can be categorized by the production of distinct and therefore countable items. They contrast to other branches especially like process manufacturing.

A typical example for discrete manufacturing is the production of automotive components, based on existing bills of material and routings. Corresponding firms act in a global market and face the challenge of reduced product lifecycles and / or customer loyalty. High quality and excellent service have to be pushed forward. This environment requires a permanent improvement effort: Main drivers for the companies are the continuous optimization of process efficiency and the sustainable reinforcement of cross-side internal collaboration as well as the partnership with external suppliers and third parties.

Within the field of practice there is a broad spectrum of options: Items can be produced in low volume with high complexity or high volumes of low complexity. Low volume / high complexity production results in the need for an extremely flexible manufacturing system that copes with the quality requirements of the customers and appropriate time-to-market speed while cutting costs. High volume / low complexity production typically leads to focusing on inventory controls, lead times and reducing or limiting materials costs and waste.

The processes deployed in discrete manufacturing are not continuous in nature. Each manufacturing process can be individually started or stopped and typically runs at varying lot sizes. As an input the final product often needs purchased units and / or parts that have to be transferred between several plants within the company.

As a concrete example the fictional company Global Bike Inc. (GBI) 2.2 has been chosen. In general it can be compared to the IDES, the "International Demonstration and Evaluation System", created once by SAP to demonstrate various business scenarios executed in an SAP ERP system. GBI is used for the SAP University Alliances (UA) program as a basis for higher education accreditation purposes. It contains application data with many realistic characteristics for a bike-producing firm and the business processes are designed to reflect real-life business requirements (e.g. the order-to-cash process and the procurement-to-pay handling).

GBI has a complete story attached to it and comprises two companies located in the US and in Germany, and a material spectrum including trading goods, raw materials, semi-finished goods, and finished goods. The actual detailed content, which requires license and use of SAP software to function, is available for SAP UA members.

For the research project the existing data structure will be enhanced for scenarios of make-to-stock production (production with no customer relation) as well as make-to-order processes.

2.2 Multilevel Requirements Coverage

This research project aims to provide sales managers as well as their counter-parts, the MRP controllers, of the operative management level with improved reporting. Therefore new opportunities within the area of multilevel requirements coverage based on SAP ERP on HANA have been chosen.

The SAP SCM approach would have been an alternative. However, it is not the idea of the project to search for further optimization heuristics or algorithms in this particular field. The paradigm is rather to support human beings from different business departments in decision making because of the highly increasing demand for flexibility within standardized business processes in practice: customer order management, purchasing, stock transfer, production, and billing have to be met in a holistic perspective. The behavior of the customers in changing quantities, schedule lines, materials, rejecting or even adding further sales order items are the cause for corresponding activities.

The concern of material requirements planning is to guarantee material availability, that is, to procure or produce (for sales and distribution on the one hand as well as for internal purposes on the other) the correct materials in adequate quantity and quality in time. Hence the requirement coverage elements for all low-level material numbers have to be identified. These comprise planned orders, production orders, purchase requisitions, purchase orders, advanced shipping notifications, physical stocks (in several plants), and scheduling agreements.

Beyond the creation of proposals and inventory monitoring especially changes during run-time operations (e.g. concerning a specific sales order item or several sales orders as well as current delivery conditions concerning the suppliers) prove to be a major challenge. The existing SAP ERP standard multi-level order report transaction (MD4C) can only be used for single sales order items and turns out to be less performant in action without using in-memory technology. In practice an MRP run is often performed once a day. Using the opportunities of the HANA platform shorter frequencies (e.g. MRP runs once an hour) are possible and offer improved decision making.

The design and the implementation of a solution which allows both a sales manager and a MRP controller to easily understand the current situation of requirements´ coverage in adequate depth act as project focus. The proposal proves to be coherent to innovations driven by SAP itself, specifically the business process solutions for MRP challenges by using fast, simple, and cross-plant MRP (see SAP Business Suite powered by SAP HANA Fact Book: Manufacturing). The result can be used globally within the SAP UA. From a practitioner´s perspective the generic prototype solution within the discrete manufacturing industry offers extensible transfer benefits.

3 Crystal Ball Report

The concept of the multilevel requirements coverage report is to create a program that forecasts for schedule lines from different customer orders (chosen by human agents) if the delivery can take place right in time or not, taking into consideration the physical stock situation and the planned requirement coverage elements. Furthermore, the sales order item value is broken down on schedule line item level to identify the schedule line items with the highest expected turnover which deserve the highest attention.

The corresponding requirements coverage elements are automatically identified across all BOM levels. Depending on the requirements coverage elements a status will be set by the system that describes the probability of fulfilling the requirements for the shipment in time. Example: The existence of the complete required quantity as a physical sales order stock for the finished product gives a 100% chance to deliver in time, whereas the existence of just a planned order is less reliable.

It is possible that comparable solutions exist beyond the SAP ERP standard as add-ons in practice. Generally the question is how companies migrating from SAP ERP to SAP ERP on HANA have to handle existing additional solutions regarding the performance.

3.1 Concept

The report is provided by a separate transaction. For the analysis, driven by human agents, the orders on hand, in terms of quantity as well as on a value basis, are required in relationship to a timeline.

To quantify the orders on hand the system automatically generates the difference between the confirmed quantity and the invoiced quantity (qoh). The corresponding value (voh) is calculated in local currency:

voh = item value * qoh / (order quantity of the item level)

Regarding the timeline the orders on hand are identified per schedule line by the confirmed date of the customer order. If a customer order is fully delivered without being invoiced the confirmed date is used as well.

For identifying the scheduled lines the system refers to different determination procedures:

- Scheduled lines relevant to material requirements planning: The main functionality is based on the SAP standard transaction MD4C (multi-level customer order report). Furthermore the crystal ball report refers to the related pegged requirements.
- Third-party deals: In general they have no MRP-reference. Therefore the SAP standard transaction MD4C is not applied. Requirement coverage elements are identified instead by using the directly related purchase requirements, purchase orders, and advanced shipping notifications. In this constellation the existing goods receipts as well as the invoice receipts have to be considered.
- Further schedule lines: All other cases as the two mentioned above are handled here. The identification of requirement coverage elements does not take place and a statement about the probability for a shipment in time by creating a corresponding status is not possible.

The report itself offers the following options for analysis:

- Customer orders with expansion of the lower-level requirements.
- Receipt elements with reverse calculation for which requirements have to be covered.

3.2 Implementation

Mass data (e.g. materials and vendors, bills of material, customers, and sales orders) for the ERP system has been permanently generated with the help of a separate program. On a daily basis 900 random changes in schedule line items of sales orders were created. Furthermore the availability check was customized within the configuration of the SAP ERP system ("Carry out Control for Availability Check").

General idea for the realization has been that the analysis of the report is available by using an ABAP report. The complexity of the functional specification in subchapter 3.1 for the crystal ball report revealed to be very high. Nevertheless the corresponding ABAP program could be established based on a predecessor from a former customer project of the PIKON International Consulting Group. This program had been created for SAP ERP without HANA and was installed during the research project within an IDES environment separate from the HPI configuration in parallel.

The existing situation gave room for performance considerations: What happens to SAP ERP customers when migrating to SAP ERP on HANA regarding existing ABAP add-ons?

To pursue this question an excerpt of the overall crystal ball report was created. The problem was reduced to the following scope: Starting from a make-to-stock scenario all production orders which are affected by a concrete change of a customer order (e. g. change of quantity) have to be identified. Thus an intervention in the current production would be possible.

The excerpt has logically been realized as an ABAP report which reads the SAP change document tables. The user has several options for varying the input parameters to find out which customer orders have been changed. Output variables for this report are sales document number, sales order item number, material number, production order number, plant, change type, new value, old value, and change date. This set can be enhanced e.g. by volume of the customer order, volume of the production order, and further customer data.

During the implementation phase several alternatives have been performed:

a) Creation of a "classical" ABAP program where the calculation (especially the processing of the change documents tables CDHDR and CDPOS) mainly takes place within the application layer of the SAP ERP system. In contrast to ERP solutions based on relational databases this configuration uses no cluster tables. A comparison to an external IDES system with a relational database showed promising performance benefits of SAP ERP on HANA. However generalizations prove to be difficult because of the different system configurations.
b) "Code pushdown" design by using a HANA stored procedure: The access to two attribute views was enabled at one time via SQL script. For the other variant Calculation Engine Plan Operators (CE functions) were implemented.

Performance checks show as expected that the conventional ABAP solution exposes to be the slowest alternative. On the contrary using stored procedures within the SAP HANA database has the anticipated benefits.

The comparison between the HANA stored procedure variants (CE functions vs. SQL script) doesn´t lead to considerable differences concerning the runtime behavior. A discussion with experts from SAP has revealed the related causes: The status of the available system for the research project is ERP 6.0, enhancement package (EHP) 6. As within the ERP constellation at the HPI a standard ABAP function module (MD_SALES_ORDER_STATUS_REPORT) is employed the performance gain by using CE functions is not as high as desired. In the meantime SAP offers EHP 7 with solutions for this problem. The new SAP standard has replaced ABAP function modules by HANA procedures.

For companies with lots of existing ABAP add-ons this means there is no silver bullet for a migration. In general performance gains can rather be reached by replacing depth-first search into breadth-first within procedures. E.g. the reading of one material record after the other should be replaced by collecting all materials within a low-level code in one step in order to minimize the number of database calls.

4 Résumé and Conclusions

The research project demonstrates that SAP ERP on HANA has the clear potential to provide practitioners with faster and better information in the sales and MRP intersection. This is highly to appreciate regarding the overall market requirements for flexibility and the quality of decision-making in a short period of time.

The described solution is focused on the operational level. Further considerations regarding strategic aspects for companies are possible. E.g. a historical view on the coverage situation could be provided with opportunities to detect relevant causes and / or typical constellations for upcoming problems. The perspective then is not the concrete process but the analysis of the constellation of customers, regions, time spans, and so on. Monitoring and predicting customer behavior could be a valuable mean for proactive management.

References

[1] Comp. e.g. Jaerschke, C.: DSAG CIO Umfrage TCO & Wirtschaftlichkeit. Ergebnisse 2013. S. 1- 69, PDF presentation at DSAG-Congress 2013: Prozesse im Wandel – Fiktion oder Fakt? 17-19. September 2013, Nürnberg, internet download from April 07, 2014, p. 12

Acknowledgements

The authors would like to thank to the active collaboration with SAP, to the involved students (Giuseppe Sicorello and Leonid Weidner), and especially to the experts of PIKON Deutschland AG (Daniel Bauer, Julian Buchholz, Benjamin Duppe, Sascha Gerhardt, Fatjon Hoxha, Stefan Kerl, André Klos, Maximilian Layes, Fabian Mosbach, Rauno Müller, Jürgen Neubronner, Christian Schlömer).

Large-Scale Hashtag Recommendation for Arbitrary Texts

Toni Gruetze
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
toni.gruetze(a)hpi.de

Gary Jiarui Yao
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
gary.yao(a)student.hpi.de

Gjergji Kasneci
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
gjergji.kasneci(a)hpi.de

Felix Naumann
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
felix.naumann(a)hpi.de

Abstract

Learning models based on continuous data streams is a major challenge in machine learning. This project deals with hashtag recommendation based on the content of linked webpages the microblogging platform Twitter.
This project aims at evaluating the horizontal scalability of a distributed implementation of an incremental machine learning model in the 1,000 *Core Cluster Future SOC Lab.*

1. Hashtag Recommendation

In recent years, collaborative tagging in Web 2.0 services such as del.icio.us and blog-spot.com has emerged as an efficient way to organize large collections of documents. The principle idea is that documents are collectively labeled with freely chosen categories (tags) by users. Recent research has been done to automatically recommend appropriate tags for an unknown document. These recommendations support the user and facilitate the organization of documents by applying more concise tags. However, tags found in such collaborative tagging systems are inherently diverse. This makes automatic tag recommendation a challenging task.
On Twitter, one of the most frequented microblogging service, hashtags have emerged as a means of classifying shared content. Because hashtags are hyperlinked to search results of equally annotated tweets, they are an important means for grouping tweets according to topics. Furthermore, the hashtag usage has proven to be guided by current events. Therefore, building an recommendation model that inherently models temporal changes of the tags is essential.
In October 2013 Twitter reported that the community, generates 500 million tweets (content shares) per day, on average. In contrast to previous work, we are investigating the link-sharing and tagging behavior of Twitter users, and are not interested in the tweet text itself. This sharing behavior is analyzed and following used to recommend hashtags for new documents. Hence, we are interested in English tweets that contain hashtags and a link to an external text resource, each of these pairs is following called alignment. Experiments showed, that the Twitter community creates over 200 of these alignments per second. However, in peak times, this throughput might even multiply. In this Project, we investigate the horizontal scalability distributed implementation of our hashtag recommender system “Array of Language Models” (ALM), that is introduced in Section 2.1. The horizontal scaling is necessary, because the data throughput of over 200 alignments per second exceeds the capabilities of current single node server systems. In conclusion, the three major challenges for the model are:

Topicality The meaning or topic behind all hashtags have to be learned, such that the model is capable of identifying them in new documents.

Diversity Hashtag might be ambiguous and thus cover different topics, whereas topical overlaps between different hashtags are possible.

Scalability The model has to be capable of tracking changes in the Twitter community behavior. Hence, it has to quickly update the meanings and topics of hashtags, to enable appropriate recommendations.

In this work the focus lies on the latter challenge. In the next section we introduce our Model “Array of Language Models” and following present some results of our scale-out experiments in Section 3.

2. ALM – Array of Language Models

The hashtag recommendation approach ALM is based on probabilistic language models. ALM assumes that every hashtag represents a topic, which has a characteristic probability distribution of terms. Further, documents containing certain topics follow a mixture of the probability distribution of the respective topics. To identify appropriate topics (i.e., hashtags) for a document, we find the probability distributions that the text most likely originated from.

2.1. Model

The intuition of ALM can be justified using Bayes' theorem as follows. Given a query document q, we are interested in the probability that hashtag h aligns with q. Following the Bayes' theorem, this probability can be written as the conditional probability of $P(h \mid q)$. To recommend hashtags, ALM computes $P(h \mid q)$ for each hashtag and returns the k hashtags with highest conditional probability. Due to the restriction to ranking hashtags, the evidence can be omitted, leading to:

$$P(h \mid q) \propto P(q \mid h)P(h).$$

The class prior $P(h)$ can is proportional to the frequency of the hashtag alignments. Hence, for each hashtag the only number of alignments $freq_h$ has to be tracked. To compute $P(q \mid h)$, language models can be employed, which are essentially probability distributions of terms. There are different types of language models. However, one of the most commonly used is the unigram language model, under which a document is modeled as a bag-of-words $\{|t_1, t_2, \ldots, t_n|\}$. The unigram language model assigns a probability to a document as follows:

$$P(t_1, t_2, \ldots, t_n \mid h) = P(t_1 \mid h)P(t_2 \mid h) \ldots P(t_n \mid h)$$

Under the unigram language model, the probability of a term being generated given hashtag h is obtained independently from other terms using maximum-likelihood estimation:

$$P(t \mid h) = \frac{freq_{t,h}}{|\mathcal{D}_h|}.$$

$\mathcal{D}_h$ denotes the bag union of terms appearing in all documents that align with hashtag h and $freq_{t,h}$ denotes the number of occurrences of t in $\mathcal{D}_h$, whereas the bag semantic gives repetitive alignments more weight.

Note that the actual value of $P(t \mid h)$ does not have to be materialized at all times because it can be computed ad hoc from absolute values during recommendation. More precisely, to represent the language model for hashtag h, for every term t only the absolute term frequency $freq_{t,h}$ and the sum of the all term frequencies $|\mathcal{D}_h|$ have to be stored.

Hence, the necessary features for the algorithm are threefold: the number of its alignments per hashtag $freq_h$, the number of occurrences of each term contained in any document $freq_{t,h}$, and the number of all terms occurring in the aligned documents $\mathcal{D}_h$. The number of stored numerical values per hashtag is dependent on the number of (unique) terms n ($size(h) = n + 2$), whereas n might vary between one and tens of millions.

2.2. Incremental Model

As mentioned earlier, Twitter is used as a communication tool to exchange messages about current events in the real world and therefore, the set of topics on Twitter are under constant development.

ALM maintains a language model for every hashtag. A hashtag represents some topic, which in turn has a characteristic distribution of terms. However, as time progresses, the topics behind hashtags can change. Therefore, the probabilities that certain terms are generated by the language models should be adapted over time. To keep the language models up-to-date regarding latest happenings on Twitter, newly generated alignments are fed into ALM. Adding a new alignment (h, d) causes document d to influence the probability distribution of the language model for hashtag h. As a result, the conditional probalility of a hashtag given a query document $P(q \mid h)$ has to be reestimated. Recall that $P(t \mid h)$ and $P(h)$ can be estimated by the three types of absolute frequencies $freq_{t,h}$, $|\mathcal{D}_h|$, and $freq_h$. Therefore, aligning a new document to the language model with a hashtag implies adding the term frequencies of the new document to the previous term frequencies and increasing the alignment frequency of the hashtag.

When more and more documents are added to a language model, $freq_{t,h}$ and $\mathcal{D}_h$ naturally grow larger. Eventually, addition of subsequent documents has smaller impact on the probability distributions of the language models. As mentioned previously, the set of topics on Twitter is under constant change which can result in hashtags changing their meaning over the course of time. For instance, the hashtag *#obama* was commonly used for texts about possible participation of Barrack Obama at the ice-bucket challenge in the summer of 2014. However, the terms "ice-bucket challenge" are not specifically relevant for the hashtag anymore. A topic change of a hashtag is referred to as *topic drift*. Because the content of old documents eventually will not fit into the current topics of Twitter, removing them is a reasonable practice. If old documents are removed, newer documents have a higher chance of influencing the probability distributions of terms. To remove a document, its term frequencies need to be subtracted from the language model (namely $freq_{t,h}$, $|\mathcal{D}_h|$, and $freq_h$). Note, the different update strategies leading to up-to-date models with a hitrate of over 90% are discussed in a separate work [1].

2.3. Distributed Model

As mentioned earlier, Twitter users share on average 500 million messages on a daily basis, leading to 200 shared Links aligned with a hashtag. This number might even increase in the future. Therefore, the model has to be able to deal with a large amount of updates. We propose a distributed version of ALM, such that it is capable to deal also with future requirements. Concurrent hashtag recommendation can be incorporated into ALM in a straightforward way. Recall that to recommend hashtags for a document q, all hashtags are ranked by $P(q \mid h)P(h)$, i.e., the language models' probability of generating q and the class prior for h. For each hashtag h, this is a task on its own and can be executed concurrently.

Implementation-wise, we decided to distribute ALM using Akka[1], an open-source toolkit for distributed applications on the JVM. The dominant approach for concurrency in Akka is the actor model. An actor encapsulates state and behavior analogously to objects in object-oriented programming, and communicates with other actors exclusively by message exchange. If an actor sends a message, it ends up in the mailbox of the respective receiver actor. In Akka, actors are managed by an *actor system*, which is a structure that is started on every physical machine. The actor system is responsible for management tasks including scheduling actors to operating system threads.

Figure 1 depicts an overview of ALM's distributed architecture. Every machine (node) in the cluster runs an actor system on which initially an actor called the *node supervisor* is created. Additionally, one node is designated as the *master node* which is responsible for distributing hashtag recommendation requests to other nodes in the cluster and reading from the Twitter stream. In particular, the master node communicates with other cluster nodes by sending messages to their node supervisor. Hence, our architecture follows a master-worker pattern for work distribution.

Master Node The master node runs an HTTP server through which hashtag recommendation requests are received. Incoming hashtag recommendation requests are forwarded by a recommender actor to all node supervisors whose partial recommendation results are aggregated and sent back as a JSON response.

During initialization of the master node connects with the worker nodes and maintains a list of actively connected workers. Let W be the number of connected workers. Subsequently, a dedicated *stream reader actor* is created, which reads from the Twitter Public Stream. When the stream reader actor is faced with a tweet that contains hashtags and URLs, it sends a request message to the distributed webpage downloader module (not focus of this work) and recieves the downloaded content. Next, for each alignment (h, d), the stream reader actor computes an integer hashcode for

[1]`http://akka.io/`

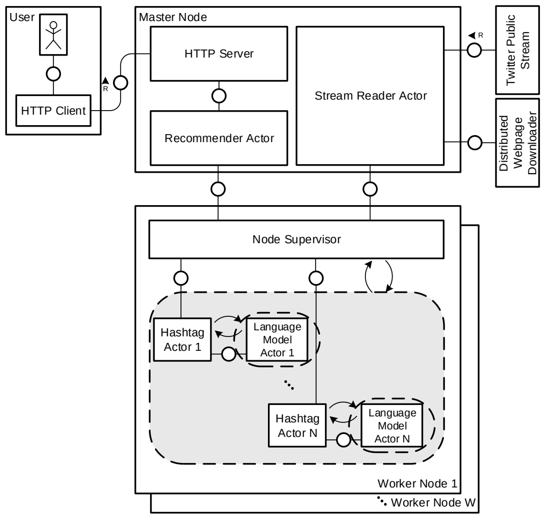


Figure 1. Architecture of the distributed ALM

h and sends the alignment to the node supervisor running on the i-th node, where $i := \mathit{hash}(h) \mod W$.

Worker Node Every worker node runs a node supervisor, which receives the alignments sent by the stream reader actor and takes hashtag recommendation requests. In the following we describe how incoming alignments are handled. The node supervisor maintains a mapping from hashtags to actor references. If h is new (not seen previously), a so-called hashtag actor is created. If h already exists, the exiting hashtag actor is used. The node supervisor sends the alignment to the hashtag actor, which in turn forwards the alignment to the assigned *language model actor (LM actor)*. The LM actor is responsible for updating the language model of the hashtag, and synchronizes the language model with its parent hashtag actor periodically. The separation between hashtag and LM actor prevents blocking due to many alignment writes. Blocking should be avoided because the hashtag actor is also responsible for computing the probability that a query document is generated by its language model.

If the node supervisor receives a hashtag recommendation request from the recommendation actor, the request is forwarded to all hashtag actors, which compute the probability of generating q and send their results back to the node supervisor. The best k hashtags ordered by probability of generation are chosen by the node supervisor and send to the recommendation actor running on the master node. Finally, the recommendation actor selects the k best hashtags chosen from all partial recommendation results sent by the node supervisors.

