Doctoral Dissertation

Virtual Machines Live Migration Cost Modeling and Prediction

Author:
Mohamed Esameldin Mohamed Elsaid

Supervisor:
Prof. Dr. Christoph Meinel

Co-Supervisor:
Prof. Dr. Tilmann Rabl

Chair:
Internet-Technologien und Systeme

German Title:
Modellierung und Vorhersage der Live-Migrationskosten für Virtuelle Maschinen

Submission Date:
February 20, 2022

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-54001
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-540013

Acknowledgements

I give this research work to the soul of my father (Esameldin Mohamed Elsaid) who was continuously encouraging me to get the PHD degree until he passed away just one year before the submission of this thesis. So I send the effort done in this project to my father's sole to achieve his dream of attending my PHD defense. Also, I express my deep gratitude and thankfulness to my mother (Eglal Hassan) who gave and still give me the great and continuous support to work and finish my PHD project and to succeed in my life. I do not find enough words actually to thank and express my gratitude as well to my wife (Rahmatalla Mohamed Noureldin) who supports me and accepted to live with my tight schedule, working on vacations and continuous business to be able to get my PHD degree in parallel with my professional career. Thank you Rahmatalla for every single moment of support. From my family, I would like also to give this research work to the soul of my uncle (Ahmed Hassan) who had a great role in my life and was encouraging me to get the PHD degree. My great thanks goes also to my father in law (Mohamed Noureldin) and my manager at Intel (Andrew Green) who showed me all the support, encouragement and help needed to work on this research project and achieve this milestone in my life.

I would like also to express my great gratitude and appreciation to Prof. Meinel who supported my enrollment at HPI as external-PHD student with his supervision and supported my research from ideas to lab access, to papers publications and also in thesis review. Thank you Prof. Meinel. I thank also Prof. Rabl for his support in the thesis supervision and in the thesis review. My great thankfulness and appreciation to Prof. Hazem Abbas who supported me with his experience and guidance in publishing some of the papers related to this PHD work as the papers co-author. Finally, I thank all the thesis reviewers and examiners for running the doctoral committee and sharing their comments on the thesis content.

أهدي هذا العمل البحثي لروح والدي (عصام الدين محمد السيد) الذي كان يشجعني باستمرار للحصول على درجة الدكتوراه حتى وفاته قبل عام واحد فقط من تقديم هذه الرسالة. لذلك أرسل الجهد المبذول في هذا المشروع لأبي لتحقيق حلمه في حضورمناقشة الدكتوراه . كما أعرب عن عميق امتناني وشكري لأمي (إجلال حسن) التي قدمت لي ولا تزال تقدم لي الدعم الكبير والمستمر للعمل والانتهاء من مشروع الدكتوراه الخاص بي وأن أنجح في حياتي. أنا لا أجد كلمات كافية في الواقع لأشكر وأعبر عن امتناني أيضًا لزوجتي

(رحمة الله محمد نور الدين) التي تدعمني وتقبل العيش مع جدولي الزمني الضيق ، والعمل في الإجازات والعمل المستمر لأتمكن من الحصول على درجة الدكتوراه بالتوازي مع حياتي المهنية. شكرا لك رحمة الله لكل لحظة دعم. من عائلتي ، أود أيضا لإعطاء هذا العمل البحثي لروح خالي (أحمد حسن) الذي كان دور كبير في حياتي وكان يشجعني على الحصول على درجة الدكتوراه. عظيم الشكر أيضًا إلى والد زوجتي (محمد نور الدين) ومديري في شركة إنتل (أندى جرين) الذي أظهر لي كل الدعم والتشجيع والمساعدة اللازمة للعمل فى هذا المشروع البحثي و تحقيق هذا الإنجاز في حياتيكما أود أن أعبر عن عميق إمتناني وتقديري (للبروفيسور كريستوف ماينل) الذي دعم تسجيلي في جامعة بوتسدام كطالب دكتوراه مع إشرافه ودعم بحثي من الأفكار و حتى الوصول إلى المختبر و إلى الأوراق والمنشورات البحثية وكذلك في مراجعة الأطروحة. شكرا لك بروفيسور ماينل أشكر أيضًا (بروفيسور رابل) على دعمه في الإشراف على الرسائل ومراجعتها. شكري العظيم وتقديري (للأستاذ الدكتورحازم عباس) الذي قدم الدعم بخبرته وإرشاده في نشر بعض الأوراق العلمية بالرسالة حيث عمل كمؤلف مشارك لهذه الأوراق البحثية. أخيرًا ، أشكر جميع مراجعي الأطروحة والممتحنين لإدارة لجنة الدكتوراه وإبداء ملاحظاتهم المثمرة على محتوى الرسالة.

Abstract

Dynamic resource management is an essential requirement for private and public cloud computing environments. With dynamic resource management, the physical resources assignment to the cloud virtual resources depends on the actual need of the applications or the running services, which enhances the cloud physical resources utilization and reduces the offered services cost. In addition, the virtual resources can be moved across different physical resources in the cloud environment without an obvious impact on the running applications or services production. This means that the availability of the running services and applications in the cloud is independent on the hardware resources including the servers, networks and storage failures. This increases the reliability of using cloud services compared to the classical data-centers environments.
In this thesis we briefly discuss the dynamic resource management topic and then deeply focus on live migration as the definition of the compute resource dynamic management. Live migration is a commonly used and an essential feature in cloud and virtual data-centers environments. Cloud computing load balance, power saving and fault tolerance features are all dependent on live migration to optimize the virtual and physical resources usage. As we will discuss in this thesis, live migration shows many benefits to cloud and virtual data-centers environments, however the cost of live migration can not be ignored. Live migration cost includes the migration time, downtime, network overhead, power consumption increase and CPU overhead.
IT admins run virtual machines live migrations without an idea about the migration cost. So, resources bottlenecks, higher migration cost and migration failures might happen. The first problem that we discuss in this thesis is how to model the cost of the virtual machines live migration. Secondly, we investigate how to make use of machine learning techniques to help the cloud admins getting an estimation of this cost before initiating the migration for one of multiple virtual machines. Also, we discuss the optimal timing for a specific virtual machine before live migration to another server. Finally, we propose practical solutions that can be used by the cloud admins to be integrated with the cloud administration portals to answer the raised research questions above.
Our research methodology to achieve the project objectives is to propose empirical models based on using VMware test-beds with different benchmarks tools. Then we make use of the machine learning techniques to propose a prediction approach for virtual machines live migration cost. Timing optimization for live migration is also proposed in this thesis based on using the cost prediction and data-centers network utilization prediction. Live migration with persistent memory clusters is also discussed at the end of the thesis. The cost prediction and timing optimization techniques proposed in this thesis could be practically

integrated with VMware vSphere cluster portal such that the IT admins can now use the cost prediction feature and timing optimization option before proceeding with a virtual machine live migration.
Testing results show that our proposed approach for VMs live migration cost prediction shows acceptable results with less than 20% prediction error and can be easily implemented and integrated with VMware vSphere as an example of a commonly used resource management portal for virtual data-centers and private cloud environments. The results show that using our proposed VMs migration timing optimization technique also could save up to 51% of migration time of the VMs migration time for memory intensive workloads and up to 27% of the migration time for network intensive workloads. This timing optimization technique can be useful for network admins to save migration time with utilizing higher network rate and higher probability of success.
At the end of this thesis, we discuss the persistent memory technology as a new trend in servers memory technology. Persistent memory modes of operation and configurations are discussed in details to explain how live migration works between servers with different memory configuration set up. Then, we build a VMware cluster with persistent memory inside server and also with DRAM only servers to show the live migration cost difference between the VMs with DRAM only versus the VMs with persistent memory inside.

Zusammenfassung

Die dynamische Ressourcenverwaltung ist eine wesentliche Voraussetzung für private und öffentliche Cloud-Computing-Umgebungen. Bei der dynamischen Ressourcenverwaltung hängt die Zuweisung der physischen Ressourcen zu den virtuellen Cloud-Ressourcen vom tatsächlichen Bedarf der Anwendungen oder der laufenden Dienste ab, was die Auslastung der physischen Cloud-Ressourcen verbessert und die Kosten für die angebotenen Dienste reduziert. Darüber hinaus können die virtuellen Ressourcen über verschiedene physische Ressourcen in der Cloud-Umgebung verschoben werden, ohne dass dies einen offensichtlichen Einfluss auf die laufenden Anwendungen oder die Produktion der Dienste hat. Das bedeutet, dass die Verfügbarkeit der laufenden Dienste und Anwendungen in der Cloud unabhängig von den Hardwareressourcen einschließlich der Server, Netzwerke und Speicherausfälle ist. Dies erhöht die Zuverlässigkeit bei der Nutzung von Cloud-Diensten im Vergleich zu klassischen Rechenzentrumsumgebungen.
In dieser Arbeit wird das Thema der dynamischen Ressourcenverwaltung kurz erörtert, um sich dann eingehend mit der Live-Migration als Definition der dynamischen Verwaltung von Compute-Ressourcen zu beschäftigen. Live-Migration ist eine häufig verwendete und wesentliche Funktion in Cloud- und virtuellen Rechenzentrumsumgebungen. Cloud-Computing-Lastausgleich, Energiespar- und Fehlertoleranzfunktionen sind alle von der Live-Migration abhängig, um die Nutzung der virtuellen und physischen Ressourcen zu optimieren. Wie wir in dieser Arbeit erörtern werden, zeigt die Live-Migration viele Vorteile für Cloud- und virtuelle Rechenzentrumsumgebungen, jedoch können die Kosten der Live-Migration nicht ignoriert werden. Zu den Kosten der Live-Migration gehören die Migrationszeit, die Ausfallzeit, der Netzwerk-Overhead, der Anstieg des Stromverbrauchs und der CPU-Overhead.
IT-Administratoren führen Live-Migrationen von virtuellen Maschinen durch, ohne eine Vorstellung von den Migrationskosten zu haben. So kann es zu Ressourcenengpässen, höheren Migrationskosten und Migrationsfehlern kommen. Das erste Problem, das wir in dieser Arbeit diskutieren, ist, wie man die Kosten der Live-Migration virtueller Maschinen modellieren kann. Zweitens untersuchen wir, wie maschinelle Lerntechniken eingesetzt werden können, um den Cloud-Administratoren zu helfen, eine Schätzung dieser Kosten zu erhalten, bevor die Migration für eine oder mehrere virtuelle Maschinen eingeleitet wird. Außerdem diskutieren wir das optimale Timing für eine bestimmte virtuelle Maschine vor der Live-Migration auf einen anderen Server. Schließlich schlagen wir praktische Lösungen vor, die von den Cloud-Admins verwendet werden können, um in die Cloud-Administrationsportale integriert zu werden, um die oben aufgeworfenen Forschungsfragen zu beantworten.

Unsere Forschungsmethodik zur Erreichung der Projektziele besteht darin, empirische Modelle vorzuschlagen, die auf der Verwendung von VMware-Testbeds mit verschiedenen Benchmark-Tools basieren. Dann nutzen wir die Techniken des maschinellen Lernens, um einen Vorhersageansatz für die Kosten der Live-Migration virtueller Maschinen vorzuschlagen. Die Timing-Optimierung für die Live-Migration wird ebenfalls in dieser Arbeit vorgeschlagen, basierend auf der Kostenvorhersage und der Vorhersage der Netzwerkauslastung des Rechenzentrums. Die Live-Migration mit Clustern mit persistentem Speicher wird ebenfalls am Ende der Arbeit diskutiert. Die in dieser Arbeit vorgeschlagenen Techniken zur Kostenvorhersage und Timing-Optimierung könnten praktisch in das VMware vSphere-Cluster-Portal integriert werden, so dass die IT-Administratoren nun die Funktion zur Kostenvorhersage und die Option zur Timing-Optimierung nutzen können, bevor sie mit einer Live-Migration der virtuellen Maschine fortfahren. Die Testergebnisse zeigen, dass unser vorgeschlagener Ansatz für die VMs-Live-Migrationskostenvorhersage akzeptable Ergebnisse mit weniger als 20% Fehler in der Vorhersagegenauigkeit zeigt und leicht implementiert und in VMware vSphere als Beispiel für ein häufig verwendetes Ressourcenmanagement-Portal für virtuelle Rechenzentren und private Cloud-Umgebungen integriert werden kann. Die Ergebnisse zeigen, dass mit der von uns vorgeschlagenen Technik zur Timing-Optimierung der VMs-Migration auch bis zu 51% der Migrationszeit für speicherintensive Workloads und bis zu 27% der Migrationszeit für netzwerkintensive Workloads eingespart werden können. Diese Timing-Optimierungstechnik kann für Netzwerkadministratoren nützlich sein, um Migrationszeit zu sparen und dabei eine höhere Netzwerkrate und eine höhere Erfolgswahrscheinlichkeit zu nutzen.
Am Ende dieser Arbeit wird die persistente Speichertechnologie als neuer Trend in der Server-Speichertechnologie diskutiert. Die Betriebsarten und Konfigurationen des persistenten Speichers werden im Detail besprochen, um zu erklären, wie die Live-Migration zwischen Servern mit unterschiedlichen Speicherkonfigurationen funktioniert. Dann bauen wir einen VMware-Cluster mit persistentem Speicher im Server und auch mit Servern nur mit DRAM auf, um den Kostenunterschied bei der Live-Migration zwischen den VMs mit nur DRAM und den VMs mit persistentem Speicher im Server zu zeigen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Problems

Live Migration is a key technology and essential feature in datacenter virtualization. With live migration, the VMs can be moved from a physical host to another with almost no impact on the running applications availability. This makes the running applications get affected by the entire physical server issues; which enhances the service availability dramatically. Many research articles have discussed live migration optimization and cost modeling, however in this work we focus on research problems that are not addressed before. The problems that we aim to solve in this research project are:

1. Live migration cost modeling for VMware platforms has few contributions in the related work compared to open source hypervisors like Xen and KVM. So it is required to perform live migration cost modeling validation on VMware clusters.

2. IT admins run live migration requests without an idea about the cost beyond each live migration request. This might lead to resources bottlnecks, higher migration cost and migration failures; especially for large memory Virtual Machines (VMs). The existing live migration cost prediction publications present complex techniques that are not practical to be integrated with the virtual datacenter clusters portals.

3. It is not obvious how to integrate a live migration cost prediction algorithm with cloud administration portal such that the IT admins can make use of this feature as an integrated feature in the clusters management portals.

4. To the best of our knowledge, no article has discussed the idea of timing optimization for VMs migration based on migration cost prediction. Therefore, a practical technique should be developed to help the IT admins in choosing the optimal timing for a VM live migration to have minimal cost.

5. To the best of our knowledge, there is no article that discusses live migration cost modelling difference with using the standard DRAM memory versus using the persistent memory as a new trend in memory technology.

Based on the problems discussed in the previous section, the raised questions that should be answered in this thesis are:

1. Does the live migration cost change from a hypervisor to another?
2. What is the estimated live migration cost for a VM before proceeding with migration?
3. What is the optimal time for a VM to be migrated?
4. How to integrate the proposed techniques with Cloud administration portals?
5. Is there a difference between DRAM and Persistent Memory in live migration cost?

1.2 Research Methodology

Our research methodology to address the research problems and answer the raised questions are:

1. To obtain empirical modeling for live migration cost using a VMware cluster test-bed and with using different VMs sizes and different benchmark tools for single and multiple VMs migration.
2. To make use of machine learning techniques in order to propose a live migration cost prediction approach. The proposed approach should be simple and practical to be implemented and used by the IT admins.
3. For live migration timing optimization, we make use of datacenter network utilization prediction that were proposed by many published articles using different prediction techniques. We use one of these network utilization prediction techniques in addition with our proposed live migration network overhead prediction to propose an optimal time for a VM live migration to the IT admins.
4. For integration with virtual datacenters and cloud computing administration portals, we integrate the proposed prediction technique with VMware vSphere User Interface (UI) as a commonly used hypervisor at enterprise datacenters.
5. To repeat the migration tests on VMware cluster with persistent memory included to compare the modeling results between DRAM and persistent memory in VMs live migration.

1.3 Thesis Contribution

The thesis contribution can be listed in the below points:

1. We provide a literature review on live migration cost modeling and prediction. Then we classify these works according to the cost parameter and show how it is modeled in the different research papers.

2. Related work articles show that cost modeling for live migration focus mainly on open source hypervisors and the contribution for VMware environments is obviously few. Hence, our first contribution is to propose empirical cost modeling for virtual machines live migration in VMware environments.