3. Results

We evaluated the scalability of the distributed ALM implementation on the $1,000$ *Core Cluster* of the HPI Future SOC Lab. The cluster consists of 25

Quanta QSSC-S4R machines, each having 4 Intel Xeon E7- 4870 @ 2.40GHz CPUs (40 cores in total) and 1024GB of RAM. The nodes are connected through 2 $\times$ Intel Corporation 82599EB 10-Gigabit network adapters.

3.1. Write performance

To evaluate the scale-out capabilities of the distributed version ALM with respect to its write performance, a collection of approximately one million alignments was collected from the Twitter public stream. The collection was then sent to the stream reader actor to simulate a typical stream of Tweets. We further included the linked document content to the collection, because this work shall omit the temporal overhead for requesting/downloading external web resources. The stream reader then send the alignments to the responsible node supervisor, which in turn forwards it to the appropriate hashtag actor. We measured the time needed to write all alignments to the model. We repeatedly run the experiment (5 times) and averaged the results.

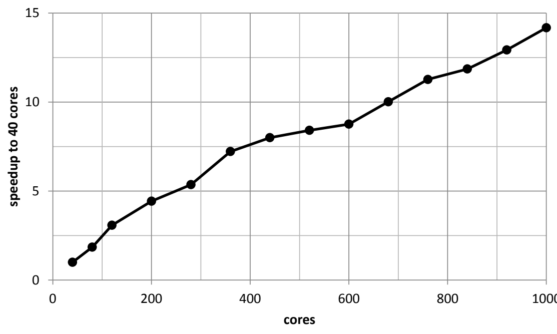


Figure 2. Speed-up of write operations in comparison to a 40-core configuration

Figure 2 shows the relative speedup of different cluster configurations in comparison to a one node setup with 40 cores. The 40 core configuration is already able to process approximately 130 alignments per second, were the complete collection is written in 2h 1min. Using 25-times more resources ($1,000$ cores), the throughput increased to approximately 1.866 alignments per second, which is a speedup of 14.2. Hence, writing new alignments to a distributed version of ALM scales-out nearly linearly, with an parallelization efficiency is approximately 57%. The difference to an ideal speedup is explainable by the additional communication overhead between the different nodes. Furthermore, the usage of hashtags follows the power law. Hence, commonly used hashtags have an increased update rate, yielding to a bottleneck due to changes of their language model. It remains to show, that the write performance behaves similar with a higher amount of nodes and thus an even increased network overhead.

3.2. Read/Write performance

Next, we try to evaluate the recommendation performance of the distributed ALM, with concurrent writes. This experiment shows the actual throughput of recommendations created by the system. The model is first trained with one million alignents, and afterwards continuously updated with further alignments with a maximal write throughput, while repeatedly triggering $1,000$ recommendation requests for different documents. Then, the resulting time for all recommendation requests are averaged. Again, we repeatedly run the experiment (5 times) and averaged the results.

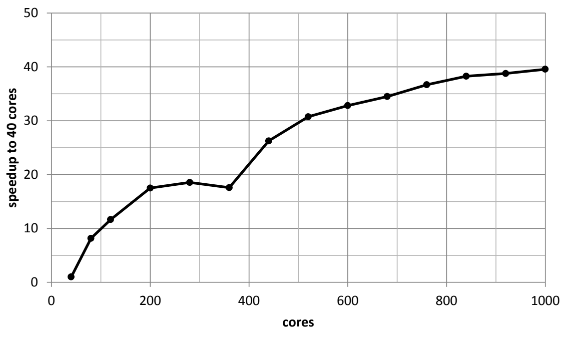


Figure 3. Speed-up of concurrent read and write operations in comparison to a 40-core configuration

Figure 3 shows the recommendation throughput of ALM. Due to the encapsulated structure of the LM actors, the read performance is not significantly influenced by concurrent writes. Most interestingly, the speedup for the recommendations is super linear to the used resources. Starting with a throughput of 3.8 recommendations per second for a single node (40 cores) configuration, the throughput reaches a value of 148.7 recommendations per second for the 25 node ($1,000$ cores) configuration. This is explainable by increased cache effects for the hashtag models on the distributed configuration. Given the fact, that fewer hashtag models are held per node, an increasing number of cache hits can be expected. Another interesting effect occurs in case of the 360 cores (9 nodes) configuration. We are not aware of the actual reason for the performance drop, but we suppose this drop originates from topological peculiarities of the server infrastructure.

4. Conclusion

In this project, we showed that ALM, a probabilistic model for recommending hashtags for text documents based on the tagging behavior of the Twitter community, is capable of learning the tagging behavior of the Twitter community (i.e., from the Twitter Firehose) while simultaniously recommending hashtags for previously unseen documents.

References

[1] G. J. Yao. Large-Scale Twitter Hashtag Recommendation for Documents. Master's thesis, Hasso Plattner Institute, Potsdam, Germany, 2014.

Open Government Data Integration with Stratosphere on the FutureSOC 1000-core cluster

Arvid Heise
Hasso-Plattner-Institut
arvid.heise@hpi.de

Felix Naumann
Hasso-Plattner-Institut
felix.naumann@hpi.de

Abstract

Integrating data from multiple data sources enhances their value for businesses and organizations. In this project, we integrate large Open Government datasets to find interesting relationships between politicians and companies, such as potential cases of nepotism.

We devised data integration operators for the parallel data analysis framework Stratosphere, which we evaluate on the 1000-core compute cluster of HPI's Future SOC Lab. In particular, we compare the scale-up to the scale-out capabilities of Stratosphere and our implementations.

1 The power of integrated Open Government datasets

In today's business landscape, data plays an important role – either directly as an asset, most prominently seen in Google, or as the main driver for business decisions. Data is typically collected through several sources and applications, such as customer relations tables, sales reports, or data derived from suppliers. For high tech companies, the acquisition of data is one of the main motivations for buying other companies; for example, when Facebook acquired WhatsApp in 2014 for 19 billion dollars[1].

However, hoarding data does not immediately help an organization. According to the Data Warehouse Institute, poor data quality costs US businesses $600 billion dollars in the early 2000s [2]. Therefore, the quality of the data must be constantly monitored and maintained. New datasets must be carefully integrated in the data warehouse of the organization to increase the value of the data and justify the supposedly expensive acquisition.

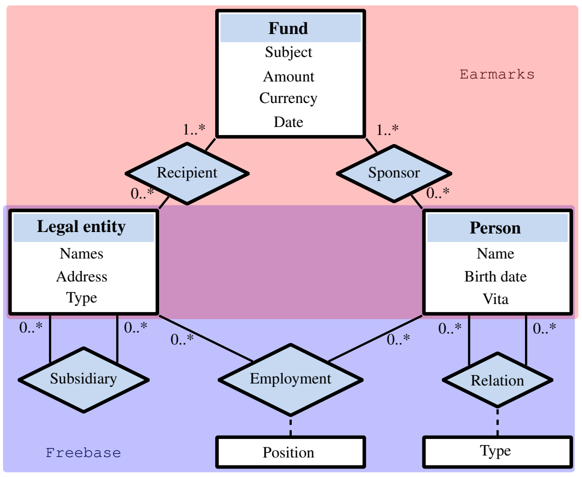


Figure 1: Entity-relationship model for the running example.

In this project, we integrate large Open Government datasets with our data integration operators implemented in Stratosphere[1]. A possible usage of the integrated dataset is to find interesting relationships between politicians and companies, such as potential cases of nepotism [3]. Figure 1 exemplarily shows the resulting data model of an integration of the US Earmarks[2] data source and Google's Freebase[3].

The first data source contains earmarks: Personal spending of a US congress member to an organization. We extract information about the receiving legal entity, the enacting congress member, and the fund itself.

To detect suspicious cases, we lack information about possible connections between the recipient and the sponsor. In Freebase, we find familiar relationships, employment records, and subsidiary information.

We now need to integrate the two data sources to find suspicious circular relationships: A congress member enacted an earmark that benefits a legal entity, at which the congress member or a relative is employed.

[1]http://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/

[2]http://earmarks.omb.gov/earmarks-public/
[3]http://freebase.org

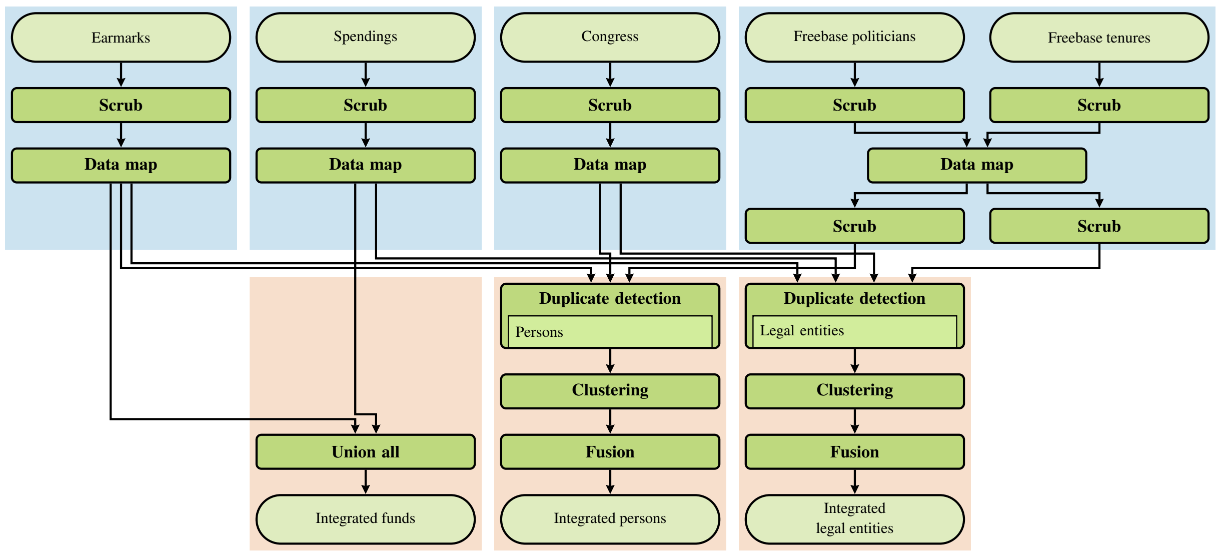


Figure 2: Data integration query for scalability evaluation.

2 Integration process

Data integration consists of several subtasks with different complexity, which addresses different types of heterogeneities. In the following, we briefly present each subtask for our Open Government Data integration query summarized in Figure 2, which integrates a total of four data sources.

2.1 Data scrubbing

One of the most underestimated challenges in the integration of several data sets is systematic heterogeneity on the value and record level. For example, differently abbreviated street names render matching techniques of records more complex as they need to be able to detect such differences and at the same time avoid being too lenient, which would results in many wrong matches.

Our data scrubbing operator allows users to declaratively define constraints on the values and repair functions that correct violations. In the script excerpt below, each politician needs to have an id, name, and party. Any record without id and name are discarded, while missing parties are repaired with a default value. Further, names are split and titles are normalized with a dictionary.

```
$pol scrubbed = scrub $politicians
  with rules {
    id: [required, type(numeric)],
    name: [required, &splitName],
    party: required or default('unkn.'),
    title: in($officialTitles),
  };
```

2.2 Data mapping

Users declaratively specify the mappings of attributes of the source relations and the target relations. Our operator finds the minimal number of transformation operators that aligns the schemata accordingly.

For example, the previously normalized and split name is now assigned to separate attributes of a person. Further, our operator supports nested expressions and arbitrary cardinalities as can be seen in the `worksFor` relationship to the parties.

```
$politician, $party = map data of
  $p in $pol scrubbed, $t in $tenure
  where $pol scrubbed.party == $t.id
  into [
    entity $politician
      identified by $p.id
      with {
        firstName: $p.name[0],
        lastName: $p.name[2],
        worksFor: [{
          legalEntity : $t.party
        }]
      } ...
  ];
```

2.3 Record linkage

The most compute-intensive and hardest part of the data integration is to find corresponding records across data sources. It usually comprises three tasks:

- Select candidate record pairs that have a high probability of representing the same real world entity.

- Apply sophisticated (string) similarity measures to the candidate to decide if it is an actual match.
- Cluster the matches to a consistent, transitively closed result. This step usually means the addition of more matches, but may also involve the deletion of borderline cases.

The following script performs all three tasks. It defines a weighted, composite similarity measure, which compares first, middle, and last name. It configures a 2-pass Sorted Neighborhood Method on first and last names. It also applies a transitive closure to the result.

```
$duplicates = detect duplicates $p in
    $earmarksPersons, $fbPersons
  where
    intDiff($p.birthDate.year) == 0
      if $p.birthDate and
    ( 2 * jaroWinkler($p.firstName) +
      2 * jaroWinkler($p.lastName) +
      1 * jaroWinkler($p.middleName))
    ) / 5 > 0.8
  sort on [$p.firstName, $p.lastName]
  with window size 20
  cluster with 'transitive closure';
```

2.4 Data Fusion

Finally, for each cluster of representations of the same real-world entity, we want to obtain a single, consistent representation. Our data fusion operator enables users to declaratively specify conflict resolution functions for attributes. These functions decide which of the potentially conflicting values of the different representations to choose in the final representations.

```
$persons = fuse $personClusters
  with weights {
    freebase: .7
  }
  with resolutions {
    *: [mostFrequent, longest, first],
    firstName: vote(&isNickNameOf),
    birth: min,
    worksFor: mergeDistinct
  };
```

3 Results

We evaluated the scalability of our integration operators on the 1,000 core compute cluster of the Future SOC Lab. The cluster consists of 25 Quanta QSSC-S4R, each having 4 Intel Xeon E7- 4870 @ 2.40GHz and 1 TiB RAM. The nodes are connected through a 2 x Intel Corporation 82599EB 10-Gigabit network. Since local storage was unavailable for us, we simulated it with a 200 GiB Ramdisk. For HDFS, we installed Hadoop 1.13 and configured it to use the Ramdisk. To execute the scripts, we used Stratosphere 0.6 (Apache Flink pre-release) and the cleansing package 0.1[4]. We assigned 25 GiB RAM to the dedicated job manager and 100 GiB RAM to each task manager.

3.1 Integration results

To put the runtime evaluation in the next section into perspective, we first review the results of the different parts of the script. Table 1 shows the number of extracted persons, legal entities, and funds per dataset.

Source	Records	Resulting Entities		
		Person	Legal entities	Fund
US-Earmark	58,751	783	9,742	11,577
US-Spending	1,219,800	-	148,647	1,219,793
US-Congress	42,621	11,734	53	-
Freebase				
US-Politician	2,978	1,900	-	-
Tenure	72,487	-	44	-

Table 1: Extracted entities per source.

`US Earmarks` are personally sponsored by the 535 congress members in one period. However, the fact that we extracted 783 persons already indicates duplicates within this dataset. Further, we identified almost 10,000 recipient entities in over 11,500 funds. We can conclude that the average congress member enacted almost 22 earmarks, but almost no legal entity received more than one fund if we assume that the number of duplicates in the legal entities is low.

In contrast, the `US Spending` dataset exhibits less obvious data quality problems. Only 7 of the over 1 million records have been filtered through the *scrub* operator. We extracted almost 149,000 legal entities, which corresponds to 8 average funds per entity. The `US-Congress` dataset contains approximately 12,000 historic person records of congress and senate members from 53 parties. `Freebase` contains 1,900, mostly recent US politicians from 44 parties.

The duplicate detection, clustering, and fusion of *persons* resulted in 1,889 final records. In particular, we matched 1,626 politicians from Freebase and 476 persons from `US Earmark` to 1,889 entries from `US Congress`, which corresponds to 213 persons that are contained in all three datasets.

In comparison, we found only 278 duplicate *legal entities*. For `US-Congress` and `Freebase`, we extracted only parties, which do not appear in `US Earmarks` and `US Spending`. Further, politicians

[4]`https://github.com/AHeise/sopremo-cleansing`

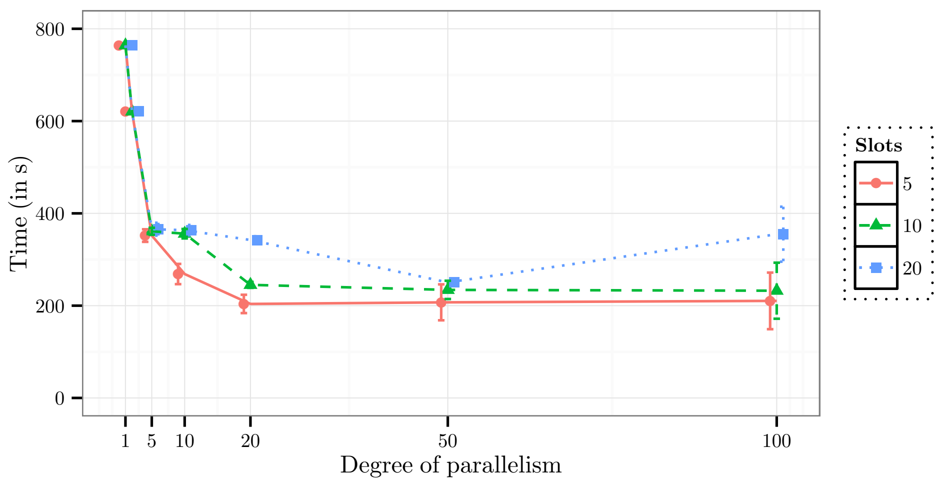


Figure 3: Runtime for the complete script executed with different degrees of parallelism and slots per node. The error bars represent the standard error.

mostly enacted earmarks for non-profit organization, which rarely appear in `US Spending`. Lastly, our conservative settings are more precision-oriented, so that we received only few false positives, but probably many false negatives.

3.2 Scalability experiments

In the first experiment, we observe the scale-out properties of the complete script. The overall degree of parallelism varies from 1 to 100 and the number of slots per node from 5 to 20. Figure 3 depicts the mean values with standard error bars. On one core, the complete script needs 13 minutes. In the best settings, the script takes 3 minutes to complete.

We can see that a higher degree of parallelism does not necessarily improve the runtime. For five slots, the runtime does not decrease after a degree of parallelism of 20. For ten and 20 slots, the best runtime is achieved on 50 cores. Surprisingly, the runtime becomes worse for 20 slots and 100 cores. Further, higher degrees of parallelism also cause more variance in the execution time as shown by the standard error.

Scalability of different operators

For our second experiment, we set the number of tasks per nodes to five. We executed each operator individually for degrees of parallelisms from one to 100. Figure 4 visualizes the average runtime for each task. The total execution time of all tasks on one core with 25 minutes is almost double as high as the execution of the complete script. We attribute the difference to the efficient physical optimization of Stratosphere and less I/O from selective materialization of intermediate data.

On one node, four tasks dominate the runtime: The *scrub* and *data map* of `US Spending` as well as the *duplicate detection* of persons and legal entities. The first two tasks mostly depend on the size of the dataset, so that `US Spending` as the biggest dataset naturally takes the longest to process. *Duplicate detection* is typically among the most expensive parts of a data integration project due to the expensive similarity measures and large search space.

When scaling out to more nodes, we see two effects: Expensive tasks become gradually faster, but cheap tasks require more time when scaling out. Both effects cancel each other after a degree of parallelism of 20, so that we cannot see an overall runtime improvement. For the future, tasks should be more intelligently distributed: If the optimizer estimates that a higher degree of parallelism does not result in a smaller runtime, it should limit the degree for the specific task. Consequently, we achieve a better overall runtime with fewer resources.

Comparison to baseline

To put these measurements into perspective, the process to create the data for the original GovWILD[5] portal can serve as a baseline. The same set of steps are performed, albeit on around double the sources and twice the total data amount. There, a mixed workflow of Jaql (on Hadoop) and Java programs have an overall runtime of approximately five hours. We suspect that the intermediate materialization of the data as well as the startup costs of the many small Hadoop jobs contribute most to the comparably higher runtime. Consequently, development cycles to improve the workflow are tremendously shortened with our approach.

[5]`http://govwild.hpi-web.de/`

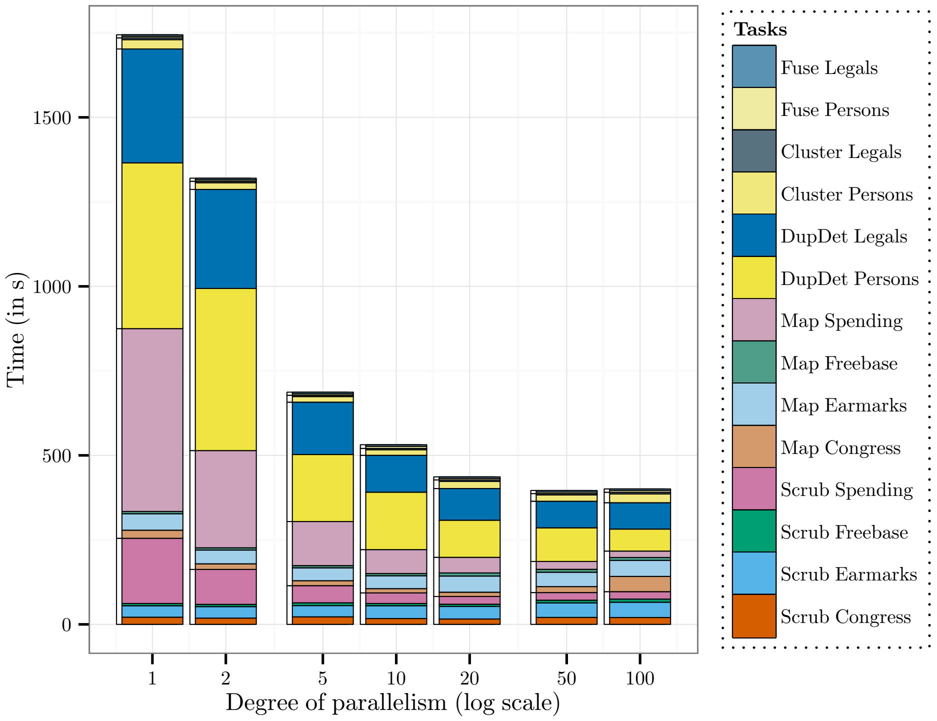


Figure 4: Runtime for the individual script executed with different degrees of parallelism.