3. Machine learning based approach is proposed for live migration cost prediction. The proposed approach considers the simplicity of the prediction technique for integration with the virtual datacenters administration portals.

4. A novel timing optimization technique is also proposed in this thesis to provide an optional guidance to the IT admins about the optimal time to run live migration for a specific VM based on the datacenter network utilization prediction.

5. The proposed cost prediction and timing optimization technique could be integrated as a built-in plug-in with VMware vSphere user interface to show an example for practical implementation for the proposed algorithm with cloud administration interfaces. This facilitates using cost prediction and timing optimization features by the IT admins as an extended option before proceeding with the VMs live migration.

6. In this thesis, we discuss also the persistent memory technology as a new servers memory technology that is supported by different hypervisors. We validate in this thesis if live migration cost modeling with the standard DRAM can be used also for persistent memory.

The proposed work in this thesis is published in the following papers:

1. Elsaid, M.; Abbas, H. and Meinel, C. Virtual Machines Pre-Copy Live Migration Cost Modeling and Prediction: A Survey, 2021. Distributed and Parallel Databases. Springer (Final revision processed)

2. Mohamed Esam Elsaid, Mohamed Sameh, Hazem M. Abbas and Christoph Meinel. Live migration timing optimization integration with vmware environments. In Donald Ferguson, Claus Pahl, and Markus Helfert, editors, Cloud Computing and Services Science, pages 133–152, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72369-9.

3. Elsaid, M.; Abbas, H. and Meinel, C. (2020). Live Migration Timing Optimization for VMware Environments using Machine Learning Techniques. In Proceedings of the 10th International Conference on Cloud Computing and Services Science, 2020 - Volume 1: CLOSER, ISBN 978-989-758-424-4, pages 91-102. DOI: 10.5220/0009397300910102

4. Mohamed Esam Elsaid, Hazem M. Abbas, and Christoph Meinel. Machine learning approach for live migration cost prediction in vmware environments. In Vıctor Mendez Munoz, Donald Ferguson, Marku sHelfert, and Claus Pahl, editors, Proceedings of the 9th International Conference on Cloud Computing and Services Science, CLOSER 2019, Heraklion, Crete, Greece, May 2-4, 2019, pages 456–463. SciTePress,2019. doi: 10.5220/0007749204560463.

5. M. E. Elsaid, A. Shawish and C. Meinel, "Enhanced Cost Analysis of Multiple Virtual Machines Live Migration in VMware Environments," 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2), Paris, France, 2018, pp. 16-23, doi: 10.1109/SC2.2018.00010.

6. M. E. Elsaid, A. Shawish, and C. Meinel. A Social Aware Approach for On-line Social Networks Data Allocation and Replication. In 2017 European Conference on Electrical Engineering and Computer Science (EECS), pages 266–271, 2017. doi: 10.1109/EECS.2017.56.

7. M. E. Elsaid and C. Meinel. Multiple Virtual Machines Live Migration Performance Modelling – VMware vMotion based Study. In 2016 IEEE International Conference on Cloud Engineering (IC2E), pages 212–213,April 2016. doi: 10.1109/IC2E.2016.9.

8. M. E. Elsaid and C. Meinel. Friendship based storage allocation for onlinesocial networks cloud computing. In 2015 International Conference onCloud Technologies and Applications (CloudTech), pages 1–6, 2015. doi:10.1109/CloudTech.2015.7336970.

9. M. E. Elsaid and C. Meinel. Live migration impact on virtual datacenter performance: Vmware vmotion based study. In 2014 International Conference on Future Internet of Things and Cloud, pages 216–221, Aug 2014.doi: 10.1109/FiCloud.2014.42.

1.4 Thesis Organization

The next chapters of the thesis are organized as following, in Chapter 2 we discuss the thesis background that covers the dynamic resource management topic and focuses on live migration process and configuration details. In Chapter 3, we provide a related survey and classification for the related articles that discuss live migration cost. Chapter 4 presents the testing environment using a VMware cluster and explains how to integrate an algorithm with VMware user interface to be added as a built-in plug-in. Live Migration cost modeling is presented in Chapter 5 as empirical models. Then the proposed cost prediction approach is discussed in Chapter 6. In Chapter 7, we propose the timing optimization approach for virtual machines live migration. Live migration for VMs with persistent memory is discussed in Chapter 8 and then we conclude the thesis in Chapter 9.

Chapter 2

Background

2.1 Dynamic Resources Management in Virtual Environments

Virtualization concept relies on resources sharing such that the datacenters resources of compute, network and storage can be shared between different operating systems and applications with a physical layer based isolation [122]. Dynamic resource management in virtual datacenters and cloud computing means that the physical resources assignment for the virtual resource is dependent on the actual demand. So the hypervisor can scale up and down the resources based on the actual needs [80]. In case of over-commitment when a high demand for the physical resources is requested and can not be fulfilled to meet the capacity requirements of the virtual resource, a virtual resource migration to another physical machine that can meet the physical resources demand is required [111], [134]. This concept applies to storage, network or in compute resources. Not only in an over-commitment situation, but also virtual resources migration can be used for other purposes like performance enhancement [57], [138] or power saving objectives [123], [83] as part of the dynamic resource management feature in virtual datacenters and cloud computing environment.

2.1.1 Storage Resources Dynamic Management

Data storage increases dramatically with the massive data demand on social networks, e-commerce, IoT, online video sharing platforms, enterprise applications and many other data sources. Users and IT admins look for a storage environment that is reliable, secure, cost efficient and meets the performance requirements. Storage virtualization is an abstraction technology of the storage layer that separates the hosts' view from the physical storage systems [30].

2.1.1.1 Virtual Storage Migration and Replication

Virtual data-centers and cloud computing environments provide a virtualization layer for the storage platform in order to meet the data storage cost efficiency, availability, reliability and rapid scaling requirements. In storage virtualization, the storage layer of the VM disk is implemented as a set of files to be stored

on the physical medium. Therefore, in the case of the VM replication or migration, the VMs disk files can be smoothly transferred and shared across different physical hosts and different physical storage servers. With virtual storage, the storage resources can be more cost efficient, rapidly scaled and be more reliable and available. Virtual storage facilitates the storage migration in cloud and virtual storage environments by abstracting the storage of the virtual machines in a set of disk files or data stores using the platform hypervisor in stead of physically attached disks to server [31] and [5]. Using storage virtualization, live storage migration feature can be supported by transferring the VMs disks files to the target storage server without applications disruptions [31], [5]. Virtual storage migration and replication can be used for different reasons depending on the applications requirements like cost saving, higher durability, higher availability, load balance and performance enhancement of the application [106].
Cost saving can be achieved also with live storage migration of the virtual storage data stores. Data can be moved to a cheaper storage tier when the frequency of the data access becomes lower [86]. Storage tiering is dependent on the access rate of the data, for high access rate of the data a hot tier should be used to afford the high storage performance requirement. When the data access rate decreases a migration to the cold tier that has slower performance can be done to save the storage cost. This concept applies for private, public and hybrid cloud [86].
For high durability, data should not be lost and for high availability there should be accessible. These two parameters are measured in multiple of nines probability; AWS S3 for example shows five nines (99.999) percentage against data unavailability and 11 nines against data loss [1], [107]. VMs storage replication and migration techniques are used in cloud environments to have multiple copies of the data in order to increase the service availability and durability levels [107]. This redundancy in data storage avoid a single point of failure in the storage environment such that the application server can be assigned to another virtual storage node when a failure happens at the primary node[11].
Storage migration and replication can be useful also for the cloud environment load balance [103], [85]. With data replication of the virtual storage resources, data is stored in multiple copies across different storage nodes and the application server chooses the storage nodes that avoids resources bottleneck situation and meet the application Service Level Agreement (SLA) requirement.
Application performance optimization can be achieved also by using the virtual storage replication and migration techniques. In [66], storage replication of VMs could be used to optimize Hadoop Distributed File System (HDFS) performance. The authors use SVM to classify the storage nodes of the HDFS cluster based on the nodes load and topology. The top nodes are selected for replication to achieve better load balance between the nodes with less replication overhead. In [70] and [72], we proposed storage migration and replication techniques to minimize the access latency for Online Social Network (OSN) cloud environment web content. In chapter 3, we discuss in details the proposed data management approaches for OSN cloud.