4 Conclusion

We evaluated our Stratosphere operators on a 1000-core cluster with an Open Government Data integration project of four data sources. The evaluation challenged Stratosphere in several ways. It is the most complex query to date. Never before was the scale up behavior of Stratosphere tested on such large nodes. The tasks are more compute- and less data-intensive. For such parallel data analytics system, our data volume and execution times are often too short to fully unleash their scalability potential.

Nevertheless, the query ran reliably on up to 100 cores and exhibited good overall runtimes. For most queries, we saw a steady decrease of overall runtime for up to 100 cores. Especially, the expensive operators profited heavily from scaling out, such that users can almost interactively tweak the various parameters of the integration queries.

The integration of our data cleansing and integration operators in Stratosphere provides users with the opportunity to declaratively formulate their data integration workflows, use functionality developed in other packages, and execute them in parallel on a cluster. In particular, our operators allow users to integrate large-scale datasets in a timely manner.

Acknowledgements

We would like to thank Fabian Tschirschnitz and Tommy Neubert who worked as part of the data integration team on the Stratosphere project.

References

[1] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *VLDB Journal*, pages 1–26, 2014.

[2] W. Eckerson. Data quality and the bottom line. *TDWI Report, The Data Warehouse Institute*, 2002.

[3] A. Heise and F. Naumann. Integrating open government data with stratosphere for more transparency. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14(0):45–56, 2012.

Implications of Non-Volatile-Memory Hardware Characteristics for In-Memory Databases

David Schwalb[1], Christopher Kaufmann[2], Martin Faust[1], and Frank Ridderbusch[2]

[1]*Hasso Plattner Institute, Potsdam, Germany*
[2]*Fujitsu, Paderborn, Germany*

Abstract

Storage Class Memory (SCM) introduces a new storage technology, which combines word granular access by single CPU instructions with the advantage of being non-volatile and offering a high storage density. It is well-suited to increase the memory capabilities of individual servers efficiently to deal with increasing memory demands of applications and also offers durable storage at the level of main memory.

In this paper, we present the in-memory storage engine Hyrise-NV, which stores table and index structures directly on NVM. Our architecture enforces atomicity and ordering guarantees and performs database changes directly on NVM using multi-version data structures without the necessity of a write-ahead log.

1 Introduction

With the emergence of persistent memories on the memory bus [2, 10, 11, 14], a transaction can be made durable by updating its data in the Storage Class Memory (SCM). Thus, the need of writing a separate log to storage devices becomes obsolete and the need for grouping transactions due to the shortcomings of disks is eliminated, improving transaction latencies and the performance of the system. However, the actual hardware characteristics of future SCM are unclear and widely differ among the used technologies [10, 16]. In this project, we will investigate expected hardware characteristics of SCM and their implications on the system architecture of in-memory databases. We use the prototypical storage engine Hyrise [5, 9] and extend the architecture to natively support SCM as the primary persistence to store table data.

Contributions. We present the in-memory storage engine Hyrise-NV that stores all table data and index structures directly on SCM and enforces atomicity and ordering guarantees using multi- version data structures without the necessity of a write-ahead log. In particular, we make the following contributions:

(1) We outline the current developments regarding emerging non-volatile memory technologies, see Section 2.
(2) We present the system architecture for the in-memory storage engine Hyrise-NV, using SCM natively as the primary database persistence, see Section 4.
(3) The SCM latencies are emulated by overwriting the Serial Presence Detect (SPD) values of the DIMMs based on the project *SAP HANA in a Hybrid Main Memory Environment* [21]. We evaluate our approach using the hardware-based emulation approach on DRAM and using the TPC-C benchmark, indicating an architectural throughput overhead of up to 20% and an additional 5% overhead for memory access latencies increased by 2.3X, see Section 5.

To support this project we used access to a server with the following specifications at the Future SOC Lab: Fujitsu RX600 (4-way) with 32 cores and 512GB memory.

2 Emerging Non-Volatile Memory Technologies and Hardware

The following section gives an overview of emerging non-volatile memory technologies, their expected performance characteristics and hardware supported operations to flush volatile caches.

2.1 Overview Storage Class Memory

In todays memory hierarchy a latency gap between the volatile memory and the non-volatile storage hierarchy

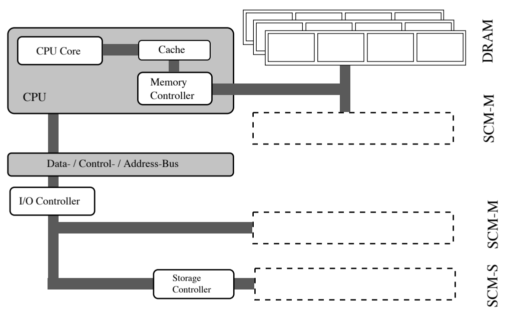


Figure 1. Attachment of SCM to the CPU

exists. The fastest non-volatile storage devices today are Solid-State-Disks that are attached via the PCIe-Bus and reach write latencies around 10μs and read latencies of about 60μs. On the other hand, the slowest volatile memory is typically Dynamic Random Access Memory (DRAM), which operates below 100ns latency, symmetrical for read- and write-access (1).
Upcoming Storage Class Memory (SCM) will be able to close this gap and provide a memory technology that combines the low latencies of volatile memory with the high densities of non-volatile-storage.
SCM can be divided into two types: SCM-Storage (SCM-S) and SCM-Memory (SCM-M). SCM-S is typically attached to the CPU via a storage controller using block access, which results in additional layers between the CPU and the SCM. Those layers need to be passed, resulting in a higher latency for storing and retrieving the data. SCM-M is attached directly to the memory controller, which enables the CPU to perform load and store operations on the level of bytes or words and is therefore able to offer the full advantages of SCM.
Besides this clear separation now devices emerge that allow not only block access, but also direct access to the attached memory. For example, the PMC-Sierra Flashtec cards use supercap buffered NV-DRAM with NAND as storage backend on power-fail that can be accessed with load and store instructions [15]. A future version of the card may replace the DRAM with Storage Class Memory. Figure 1 shows the different SCM types and their typical connection to the CPU.
There are different emerging technologies for Storage Class Memory, each with particular advantages or disadvantages. For the purpose of this project we have focused on the technologies that seem to be suitable as a non-volatile, high density DRAM expansion or even as a replacement for DRAM.
The International Technology Roadmap for Semiconductors (ITRS) lists the following properties as desirable for SCM-M [4].

(1) Cost per bit should be comparable to DRAM or better
(2) Read/Write latency below 100ns
(3) The persistence should be sufficient to survive power failure and the need for refresh power should be eliminated.
(4) It should include integrated hardware based failure remapping and ECC.
(5) The architecture should possibly be merged with DRAM L4 cache or be directly manageable.

SCM-M should also provide a data retention for more than five days without power and endurance larger than 10^9 write cycles.

2.1.1 Ferroelectric RAM (FRAM or FeRAM)

FRAM memory cells are built like DRAM memory cells, but instead of using a common capacitor, a ferroelectric capacitor is used. Ferroelectric material is able to maintain the electric polarization without an external electric field. By using an electric field, the polarization can be reversed. This behavior is used to store the data.
The positive characteristics of FRAM are a low latency, low power consumption and a high endurance. On the other side, the scalability of FRAM is not sufficient and a 3D integration and multi-layer cells seem difficult to achieve. This results in a low density, making FRAM very unlikely to be a DRAM replacement [4].

2.1.2 Spin-Torque-Transfer Magnetic-RAM (STT-MRAM)

The main component of STT-RAM is a small magnetic element called magnetic tunnel junction. It consists of two magnetic layers that are separated by an isolation layer and the information is stored in the magnetic state of one of the layers. Reading is realized using the tunneling magneto resistance effect, and the spin-transfer torque effect is used for writing.
STT-MRAM also features high endurance, low latencies and has better properties regarding scalability and 3D integration than FRAM. However, the creation of multi-layer cells seems to be difficult. Provided that a high density can be reached, STT-MRAM might become a valid DRAM challenger [4].

2.1.3 Phase Change Memory (PCM)

PCM works by using the unique behavior of chalcogenide materials. Depending on the state of the material (amorphous = high resistance and crystalline = low resistance) the electronic resistance changes which is then interpreted as a zero or a one.
Scalability, multi-layer cells and 3D integration are feasible for phase-change memory but besides the inferior values in power consumption in contrast to RRAM, there are some problems in the area of write endurance [12]. Because of the way information is written to a PCM cell (melting and slow, controlled cooling down) the write latency and the power con-

	DRAM	PCM	RRAM	STT-MRAM	FRAM
Rd. Lat.	< 10ns	12ns	< 10ns	35ns	45ns
Wr. Lat.	< 10ns	100ns	< 10ns	35ns	65ns
Wr. Endur.	$> 10^{16}$	10^{9}	$< 10^{10}$	$> 10^{12}$	10^{14}

Table 1. Overview over read latencies, write latenies and write endurance of different SCM technologies and DRAM [4]

sumption seems too high for the application as a DRAM replacement [4].

2.1.4 Resistive RAM (RRAM or ReRAM)

In RRAM a memory cell consists of two electrodes at which ions dissolve and then precipitates again. This results in the change of the electric resistance, which then can be used to store data.
To date RRAM seems to be the most promising technology family for SCM-M and different companies developed their own techniques to build actual components. In contrast to FRAM and STT-MRAM it should be possible to produce it with high density and a latency that is low enough to serve as a DRAM replacement [4].

2.1.5 Other Technologies

Besides the mentioned technologies other SCM implementations exist. NRAM$^{\text{TM}}$, a technology developed by the company Nantero Inc., is based on Carbon Nanotubes and should also show performance characteristics that are able to compete with DRAM [13]. Other technologies like Mott Memory, Macro-molecular memory and molecular memory are too early in development process to make a profound evaluation [4].

2.2 Expected Performance Characteristics

In the following, we outline expected performance characteristics of SCM, focussing on latency, bandwidth and write endurance.

2.2.1 Latency

It is expected that the latencies of Storage Class Memory will settle down in the order of between 100ns and 10μs and are therefore significantly faster than established NAND flash [3]. Some technologies, like STT-MRAM might also reach latencies that make it sufficient as a DRAM replacement. Table 1 lists the latencies for different SCM technologies and the known latency values for DRAM as a reference. It should be mentioned, that these are the values for single cells. The read and write latencies may vary between the final implementation and the coupling to the CPU.

2.2.2 Bandwidth

To replace or extend DRAM, SCM has to provide similar bandwidth characteristics that the current DDR4 memory interface provides. As with the progression from DDR to DDR4 DRAM, where specific optimization techniques were used to improve performance, it is not too farfetched to assume that alternative techniques are going to be developed to optimize the bandwidth characteristics of SCM as well.

2.2.3 Write Endurance

Write Endurance of the various SCM technologies is much better when compared to NAND flash and does in some implementations approach DRAM. For example STT-MRAM is expected to reach more than 10^{15} cycles which would be sufficient for replacing DRAM [4].

2.2.4 Expected Timeline for availability

SCM implementations that are usable in the scope of this project are currently not available, but viable alternatives to current storage technologies are expected to appear in the next years. While there are already some commercially available SCM components using PCM or FRAM technology, these are typically only suitable for the usage in small scale or embedded devices due to the limited capacities of these components. The scope of this project looks at large scale “In Memory” databases. It will take some time until SCM components with sufficient capacity have been developed.
Until the full commercial availability of SCM, bridge technologies like, for instance, NVDIMMs, which exist in different flavors or hybrid SSDs, which implement specific characteristics of SCM, are already available.

2.3 Hardware support to flush volatile caches

With Storage Class Memory volatile buffers and caches will still be required in the memory architecture and measures need to be taken to ensure that the persistent data is kept consistent. The software may explicitly and directly need to use the cache line flush instructions provided by the CPU.
Depending on the processor architecture different instructions exists:
(1) x86: clflush (Flushes and invalidates cache line)
(2) PowerPC: dcbst (Flushes a data cache block), dcbf (flushed and invalidates a data cache block)
(3) UltraSPARC: block store (flushes and invalidates)
The current implementations of the cache flush instructions do not acknowledge the intended semantics of the data (simple data vs. indexes vs. log buffers or similar). It must also be tracked, which data has to

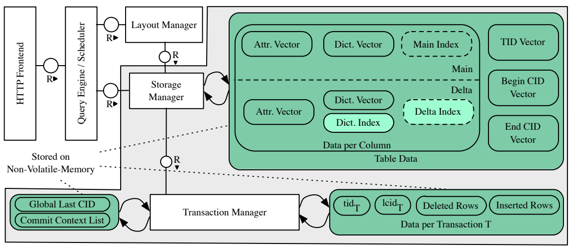


Figure 2. Hyrise architecture overview outlining data structures stored on non-volatile-memory. Highlighted data in green is stored on SCM.

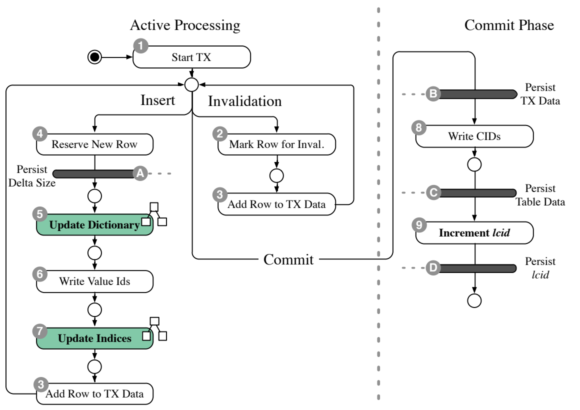


Figure 3. Additional barriers are required during transaction processing to explicitly flush data from caches.

be flushed and the flushes must provide a fine enough granularity [1]. As current CPU generations are not designed for SCM, we expect optimizations and extensions to better support SCM.

3 System Overview

Hyrise[1] is an in-memory storage engine specifically targeted to mixed workload scenarios [6] and a balanced execution of both analytical and transactional workloads using task based scheduling [23], while optimizing for the set processing nature of business applications [8]. This section briefly outlines the general system architecture and then presents indexing, concurrency control and persistency mechanisms in more detail in order to provide the foundation for Section 4 that describes architecture adaptations to move the primary data persistency to SCM.

Data modifications follow the insert-only approach and updates are modeled as inserts and deletes. Deletes only invalidate rows. Inserts keep the insertion order of tuples and only the lastly inserted version is valid. Although Hyrise supports flexible hybrid storage layouts [6], this paper focuses on the columnar storage of tables and tables are stored as a collection of columns and meta-data in main memory. Queries are formulated directly as physical query plans.

Figure 2 gives a high level overview of the system architecture. Each column consists of two partitions: main and delta partition. The main partition is dictionary compressed using an ordered dictionary, replacing values in the tuples with encoded values from the dictionary. In order to minimize the overhead of maintaining the sort order, incoming updates are accumulated in the write-optimized delta partition as described in [8,19]. In contrast to the main partition, data in the write-optimized delta partition is stored using an unsorted dictionary. In addition, the delta dictionary maintains a tree-based index with all the unique uncompressed values of a column's delta partition to allow for fast value searches and to speed up dictionary inserts [18]. The attribute vectors of both partitions, storing the dictionary encoded values, are further compressed using bit-packing mechanisms [22]. To ensure a constantly small size of the delta partition, Hyrise executes a periodic merge process to combine all data from the main partition as well as the delta partition into a new main partition that then serves as the primary data store [8].

[1]Source available at `https://github.com/hyrise/hyrise`

4 Adding SCM-Support to Hyrise

This section focuses on the changes required to the overall architecture of Hyrise to move the database persistency to SCM and the aspects of transaction management. The goal is to have the complete database state on SCM, including all table data and index data structures, in order to enable instant restarts of the system. Looking at the criteria of atomicity, consistency, isolation and durability for the transaction management of database systems, the shift of keeping the primary persistency on SCM does not interfere with the consistency of the database or isolation of transactions during runtime. However, it is necessary to carefully design the system in a way that transactional changes reach SCM in an atomic way and that all transactional changes are guaranteed to be durable at the point of a transactional commit.

In order to keep the database state directly on SCM and to be able to resume execution, the system needs the following information on SCM, see Figure 2: (i) the table data including main and delta partitions with attribute vectors and dictionaries, existing index structures and MVCC vectors, (ii) the state of the transaction manager including the global last visible commit id and commit context list as well as (iii) a transaction state per transaction and (iv) table meta-data.

Moving all data structures (i) to (iv) from volatile memory to SCM requires a careful system design to guarantee the consistency of all data structures, explicit barriers in the system to guarantee the write order on SCM, as well as mechanisms to provide atomicity and durability for transaction management also on SCM.

The basic data structures used in the system are either based on vectors or trees. All vectors in the system are designed to be append only and do not execute in-place updates of values. Appending new values is straightforward as long as no re-allocation needs to be performed. Managing consistent updates of tree-based data structures is more complicated, using a multi-versioned tree designed explicitly for storage and direct manipulation on SCM [20]. For the remainder of this section, we assume vector and tree structures that support consistent updates.

The transaction management mechanism of snapshot isolation with multiple versions of rows nicely lays the foundation of moving the transaction management to SCM. The principle of appending updated versions in combination with invalidations of old versions without using in-place updates is well suited for direct persistency on SCM. Therefore, atomicity of transactions can easily be ensured by using MVCC visibility mechanisms. Building on this mechanism, a transaction can work on its private data space on SCM until processing has finished and all changes are made visible by writing the last visible commit id. This step consists of the incrementation of one single integer and can therefore be written atomically to SCM.

Figure 3 outlines the single steps of our system during transaction processing and explicitly needed barriers to guarantee consistency on SCM. As an example, let us assume a transaction T updating a single row, resulting in one invalidation and one insert:

(1) Transaction T starts with id tid_T and the last visible commit id $lcid_T$.
(2) The row to be updated is marked for deletion locally by setting $vbeg$ of the row.
(3) The invalidated (or newly inserted) row is added to the local list of deleted (or inserted) rows from T.
(4) After invalidating the old version, the updated version is inserted as a new row. For this, an isolated write spaced owned by T is reserved in the delta. The write space consists of one value per attribute vector of the table and is maintained using a single integer representing the delta size. In case the allocated memory of the vectors is exhausted, a more complex mechanism guaranteeing atomic resizes of all vectors needs to be executed. *Barrier A* ensures that the size of the delta partition is flushed to SCM.
(5) For every column, the value is encoded as a value id using the delta dictionary. If no entry exists in the delta dictionary for the new value, it is inserted into a persistent unordered dictionary and a persistent tree structure providing a consistent insert operation.
(6) The value ids are then written into the attribute vectors.
(7) The new row is now inserted into existing tree-based index structures.
(8) When T enters the commit phase, *barrier B* ensures that the transaction context is persisted on SCM, containing a list of all inserted and updated rows. Afterwards, T finalizes all changed rows by writing the respective commit ids.
(9) Before the last step, *barrier C* persists all table data and flushes all dirty changes on attribute vectors and $vbeg$ and $vend$. Then, the global $lcid$ is incremented, making changes of T visible for subsequent transactions. Afterwards, a final *barrier D* ensures that the changed $lcid$ is written to SCM.

The combination of append only updates, explicit persistency barriers and consistent tree structures enables the system to guarantee a consistent state on SCM at any time.

4.1 Recovery

If the system crashes, the information persisted on SCM is the only version of the data and needs to allow recovery to the latest consistent database state. The recovery process works by (i) re-initializing the system with persisted data structures, (ii) recover in-flight transaction and (iii) recover index structures.

After system restart, the state of the transaction managed, table meta-data and table data are re-mapped into the address space and initialized respectively based on meta-data.

As the system might crash while a transaction T has already written its commit id to $vbeg$ or $vend$ but before it incremented $lcid$, T needs to be reverted on recovery in order to allow further increments of $lcid$. Instead of scanning the complete vectors $vbeg$ and $vend$ of all tables on recovery to find changes of in-flight transactions that have to be reverted, the transaction context of committing transactions is persisted when entering the commit phase. Therefore, the recovery can easily traverse all in-flight transactions and revert potential changes by iterating through the lists of inserted and deleted rows.

Index structures are as well recovered directly from their persistency on SCM. The tree has to be verified and, if necessary, repaired. As soon as the lastCID in the transaction manager and the meta-data have been loaded, the rest of the steps have no further dependencies and can be executed in parallel.

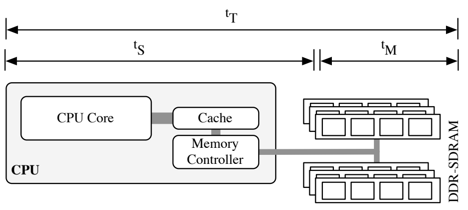


Figure 4. Overview of the modified memory latency. The real hardware latency t_M was modified whereas the system overhead t_S for requesting and transferring the data was unchanged, resulting in an observable total latency t_T.

Total Latency t_T	188.5	234.7	250.2	256.6
Memory Latency t_M	36.0	82.2	97.7	113.1
System Overhead t_S	152.5	152.5	152.5	143.5
Total Latency Factor	$1.0X$	$1.2X$	$1.3X$	$1.4X$
Memory Latency Factor	$1.0X$	$2.3X$	$2.7X$	$3.1X$

Table 2. Modified system hardware latencies to simulate SCM characteristics. Symmetric read and write latencies, all numbers in nanoseconds.

5 Experimental Evaluation

This section presents the experimental evaluation of Hyrise-NV. We evaluate the following aspects: 1) recovery time, 2) SCM latency and architectural influence on runtime performance and 3) micro-benchmarks outlining performance characteristics of the presented index structures on SCM. We compare Hyrise-NV with the traditional log-based version Hyrise-Log and Hyrise-None without any durability guarantees.

All benchmarks were executed on a machine with four Intel® Xeon® E7-8870 processors (with 10 cores running at 2.4 GHz) and 1.5 TB of DDR3 1067 MHz RAM. We limited all experiments to one NUMA node in order to eliminate NUMA effects as they are considered outside the scope of this work. Hyrise-Log persists to a 1 TB PCIe-attached flash drive, featuring a maximum theoretical read bandwidth of 1.5 GB/s and a read latency of 68μs.

We use the TPC-C[2] workload to evaluate the performance characteristics of Hyrise-NV, reflecting a transactional enterprise scenario modeling the order management for a company. The generated data set consists of 20 warehouses. The benchmark is executed in burst mode without any think-times and throughput is reported as the total number of successfully completed transactions per minute. If not specified otherwise, 300 parallel users are used for the presented benchmarks, where each user reflects one database connection and executes a stream of transactions by triggering a transaction and waiting for its commit. Clients and server are running on the same machine; queries are transmitted via HTTP using a modified Apache Benchmarking Tool.

5.1 Methodology

As there is currently no SCM hardware attached to the memory bus on the market, we use hardware emulation with modified DRAM to simulate the memory latency of SCM hardware and measure its impact on performance. The latency is changed by modifying the Serial Presence Detect (SPD) of the memory DIMMs, resulting in an increased real memory latency t_M as shown in Figure 4. The system overhead t_S for requesting and transferring the data is unchanged for configurations A, B and C, resulting in an observable total latency t_T as measured when accessing memory from the CPU. The presented method achieves the different latency configurations as outlined in Table 2, with a maximum memory latency factor of $3.1X$. Configuration A is the standard latency of the used system, configurations B and C are achieved through the modified SPD, whereas configuration D is the same as C but with activated memory interleaving in order to add additional latency. Additionally, the memory interleaving has an impact on t_S, resulting in a small reduction of the system overhead for configuration D.

We expect future SCM technologies to use the same memory access hierarchy as current DRAM and therefore expect t_S to remain unchanged. Thus, we consider this approach as a valid first evaluation of future SCM hardware characteristics. Unfortunately, it only allows for a limited latency slowdown up to $3.1X$ and does not allow for asymmetric read and write latencies although some future SCM technologies [7,17] are expected to exhibit such characteristics.

5.2 Recovery Time

The main design goal for Hyrise-NV is the capability of an instant system restart which is why all data structures are directly stored and manipulated on SCM. In order to verify this feature, we measure the recovery time for a table from the TPC-C dataset. Hyrise-NV achieves instant recovery times of approx. 100ms, independently of the table's main and delta sizes. As data is directly persisted on SCM, the system only needs to be restarted by initializing the respective data structures and rolling back in-flight transactions. Figure 5 shows the recovery times in seconds for varying sizes of the TPC-C stock table when the total row count of the table is varied from two million to ten million rows. Additionally, the distribution of rows into main and delta is varied resulting in delta sizes from

[2]TPC-C Benchmark: `http://www.tpc.org/tpcc/`

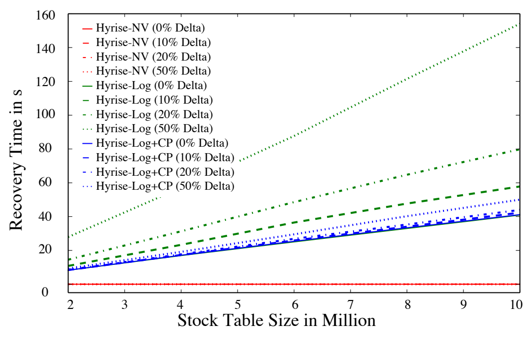


Figure 5. Comparison of recovery times for Hyrise-NV and a log-based recovery approach with and without delta checkpoint.

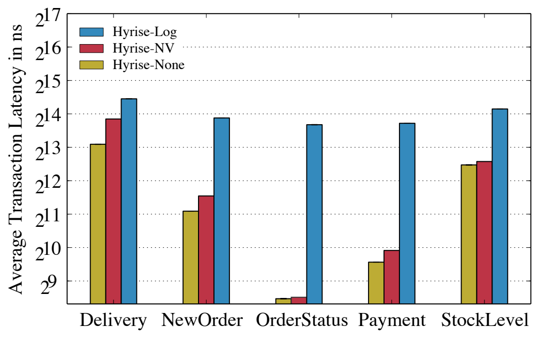


Figure 7. Latencies of TPC-C transactions. Group commit window for Hyrise-Log was 10ms.

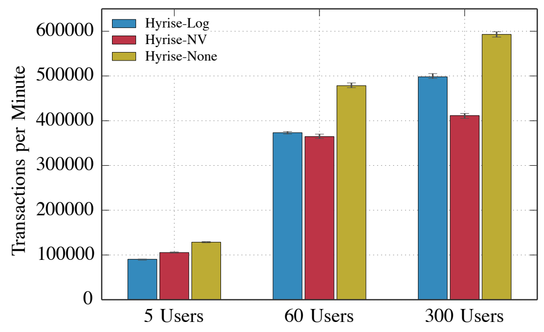


Figure 6. TPC-C throughput for Hyrise-NV, Hyrise-Log and Hyrise-None. Group commit window for log with 300 user was 10ms, 60 user was 4ms, and 5 user was 1ms.

0% to 50%.

In contrast, the implemented traditional log-based recovery mechanism loads the persisted data structures form binary dumps stored on a PCIe attached SSD and replays the respective log file. The recovery mechanism is reasonably optimized and distributes work across all available cores to saturate the available bandwidth and to provide a reasonable baseline. As replaying the log is more expensive as loading a binary dump, we expect the recovery costs to increase with the percentage of rows in the delta that are not part of a checkpoint. To measure this, we recover a table with varying total and delta sizes.

We differentiate between the two log-based recovery cases with (Log+CP) and without a delta checkpoint (Log). If no delta checkpoint exists, recovery takes up to 150 seconds as the required data structures for the main partition need to be loaded and the complete delta log needs to be replayed. For the Log+CP benchmark, a delta checkpoint was created directly before killing the database process. As a result, no log replay is required since the delta checkpoint reflects the complete state of the delta as a binary dump on SSD. This reduces the recovery time to approx. 50 seconds, still depending linearly on the total table size. Although the delta checkpoint reduces (or eliminates) the number of rows having to be read from the delta log, the delta size still influences recovery times as index structures need to be recreated.

5.3 Runtime Performance

In order to evaluate the runtime performance of Hyrise-NV, we compare it to Hyrise-Log (which uses a traditional log-based approach) and differentiate the identified overhead for system throughput into architectural and hardware overhead. *Architectural overhead* describes the additional complexity of ensuring the required write atomicity and ordering to support the described SCM-only architecture. We quantify the architectural overhead by comparing Hyrise-NV to Hyrise-Log using normal DRAM without any increased latencies. In contrast, *hardware overhead* describes the respective overhead that is introduced by increased memory latencies of future SCM hardware. Additionally, we provide the maximally achievable throughput with Hyrise by comparing the results to Hyrise-None without any durability guarantees that only executes transactions on volatile memory without separate log files on storage or special enforced guarantees for the use of SCM.

Figure 6 reports the total number of successfully completed TPC-C transactions per minute for 5, 60, and 300 parallel users. The used group-commit window was tuned to perform best with a given number of parallel users, resulting in a 10ms window for 300 users, whereas the window size was reduced to 4ms for 60 users, and 1ms for a single and 5 users. With 300 users, Hyrise-None achieves a maximum of 300K transactions per minute, whereas Hyrise-Log and Hyrise-NV reach 250K and 200K transactions per minute, re-

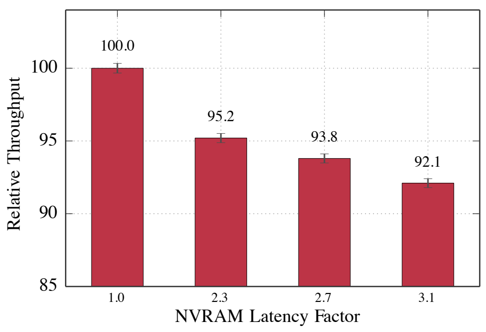


Figure 8. SCM Latency overhead on Throughput

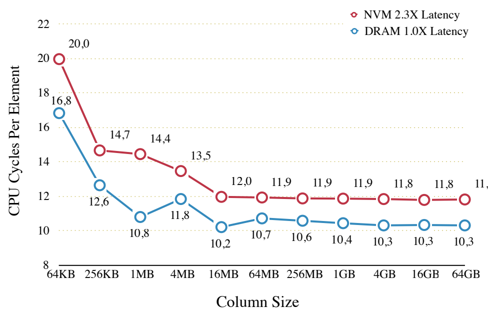


Figure 9. Sequential column scan speed with increased hardware memory latency.

spectively. For 300 parallel users, we report a relative architectural overhead of Hyrise-NV compared to Hyrise-Log of 20%.

In contrast, if the number of parallel users in the system is low, the overhead of the log-based approach increases as the efficiency of batching transactions is limited by the small number of parallel users. For 5 users, this results in a relative overhead compared to Hyrise-None of 30% and even 40% overhead with one single user. We report a throughput overhead of Hyrise-NV compared to Hyrise-None of 20% for 5 users and 28% with one single user. Directly comparing Hyrise-NV and Hyrise-Log, Hyrise-NV has a 14% and 18% higher throughput for 5 users and 1 user respectively.

Figure 7 shows a comparison of the average transaction latencies grouped by the TPC-C transaction type. For Hyrise-Log, latencies of all transactions are above the group-commit window of 10ms. Hyrise-NV achieves better latencies throughout all transactions, profiting particularly from short running transactions like OrderStatus or Payment.

The results of the previous benchmark were obtained using regular DRAM to simulate SCM and thus do not reflect differences in hardware, taking only the architectural overhead into account but not considering the hardware overhead of potential SCM technologies. As latencies of SCM in the near future are expected to be higher than latencies of today's DRAM [17], we evaluate the impact of increased memory latencies for Hyrise-NV by using a hardware-based emulation as described in Section 5.1.

Figure 8 displays the total transaction throughput per minute for TPC-C relative to Hyrise-NV on the same system with unaltered memory latencies. The experiment was executed on a Fujitsu RX600 with 500GB main memory and 4x10 cores. Increased memory latencies by 2.3X result in a throughput overhead of approx. 5%, due to the caching hierarchy, mitigating the effects of the slower memory. Figure 9 outlines the effect of higher memory latency for an unparallelized column scan, sequentially iterating over a complete column in memory and computing the sum of all values, reporting approx. 13% overhead by 2.3X increased memory latencies.

In summary, Hyrise-NV allows for almost instant database restarts and improved transaction latencies. For small numbers of parallel users and comparing with Hyrise-Log, we report a throughput advantage of up to 18% for a single user. For typical database scenarios with a large number of parallel users allowing to efficiently batch transactions, we report an architectural throughput overhead on today's systems of up to 20%. However, processor manufacturers are working on improved hardware support for SCM and we believe that this will significantly reduce the architectural overhead for future systems.

6 Conclusion and Future Work

In this paper, we presented an outline of emerging SCM-technologies and presented Hyrise-NV. Hyrise-NV is a columnar in-memory database engine using SCM as the primary persistence for tables and index structures. Our architecture enforces atomicity and ordering guarantees and performs database changes directly on NVM using multi-version data structures without the necessity of a write-ahead log. We evaluate our approach using a hardware-based emulation approach – emulating SCM latencies by overwriting the SPD values of the DIMMs – using the TPC-C benchmark, indicating an architectural throughput overhead of up to 20% and an additional 5% overhead for memory access latencies increased by 2.3X.

Future work includes research on optimized data structures for SCM, reducing the need to flush caches and to enforce barriers. Further evaluations using more sophisticated emulations and first real hardware promise additional interesting results. The topics of high availability and disaster recovery are also of great interest, raising the question if log structures are still required for such purposes or if other mechanisms are know more appropriate as log files are no more required for durability and atomicity purposes.

References

[1] D. Chakrabarti. Non-volatile Memory in the Storage Hierarchy: Opportunities and Challenge. `http://www.snia.org/sites/default/files2/SDC2012/presentations/Gen_Sessions/DhruvaChakrabarti_Non_Volatile_Memory_revised.pdf`, 2012.

[2] S. Chen, P. B. Gibbons, and S. Nath. Rethinking Database Algorithms for Phase Change Memory. *CIDR*, 2011.

[3] S. Chung. RRAM Opportunity for High density memory application. `http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2014/20140806_203A_Chung.pdf`, 2014.

[4] I. T. R. for Semiconductors. Emerging Research Devices (ERD) 2013 Tables. `http://www.itrs.net/Links/2013ITRS/2013Tables/ERD_2013Tables.xlsx`, 2013.

[5] M. Grund, J. Krueger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. HYRISE—A Main Memory Hybrid Storage Engine. *VLDB*, 2010.

[6] M. Grund, J. Krueger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. HYRISE—A Main Memory Hybrid Storage Engine. *VLDB*, 2010.

[7] H. Kim, S. Seshadri, C. L. Dickey, and L. Chiu. Evaluating Phase Change Memory for Enterprise Storage Systems: A Study of Caching and Tiering Approaches. *FAST*, 2014.

[8] J. Krüger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, P. Dubey, H. Plattner, and A. Zeier. Fast updates on read-optimized databases using multi-core cpus. *VLDB*, 2011.

[9] J. Krüger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, H. Plattner, P. Dubey, and A. Zeier. Fast Updates on Read-Optimized Databases Using Multi-Core CPUs. *VLDB*, 2011.

[10] D. Li, J. S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu. Identifying Opportunities for Byte-Addressable Non-Volatile Memory in Extreme-Scale Scientific Applications. *IPDPS*, 2012.

[11] D. Narayanan and O. Hodson. Whole-system persistence. *ASPLOS*, 2012.

[12] R. Neale. EETimes: Latest Updates on Phase Change Memory Problems. `http://www.eetimes.com/author.asp?section_id=36&doc_id=1320328`, 2013.

[13] J. Oshita. CNT-based 'NRAM' Shows Potential as Universal Memory. `http://techon.nikkeibp.co.jp/english/NEWS_EN/20140613/358460/`, 2014.

[14] S. Pelley, T. F. Wenisch, B. T. Gold, and B. Bridge. Storage Management in the NVRAM Era. *VLDB*, 2013.

[15] PMC. FLASHTEC NVRAM DRIVES. `http://pmcs.com/products/storage/flashtec_nvram_drives/`, 2013.

[16] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. *IPDPS*, 2009.

[17] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable High Performance Main Memory System Using Phase-change Memory Technology. *ISCA*, 2009.

[18] D. Schwalb, M. Faust, J. Krueger, and H. Plattner. Physical Column Organization in In-Memory Column Stores. *DASFAA*, 2013.

[19] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-store: A column-oriented dbms. *VLDB*, 2005.

[20] S. Venkataraman, N. Tolia, P. Ranganathan, and R. H. Campbell. Consistent and Durable Data Structures for Non-volatile Byte-addressable Memory. *FAST*, 2011.

[21] A. Waizy, B. Höppner, R. Liesegang, O. Lilienthal, K. Büker, H. Schmitz, D. Kasper, and J. Schrage. SAP HANA in a Hybrid Main Memory Environment. *HPI Future SOC Lab Report*, 2013.

[22] T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. SIMD-Scan: Ultra Fast in-Memory Table Scan Using on-Chip Vector Processing Units. *VLDB*, 2009.

[23] J. Wust, M. Grund, K. Hoewelmeyer, D. Schwalb, and H. Plattner. Concurrent Execution of Mixed Enterprise Workloads on In-Memory Databases. *DASFAA*, 2014.

Setting up Customized Genome Data Analysis Pipelines with Analyze Genomes

Matthieu-P. Schapranow Cindy Fähnrich
Hasso Plattner Institute
Enterprise Platform and Integration Concepts
August–Bebel–Str. 88
14482 Potsdam, Germany
schapranow|cindy.faehnrich@hpi.de

Abstract

Next-generation sequencing enables sequencing the human genome at reduced costs and time. Nowadays, subsequent interpretation and analysis of the generated genome data is a computer-aided task. A broad range of tools is available for the distinct analysis steps. However, their composition to a complete analysis pipeline is a cumbersome task, which requires IT support for setting up and maintaining the pipeline.
Our work focuses on the efficient processing of human genome data produced by next-generation sequencing machines. As functionality integrated into our Analyze Genomes platform, we enable researchers to choose from a range of analysis tools to design their own customized analysis pipeline that is then executed within our distributed execution framework. We made use of the Future SOC lab resources by evaluating a customized pipeline in two settings each applying a different tool for a particulate analysis step, whose results are shared in this report.

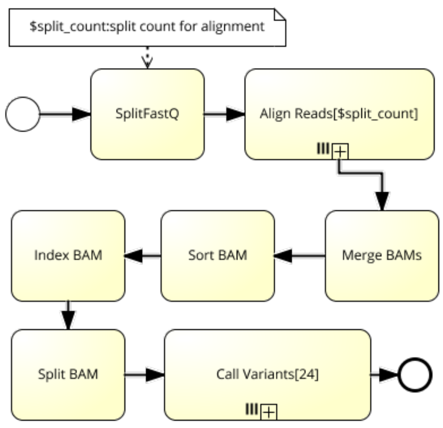

$split_count:split count for alignment
SplitFastQ
Align Reads[$split_count]
Index BAM
Sort BAM
Merge BAMs
Split BAM
Call Variants[24]

Figure 1. Genome Data Analysis Pipeline modeled with the presented notation subset and extensions using subprocesses, parallel execution, parameters, and variables.

1. Project Idea

Latest Next-Generation Sequencing (NGS) devices enable the processing of whole genome data within hours at reduced costs [1]. As a result, the time consumed for sequencing is meanwhile a comparable small portion of the time consumed by the complete workflow. Data processing and its analysis now consume a significantly higher portion of the time and accelerating them would affect the overall workflow duration. During this phase, the NGS output, i.e. short chunks of Deoxiribonucleic Acid (DNA) in digital format, need to be aligned to reconstruct the whole genome. Afterwards, variants compared to a reference, e.g. normal vs. pathologic tissue, are identified during variant calling. For both alignment and variant calling, a wide range of tools have been implemented and are available for free usage [4, 5, 6, 7, 8]. In addition, a number of intermediate steps must be conducted to improve data quality and transform data to meet requirements posed by analysis tools [2].

When building a genome data analysis pipeline, researchers have to choose carefully what tools to use as each of them is eligible for a different use case. In addition, designing and setting up an analysis pipeline requires technical support by IT administrators to install and combine tools as well as to run pipeline scripts that are depending on each other. The implementation of such pipelines in form of directly connected scripts has significant drawbacks, e.g. each time the structure of a pipeline changes a software developer needs to be involved to adapt the program code.

In the scope of the *Analyze Genomes* project, we have built an analysis platform providing tools for setting up and executing analysis pipelines, assessing their results, and combining these with scientific data from distributed data sources [10]. We provide a modeling environment to create customized pipelines and choose from a range of ready-to-use third-party tools. We build on the previously created execution framework incorporating cloud computing and in-memory technology to accelerate processing of genome data [10]. In our experiments, we create a customized analysis pipeline and apply two distinct tools — a state-of-the-art tool and an in-memory-based approach — for read alignment. We comopare runtime execution of both tools and share our findings in this work.

2. Modeling of Genome Data Analysis Pipelines

Nowadays' genome data analysis pipelines are commonly implemented as a number of software scripts invoking corresponding analysis tools that are executed one after another. We refer to a concrete implementation of a processing and analysis workflow as Genome Data Analysis Pipeline (GDAP). We refer to the atomic unit of a GDAP as job, which encapsulates a concrete script that can be executed to perform a specific task. The abstract representation of a job is an activity. Those activities are combined by researchers and clinician to set up a complete GDAP. For modeling GDAPs in our platform, we use a minimal subset of Business Process Model and Notation (BPMN) 2.0 modeling capabilities [9]. We incorporate the existing XML Process and Definition Language (XPDL) standard to store and exchange our GDAP models. The overall *Analyze Genomes* Platform with its specific execution environment for GDAPs has been set up in former lab periods.

2.1 Hierarchy of Activities

GDAPs can be hierarchically nested to any level of depth. Any self-contained part of a process model can be represented as a separate process model, which can be referenced from the overall process model by creating an activity with the subprocess' name. For example, Figure 1 depicts the process model of a GDAP with the two subprocesses *Align Reads* and *Call Variants*, which are marked by the cross at the bottom of the activity. During runtime, these activities are automatically replaced by the actual subprocesses, e.g. the one shown in Figure 2 for using BWA alignment.

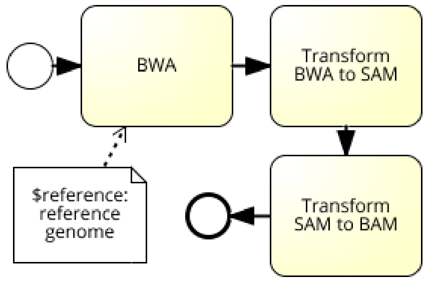


Figure 2. Alignment subprocess with all activities that must be conducted when applying BWA as alignment algorithm.

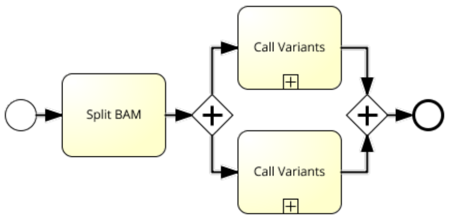


Figure 3. Explicit modeling of parallelization with parallel gateways. The subprocess "Call Variants" is executed twice in parallel.

2.2 Parallel Processing of Activities

Parallel execution of activities in BPMN can be defined in two ways, which depend on the parallel activities being identical or not. Multiple parallel executions of a single activity or subprocess are modeled via a dedicated attribute of the activity or subprocess. This attribute is called parallel multiple instance and is depicted by three vertical lines at the bottom of an activity as shown for *Align Reads* and *Call Variants* subprocesses in Figure 1. The number of parallel instances is defined by the number in square brackets that follows the activity's name. In the example, *Call Variants* is executed 24 times in parallel.
Parallel execution of differing activities or subprocesses are modeled via parallel gateways. These gateways explicitely model parallelization and its quantity, e.g. by modeling two branches. Figure 3 illustrates an example for parallel gateways with two variant calling activities being executed in parallel. When the gateway receives a signal from its incoming edges, all of its outgoing edges are signaled. These are only signaled once the gateway has received signals from all incoming edges, i.e. gateways synchronize parallelization.

2.3 Parameters and Variables

We distinguish between parameters and variables as follows. Parameters are set during design time of the GDAP model and cannot be changed afterwards. Variables are placeholders that are assigned at the latest point in time prior to the execution of a GDAP model.
Some activities require particular input parameters for correct task execution, e.g. which reference genome to use or how many threads to apply. We use BPMN data objects to model those input parameters for activities [9]. A parameter is stored in a data object that contains the parameter name followed by a colon and the parameter value. Multiple parameters are listed in a single data object and comma-separated.
We support usage of variables in GDAP models by using a specific data object that is identified by a dollar sign and followed by the variable's name. Figure 1 depicts the use of variables in a GDAP model, i.e. the variable *split_count* has to be set to a concrete value prior to the execution of the concrete GDPP model instance.
Parameters and variables can be assigned to multiple activities. For example, Figure 1 depicts multiple usage of parameter *split_count*. On the one hand, it is required by the first activity to know how many splits to create. On the other hand, the parameter defines the amount of alignment subprocesses that will be executed in parallel.