2.1.2 Networks Resources Dynamic Management

Software Defined Network (SDN) is another virtualization layer for the network infrastructure that grew up rapidly during the last decade and shows many benefits for cloud network resources management. SDN virtualizes the network resources to abstract the physical hardware components of the network infrastructure such as the switches and routers [12] and [33]. Using SDN IT admins can dynamically manage the datacenter networking resources to fulfill the workload and applications requirements. They can also implement the network policies consistently and at large scale using a centralized control console. In addition, workloads can be migrated across different virtual or physical networks with almost no impact on the application availability.
Network infrastructure can be deployed including network controllers, software load balancers, and gateways using the SDN controller. IT admins can create virtual network policies centrally and assign that to the corresponding applications such that when the application is migrated, the network configuration adjusts itself automatically. This means that the network manual reconfiguration is not needed, which saves the operational complexity and operational cost. Each Virtual Network (VN) control logic can run on a virtual controller as a set of rules instead of configuring the physical switch [110]. SDN consists of 3 layers as shown in Fig. 2.1. Layer one is the infrastructure layer that has the physical network infrastructure equipments like switches and routers. The second layer is the SDN controller layer where the network configuration, switching, routing, L2 VPN, L3 VPN, firewall security rules, DNS, DHCP, and clustering are applied [12], [33]. The controller services expose their interfaces to the upper layer; which is the application layer.
The application layer is the top layer that includes network automation, network management, network monitoring, troubleshooting, network policies and security. The application layer services differentiate between a SDN platform to another depending on the SDN platform vendors interest to add application layer features for their customers. The application layer services facilitate offering an application-driven network (ADNO) architecture that defines logic structure of the network based on the application requirement [98]. This is considered as a new paradigm in networking management. In ADNO, the physical network resources are sliced into many logical network such that each application is assigned to a logical network that is optimized for the application requirements. Live migration of the logic network segment and the application assigned to it is a feature in SDN. In this case the VM where the application run, the logical network assigned to it and the related management modules can be migrated from a physical resource to another [96]. In [96], the authors propose a live migration algorithm for applications VMs and their assigned networks with down time avoidance for SDN environments.

2.1.3 Compute Resources Dynamic Management

Dynamic management of the compute resources is called live migration of the virtual machines. Cloud computing and virtual datacenters admins can move the VMs from a physical host to another without applications interruptions. Live Migration is one of the most important features and a powerful tool in machine virtualization. It allows an entire running and active virtual machine

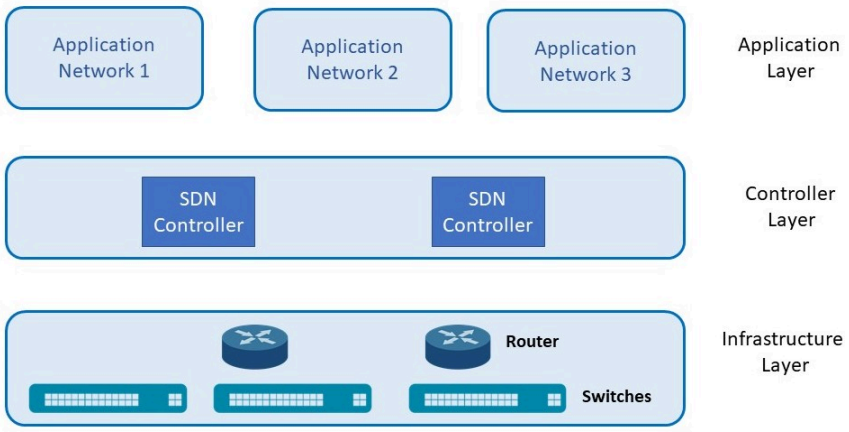

Figure 2.1: SDN Architecture

(VM) to be transferred from one physical host to another with a very little interruption which allows seamless movement of online servers in LAN or in MAN scale without asking clients to disconnect and reconnect [4]. Live migration is supported by VMware (vMotion), Xen (XenMotion), Microsoft Hyper-V and Redhat KVM [3]. Servers load balancing, online maintenance, fault tolerance and power saving are all dependent on VMs live migration feature. So live migration is an essential feature in virtual datacenter and cloud computing environment to have dynamic resource management. On the other hand, it is important also to study the drawback of live migration overhead on the datacenter performance.
The rest of this thesis focuses on the live migration process. We start with detailed background on live migration types, phases, configurations, networking and cost. Then we propose cost modeling, prediction and timing optimization for VMs live migration.

2.2 Live Migration Types

From migration processes point of view, live migration has three different types; as shown in Fig. 2.2 The three types are Pre-copy, Post-copy and Hybrid-copy.

- In Pre-copy, live migration starts with transferring the whole content of the source host memory to the target host, however due to the fact that the application is still writing data on the source host memory, this new data is called dirty pages that should be transferred also to the target host in other iterations. This iterative copy runs until a stopping condition is met. There are different stopping conditions, as we will discuss later. When the stopping condition is met, the last copy of the memory and the CPU state are transferred to the target host and the this is the time when the VM is handed-over to the target host. During this handover, there is a down-time that should be very short to avoid the running application interruption. This means that in Pre-copy, the handover of the

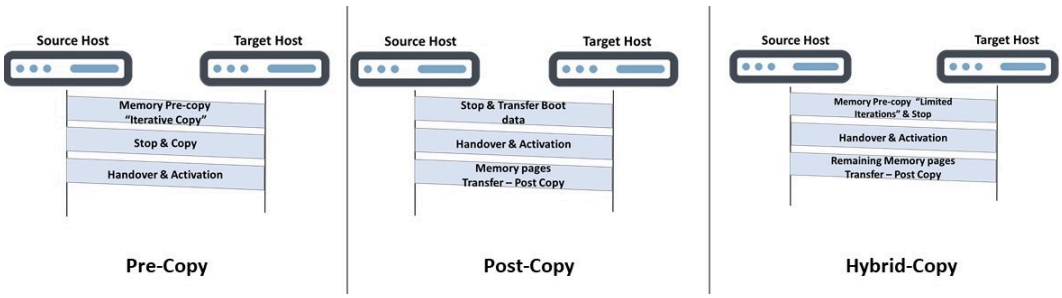

Figure 2.2: Live Migration Types

VM only occurs when there is little amount of data to be transferred to minimize the down-time and to have robust migration. That is the the reason for considering Pre-copy live migration as the most reliable live migration type. VMware, KVM, Hyper-V and Xen are all using Pre-copy live migration. The dis-advantage of Pre-copy live migration is the migration time which is not predictable because the number of copy iterations can not be predicted. It depends basically on the dirty pages rate and the network transmission rate. In some cases the migration might take too long time or even fails due to high dirty pages rate with lower network transmission rate. But when this case happens, the VM continues running on the source host without disruption, which makes Pre-copy as the most reliable technique.
The stopping condition in Pre-copy differs from a hypervisor to another. The number of pre-copy iterations, the residual amount of data to be migrated in the source host memory, or the ratio between the transferred data and the memory content to be migrated are the main stopping conditions for pre-copy. The stopping conditions in the Xen platform are [50]:

1. Less than 50 pages are dirtied during the last pre-copy iteration.
2. To have 29 pre-copy iterations been carried out.
3. More than three times the total amount of RAM allocated to the VM has been copied to the destination.

While the stopping conditions for VMware are [87]:

1. Less than 16 megabytes of modified pages are left.
2. There is a reduction in changed pages of less than one megabyte.

- In Post-copy migration, the source host transfers only the data required for the VM boot to the target host and then stops the VM at the source host to hand it over. After the VM activation at the target host, the source host starts sending the memory data in one iteration to the target host. This means that the memory copy is done in a one shot after the VM handover, and so post-copy migration time is predictable. However, this means that if the memory content transfer fails for any reason, the VM will be destroyed and data loss might occur [79]. So, it is not a reliable migration technique as Pre-copy. Hence, post-copy is not used

by any commercial hypervisor. In [100], a detailed comparison between pre-copy and post-cop migration is presented. The comparison in [100] shows that pre-copy technique is more safe and reliable and so it is the commonly used technique by the hypervisors Xen, VMware, Microsoft Hyper-V, Oracle VM server, KVM, Virtuozzo, OVirt and Google Compute Engine. Post-copy migration is used also in KVM, Virtuozzo, OVirt and Google Compute Engine.

- Hybrid-copy technique has several algorithms that try to mix steps of pre-copy and post-copy to achieve higher robust migration with migration time prediction. One of these algorithms [88] and [115] firstly migrates the memory content of the VM to the target host and during this migration, new dirty pages are written to the source host memory, and several pre-copy iterations are run but with limited number to keep the migration time predictable. Then the VM state is transferred and the handover occurs to activate the VM at the target host. The residual memory pages are transferred to the target host in a post-copy manner. Hybrid-copy depends on having low amount of residual memory pages in the post-copy phase to enhance the migration robustness compared to post-copy. However it does not show the same reliability and robustness level of pre-copy. So, in case of transfer failure in the post-copy phase, data loss might occur.

2.3 Pre-Copy Live Migration Phases

Live migration is a key technology for data-centers and cloud computing environments, however there is a cost for the live migration process. The cost includes down time, migration time, network overhead, power consumption overhead, memory overhead and CPU utilization overhead.
Pre-copy live migration has mainly six phases; as shown in Fig. 2.3 [63]. These phases are:

1. Initialization: Initiating the migration by selecting the VM to be migrated and selecting the target machine.

2. Reservation: The source machine sends a request to the target machine for resources reservation and the target machine answers with an acknowledgment after reserving the required resources for the migration.

3. Iterative pre-copy: The entire RAM is sent in the first iteration, then pages modified during the previous iteration are transferred to the destination. Using shadow page table for memory dirty pages mapping.

4. Stop-and-Copy: When the stop conditions are met, the VM is halted on the source for a final transfer round. At the same round of stop- and-copy while transferring the final dirty pages, the migrated VM's CPU state is transferred to the destination.

5. Commitment: the destination host checks if it has received successfully a consistent copy of the migrated VM. Then the target machine sends a message telling the source that it has successfully synchronized the migrated VM states.

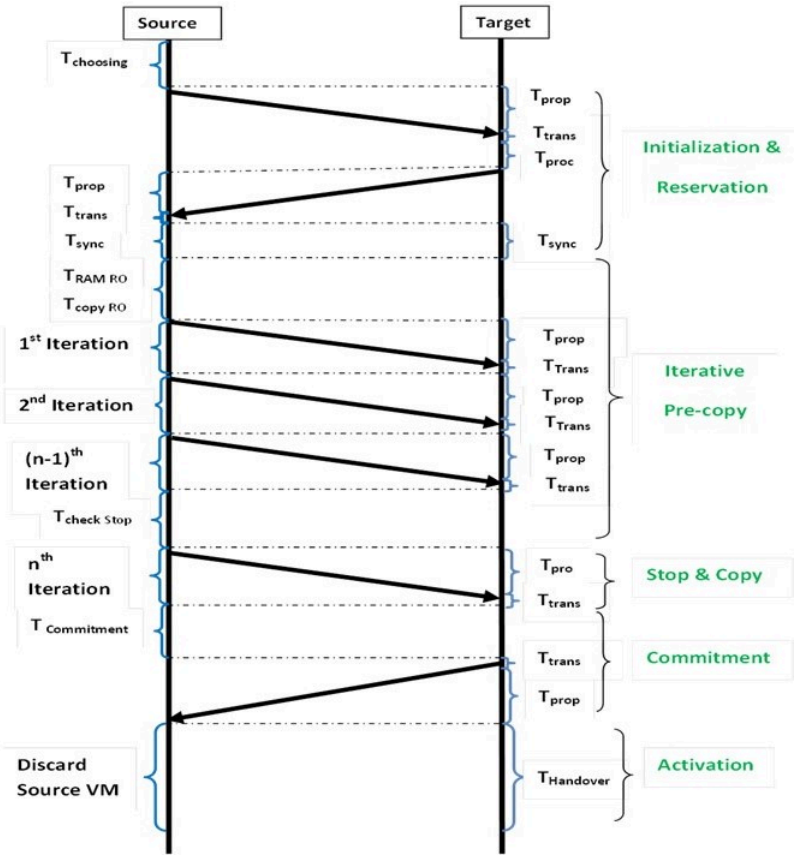

Figure 2.3: Pre-copy Live Migration Phases

6. Activation: After target host informs source host that it has synchronized their states, source VM can be discarded. The migrated VM running on target host is the primary host now and takes over the services offered by source VM.

There are many research articles that analyze and modeled live migration cost [69], [73], [89], [116], [129]. In addition, there are also many other papers that predict the migration cost for pre-copy migration in order to resolve the disadvantage of not having a predictable migration cost [124] - [51]. However, it is a challenge for readers of these papers to track the differences and map the suitable use-case for each model of these papers.
In this work, we review, compare, classify and summarize different up to date research articles in the area of live migration overhead modelling and prediction. Also, we cover live migration overhead for servers with persistent memory inside; as an emerging memory technology recently used in modern datacenters. Finally, we discuss the outstanding research topics in the area of live migration cost analysis and prediction. In more details, this project contribution can be summarized in the following points:

1. Analyze the other related survey papers that cover pre-copy live migration

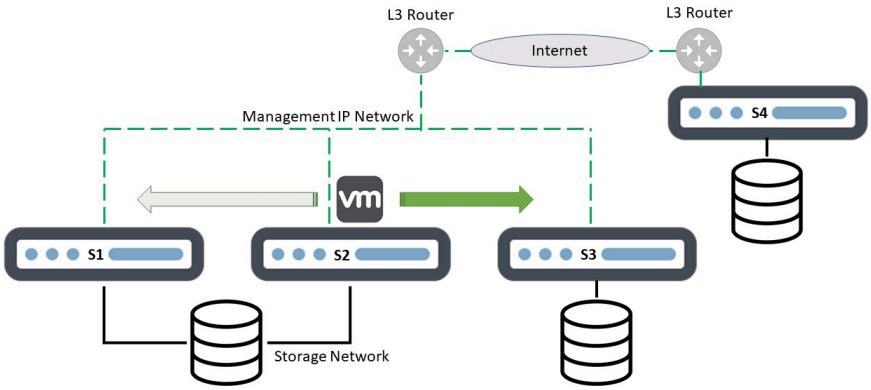

Figure 2.4: Live Migration Configurations

topic and show their differentiation compared to this project.

2. Categorize and compare related live migration cost papers based on the papers focus (analysis or prediction), the cost parameters that are discussed and the hypervisors used.

3. Discuss live migration for modern datacenter servers with persistent memory technology.

4. Share the outstanding research problems in the area of live migration cost.

In this work, we focus on the pre-copy live migration cost as the mainly used technique by all virtualization hypervisors due to higher robustness; as mentioned in the previous section. In the next section, we discuss in more details pre-copy live migration phases, the migration cost parameters considering their root causes, and the different methods for migration cost analysis and prediction.
Pre-copy live migration has mainly six phases; as shown in Fig. 2.3 [63].

2.4 Live Migration Configurations

Live migration can be achieved with different configuration scenarios; as shown in Fig. 2.4.

- The first scenario is to migrate the VM compute resources to another physical host without the VM storage virtual disk migration. This can be applied only under the condition of having a shared storage environment between the source and the target servers. In this case, mainly the memory content is migrated. For example the VM in Fig. 2.4 can be migrated with this scenario only between S1 and S2 hosts through the management IP network of the cluster.

- The second scenario is to migrate the compute and storage resources of the VM from a source to a target host through the management IP network of the cluster. In this case the memory and the virtual disk storage content should be migrated. So, the VM in Fig. 2.4 can be migrated from S1 or S2 to S3 host or vice versa.

- The third scenario is to migrate the compute and storage resources of the VM from the source to the target host through the WAN or the Internet network. This scenario is useful for datacenter migrations or disaster recovery solutions between datacenters in different locations. So, in this scenario, the VM S1, S2 or S3 can be migrated to S4; as shown in Fig. 2.4.

- The fourth scenario of live migration is to have multiple VMs migration simultaneously. The number of simultaneous VMs to be migrated has a maximum limit. This limit is defined by the source host of the migration that is responsible for resources allocation and migration success verification process. Referring to Fig. 2.4, in multiple simultaneous VMs migration, there can be many VMs in different hosts that can be migrated from any of the hosts to another.