ID	Size [Gbp]	Size [GB]	Reads [Billion]
1	1.0	2.4	10.8
2	1.9	4.8	21.7
3	3.9	9.6	43.4
4	7.9	19.2	86.8
5	15.8	38.5	173.7
6	31.6	78.0	345.8

Table 1. Detailed specification containing the amount of Giga base pairs (Gbp), number of reads, and storage size.

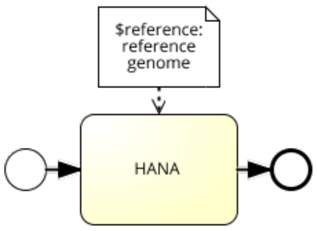


Figure 4. Alignment subprocess when using HANA alignment in the GDAP. Contrastly to when using BWA, HANA alignment does not require any other data transformation steps.

3. Benchmark Setup

All benchmarks were performed on the Future SOC cluster with 25 computing nodes. We used NGS data, i.e. the FASTQ file of patient HG00251 from the 1,000 genomes project, for our measurements [11]. The FASTQ file consumes 160 GB of disk space, consists of approx. 63 Giga basepairs (bp), approx. 695 M reads with 91 bp individual read length, forming an average 20x coverage. We split up the data into six data sets equally increasing in size, whose specifics are shown in Table 1. We set up the GDAP depicted in Figure 1 with the presented BPMN subset and extensions. We modeled the alignment part, which is one of the mayor analysis tasks, as subprocess to enable flexible tool exchange. We ran the GDAP with two settings each using a different alignment tool and measured execution times of these tools.

3.1 Burrows Wheeler Aligner

We used Burrows Wheeler Aligner (BWA) version 0.6.2 as alignment algorithm [6]. BWA is capable of multithreading, and we configured it to use a maximum of 80 threads. This relates to the maximum available hardware resources of our benchmark infrastructure. The tool creates outputs in SAI format, which is a binary format that needs to be converted into Sequence Alignment/Map (SAM) format for further processing [7]. In addition, subsequent analysis steps require Binary SAM (BAM) files, i.e. we further have to compress the SAM files. These format transformation steps are additionally included in the alignment subprocess shown in Figure 2.

3.2 HANA Alignment Server

As second tool we used an in-memory-based approach for alignment, i.e. the HANA alignment server [10]. The alignment algorithm is integrated into our in-memory computing platform, i.e. it can directly access native database operations. This algorithm was configured to use a maximum of 80 threads and emits alignments in BAM format. As a result, the corresponding alignment subprocess as shown in Figure 4 only contains the actual alignment without any additional format transformations.

4. Results and Findings

Table 2 depicts execution times for BWA and HANA alignment server for increasing file sizes, respectively. t_{BWA} refers to alignment times of BWA, whilst t_{HANA} refers to alignment times of HANA alignment server. In general, aligning reads with BWA takes longer than with HANA alignment server. For the smallest file size, BWA aligns reads within 1,395 seconds, whilst HANA alignment server carries out the same task more than 21x faster within 66 seconds. For the largest file size, BWA aligns reads within 24,417 seconds, which is a performance increase of factor 17 for a file size increase of factor 31.6. In contrast, HANA alignment server aligns reads for the largest file size within 2,000 seconds, which is a performance increase of factor 30. We compute relative runtime improvements from BWA to HANA alignment with $R_{BWA,HANA} = \frac{(t_{BWA}-t_{HANA})}{t_{HANA}}$. On average, applying HANA alignment server instead of BWA for alignment brings a relative runtime improvement of 94 percent for all file sizes as listed in Table 2.
In general, HANA alignment server delivers alignments much faster than BWA. A large contribution to that is the fact that the alignment server indexes the reference genome once at server startup, which is not considered in the performance numbers. In contrast, BWA indexes the reference anew for each alignment task. Efficiently indexing the reference genome at a reasonable memory footprint is an issue of nowadays' alignment algorithms. The results presented let us draw the conclusion that the strategy followed in HANA alignment server poses an alternative to current approaches.
However, in relation to the increasing file sizes, BWA performs better than HANA alignment server. The latter performs straightly proportional to file size increase, i.e. doubling the file size leads to doubling runtime performance. In contrast, runtime performance of BWA increases by a factor that is almost half of the file size increase for the largest data set. Reasons for that will have to be investigated in further experiments and runtime performance measurements. However, this fact never translates into better runtime performances for BWA instead of HANA alignment server for currently analyzed file sizes.

Size [Gbp]	$t_{BWA}[s]$	$t_{HANA}[s]$	$R_{BWA,HANA}$
1.0	1,395	66	95
2.0	2,446	129	95
4.0	4,250	252	94
7.9	8,099	501	94
15.8	14,504	1,001	93
31.6	24,417	2,000	92

Table 2. Comparison of execution times for HANA and BWA alignment for increasing data set sizes.

5. Next Steps

We constantly aim at extending our system by new functionality and improve methods already applied. In addition, we search for new data sources, e.g. data from latest releases of the 1,000 genomes project or the personal genome project, to be integrated into our knowledge base and used in real-time analyses [3].
We intend to set up additional GDAPs for more specific use cases, e.g. for targeted sequencing, and integrate a broader range of tools to choose from when building up a GDAP. In addition, we aim at integrating more parts of the analysis process into our in-memory computing platform, e.g. variant calling.

References

[1] W. J. Ansorge. Next-Generation DNA Sequencing Techniques. *New Biotechnology*, 25(4):195–203, 2009.

[2] Broad Institute. GATK Best Practices. `http://www.broadinstitute.org/gatk/guide/best-practices`. Accessed Oct. 10, 2013.

[3] G. M. Church. The personal genome project. *Molecular Systems Biology*, 1(1), 2005.

[4] Heng Li. Bcftools. `http://samtools.github.io/bcftools/` [retrieved: May 30, 2014].

[5] S. S. Langmead B. Fast Gapped Read Alignment with Bowtie 2. *Nature Methods*, 9(357–359), 2012.

[6] H. Li and R. Durbin. Fast and Accurate Short Read Alignment with Burrows-Wheeler Transformation. *Bioinformatics*, 25:1754–1760, 2009.

[7] H. Li et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16), 2009.

[8] A. McKenna et al. The Genome Analysis Toolkit: A MapReduce Framework for Analyzing Next-Generation DNA Sequencing Data. *Genome Res*, 20(9), 2010.

[9] M. Owen and J. Raj. BPMN and Business Process Mgmt. `http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf` [retrieved: May 30, 2014], 2003.

[10] M.-P. Schapranow, F. HÃďger, C. FÃďhnrich, E. Ziegler, and H. Plattner. In-memory computing enabling real-time genome data analysis. *International Journal on Advances in Life Sciences, Vol 6, Nr 1-2*, 2014.

[11] The 1000 Genomes Project Consortium. A Map of Human Genome Variation from Population-scale Sequencing. *Nature*, 467(7319):1061—1073, Oct. 2010.

Distributed-memory Simulation of Seismic Events following Earthquakes

Fahad Khalid[a], Camilla Cattania[b], Andreas Polze[a]
[a]Hasso Plattner Institute for Software Systems Engineering, Potsdam, Germany
{fahad.khalid, andreas.polze}@hpi.uni-potsdam.de
[b]GFZ German Research Centre for Geosciences, Potsdam, Germany
camcat@gfz-potsdam.de

Abstract

The significance of effective forecasting techniques for predicting natural hazards is obvious; given the devastation caused by such disasters. Earthquakes constitute one category of such events. Numerical simulations of seismic events can be demanding both in terms of memory consumption and processing requirements. Therefore, in order to scale such simulations, it is imperative that distributed-memory implementations are available.

In this report, we present our experience in extending an MPI based parallel simulations for clusters that was first ported from a shared-memory implementation during the fall 2013 session of the FutureSOC Lab. The simulation is distributed over 1000 cores across 25 nodes.

Our results show that the simulation scales up to 1750 processes. However, given the current configuration of the FutureSOC cluster, we see significant overhead in the I/O parts of the simulation. We highlight these using results obtained from running the simulation with different parameters on the 1000 core FutureSOC cluster.

1 Introduction

Note: This report is an incremental update to our FutureSOC project report submitted for this project in March 2014. We reproduce text from our previous report where information from the earlier report needs to be presented for the readers' convenience.

Earthquakes (seismic events) are caused by stresses that build up in the Earth's crust. Once an earthquake occurs, it induces changes to the stress; which can lead to subsequent events. The changes in stress that follow earthquakes can be simulated, making it possible to estimate the locations where subsequent earthquakes are more likely to occur.

Mathematical models for the rate of earthquake production [1] have been proposed and are currently in use. The current research at GFZ includes improving the predictive power and accuracy of the models with the help of simulations. A *C* language based shared-memory parallel (multicore CPU-based) simulation code – called *Coulomb Rate-State (CRS)* – was developed at GFZ that has enabled researchers to improve the existing mathematical models and devise better prediction strategies. However, performance of this implementation is constrained by the number of threads and the amount of main memory available in a single shared-memory machine.

In order to scale the simulation to larger problems and/or fine-grained models, it is important that the code be able to harness the power of multiple machines. This requirement naturally led us to this collaborative effort, where we successfully extended the existing simulation code to an MPI based distributed-memory implementation, specifically designed for execution on commodity clusters.

In the sections to follow, we focus on the unstable performance of the I/O modules of the simulation likely caused by the current configuration of the FutureSOC cluster.

2 Implementation

The simulation code consists of several modules, where each computes complex mathematical models. For brevity, we discuss only those aspects of the simulation that are important for the distributed-memory implementation. From this point onwards in the document, we will refer to the distributed-memory implementation as *CRS-MPI*.

CRS-MPI comprises of three major parts. For the purpose of this report though, we present on the File I/O and broadcast module design.

2.1 I/O and Communication

In the current *CRS-MPI* implementation, input files are read only by the root process. The values read by the root process are then *broadcast* to all other processes.

A more efficient solution would utilize a shared file system. However, since clusters with shared file systems are generally not available to the targeted end-users – Geophysicists –, we decided not to rely on the availability of a shared file system. The current implementation is capable of execution on commodity clusters supporting a minimal feature set.

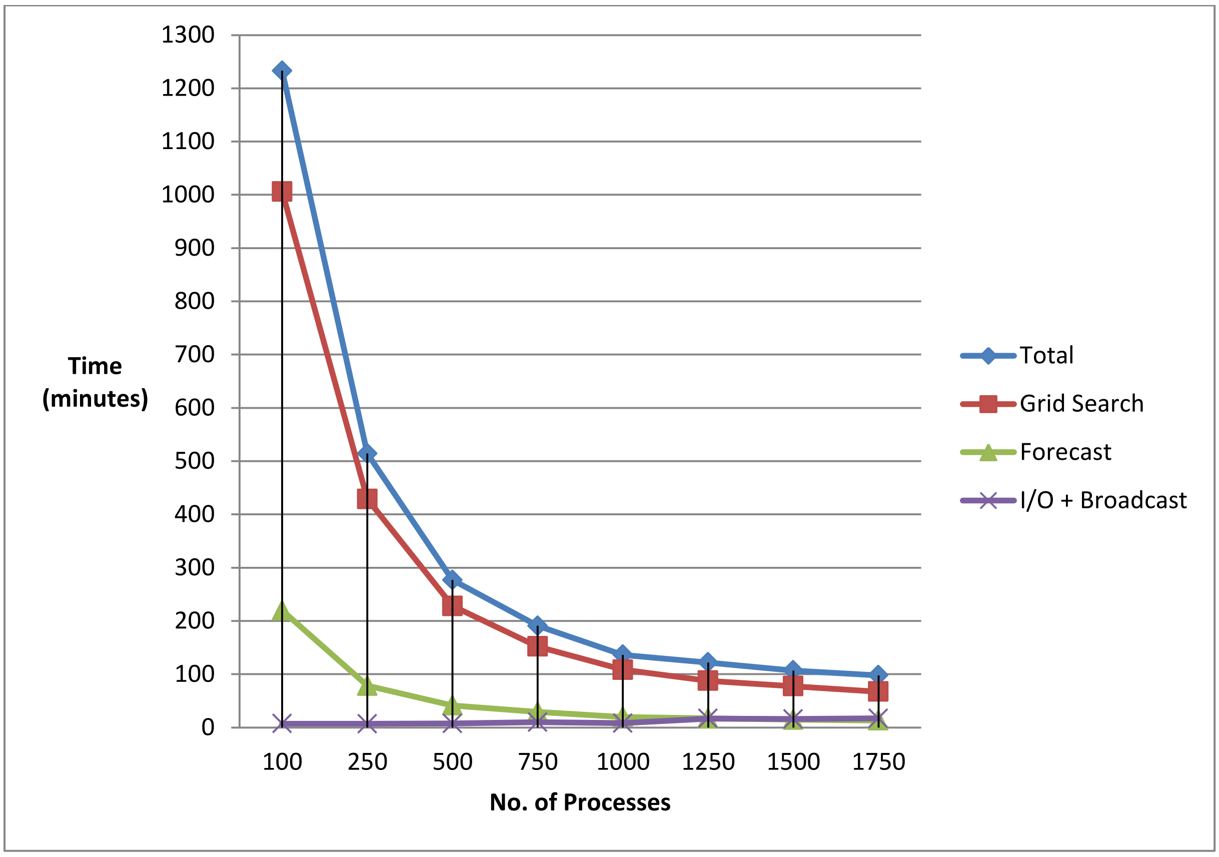


Figure 1: Time taken by different parts of the simulation, as well as the entire simulation when executed with different numbers of processes.

3 Evaluation

In this section we present and analyze results obtained by running the simulation on the *1000-core* cluster with different numbers of processes. The same input data and parameters have been used in all simulation runs. The results have been filtered to highlight issues with file I/O and message broadcast performance on the cluster.

3.1 Test Environment

The machine used for testing is the *1000-core* cluster available in the FutreSOC Lab. The cluster consists of 25 nodes, where each node supports a maximum of 80 hardware threads when hyper-threading is enabled. Moreover, each node is equipped with 1 TB of RAM; making it a total of 25 TB distributed across all nodes. The cluster is homogenous, i.e., all nodes have the same hardware and software configuration. The code was compiled with GCC 4.7.3. During the tests, each node was running SUSE Enterprise 11 SP2 with Open MPI 1.7.4.

3.2 Results

Figure 1 plots execution times against the number of processes used for each simulation run. The plotted results depict total execution time of the simulation, as well as the time consumed by file I/O and broadcast operations.

Total execution time was measured using the Linux *time* utility. The code was instrumented with *MPI_Wtime()* to calculate execution time for the file I/O and broadcast modules of the simulation.

Figure 1 shows that the file I/O and broadcast tasks only contribute to a fraction of the total execution time for up to about 1000 processes. As the number of processes grows larger, the broadcast incurs increasing overhead. This is due to the fact that an increasing number of processes imply an increase in the amount of message communication between processes. For a dataset that results in an execution path intensive enough in arithmetic operations, the broadcast cost is overshadowed by computation. It is such a dataset that was used to generate results shown in Figure 1. However, the situation is different when the execution path is low in arithmetic intensity.

Figure 2 shows results from a dataset that triggers an execution path with low arithmetic intensity. In this case, the perturbation in performance caused by the file I/O and broadcast phases is much more visible. An investigation of the phenomenon revealed that it is the file I/O module that causes this unstable behavior. We can see that the perturbation in overall performance is directly correlated to the I/O performance behavior.

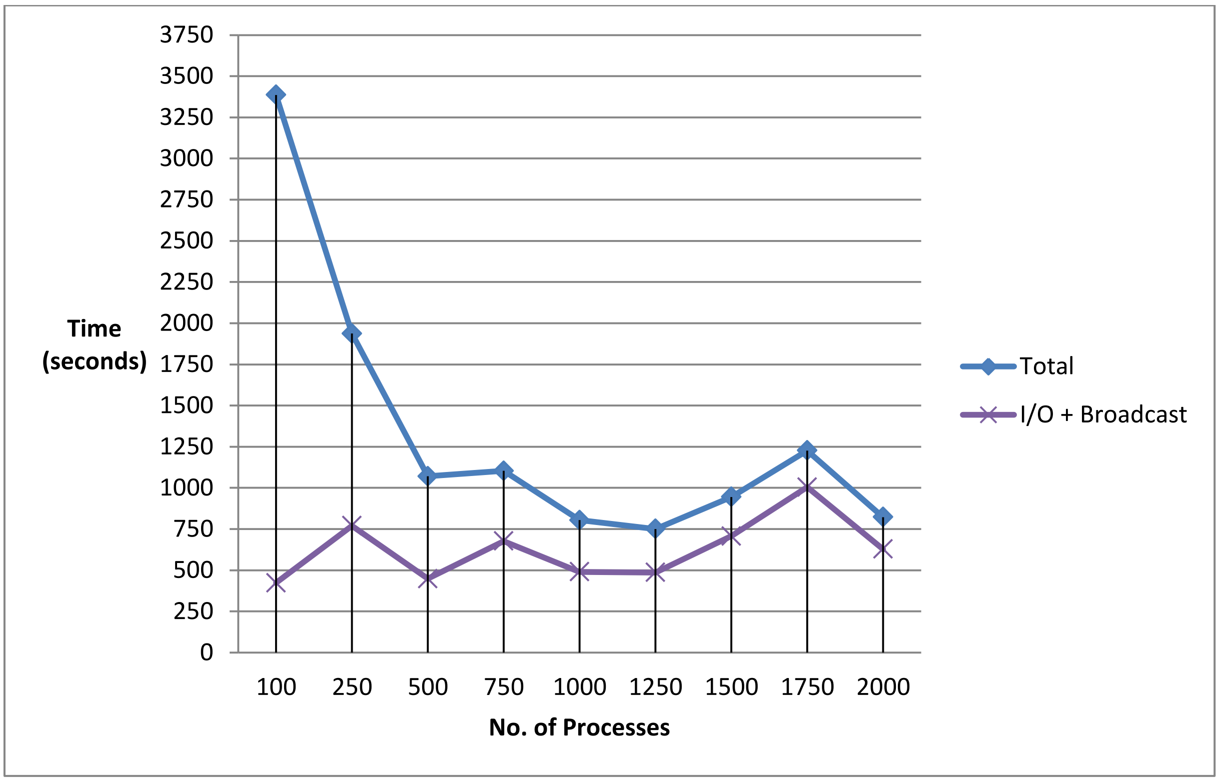


Figure 2: Speedup obtained with each increment in the number of processes. The first increment is from 50 processes to 100 processes.

We analyzed the simulation source to identify the root cause. We found that it is not the simulation source, but rather the cluster file system that causes these perturbations.

4 Conclusion

In order to fully optimize a large simulation such as *CRS-MPI*, the stability of the underlying hardware and middleware infrastructure is vital. In the upcoming session of the FutureSOC Lab project, we intend to investigate and resolve this issue in collaboration with the cluster administration staff.

Moreover, we are currently working on an optimized communication model that would reduce the performance overhead incurred by broadcast. Since it is not possible to customize the interconnect topology, we are working on a solution that is based solely on changes to the application. We intend to test our updated implementation in the upcoming session.

References

1. Dieterich, J., “A constitutive law for rate of earthquake production and its application to earthquake clustering”. *Journal of Geophysical Research*, 1994. 99(B2): p. 2601-2618.
2. Kurt Keutzer, Berna L. Massingill, Timothy G. Mattson, and Beverly A. Sanders. “A design pattern language for engineering (parallel) software: merging the PLPP and OPL projects”, 2010.

Simulation of Quantum Annealing of the Transverse Field Ising Model in Classical Parallel Hardware

Fabian Bornhofen, Fahad Khalid, and Andreas Polze
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
fabian.bornhofen@student.hpi.de,
{fahad.khalid, andreas.polze}@hpi.de

Abstract

Trying to understand the way Adiabatic Quantum Computers work, we are looking into ways of solving Ising models using algorithms inspired by quantum simulations on massively parallel hardware. We present a parallel implementation of the Path Integral Monte Carlo algorithm of the Edwards-Anderson Ising model that is based on the local optimization of tiles using single spin updates. In a system with $3072 \times 3360 \times 25$ *spins, we achieved a 25x speedup over a serial implementation using 32 annealer threads working in parallel.*

1 Motivation and Scope

Adiabatic Quantum Computation (AQC) is an emergent branch of Quantum Computing research that is already gaining traction outside pure academic research. Google and Lockheed Martin have invested in the first commerical chips by D-Wave[1], with the former starting its own chip-making efforts[2]. As software engineers, we are interested in the capabilities and limitations as well as programming models for this type of chip, since its inner workings differ significantly from conventional CPUs. Essentially, an AQC chip is a hardware implementation of an optimization algorithm called Quantum Annealing (QA) [1]. Existing implementations use Ising models (see Section 2) as their native representation of problems, which can be thought of as the chip's machine language.
In order to understand how to encode computational problems for AQC chips, we want to simulate this algorithm on classical hardware. A problem with simulations of quantum systems is that exact calculations of quantum states become computationally intractable on classical computers and researchers resort to Monte Carlo simulations when dealing with larger systems. Research has shown that the Path Integral Monte Carlo algorithm (PIMC) [3, 2, 4] yields results that are more consistent with the actual chips's results than classical Monte Carlo approximations [1].
Our contribution consists in leveraging parallel classical hardware for scaling and speeding up the simulation of QA of a simple type of the Ising model. While the model itself may be too simplistic to be useful for actual computations, we expect some insights about the general feasibility of classical AQC simulators and the potential for parallelism in the PIMC algorithm.