2.5 Live Migration Networking

Virtual networking is an essential requirement for virtualized datacenters and cloud computing platforms [82]. Each VM has a virtual network adapter and at least one virtual port. The VMs are inter-connected to virtual switches (vSwitches) that use physical Ethernet switches in the back-end. In this section, we discuss in more details the concept of network virtualization and how live migration is implemented in the four hypervisors; VMware vSphere, Microsoft Hyper-V, Xen and KVM.
In virtual networking, each VM has virtual Network Interface Cards (vNICs). Each vNIC has one or more virtual ports (vPorts). Each vPort is connected to a vSwitch. This virtual switch can be a local switch inside the physical host only to connect the VMs within this host, or can be a cluster virtual switch to connect between the VMs in a cluster. Each vSwitch has at least one uplink port which is mapped to a physical switch port. Each group of ports in the vSwitch can create a separate vLAN or port group that can be labeled. For one or more physical hosts connection, a cluster vSwitch is used as a centralized vSwitch that connects all the VMs of the cluster physical nodes. This vSwitch concept applies to all hypervisors [128]. However, the hypervisors are different to each other when it comes to live migration networking set up. We discuss in this section live migration networking configuration details for VMware vSphere, Microsoft Hyper-V, Xen and KVM.

2.5.1 VMware vSphere Live Migration Networking

Live migration feature in VMware is called vMotion. Fig. 2.5 represents a commonly used scenario in enterprise datacenters where a storage array is shared between the cluster servers using FC-SAN network. The example shown in Fig. 2.5 is a best practice for vMotion networking. Fig. 2.5 shows a cluster of two physical machines that are connected to a shared storage using FC-SAN

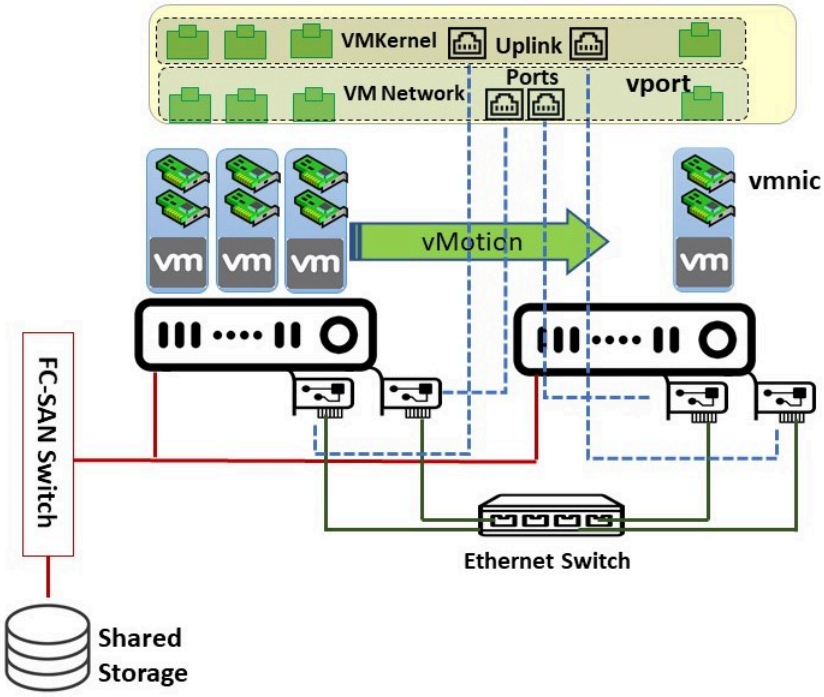

Figure 2.5: Network Topology for VMware vMotion

switch and connected to the IP network using an Ethernet switch. The solid lines represent physical connections and the dotted lines represent the virtual connections for the virtual distributed switch. Live migration uses TCP/IP protocol and so it utilizes the IP network. From best practice point of view, each physical host should have at least 2 physical NICs and each VM should have at least two NICs [45]. The VMs in the cluster are connected to a virtual distributed switch. Using port groups, the IO traffic of the VMs can be isolated. There are two types of port groups in VMware; VMkernel distributed port group and VM network distributed port group. VM network port group is responsible for the production traffic like applications traffic. VMkernel port group is responsible for the special classes of traffic like vMotion, management, iSCSI, NFS, Fault tolerance, replication traffic and VMware vSAN as a Software Defined Storage (SDS) traffic [32]. The physical machines NICs ports should be mapped to the distributed switch as uplink ports. The uplink port is responsible for the in-going and the out-going traffic into and from the distributed switch. Each port group should have at least one uplink port from each physical host. Each uplink port can be shared between many port groups. For vMotion traffic, it is a best practice to create a dedicated VMkernel port group between the VMs in the cluster. This vMotion distributed port group should include at least one uplink port from each physical host [45]. This uplink port assignment is actually not only for vMotion port group, but also for any other VMkernel based port group. From physical port isolation, vMotion traffic is physically isolated on the host port level from the applications traffic. However, depending on the

backend network topology, vMotion and workload traffic might compete on the backend network bandwidth.

2.5.2 Microsoft Hyper-V Live Migration Networking

Virtual layer two switches in Hyperv-V have the same concept like VMware. It is basically a software based switch that connects the VMs vNICs with the physical ports uplinks [46]. Also, live miration in Microsoft Hyper-V has the same concept like VMware vMotion. The best practice for Hyper-V is to configure a separate virtual network or VLAN for live migration in order to isolate the migration network traffic from the applications traffic [8].

2.5.3 Xen Hypervisor Live Migration Networking

In Citrix Xen, vSwitch concept is also used as in vSphere and Hyper-V, so each VM has at least one vNIC and vports that are connected to a distributed vSwitch which connects that VMs across the cluster and includes the hosts physical NICs as the vSwitch uplinks. The difference in Xen compared to vSphere and Hyper-V is having a separate OpenFlow controller. This OpenFlow controller is a centralized server that controls the Xen servers virtual network and is responsible for the vSwitches configuration, traffic management and performance monitoring [36]. Live migration feature in Xen is called XenMotion. XenMotion networking best practice is to create a cross server private network that isolates XenMotion traffic from other other management or workload traffic. This private network provisions dedicated virtual management interface of the VMs for live migration traffic [16].

2.5.4 KVM Live Migration Networking

Libvirt is used for KVM Hypervisor virtual networking [40]. Libvirt uses APIs that talks to Quick EMUlator (QEMU) for network and storage configurations. Each VM has its own QEMU instance. The vSwitch that is created by libvirt can connect the VMs vNICs across the KVM cluster with the physical hosts uplink ports. For KVM live migration networking, Redhat best practice is to create separate the storage network from the migration network. So live migration isolation from other management traffic or workload traffic is not mentioned [28]. This means that live migration network traffic might be in contention with the workload traffic or with other management traffic.

2.6 Summary

In this chapter, we have discussed in detail the dynamic resource management concept for storage, network and compute resources of the virtual datacenters and cloud computing environments. Then, we focused on the live migration feature that is used for compute resources dynamic management. We covered in this chapter, the definition of live migration phases, configuration types, supported scenarios and networking setup. This background is important for the next sections, where we discuss live migration cost, cost modeling and migration cost prediction techniques.

In the next sect3ion, we discuss live migration cost and the related work that discuss the cost modeling and prediction for VMs live migration.

Chapter 3

Online Social Networks Cloud Storage

In this chapter, we discuss storage management in OSN as a use-case for dynamic storage management in cloud computing environments. The motivation toward our OSN optimization is the rapid growth in OSN users' during the last decade, the latency challenge to access the feeds data by their users' friends and the different nature of the OSN architecture where the users can share huge content of data with each other across the globe based on each user's friendship matrix. The friendship matrix of a user is the social network that the user has with the other users in the network, so either to be a friend on not.

3.1 Storage Management in Online Social Networks

Social networks providers have many datacenters that are located across the world to cover their users. Each user should store his/her data in one of the datacenters storage servers where his/her friends can access this data. To have better quality of service, the social networks users should access their friends' feeds with minimum access delay. The problem is how to map the social networks provider users to datacenters allocation such that the average user' delay is minimized in reading/accessing their friends' feeds. As shown in Fig. 3.1, user A data is assigned to Australia datacenter and user B data is assigned to US datacenter. If user A and user B are friends and user-B accesses user A's feeds, the data uploaded from user A to Australia datacenter will be transferred by the Internet to user B's station. If user A has many friends, the data will be transferred multiple times via the Internet from Australia datacenter user A's friends stations.