2 The Ising Model and PIMC

Ising models are lattices of so-called *spins* that can take values of 1 or -1, are weighted, and interact with each other. A simple example of such a model is the 2-dimensional Edwards-Anderson (EA) model. In this model, spins are arranged in a square lattice and each spin interacts with its nearest neighbors. Its *energy* is given by the *Hamiltonian* function

$$H(\mathbf{s}) = -\sum_i h_i s_i - \sum_{<i,j>} J_{ij} s_i s_j$$

where $< i, j >$ denotes a pair of neighboring spins, J_{ij} the interaction strength between spins i and j, and h_i the weight of spin i.
Existing AQC chips try to find the lowest energy state (*ground state*) of an Ising model. That means, given J and h values, they attempt to find a vector $\mathbf{s}$ of spin values that minimizes $H(\mathbf{s})$.
In order to be able to solve other kinds of optimization problems using the chip, programmers (or compilers) have to reduce these problems to the chip's Ising model. In case of the D-Wave chip, the hardware graph allows not only nearest neighbors, but also next-nearest neighbors to interact.
Our first step towards simulating AQC hardware is to simulate Quantum Annealing on the simpler EA type of Ising model. We did this because literature about

[1]http://www.dwavesys.com/
[2]http://www.technologyreview.com/news/530516/google-launches-effort-to-build-its-own-quantum-computer/

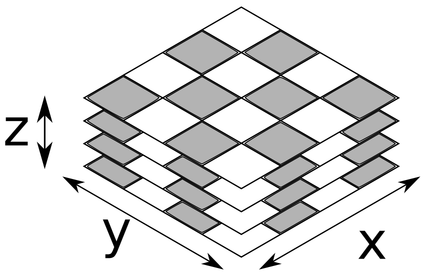


Figure 1. 3-Dimensional system decomposed into sublattices

that model is abundant, making it a convenient starting point for non-physicists.

Research on the EA model includes simulations of Quantum Annealing on this model using the Path Integral Monte Carlo (PIMC) method [3, 2, 4].

In PIMC, one replicates the original lattice P times in an additional dimension z. The i-th spin in a replica then interacts with its nearest neighbors in the same replica and also with the i-th spins in both of its neighboring replicas.

The resulting Hamiltonian [3, 2, 4] becomes

$$H_{ST} = -\sum_{k=1}^{P}\Big(\sum_i h_i s_i^k + \sum_{<i,j>} J_{ij} s_i^k s_j^k + J_\perp \sum_i s_i^k s_i^{k+1}\Big)$$

with

$$J_\perp = -\frac{PT}{2}\ln\tanh\frac{\Gamma}{PT}$$

and

$$s_i^{P+1} = s_i^1$$

Γ denotes a field that is decreased from an initial value Γ_0 to 0. During that process, the algorithm tries to flip single spins. A flip is accepted if it decreases H_{ST} or if a uniformly chosen random number r satisfies the Metropolis condition $r > \exp(-\beta\Delta H_{ST})$ where β is the inverse of the constant temperature T that has to be chosen beforehand.

3 Parallelization Stragegy

The problem with the described PIMC method is that it attempts to flip single spins while the effect of a single spin flip is only local.

Yavors'kii and Weigel [5] described a way of decomposing spin lattices with local interactions into a checkerboard pattern. We adapt the idea for performing local optimizations on sublattices in parallel.

As aforementioned, the system is replicated along the additional z-dimension. We decompose the 3-dimensional system into coarse-grained sublattices across the x- and y-dimensions of the original lattice (Figure 1). The algorithm then works in two phases, in which it does a sweep on either all white or all black sublattices. A sweep consists of $k > 1$ flip attempts on a sublattice.

The division into black and white phases ensures the absence of data races between spin updates in neighboring sublattices.

Literature suggests the possibility of using different update patterns as well [5, 1]. In this work, we include global moves [3] as a way of improving the algorithm on certain problem instances. A global move is an attempt to flip all spins across the z-dimension at a given (x, y)-coordinate. In a purely serial implementation, the time to perform a global move depends on the choice of the number P of replicas and becomes more expensive with increasing P. We explore the potential for parallelizing this operation as well.

In addition to that, we try to factor out parts of the update loop that do not have to be performed in order. A candidate is the pseudo-random number generator that can work in a separate thread and feed buffers of random bytes into the actual simulation.

4 Experiments and Results

We measured scalability of the checkerboard decomposition strategy on a Hewlett Packard DL980 G7 machine with 8 x Xeon X7560 (64 CPU cores) and 2048 GB RAM running Ubuntu Server 12.10.

As a benchmark, we used a lattice sized 3072×3360 spins, $P = 25$, resulting in a 3-dimensional system of just over 25,800,000 spins. Values for h_i and J_{ij} were randomly chosen from $[-1, 1]$. Using different widths and heights of sublattices, we tested ten different numbers of sublattices between 1 and 64 and measured the time it took to perform 10^8 spin flips. Note that with this relatively low number of updates we did not attempt to find the system's true ground state but to measure the number of spin updates per second.

We expect the single-spin update scheme to have a high instruction overhead. Therefore, we tested two different encodings for spins. In the first case, we encoded spins as floating-point numbers (`double`), so that no additional bit operations were needed to get a spin's value once it is loaded from memory. In the second case, we packed 64 spins into a 64-bit integer, with 0 representing -1 and 1 representing 1 states of the spin.

Figure 2 shows the speedup of the checkerboard tiling approach when scaling from a serial run to 64 tiles. Due to the checkerboard decomposition, 64 tiles mean that 32 threads are working at a time. With these 32 threads, simulations of the 3072×3360 lattice ran 25 times faster compared to serial runs when using bit-encoded spins. When we encoded spins as doubles, we observed a mere 17x speedup. Comparing runtimes per number of tiles, bit-encoded spins were con-

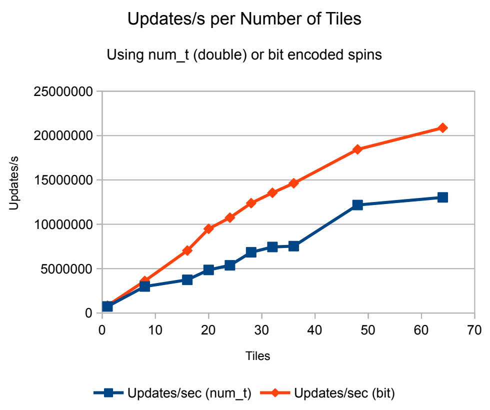


Figure 2. Updates per second

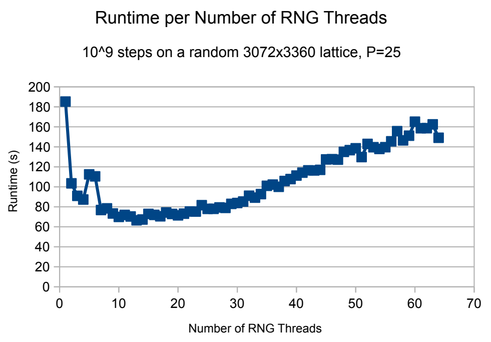


Figure 3. Effects of RNG Parallelization

sistently faster and yielded 38% speedup over double-encoded spins in the experiment with 64 tiles.

Based on the best result of this scalability experiment, we figured out the optimum number of pseudorandom number generator (RNG) threads. We let each RNG thread generate buffers of 16k random bytes from which the simulator threads read their random numbers. Figure 3 shows that when using 32 threads to perform 10^9 updates, the optimum number of RNG threads is 13. We expect that this number could easily be halved in an implementation that used random numbers more economically. For the sake of implementation simplicity, we use 64 random bits for each random number we use, even though this is not always technically necessary, e.g., when choosing a random site (x, y) in a much smaller lattice.

We tried to speed up global moves in a serial implementation by applying an OpenMP reduction to the calculation of ΔH_{ST} during a global move. In our experiments, we chose P between 20 and 30, which is within the range of values suggested in literature [2]. The resulting speedup was either negligible or negative. This does not come as a surprise, though, since we suspect synchronization overhead among CPU threads to outweigh the benefits of parallel processing at this small scale. For larger choices of P, this optimization may however be viable. On GPUs, where synchronization overhead between threads is lower and more threads are available, this optimization is actually beneficial.

5 Discussion

We implemented a parallelization scheme that helped speed up the optimization of large instances of the EA model using the PIMC algorithm. The findings we present in this report are applicable to the EA model. Graphs that are less dense than the EA model will likely not fully benefit from the checkerboard decomposition. They may require more sophisticated schemes of synchronization when trying to locally optimize subgraphs in parallel. On the other hand, we expect simulations of graphs with a larger number of connections to become increasingly compute-bound. The most extreme case is the Sherrington-Kirkpatrick (SK) Ising model, a generalization of the EA model. It is represented as a complete graph, i. e. every spin interacts with every other spin. In a Monte Carlo simulation of the SK model, we expect single step to become more computationally expensive so that there is potential for parallelism within single steps.

As for the EA model, we are dealing with a Markov-chain Monte Carlo algorithm with a relatively high instruction overhead. Due to the Markov-chain nature of the algorithm, it has a considerable serial portion of code. This serial part benefits from having a large number of fast processors to run on. Our CUDA implementation showed that global moves in particular work faster on GPU than on CPU. However, up-to-date GPUs typically have less memory than massively parallel CPU systems. This makes the architecture and implementation of such simulations much more complicated if we are to deal with problem instances too large to fit into GPU memory.

References

[1] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nat Phys*, 10(3):218–224, Mar 2014. Article.

[2] R. Martoňák, G. E. Santoro, and E. Tosatti. Quantum annealing by the path-integral monte carlo method: The two-dimensional random ising model. *Phys. Rev. B*, 66:094203, Sep 2002.

[3] G. E. Santoro and E. Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General*, 39(36):R393, 2006.

[4] M. Sarjala, V. Petj, and M. Alava. Optimization in random field ising models by quantum annealing. J. Stat. Mech. (2006) P01008, 2005.

[5] T. Yavors'kii and M. Weigel. Optimized GPU simulation of continuous-spin glass models. *Eur. Phys. J. Special Topics*, 159, 2012.

Project Report: Dispersed Data Processing Services for Third-Party Applications

Josef Spillner
Technische Universität Dresden
Fakultät Informatik
Nöthnitzer Str. 46
josef.spillner@tu-dresden.de

Abstract

The Cloud Storage Lab at Technische Universität Dresden is exploring novel storage and processing architectures for distributed, untrusted cloud environments. From July to October 2014, we used the infrastructure of the Hasso Plattner Institute's Future SOC Lab to run larger-scale experiments with StealthDB, a database management system which supports both dispersed and encrypted storage of data and partially even the execution of operations over this data. The findings of these experiments are summarised in this report.

1 Background

Distributing data across several storage targets (e.g. devices or services) introduces advantages, such as higher availability or increased access performance, at the expense of higher capacity and transmission requirements. Dispersed storage offers a balanced compromise with minimal capacity overhead for a freely selectable redundancy [4]. We have previously explored and statistically analysed the properties of dispersing cloud controllers in the Future SOC Lab [2].
When large volumes of data are dispersed to non-cooperating cloud providers, the question arises how to productively use the data without having to transfer it back to the client. Depending on how the data was encoded, its dispersion and encryption may have rendered it unsuitable for any further processing directly on the storage targets. However, a conscious selection of coding algorithms retains the possibility to run at least a subset of processing algorithms directly on the fragments [3].
To exploit this possibility and see how far down we can drill on such fragments, we took our experience on dispersed storage and lifted it to the processing level on fine-grained data units. Thus, we introduced StealthDB, a novel database management system [1]. In the following sections of the report, we first describe the system in its current state and then explain how its development was supported by the experiments.

2 System Description

StealthDB, as shown in Fig. 1, is a column-store system which flexibly combines in-memory, file and cloud storage. An arbitrary number of mixed storage areas is connected through one out of four distribution schemes: Round-robin placement, hash ring replication, full replication or dispersion. Furthermore, each chosen distribution scheme can be combined with encryption. Depending on the type of a column and the operations which should be run remotely on it, homomorphic, order-preserving and (still in development) fuzzy encryption can be applied. Distribution schemes can be set globally with a `USE CLOUDS ...` statement which extends SQL. They can also be modified at run-time for each column by calling `ALTER TABLE table ALTER COLUMN column USE CLOUDS ....` In this case, data migrates between the clouds via the client so that the evolution of cloud services will not break applications which use the database system.

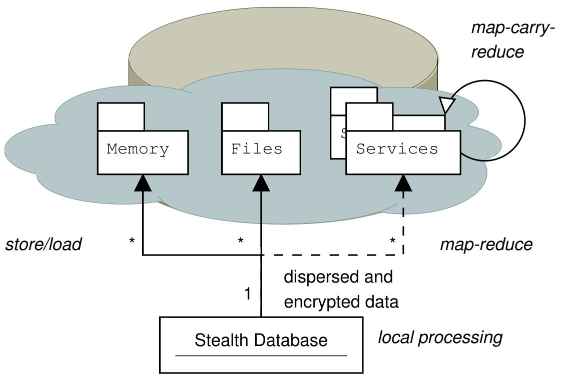


Figure 1. Stealth database design with multiple storage locations

Processing is implemented in a map-reduce style.

Whenever a column needs to be aggregated, an evaluation of the remote execution possibility is performed. For instance, on dispersed and encrypted data, the following statement can be run on each cloud: `SELECT SUM(id) WHERE id < 100`. However, the following query will not work due to the still missing support for searchable encryption: `SELECT name WHERE name LIKE '%news%'`.
StealthDB further increases the safety and robustness in multi-cloud environments by supporting `SELECT ... OPTIMIZE FOR goal` queries. Users are given the possibility to prioritise performance over reliability, or energy efficience over performance, for instance. The combination of security features and conscious coding led to the name *Stealth Computing* from which the name of the system is derived.
The development of StealthDB started in April 2014. As a moving target, its functionality grew along many experiments. Only in early October 2014, it passed its test suite for the first time which is a combinatorial iteration through 36 configurations involving all storage target types, distribution and encryption schemes, and different numbers of storage targets. Its balance of cloud and security features and its associated performance drawbacks and relational simplicity will position the database mainly as backend for niche cloud applications, for instance in the personal data domain or for quality- assured sensor data processing, which are insufficiently covered by existing database management systems. One interesting aspect of the implementation is its small footprint: Currently, it consists of less than 2000 lines of code with a total footprint of 100 kB.

3 Procedure Description

In preparation of our first submission on stealth databases [1], we came up with a simple *million movies benchmark* which shall serve as baseline for all experiments in different environments. Compared to TPC-H and other industrial benchmarks, it requires only the most basic language features of SQL.
The benchmark generates one million artificial movie names, inserts them into a text column in conjunction with a unique numeric identifier column, and selects all entries both ordered and unordered. Furthermore, it selects the sum of all identifiers which is a constant $\frac{n(n+1)}{2}$ for $n = 1000000$ equal to 500000500000. In a second experiment, it migrates the identifier column between different cloud configurations, including a hashring with two replica per value.
The experiments involve the correctness and performance measurements of data insertion as well as migration between different locations. The correctness property refers to functional correctness, without considering hardware or communication faults, and can be considered as mostly fulfilled at the end of the experiment season. Thus, the main interest shifted towards the performance measurements.

4 Experimental Setting and Results

StealthDB has first been installed on a server system provided by the Future SOC Lab. The hardware consists of four out of 32 HP Converged Cloud blades, each with two Intel Xeon E5-2620 CPUs (à 6 cores/12 threads) running at 2.0 GHz and 64 GiB of memory and an NFS-connected 3PAR disk array with 3 TB capacity. The blades are interconnected with 10G Ethernet. The system runs Ubuntu 14.04 with Linux kernel 3.13.0 and Python 3.4.0.
StealthDB has then been configured to run on the first node together with the Pyro RPC name server (`./pyro-ns -cluster`). On each of the other nodes, the cloud backend was started (`./stealthdb-cloud `hostname` -cluster`). Then, the database system has been instructed to store its data on all of them (`USE CLOUDS 'auto';`) which resulted in the distribution list `['cloud://ubuntu-tud-0104', 'cloud://ubuntu-tud-0102', 'cloud://ubuntu-tud-0103'] with ['replication']`.
The results of running StealthDB in the cloud are consolidated in Table 1. They compare all-local dispersed files with dispersed cloud access from one blade to the three other ones. Additionally, homomorphic encryption of fragments in the cloud is used for higher protection of their sensitive contents. For the dispersion, the Bitsplitter library written in C with a Python interface is used. For the encryption, the Paillier module for Python is used. The principal observations are that (1) transmitting RPC messages massively decreases the insertion performance but is otherwise competitive against the local disk speed, and (2) the parallel dispersed processing over encrypted numbers speeds up the execution considerably compared with the local decryption process.

Table 1. Million movies management results in the converged cloud

Benchmark step	Backend	Runtime
INSERT	file:disp	815.30s
INSERT	cloud:disp	17972.00s
INSERT	cloud:disp:crypt	224877.00s
SELECT	file:disp	11.99s
SELECT	cloud:disp	13.54s
SELECT	cloud:disp:crypt	444.68s
SELECT ORDER BY	file:disp	12.82s
SELECT ORDER BY	cloud:disp	14.11s
SELECT ORDER BY	cloud:disp:crypt	444.50s
SELECT SUM(id)	file:disp	14.51s
SELECT SUM(id)	cloud:disp	14.92s
SELECT SUM(id)	cloud:disp:crypt	80.67s

Another interesting metric is the peformance of data migration. In this experiment, we migrate from an initial configuration to a different configuration which could be modelled as property-annotated state transitions. The processing times are encompassing the retrieval, re-coding and re-insertion. In Table 2, some examples are presented.

Table 2. Million movies migration in the converged cloud

Initial conf.	Migration conf.	Runtime
cloud:replication	cloud:hash	9405.90s
cloud:hash	cloud:hash:crypt	79120.00s
cloud:hash:crypt	cloud:disp:crypt	148078.00s

There is a lot of optimisation potential in StealthDB by adding native methods (implemented in C), skipping sub-column aggregation if there is only one subcolumn, and using persistent file handles and proper binary file formats. During the project, we have reimplemented the Paillier library in C. However, due to the restricted native integer sizes even on 64 bit architectures and the requirement to use only half of the bitwidth for the input numbers, this library only supports the encryption of values up to 32 bits (when using unsigned long long types) or even 16 bits (when using plain integers). There is an enormous speed-up with a factor of around 70 for the 16-bit flavour, but hardly any speedup for the 32-bit flavour. Therefore, for our experiment we kept using the Python implementation which handles 128 bits by default or more on demand.

Fig. 2 visualises the overheads in the current StealthDB implementation when traversing different paths from plain storage in memory with local processing to distributed processing over encrypted dispersed data fragments in the cloud. The overheads prevent the system from many practical use cases where performance matters. Nevertheless, even in its current initial state, StealthDB shows potential for cases in which confidentiality, availability and other service quality properties matter. We expect production-level database systems to pick up sub-column and dispersed processing concepts in the near future.

5 Conclusions

Stealth computing is a promising concept to manage confidential and privacy-sensitive data in the cloud. With StealthDB, a database system for structured data management has been created. The experiments clearly show both the expressive power of dynamically assigning clouds and distribution schemes and the necessity to conduct further research and engineering in order to speed up the algorithms and cover more aggregate functions in the dispersed processing model.

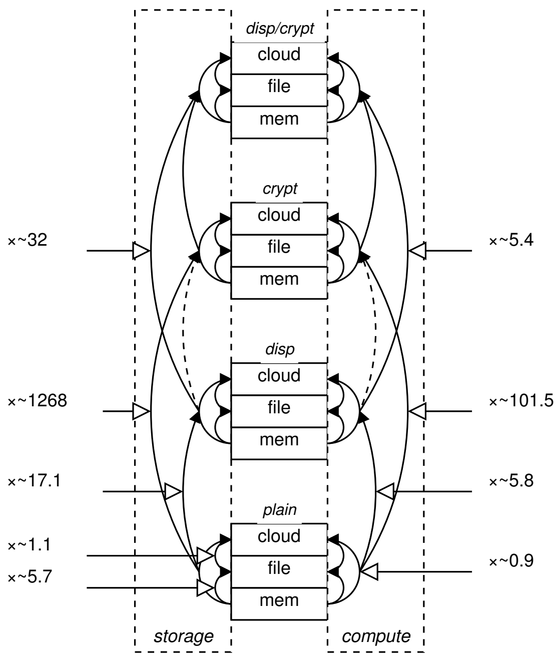


Figure 2. Overhead factors comparison figure

The Future SOC Lab infrastructure offered sufficient computing resources for the experiments. There are two suggestions on how to improve it in the future. First, having an extremely distributed platform similar to the hundreds-of-nodes Raspberry Pi clusters would be an interesting addition. With the proliferation of the Internet of Things as well as current data centre trends towards smaller and more energy-efficient compute units, such a platform could cover many use cases. The second, more modest, suggestion is to improve the accounts system as there were sporadic issues with the shell login and root/user switches (`su - josef.spillner: No passwd entry for user 'josef.spillner'`). Overall, the infrastructure reliability has been very good.

References

[1] J. Spillner and M. Beck. Stealth Databases: Ensuring Secure Queries in Untrusted Cloud Environments. Rio de Janeiro, Brazil, November 2014. Submitted to 3rd Latin American Conference on Cloud Computing and Communications (LatinCloud).

[2] J. Spillner and J. Müller. Project Report: Statistical Analysis of Cloud Storage. HPI FutureSOC Lab, Fall 2013 project report, March 2014.

[3] J. Spillner and A. Schill. Algorithms for Dispersed Processing. In *1st International Workshop on Advances in Cloud Computing Legislation, Accountability, Security and Privacy (CLASP)*, London, UK, December 2014.

[4] J. Spillner and A. Schill. Towards Dispersed Cloud Computing. In *2nd IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 175–179, Chișinău, Moldova, May 2014.

Evaluation of State-Of-The-Art Hybrid Hardware Architectures based on Application Bottlenecks

Frank Feinbube
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
University of Potsdam, Germany
Frank.Feinbube@hpi.de

Peter Tröger
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
University of Potsdam, Germany
Peter.Troeger@hpi.de

Abstract

Trends in hardware developments emphasize the every-increasing importance of hybrid computing for future computer architectures and software systems. Competitive applications need to leverage the performance opportunities provided by emerging generations of accelerator technology. With the introduction of its K20 architecture, NVIDIA takes a big step towards a wider applicability of GPU computing by supporting new concepts like dynamic programming, on-device grid management, and direct data exchange. Intel's novel Xeon Phi accelerator, being a stand-alone PCI-Express card like GPUs, but still being x86 compatible, is approaching the field of hybrid computing from the general purpose side.

The purpose of this study is evaluate the feasibility of the K20 architecture and the Xeon Phi accelerators for well-known application bottlenecks.

1 Introduction

In the course of our studies in FutureSOC Lab of the Hasso Plattner Institute, we looked into the capabilities of modern hardware architecture to improve the performance of algorithms with well-known application bottlenecks. Due to the novelty in their processing characteristics and memory hierarchy Nvidia’s K20 GPU Compute Device (1) and Intel’s Xeon Phi Accelerator Board (1) were of special interest for our studies. In order to evaluate their capabilities for improved scalability, execution performance and programmability we studied the optimization of well-known algorithms from the Berkeley Dwarfs (1) collection. This report summarizes our work and findings.

2 Concurrent Tasks with Dynamic Parallelism on NVIDIA’s GK110 Architecture

Our first study in the HPI FutureSOC Lab was conducted by M. Linkhorst in the HPI Operating Systems and Middleware Group and resulted in his master thesis. (2) It focussed on the applicability of Nvidia’s K20 GPU architecture for divide-and-conquer algorithms and graph algorithms. Special attention was given to a novel concept that was introduced in the said hardware generation and standardized in OpenCL 2.0: Dynamic Parallelism. (1)

A list of criteria for algorithms is presented in the thesis that allows to assess opportunities and potential problems of novel accelerator technologies. Based on these criteria, a survey of the Berkeley dwarves and the algorithms of the Parboil hybrid benchmark was created with respect to their suitability to benefit from Dynamic Parallelism. We found, due to its high rating in the survey, the class of Divide-And-Conquer algorithms to be especially suited for an elaborate study of the applicability of Dynamic Parallelism. For further investigation we selected two algorithms: The Breadth-first Search (BFS) algorithm and the All-pairs Shortest Path (APSP) algorithm.

The state-of-the-art BFS algorithms were assessed and compared; a number of benchmarks were executed on the Future SOC hardware. Based on the assessment, we created an optimized new BFS algorithm that uses Dynamic Parallelism to allow for concurrent usage of the resources of GPU Compute Devices. Employing the Stream concept to the BFS algorithm enabled us to create an APSP implementation with significantly better execution performance than APSP implementations based on the best state-of-the-art BFS algorithms. This demonstrates the great advantages that state-of-the-art accelerator technologies like Dynamic Parallelism can provide even for task-based algorithms. In contrast to existing solutions, the scaling behavior of our approach makes it very attractive for future generations of accelerators.

Furthermore, we show that Dynamic Parallelism can be used to provide a concurrent graph library directly on the GPU.

This work demonstrates the applicability of Dynamic Parallelism for demanding algorithms and illustrates ways in which programs can be restructured in order to benefit from it.

3 Audio Signal Processing on GPU Compute Devices

The results of our second study in this HPI Future SOC Lab project are the master thesis of M. Plauth and a publication of the results at the 15th International Conference on Parallel and Distributed Computing, Applications and Technologies. (3) The work was supervised by the HPI Operating Systems and Middleware group. The thesis explores the applicability of state-of-the-art accelerator technology for real-time audio signal processing. Studies were conducted with NVIDIA's K20 GPU Compute Devices and Intel's Xeon Phi Accelerator Board.

The thesis evaluated the feasibility of using GK110-based GPU compute devices for the application of the FastICA algorithm in a live audio signal processing scenario. Furthermore, the benefits of leveraging GPU hardware for a batch processing implementation of the FastICA algorithm were investigated as well.

For the tested range between 2 and 8 signals, the batch processing mode achieved a median speedup of factor 18.63 and 13.66 using single precision and double precision, respectively. The speedup was determined in comparison to a parallel CPU-based implementation using the MKL and IPP libraries. For the live processing mode of operation, maximum execution times between 19.33 and 130.32 milliseconds were determined for the same number of signals. With one chunk representing 1365 milliseconds of audio, all tested chunks were processed within the deadline of 170 milliseconds.

Even though previous attempts of using GPU compute devices for the acceleration of FastICA (and other ICA algorithms) have been published, this work covered two major aspects which have not been covered by preexisting work. First of all, the capabilities of the compute-centered Kepler GK110 architecture were evaluated. From the point of view at the time of writing this thesis, even the latest of the preexistent publications uses a comparatively outdated GT200 GPU. Secondly, prior work mostly dealt with processing Electroencephalogram (EEG) data, which differs a lot from the characteristics of audio data.

We demonstrated the feasibility of GPU Compute Devices for live processing of complex audio signal processing tasks such as Blind Signal Separation using FastICA.

4 Creation of Prioritization Schemes for Real-Time Multi-Processor Scheduling Policies based on Evolutionary Algorithms

The third study was conducted by C. Kieschnick and resulted in a master thesis at the HPI Operating Systems and Middleware Group. (1) In this study we evaluated the novel Xeon Phi accelerator board from Intel in the HPI FutureSOC Lab with respect to its applicability for a complex simulation based on evolutionary algorithms.

Real-time scheduling method on multi-core or multi-processor systems often follow the Time-and Space-partitioning approach. This partitioning approach carries disadvantages, as the known as Dhall's anomaly effect, resulting in n processors to a missed deadlines even with a CPU load > 1 / n. Therefore, many works deal with heuristics that examine global (i.e. non-partitioned) scheduling problems with multiple processors. In this work, a method is proposed to generate the prioritization functions by genetic programming. These prioritization functions are then examined in simulation runs for scheduling hypothetical, generated task sets.

We investigated two such sets: (a) A set of algorithms known from the literature and their combinations; (b) The complete set of all possible task sets for a given quantum number. In both cases very large task quantities are generated for the simulation. As with many large-scale simulations, the huge amount of independent tasks are an ideal show case for the Xeon Phi acceleration board. This device combines a large number of processor cores with moderate clock speed and a large number of vector processing units per core.

We evaluated the performance of three different implementation strategies and a variety of optimizations for each of them: the CPU-exclusive, the Xeon Phi-exclusive and the combined simulation. While the native execution on the Xeon Phi was twice as fast as the execution on the CPU, we analyzed the impact of further optimizations for the CPU side and the Xeon-Phi side and managed to reduce the execution time on each platform to a third depending of the workload.

Furthermore, this work evaluated the applicability and performance impact the offloading feature of Intel's Xeon-Phi-focused programming models for our complex simulation. Despite different offloading optimizations, the performance improvement using both platforms was moderate compared to the native execution on the Xeon Phi accelerator board for most workloads.

We present an elaborate study of the execution characteristics and an analysis of the bottlenecks of the hardware architecture for this class of algorithms. Our studies demonstrated that the performance opportunities on Intel's Xeon Phi accelerator board demand the algorithm not only to provide a large number of independent tasks but also to highly utilize its vector units. Furthermore it is very sensitive to data access patterns and the deliberate utilization of the cache hierarchy.

5 Load Balancing on GPUs using Dynamic Parallelism

Our latest research in the HPI FutureSOC Lab was conducted by F. Schlegel in his master thesis at the HPI Operating Systems and Middleware Group. (1) The focus of this work is on the evaluation of the Dynamic Parallelism technology as described by the OpenCL 2.0 standard (1) and introduced by Nvidia's K20 GPU architecture for load balancing of task-based algorithms.

In the last years accelerators like graphics cards became increasingly popular for general computation. The hardware architecture and programming model of graphics processors are well suited for solving data-parallel problems. However, the SIMD-like hardware found in graphics cards is not suited for calculations with irregular workload like the search of unbalanced trees.

For the first time, the recently introduced extension to the CUDA programming model by NVIDIA called Dynamic Parallelism allow the spawning of new work items for the graphics card from inside a program running on the graphics card itself. This makes it possible to adapt the amount of work items to intermediate results in order to achieve a better load balancing.

In this work novel approaches for load balancing of tree search algorithms on graphics processors using Dynamic Parallelism are presented and compared.

The N-Queens Problem is used as an example application. Runtime comparisons with existing solutions show that Dynamic Parallelism introduces a high management overhead and runtime limitations, which cannot be compensated by the achieved load balancing for the given examples.

However, the presented algorithms are not limited to be applied to backtracking and tree search; especially the presented approach for using the shared memory as buffer for work items that are distributed to new grids using Dynamic Parallelism can be applied in many load balancing scenarios. Also the newly presented concepts for efficient memory management, which is necessary for passing parameters to child grids, can be applied to other algorithms that use Dynamic Parallelism.

In the course of this work we introduced and evaluated four different implementation strategies for load balancing using Dynamic Parallelism and compared them against existing efficient GPU algorithms.

6 Acknowledgments

This work, which resulted in four master theses and conference publication, was only possible because of the generous offer to conduct our studies on modern hardware within the HPI FutureSOC Lab. Hereby we want to show our gratitude to the FutureSOC Lab steering committee which accepted our project proposal and the FutureSOC Lab team for the daily cooperation. Special thanks go to Bernhard Rabe for its friendly and timely support, advice and help.

7 References

1. **Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbs, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf,Samuel Webb Williams, Katherine A. Yelick.** *The Landscape of Parallel Computing Research: A View from Berkeley.* s.l. : Electrical Engineering and Computer Sciences, 2006.

2. **Linkhorst, Martin.** *Concurrent Tasks with Dynamic Parallelism on NVIDIA's GK110 Architecture.* s.l. : Hasso Plattner Institute at the University of Potsdam, 2013.

3. **Plauth, Max.** *Audio Signal Processing on GPU Compute Devices.* s.l. : Hasso Plattner Institute at the University of Potsdam, 2014.

4. **Corporation, NVIDIA.** *Whitepaper: NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110.* s.l. : NVIDIA Corporation, 2012.

5. **Corporation, Intel.** *Intel Xeon Phi Coprocessor x100 Product Family: Specification Update.* s.l. : Intel Corporation, 2014.

6. **Kieschnick, Christian.** *Creation of Prioritization Schemes for Real-Time Multi-Processor Scheduling*

Policies based on Evolu-tionary Algorithms. s.l. : Hasso Plattner Institute at the University of Potsdam, 2014.

7. **Schlegel, Frank.** *Load Balancing on GPUs using Dy-namic Parallelism.* s.l. : Hasso Plattner Institute at the University of Potsdam, 2014.

8. **Inc., Khronos Group.** *OpenCL 2.0 Standard.* s.l. : Khronos Group Inc., 2013.

Predicting the Availability of an In-Memory Computing Cluster

Sascha Bosse and Klaus Turowski
Center for Very Large Business Applications
Magdeburg Research and Competence Cluster VLBA
Faculty of Computer Science, Otto von Guericke University Magdeburg
{sascha.bosse|klaus.turowski}@ovgu.de

Abstract

In-Memory technologies provide the performance to deal with the emerging problems of Big Data. However, the availability of these computing systems is a crucial issue. One way to provide high availability is to form clusters of active and standby In-Memory computing nodes. By analyzing and modeling a SAP HANA® cluster provided by the Future SOC Lab, an availability prediction model for $n + m$ clusters was derived in this project. The obtained results are supposed to support decision-makers on designing In-Memory clusters in terms of availability. In the future, these models can be utilized to analyze other high availability strategies.

1. Introduction

1.1 Motivation and Project Idea

The IT industry evolves rapidly, new concepts such as Cloud Computing and Social Media change the way of business operation. However, each new technology has its disadvantages and risks which should be considered before it is implemented within an enterprise. In the last years, the concept of In-Memory Computing has emerged as a new technology in the IT sector. Considering the continuing growth of measured data and its increasing importance for decision-makers in enterprises, this concept is promising to deal with the problems and challenges of Big Data. With the introduction of SAP HANA® in 2010, In-Memory Computing reached the commercial market and was introduced in many companies, increasing the performance of data processing significantly, especially when supported by a cluster of distributed In-Memory nodes (referred to as Scale-Out by SAP).
Nevertheless, in addition to performance, availability is also a crucial non-functional aspect of IT systems [6]. Availability is defined as the "ability of a service [...] to perform its agreed function when required" [5]. Non-available systems may lead to revenue loss for the customer and to loss of reputation for the vendor of In-Memory systems [4]. Since no complex system can be free of faults, possible strategies to deal with unavailability are fault prevention, fault removal, fault forecasting and fault tolerance [7]. This project concentrated on the latter strategy, particularly achieving fault tolerance by introducing spare nodes to take over operation if active nodes fail. In case of SAP HANA®, this strategy is referred to as Host Auto-Failover [11]. While response time and space requirements determine how many active nodes are needed to reach the desired performance level, the availability of the whole distributed system strongly depends on the number of spare nodes. In this project, experiments were carried out combining analytical availability prediction models with measured data from real SAP HANA® instances. The obtained results were generalized in order to provide decision-support for In-Memory cluster designers in terms of availability.

1.2 Provided Future SOC Lab Resources

This project was conducted in cooperation with the Future SOC Lab at the Hasso Plattner Institut in Potsdam and its industry partners SAP, Fujitsu, EMC and HP. In the lab's 2014 summer term, a single SAP HANA® instance as well as a two-node distributed SAP HANA® system were provisioned in order to support this project. The provided resources and contacts were used to create and parametrize the availability prediction models for an In-Memory computing cluster.

2. Theoretical Background

2.1 In-Memory Computing Cluster Using the Example of SAP HANA®

The content of this subsection is mainly derived from the HANA cookbook, available at `https://cookbook.experiencesaphana.com`, and the HANA High Availability Guide [11].

In order to optimize In-Memory computing performance and availability, several computing nodes can be arranged in a cluster. In the case of SAP HANA®, this is called Scale Out Landscape. Each node consists of software and hardware components while the initial state of the software components is determined by the cluster architecture. In figure 1, a cluster architecture of a $2 + 1$-solution is presented.

The master and each of the worker servers host a name server as well as an index server. The master name server administrates the cluster's topology and connects the shared data and log volumes to the worker nodes, the index servers are responsible for data processing. A statistics server, collecting information about the status of each node, is run only on the master node and is deactivated on other nodes (represented by a dashed line in figure 1). Nodes that are declared as standby nodes are completely deactivated.

When a failure occurs, different mechanisms are implemented depending on the failed component. If the master name server fails, a slave name server can take over with nearly zero downtime. A similar mechanism is performed if the statistics server fails. The outage of a slave name server does not affect the cluster availability unless it has to take over for the master name server. If an index server or a complete host fails, one of the declared standby nodes will be started to take over operation of the failed node. In this case, the master name server connects the corresponding data and log volumes to the standby node. The outage of the data and log volumes as well as of the network will cause the whole cluster to be unavailable. Therefore, these components are fully redundant within the SAP HANA® cluster.

Figure 1 also presents the failure and recovery time for each modeled component. These times are modeled as exponentially distributed random variables characterized by the mean time to failure (MTTF) and the mean time to recover (MTTR). Since empirical values about the SAP HANA® components over long-time periods were not available, the parameters were derived from field studies, e.g. [12, 8, 9].

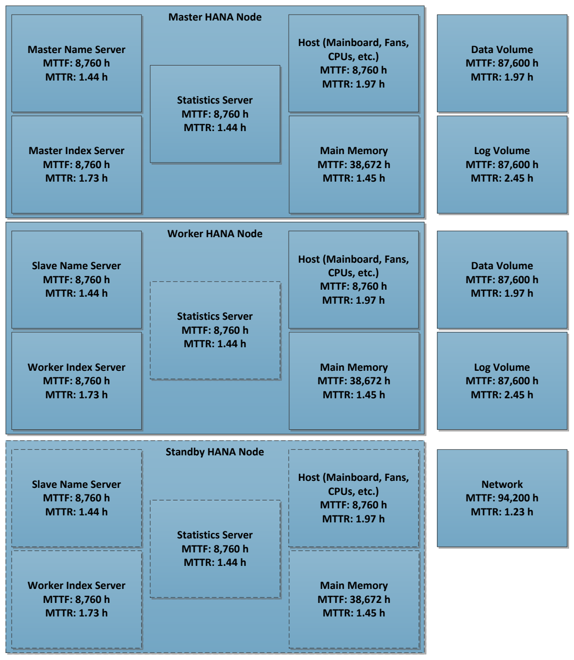

Master HANA Node
Master Name Server MTTF: 8,760 h MTTR: 1.44 h
Statistics Server MTTF: 8,760 h MTTR: 1.44 h
Host (Mainboard, Fans, CPUs, etc.) MTTF: 8,760 h MTTR: 1.97 h
Master Index Server MTTF: 8,760 h MTTR: 1.73 h
Main Memory MTTF: 38,672 h MTTR: 1.45 h
Data Volume MTTF: 87,600 h MTTR: 1.97 h
Log Volume MTTF: 87,600 h MTTR: 2.45 h
Worker HANA Node
Slave Name Server MTTF: 8,760 h MTTR: 1.44 h
Statistics Server MTTF: 8,760 h MTTR: 1.44 h
Host (Mainboard, Fans, CPUs, etc.) MTTF: 8,760 h MTTR: 1.97 h
Worker Index Server MTTF: 8,760 h MTTR: 1.73 h
Main Memory MTTF: 38,672 h MTTR: 1.45 h
Data Volume MTTF: 87,600 h MTTR: 1.97 h
Log Volume MTTF: 87,600 h MTTR: 2.45 h
Standby HANA Node
Slave Name Server MTTF: 8,760 h MTTR: 1.44 h
Statistics Server MTTF: 8,760 h MTTR: 1.44 h
Host (Mainboard, Fans, CPUs, etc.) MTTF: 8,760 h MTTR: 1.97 h
Worker Index Server MTTF: 8,760 h MTTR: 1.73 h
Main Memory MTTF: 38,672 h MTTR: 1.45 h
Network MTTF: 94,200 h MTTR: 1.23 h

Figure 1. Modeled Components of a SAP HANA® Cluster on the Example of a $2{+}1$-Solution

2.2 Availability Prediction for Distributed IT Systems Using Petri Net Simulation

In order to predict the availability of a complex system without reliable data about system-availability, analytical models can be applied. In these methods, the system is modeled as an interplay between its components for which availability data can be estimated. The simplest approaches are combinatorial ones, assuming independence between the single components. However, this assumption is not appropriate in modern IT systems [2]. Especially the standby dependency between components cannot be mapped in these models, so state-space-based models must be used where each possible system state and the transition probabilities between them are described. Nevertheless, this procedure leads to the problem of state-space explosion for complex systems, limiting their scalability to industrial cases [10].

In order to improve scalability, the state-space can be encoded, for instance, in a Petri net and the model is solved by simulation [13]. In this project, one of these approaches presented in [1] was used and is briefly described in the following.

The availability prediction model is formulated as a colored generalized stochastic Petri net (GSPN), cf. e.g. [3]. A GSPN is a bipartite graph consisting of places (represented by void circles) and transitions (rectangles) which can either be immediate (filled black rectangles) or timed transitions (void rectangles). Places can be marked by colored tokens (filled circles) to indicate that the state or condition the place represents is fulfilled. Depending on the current marking of the GSPN, transitions are activated and fire immediately or after a stochastic time according to their type. If a transition fires, the marking of GSPN is changed indicating a new system state.

In figure 2, the availability model of an example component is displayed. The two places "Online" and "Offline" represent whether the component is currently available or not. The two timed transitions "Failure" and "Recovery" are responsible for changes in the component's state by firing the token after an exponentially distributed time parametrized by the MTTF

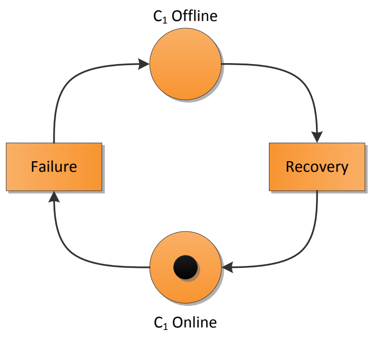


Figure 2. GSPN of a Simple Component

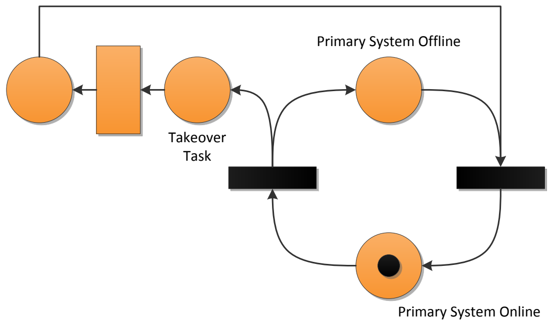


Figure 3. GSPN of a Standby Dependency

and MTTR. It is also possible to define more than one type of failure for a component which can be modeled by a token of different color, leading to different firing times of the transitions.

Single components can be arranged to component systems. These systems are modeled similar to a single component with the difference that the transitions between the two places are immediate. A so-called guard function, a Boolean expression, is assigned to these transitions so they do not fire unless the guard function is fulfilled. For example, in a serial system the failure transition is fired if one of the assigned components fails. In a k-of-n-parallel system, k components have to fail at the same point in time to cause the failure of the component system.

Between two components or component systems, a standby dependency can be defined. In this case, after the primary system has failed, a second token will be created by the failure transition to activate the timed takeover transition, as presented in figure 3. The standby component will be set online in the moment as the takeover transition fires and will be set offline again when the primary system has been recovered.

With the model elements described above, a prediction model for a $n + m$ SAP HANA® cluster with Host

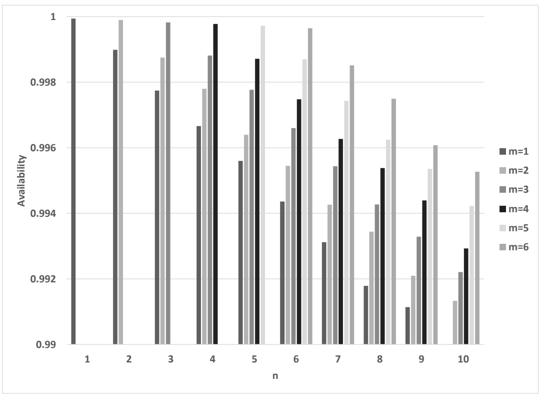


Figure 4. Availability of the Experimental SAP HANA® Cluster in Dependence to Worker (n) and Standby (m) Nodes

Auto-Failover can be generated.

3. Experiments

On the basis of the conceptual Petri net model, a simulation was implemented in the Java-based simulation framework *AnyLogic 6.9.0*. Therefore, a generic $n+m$ SAP HANA® cluster was modeled as a n-of-$(n+m)$ parallel system which means that at least n nodes have to be online to consider the cluster as available. A cold standby dependency as described above was implemented between each of the defined worker nodes and the m standby nodes, so in the case that the operation of a worker node fails a standby node takes over. The experiments were then conducted under the condition that the number of worker nodes is equal to or greater than the number of standby nodes, because the positive effect on the cluster availability would not justify the costs for an additional server. The model was simulated for a time frame of one year in 100 replications in order to get to statistically significant results. In figure 4, the results of the experiments in terms of mean availability are displayed for one to ten worker nodes and one to six standby nodes considering the condition that $m \leq n$.