In [70], we have proposed a novel friendship matrix aware storage allocation that migrates the user's hot data to another node that is close to the majority of the user's active friends. The idea of Friendship Matrix based Storage is to minimize the Internet traffic for OSN and maximize the probability of data access locality within the same friends' datacenters. This objective minimizes the

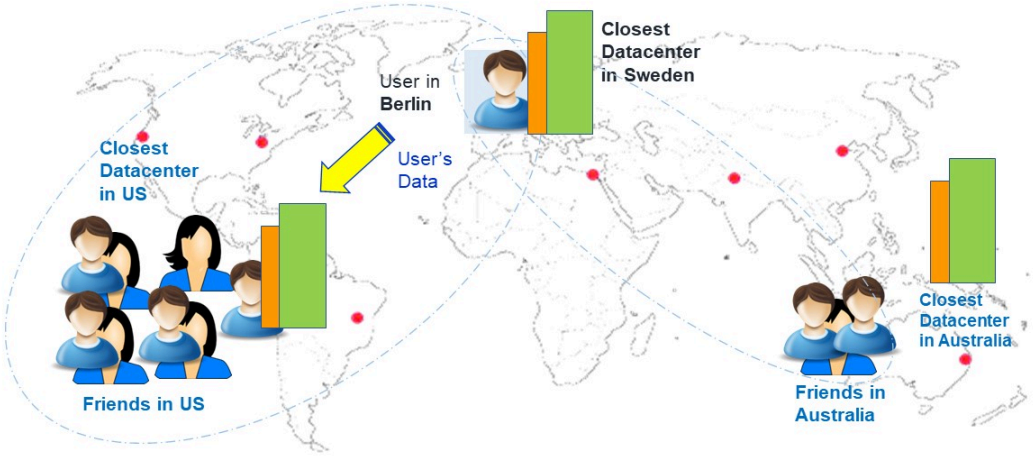

Figure 3.1: Online Social Network Provider Cloud Environment

data-access delay because the Internet is bottleneck of the data routing from sender to receiver station. As shown in Fig. 3.1, initially if Min-Distance storage is used, we assume that user-A is located in Australia and stores his/her data is Australia datacenter. User-A's friends are one friend in US and three friends in Sweden. If user-A's friends stream or access user- A's feeds, the data will be copied one time from Australia to US and three times from Australia to Sweden; which increases data transfer through the Internet and increases the data access delay by the user's friends. In order to minimize the dependency on the Internet, friendship matrix based storage is proposed. If we apply the same example using the proposed algorithm, user-A searches for his friends' locations and finds that most of the friends are located in Sweden, so user-A's feeds will be uploaded and stored in Sweden datacenter.

This algorithm will increase the upload delay for user-A, but will significantly save the download time for his/her friends; which happens many time by many friends. So the average users' data access delay will be minimized. Fig. 3.2 shows the flowchart of the proposed friendship matrix based storage allocation. As shown, the proposed algorithm starts with the user with largest number of friends in the social network, because this is the most significant user in the network performance. The number of friends for this user is calculated to check if they are less than 100 users or not. A number 100 users is an assumed value for the number of friends limit to make significant effect on the network performance that overcomes the data-relocation overhead. If the number of friends is more than 100, the proposed friendship based will be used and the user's data will be allocated based on the friends' locations; otherwise the Min-Distance allocation will be kept. When the proposed social aware data storage algorithm runs, the users will be assigned to the datacenter when most of their friends

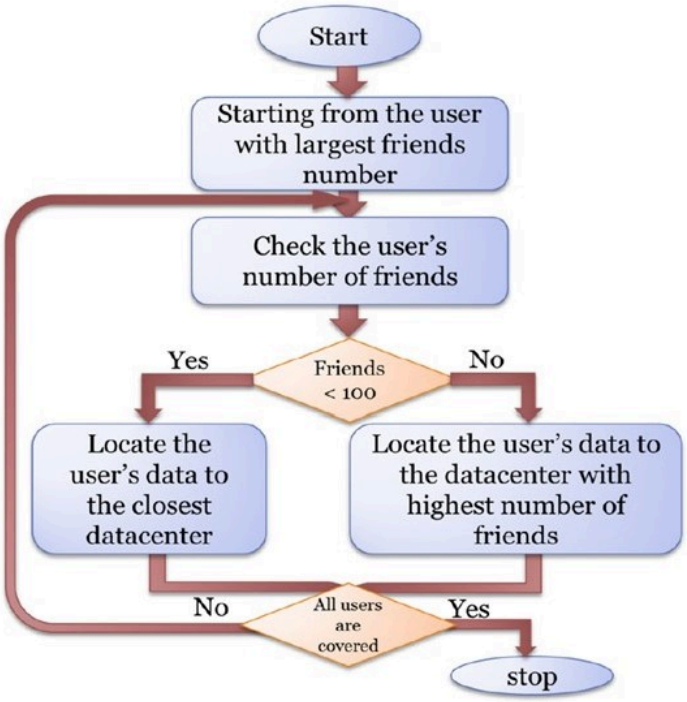

Figure 3.2: Friendship Matrix based Storage Allocation

store their data; which minimizes the access latency for their friends' feeds. The algorithm should be run periodically for all the network users to be updated with the friendship matrix and network topology changes.
It is well known in OSN that many users and their friends are normally located in the same region and most probably they are mapped to the same datacenter. In this case, the proposed algorithm will keep these users and their friends mapped to the same datacenter as this is already the optimal for access delay. Comparing the proposed algorithm with Apache Cassandra and Min-Distance based storage allocation; the friendship based storage shows less average users' data access delay. In terms of the users' load balance across the datacenters, the proposed algorithm shows less users difference between the network datacenters. So it offers better load balance between storage servers. The drawback of the proposed algorithm is its complexity in data storage decision compared to the other two algorithms.

In [72], we propose a novel social aware algorithm for OSN data storage and replication. The proposed algorithm (shown in Fig. 3.3) considers both users and data centers locations, the friendship matrix and the activity between each user and his/her friends with considering the computation simplicity and stability. As shown in Fig. 3.3, the proposed algorithm runs every T period of time to check if there might be a required redistribution for the OSN users' across the datacenter. The values of T should be defined by the OSN provider and should differ from OSN provider to another based on both users' growth

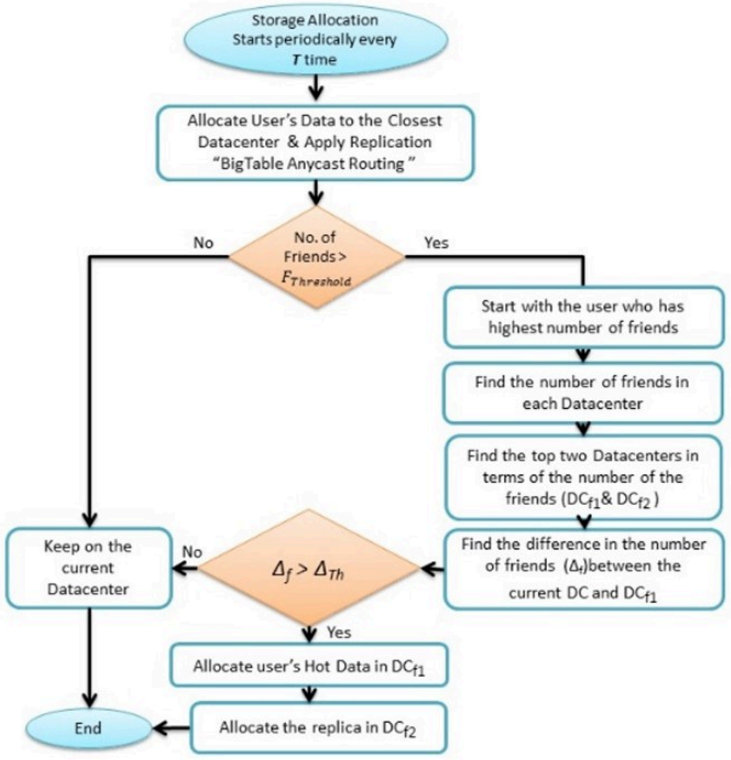

Figure 3.3: Proposed OSN Storage and Replication Approach

and their activity. Therefore, is dealt with T as a variable. The algorithm can also be initiated when OSN provider needs to optimize and reallocate the users' hot data for less access delay and hence better QoS. After the algorithm starts, initially the datacenter for a new user is the closest data center using Anycast routing; as used by Google. Once, the number of the user's friends exceeds a certain limit $F_{Threshold}$ the social aware allocation should be considered. The value of this threshold is also left variable to be decided by the OSN provider1. For each user who has number of friends exceeding $F_{Threshold}$ and starting from the user with highest number of friends, we find the number of friends in each datacenter among the OSN provider Cloud-datacenters. The algorithm then finds the top two datacenters with the maximum number of friends, noted as DC_{f1} DC_{f2}. The algorithm compares the difference between the number of friends in their current datacenter and the number of friends in DC_{f1}, if this difference is greater than $F_{Threshold}$, this means that hot data migration and storage reallocation for this user will have significant enhancement in access delay for this user's friends. It is a significant performance enhancement despite the hot data migration overhead. This check point also makes the algorithm more stable by avoiding data-relocation due to few changes in the friendship matrix. If the previous check is met, the user's hot data and his/her future data allocation should be directed to DC_{f1}.
For high availability, the replica of this data should be allocated as DC_{f2}; which