It can be stated that the availability of a $1 + 1$ cluster (99.9941%) is the upper bound for the cluster availability. The introduction of additional worker nodes increases the probability of a single node failure and, therefore, the cluster downtime since for the takeover time the cluster is not available. In a $n + 1$ system, the availability decreases subsequently to 99.0% for ten worker nodes which can no longer be perceived as high availability. However, installing the same number of standby nodes as worker nodes can preserve the upper bound availability with only slight losses, cf. table 1.

After performing the experiments for n and m from

$n = m$	C_min	C_mean	C_max
1	0.99993341	0.999940606	0.999947802
2	0.99988820	0.999896902	0.999905599
3	0.99981158	0.999825997	0.999840413
4	0.99975589	0.999777018	0.999798148
5	0.99969923	0.999722553	0.999745875
6	0.99961492	0.999645600	0.999676281
7	0.99958171	0.999609952	0.999638199
8	0.99951511	0.999550188	0.999585272
9	0.99946182	0.999492152	0.999522481
10	0.99940598	0.999436638	0.999467299

Table 1. Confidence Intervals for the Cluster Availability for $n = m$

one to ten, it was tried to generalize the obtained results in a functional dependency. Since the results indicate a linear relation, multilinear regression was applied to the results for $n \leq 6$ using the method of least squares for parameter estimation. The derived formula, with a mean square error of $1.58 \cdot 10^{-8}$ for the simulation results, is presented in equation 1.

$$A(n,m) = 1.000015 - 0.001109n + 0.001052m \quad (1)$$

In order to avoid overfitting, this formula was tested against the simulation results for $6 < n \leq 10$. For these values, the mean square error is $7.67 \cdot 10^{-9}$ which confirms the regression formula. The formula also supports the previous observations: the introduction of additional worker nodes has a negative influence on cluster availability that can be compensated by the introduction of additional standby nodes. This can be accomplished at least to a certain degree because the 'm factor' is less than the 'n factor'.

4. Conclusion

In this project, an availability prediction model for an In-Memory computing cluster was constructed. The defective components of a single node were considered as well as a $n + m$ active/standby cluster architecture. Therefore, a distributed SAP HANA® cluster provided by the Future SOC Lab was analyzed. The followed modeling approach bases on Petri net simulation which ensures scalability to more complex scenarios. The developed model was used to simulate different cluster scenarios in order to generalize the dependency between cluster architecture and availability in a functional dependency. The so derived formula is desired to provide decision-support for the designing of highly available In-Memory computing clusters.

However, the modeled active/standby system is only one approach to achieve high availability in these clusters. For instance, in addition to the Host Auto-Failover, the SAP HANA® system provides other high availability strategies such as storage and system replication in which a mirror cluster site is used to perform disaster recovery. These strategies have to be added to the developed prediction approach in order to increase the accuracy of the approach and, therefore, its suitability for decision-support.

While availability was defined in this project only by analyzing the ratio of downtime to operational time, another crucial aspect of In-Memory cluster availability is the avoidance of data losses. Therefore, the probability and the effect of data losses are important for decision makers when designing an In-Memory computing cluster. Hence, these aspects should also be considered in future research.

Nevertheless, this project provides a first step towards a decision-support system to design highly available In-Memory computing clusters with respect to costs. Together with the high performance of these clusters this may lead to a higher acceptance for this novel and promising technology.

References

[1] S. Bosse. Predicting an it service's availability with respect to operator errors. In *Proceedings of the 19th Americas Conference on Information Systems (AMCIS)*, Chicago, IL, USA, 2013.

[2] G. Callou, P. Maciel, D. Tutsch, J. Araújo, J. Ferreira, and R. Souza. A petri net-based approach to the quantification of data center dependability. In P. Pawlewski, editor, *Petri Nets - Manufacturing and Computer Science*, chapter 14, pages 313–336. InTech, 2012.

[3] G. Ciardo, J. Muppala, and K. Trivedi. Spnp: Stochastic petri net package. In *Proceedings of the 3rd International Workshop PNPM*, pages 142–151. IEEE Computer Society, 1989.

[4] V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya, and C. A. F. D. Rose. Towards autonomic detection of sla violations in cloud infrastructures. *Future Generation Computer Systems*, 28(7):1017–1029, 2012.

[5] L. Hunnebeck. *ITIL Service Design 2011 Edition*. The Stationery Office, Norwich, UK, 2011.

[6] A. Keller and H. Ludwig. The WSLA Framwork: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11:57–81, 2003.

[7] J.-C. Laprie. Dependable computing: Concepts, limits, challenges. In *Proceedings of the 25th IEEE International Symposium on Fault-Tolerant Computing*, pages 42–54, Pasadena, CA, United States, June 27-30 1995.

[8] N. Milanovic and B. Milic. Automatic generation of service availability models. *IEEE Transactions on Service Computing*, 4(1):56–69, 2011.

[9] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)*, 2007.

[10] A. Sachdeva, D. Kumar, and P. Kumar. Reliability analysis of pulping system using petri nets. *International Journal of Quality & Reliability Management*, 25:860–877, 2008.

[11] SAP HANA Development Team. *SAP HANA(tm) - High Availability*. SAP, January 2014.

[12] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM Errors in the Wild: A Large-Scale Field Study. *Communications of the ACM*, 54:100–107, 2011.

[13] V. Zille, C. Bérenguer, A. Grall, and A. Despujols. Simulation of maintained multicomponent systems for dependability assessment. In P. Faulin, A. Juan, S. Martorell, and J. Ramírez-Márquez, editors, *Simulation Methods for Reliability and Availability of Complex Systems*, chapter 12, pages 253–272. Springer, Berlin, Heidelberg, 2010.

Offloading decision based on network security condition under the specific threat of timing attacks

Tianhui Meng
Freie Universität Berlin
Institut für Informatik
Takustraße 9, 14195 Berlin
tianhui.meng@fu-berlin.de

Abstract

Mobile devices have limited computation resources and battery capability, which limit the development of mobile technology. To alleviate these restrictions, computation offloading technics emerge by sending heavy computation to resourceful servers. For the sake of improving the offloading security, offloading decisions will be made to decide whether it is secure to offload in a certain period of time, according to the success probability of timing attacks. Models will be built to decide whether to offload and the retry time if the network is under the threat of timing attacks.

1 Introduction

Mobile devices (e.g., smartphones, tablets, etc) are increasingly becoming an essential part of human life as the most effective and convenient communication tools not bounded by time and place. However, mobile systems have limited resources, such as battery life, network bandwidth, storage capacity, and processor performance. These restrictions may be alleviated by computation offloading: sending heavy computation to resourceful servers and receiving the results from these servers. But when the network is under threat of timing attacks, data secrecy and user privacy cannot be guaranteed.

Recently, timing attack is becoming one of the main security concerns for mobile offloading [1]. In timing attack, a class of side channel attack, the attacker attempts to break a cryptographic algorithm running on the server by timing the operations of a certain cloud offloading system. It was commonly believed that timing attacks can be taken only on smart cards or inter-process locally, but later research relieve that remote timing attacks are also practical and should be taking care of. The current study mainly contributes to the timing attack on RSA-CRT. Many technologies shall be developed to handle the security risks in offloading.

In this project, offloading decision based on network status will be presented to enhance the safety performance in offloading technic. Mobile devices can decide whether to offload the heavy workload to powerful servers according to timing attack success probability. A Petri-net model will be built to calculate the success probability of timing attack in a certain period of time when heavy computation work should be done by offloading. If it is not secure to offload at that time, the mobile device will do a retry after a certain amount of time. The restart time will be decided by another model. The remainder of this report is structured as follows: Section 2 gives an overview of the timing attack and RSA implementation. The general idea of remote timing attack is introduced in Section 3. In Section 4, implementation details and experiment results are addressed. Section 5 present the conclusion and future work.

2 Timing attacks and RSA implementation

Timing attacks enable an attacker to extract secrets maintained in a security system by observing the time it takes the system to respond to various queries. The idea of timing attack was first suggested by Kocher in 1996 [3]. This attack method is trying to attack the implementation of cryptographic algorithms, rather than the algorithm itself. As long as there is a difference among the execution time of the encryption system during the implementation of the different cipher text decryption, the key is likely to be cracked.

As Kocher suggested, the attack can be treated as a signal detection problem. The attacker get the computation timing measurements T_0, T_1, ..., T_{j-1} for j messages y_0, y_1, ..., y_{j-1}. To get the secure key $x = [1001011010\ldots]_2$, he guess the first b bites of x and he knows the computation time for the first b iterations of the y_i^x mod n. After subtracting known part of time, the attacker guesses the next bit x_b by computing the variances. The correct guess will be identified successfully if its adjusted values have the smaller variance.

Since then, several papers have presented new theoretical timing attack algorithms or have extended the existing ones. One problem of the attack presented by Kocher is that the attacker needs a very detailed knowledge of the implementation of the system, as he has to be able to compute the partial timings due to the known part of the key.

2.1 Local timing attack

In 1998, Dhem et al. [4] employed timing attack into a Smartcard that stores an RSA private key, which shows that the timing attack represents a practical, important threat against cryptosystems implementations, where the attacker can quite easily collect large amount of message decryptions and measure time with high precision.

It was believed that common implementations of RSA , using Chinese Remainder Theorem (CTR) and Montgomery Multiplication (MM), are not vulnerable to timing attacks, but Schindler [5] proposed a timing attack against the implementation of RSA exponentiation which employs CRT. However, Schindler's attack has many assumptions for the implementation of the cryptosystem, like the attacker has to know the modulus *n*. And Schindler could only start predicting the bit of q from the fifth bit, and the first few bits are assumed to be determined by the BB-attack.

2.2 Remote timing attack

What was also generally believed is that timing attacks cannot be used to attack general purpose servers, such as web servers, since decryption times are masked by many concurrent processes running on the system. But some papers focused on putting the results of such theoretical research into practice remotely.

OpenSSL is a well-known open-source crypto library which is often used on Apache Web Servers to provide SSL functions. Brumley and Boneh [6] [9] demonstrated that timing attack can reveal RSA private keys from an OpenSSL based web server over a local network. Aciicmez and Schindler [7] proposed an efficient attack on RSA implementations that use CRT with the Montgomery Multiplication (MM) algorithm, and suggested a general improvement of the decision strategy. Moreover, Crosby, Wallach and Riedi presented a carefully designed filter which can allow an attacker to measure events with 15-100 *μs* accuracy across the Internet and as good as 100 *ns* over a local network [8]. But none of these studies has applied these statistical methods with such efficiency. In 2013, Chen, Wang and Tian [10] uses the error detection mechanism and correction strategy proposed in the present study to improve timing attack on RSA algorithm in OpenSSL, in which the precision of the decision in improved and a 1024-bit RSA key is recovered completely for an inter-process timing attack. The current study mainly contributes to the improvement of feasibility of the timing attack on RSA-CRT.

We are much interested in the remote timing attacks, in the next section, we will test and verify the feasibility and practicability of this type of timing attack.

2.3 RSA implementation used in OpenSSL

To date, the RSA algorithm is still the most popular and secure public-key cryptographic system [2]. RSA algorithm was proposed by Rivest, Shamir and Adleman in 1977. Let p and q be two distinct large random primes. The modulus n is the product of these two primes: $n = p \times q$. Euler's totient function of n is given by $\phi(n) = (p-1)(q-1)$. A number e is selected, $1<e<\phi(n)$, such that $\gcd(e, \phi(n)) = 1$, and d is computed with $e \times d \bmod \phi(n) = 1$, using the extended Euclidean algorithm. Here e is the public exponent and d is the private exponent. The encryption is performed by computing $C = M^e \pmod n$, where M is the plaintext and C is the ciphertext, the decryption is performed using $M = C^d \pmod n$.

The core of RSA decryption is a modular exponentiation $M = C^d \pmod n$. OpenSSL uses CRT to perform this exponentiation. Since RSA with CRT uses the factors of N, a timing attack can expose these factors. Once the factorization of N is revealed it is easy to obtain the decryption key by computing $d = e^{-1} \bmod (p - 1)(q - 1)$, where e is the public encryption exponent. On the other hand, the Sliding Windows Exponentiation (SWE) algorithm performs a modular multiplication at every step. MM, discovered by Montgomery in 1985, is the most efficient algorithm for the computation of modular multiplications during modular exponentiation. It is where the implementation will reveal the time characteristics that the attacker needs.

3 A remote timing attack on RSA

RSA compute many modular exponentiation $g^d \bmod q$. That is where attackers get the information about q. In Montgomery reduction, an extra step called *extra reduction* causes a timing difference for different inputs [5]. Schindler noticed that the probability of an extra reduction during an exponentiation $g^d \bmod q$ is proportional to how close g is to q. Schindler showed that the probability for an extra reduction is:

$$\Pr[\texttt{Extra reduction}] = \frac{g \bmod q}{2R}. \qquad (1)$$

Consequently, as g approaches either factor p or q from below, the number of extra reductions during the exponentiation algorithm greatly increases. At exact multiples of p or q, the number of extra reductions drops dramatically. Figure 1 shows this relationship. By detecting timing differences that result from extra reductions we can tell how close g is to a multiple of one of the factors q. Another reason that causes the implementation time difference is multiplication routines. Time measurements will reveal how frequently the operands given to the multiplication routine have the same length.

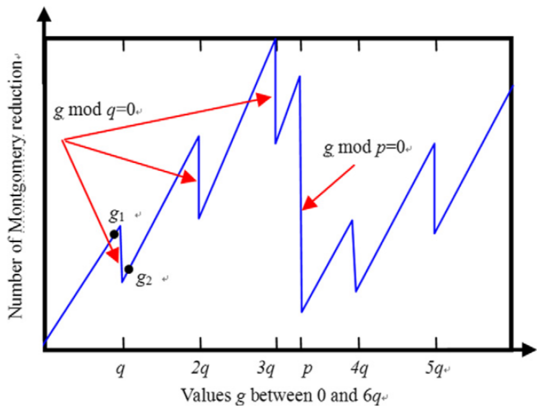


Figure 1: Number of extra reductions in a Montgomery reduction as a function of *g*

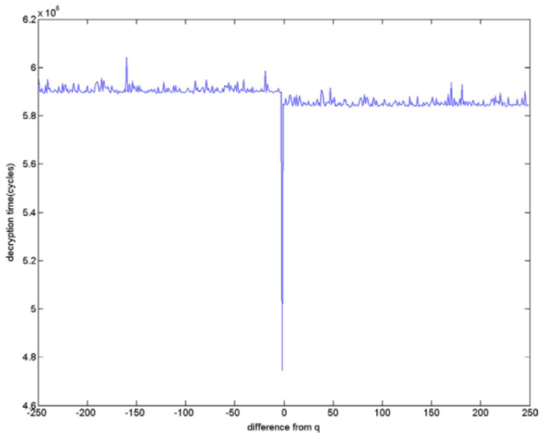


Figure 2: Timing the RSA decryption for different number near *q*

Here is how this attack is made: Firstly, the attacker guesses g lies between 2^{512} and 2^{511}, and then he times the decryption of all possible combinations of the top few bits. When plotted, he can get the top few bits of q from the first peak in the plot. After that, they get q bit one by one by comparing the decryption time difference Δ between different inputs g and g_{hi}. Where

$g = (q_0,q_1,...,q_{i-1},0,0,\dots,0)$

$g_{hi} = (q_0,q_1,...,q_{i-1},1,0,\dots,0)$

$\Delta = \text{DecryptTime}(g) - \text{DecryptTime}(g_{hi})$

If $g<q<g_{hi}$ then, the difference Δ will be ‘‘large’’, and bit i of q is 0. If $g<g_{hi}<q$, the difference Δ will be ‘‘small’’, and bit i of q is 1. The time distribution of the RSA decryption for numbers near q is shown in Figure 2.

4 Implementation details and results

We have implemented a file-upload server webpage based on Apache Tomcat 7.0.54. The server is deployed on a HP ProLiant DL980 G7 in Future SOC Lab. A Client is also developed to upload a text file to server and record the response time. When a file is uploaded to the server, the content in it will be read and encrypted, then decrypted to show the original text. The encryption and decryption time are recorded and sent back to the client. The client visited the server through VPN and measured the time from uploading the file to receiving the response from server. Both the server and client are developed in JAVA.

4.1 Response time characteristics

Firstly, we want to see the feature of response time. We implement a simple client to send http request and time the responses receive. We do the experiment under three different network environments: one is the client and the server are deployed on the same machine, so it shows us the local response time; another one is the server is in Potsdam and many routers far away from the client which is in Berlin, the client visits the server via VPN through the university network ‘eduroam’; the last one is the client visits the same remote server via VPN from a commercial networks, for which the carrier is ‘O2’. The results are shown in the following tables.

local visit				
start time	delay	number of samples	average (ms)	variance (ms)
17:15	no delay	1000	1.07	0.74
17:45	1s	1000	1.31	1.09

Table 1 local visit

remote visit from eduroam				
start time	delay	number of samples	average (ms)	variance (ms)
12:15	5s	700	11.45	100.663
15:15	5s	700	11.4	151.62
17:20	3s	400	7.8	68.97
16:40	2s	500	74.55	11190.1
17:00	2s	500	77.27	11803.6

Table 2 remote visit from eduroam

In the local visit scenario, the variance of the server response time is around 1ms. From [6], the network with less than 1 ms of variance is vulnerable to timing attack. So the local server is not safe under the threat of this type of attack. While, in the remote attacks, the response time are far more astable, of which variances are approximately 100 ms. In the best condition, the variance is about 20 ms. With a well-designed filter, it is also possible to conduct a timing attack.

In the remote visit scenario from ‘eduroam’, at 16:40 in the afternoon, the response time have the variance of more than 10 s. That is due to another download task is running on the client machine.

remote visit from o2 wlan				
start time	delay	number of samples	average (ms)	variance (ms)
0:15	2s	100	43.76	241.77
0:40	10s	1000	70.19	370.14
10:40	2s	200	36.06	21.85
10:40	4s	100	36.58	27.12
4:20	5s	500	108.11	12826
5:20	5s	500	112.01	3392.8

Table 3 remote visit from O2 WLAN

4.2 Time difference from response time

The second experiment is to evaluate the feasibility of timing attacks in the WiFi-VPN environment. For a HTTP client, the response sent back by server contains the encryption/decryption time and the original text in the file. The response time is consist with five parts shows in Figure 3.

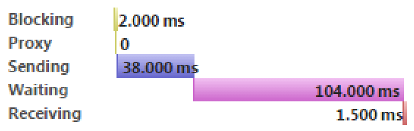


Figure 3: Response time composition

We want to verify if remote timing attacks are practical in our offloading scenario. The client sends two different file to the server to conduct RSA decrypt. We try to compare the response time difference that the client measured with the decryption time difference sent by the server. The results are shown in Table 4. For each data in the table, we sample the response time and decryption time multiple times and compute the mean time.

	in (ms)	1.txt	2.txt	time diff.
local	response time	0.82	1.31	0.49
	decryption time	0.32	0.85	0.53
eduroam	response time	7.8	8.64	0.84
	decryption time	0.11	0.91	0.8

Table 4 Measured time differences

When the client and the server are on the same machine, the response time difference is 0.49 ms, and the decryption time difference is 0.53 ms. So, in the local visit environment, timing the responses is sufficient to give an attacker the information he need to steal the secret key from the server. While in the remote visit scenario, in the optimal condition, the client can get a response time difference of 0.84 ms from average of several samples and the corresponding decryption time difference is 0.8 ms. But it is really hard to get a sufficient time different in the commercial network condition because the time variance is large. A filter is needed in this situation to help us to get rid of the noise in order to conduct a efficient timing attack.

5 Conclusions and future work

From our implementation of a timing attack client and the server running RSA encryption algorithm, we verify the feasibility of timing attack on the same machine and also on the remote server. The timing results can also be used in the models of [11] to calculate an attacker’s uncertainty about a secret.

In future work, a Petri-net model will be built to calculate the success probability of timing attack in a certain period of time when heavy computation work should be done by offloading. The mobile devices continue to record the response time of the server to give parameters to the model. A mobile device will conduct a retry after a certain amount of time if it not secure at that moment to offload. The restart time will be decided by another model.

References

[1] http://blog.astrumfutura.com/2010/10/nanosecond-scale-remote-timing-attacks-on-php-applications-time-to-take-them-seriously/

[2] A.J. Menezes, P.C. Oorschot, S.C. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, AM, 1997

[3] P. Kocher, Timing attack on implementations of Diffie–Hellman, RSA, DSS, and other systems, in: *Advances in Cryptology – CRYPTO’96*, LNCS, vol. 1109, 1996, pp. 104-113.

[4] J.F. Dhem, F. Koeune, P.A. Leroux, et al., A practical implementation of the timing attack, in: *International Conference on Smart Card Research and Advanced Applications – CARDIS 2000*, LNCS, vol. 1820, 2000, pp. 167–182.

[5] W. Schindler, A timing attack against RSA with the Chinese Remainder Theorem, in: *Cryptographic Hardware and Embedded Systems – CHES 2000*,LNCS, vol. 1965, 2000, pp. 109-124

[6] D. Brumley, D. Boneh, Remote timing attacks are practical, *Computer Networks* 48 (5), 2005, pp. 701–716.

[7] O. Aciicmez, W. Schindler, Improving Brumley and Boneh timing attack on unprotected SSL implementations, in: *Proceedings of the 12th ACM Conference on Computer and Communications Security – CCS 2005*, ACM, 2005, pp. 139–146.

[8] CROSBY, Scott A.; WALLACH, Dan S.; RIEDI, Rudolf H. Opportunities and limits of remote timing attacks. *ACM Transactions on Information and System Security (TISSEC)*, 2009, 12. Jg., Nr. 3, S. 17.

[9] B.B. Brumley and N. Tuveri. "Remote timing attacks are still practical." *Computer Security–ESORICS 2011*. Springer Berlin Heidelberg, 2011. 355-371.

[10] CaiSen Chen, Tao Wang, and Junjian Tian. "Improving timing attack on RSA-CRT via error detection and correction strategy." *Information Sciences* 232 (2013): 464-474.

[11] Köpf, Boris, and David Basin. "An information-theoretic model for adaptive side-channel attacks." In *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 286-296. ACM, 2007.