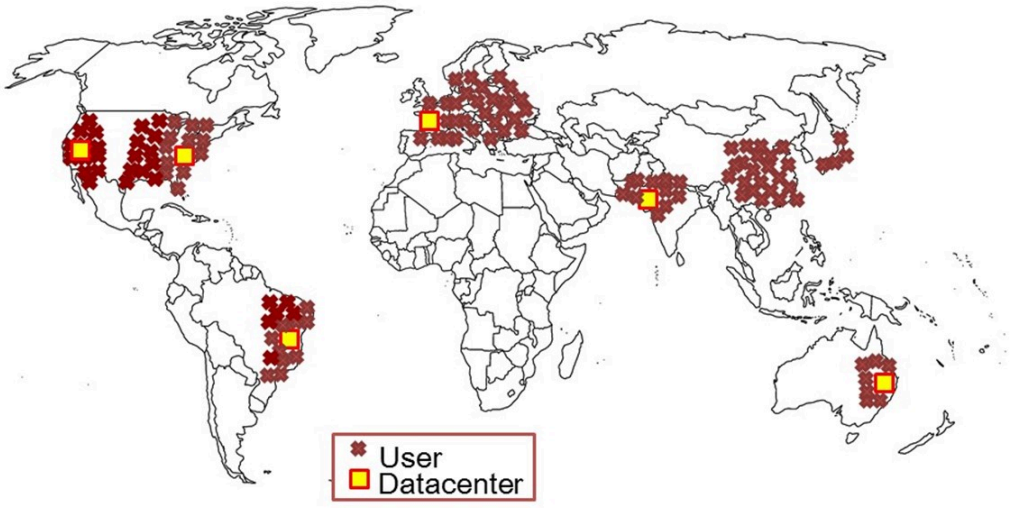

Figure 3.4: Datacenters Locations in the Simulation Network

is the second top ranked datacenter in terms of the number of friends. The idea beyond the proposed algorithm is to have the user's data and its replica assigned to the datacenters that have minimum access delay to his/her friends. This minimizes the overall average users' access delay. It is well known in OSN that many users and their friends are normally located in the same area, city or country and so most probably they are mapped to the same datacenter which is a special case of the current algorithm. In this case, the proposed algorithm will keep these users and their friends mapped to the nearest datacenter as this is already the optimal for access delay.
Thus, the proposed algorithm data storage and replication optimization is considered if the users and their friends are in the same area or if they are distributed around the globe. We may highlight also that in the case of having the user and her/his friends in the same area, the proposed algorithm will exactly operate as the Anycast routing algorithm. As, the closest datacenter of the friends is exactly the closest datacenter to the user.

3.2 Testing Environment

In order to test the proposed algorithms for storage management in OSN [72] and [70], we have used MATLAB tool to build and simulate a social network cloud with several datacenters [14]. We simulate a social network provider datacenters in a cloud environment and worldwide located users; as shown in Fig. 3.4. The area Length is 36750 km and Width is 19500 km. These values are approximately the world map length and width. As shown in the next page, the

Table 3.1: Datacenters Locations

Datacenter	X (x1000km)	Y (x1000km)
DC 1	10.719	6
DC 2	6.891	11.25
DC 3	6.125	15
DC 4	20.672	9.75
DC 5	18.375	12.75
DC 6	31.391	4.5
DC 7	27.563	10.5
DC 8	29.094	15

number of users varies with the following values 100, 500, 1000 and 5000 users that are uniformly distributed across specific areas that are dense with social network users; based on Facebook engineering active users' worldwide map [3]. Datacenters are located as mentioned in 3.1 to be as similar as possible to Google datacenters locations. The transmission rate is assumed to be 10Mbps, and the frame size is 1500B; which is the maximum Ethernet frame size. $F_{Threshold}$ is assumed to be 100. For migration condition, we assumed that = 25% of the number of friends in the current datacenter. The friendship matrix that controls the storage allocation decision for our proposed algorithm considers not only the one-to-one relationship between the OSN users, but also the read activity between them. This is because any user might have many friends in a certain area, but these friends are not active with such user, so they are not interested to read his/her feeds. In this case, the allocation decision should not consider optimizing the delay for these friends, as most probably they will not read the data in the future. This activity relationship can be obtained by the historical correlation of the read ratio between users. Table 3.1 shows an example of the activity relation of some OSN users and their friends. In this table if the value in the table equals zero it means that these two users are not friends. If the value is greater than zero, this means that the two users are friends and the higher activity between them the higher value in the matrix. The maximum activity value in the matrix is 1. Equation 3.8 shows how the activity factor of user u is calculated with a friend . The diagonal of the friendship activity matrix is zero, because the user can't be a friend with him/herself.
For each number of users, the simulation runs 10 times, and the access delay is calculated based on equations 3.1 to 3.8. Then for each algorithm, the average users' delay is calculated over the number of runs. Five thousands users is a low number of users compared to the actual OSN users, but this is the limit that we could reach based on our hardware capabilities. In future work, more number of users can be tested using more powerful machines. Replication factor of two is considered in our testing, so for each primary copy in a datacenter, there is one replica copy at another datacenter.

3.3 Simulation Results

Based on the testing environment that is used in the previous section, we have focus to measure the average users' access delay and the load balance across the

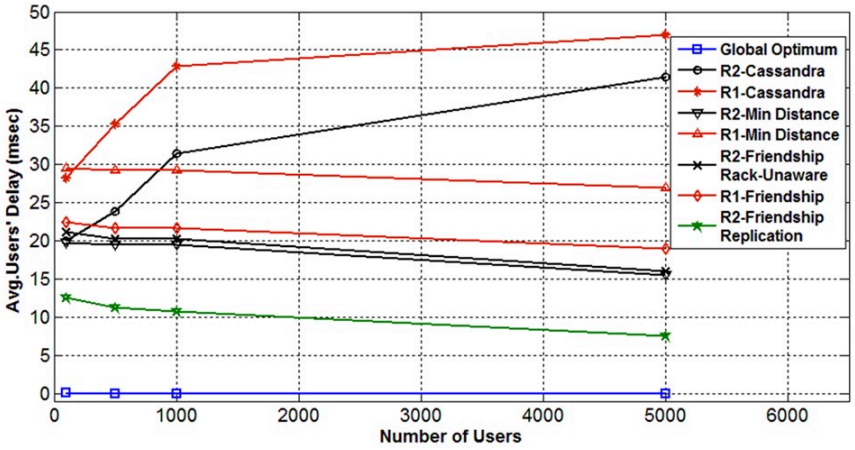

Figure 3.5: Average Users' Delay Comparison

cloud datacenters. The simulation Results show the following:

1. Avg. Users Delay: As discussed, we compare between Apache Cassandra, Min-Distance, Integer Optimization and Friendship based storage allocation algorithm as well as Rack-Unaware and Friendship based replication. Here, we compare these algorithm in terms of the average users' delay; as calculated in 3.6 from 100 to 5000 users.

$$Obj.min.(Avr.D) \tag{3.1}$$

$$D_{u,f_u} = T_{prop} + T_{proc} + T_t + T_s \tag{3.2}$$

$$T_{prop} = \frac{L_{u,i_{f_u}}}{V_{prop}} \tag{3.3}$$

$$T_t = \frac{S}{R} \tag{3.4}$$

$$T_{proc.} = T_i.U_i \tag{3.5}$$

$$Avr.D_u = \sum_{f_u=1}^{F_u} D_{u,f_u} \tag{3.6}$$

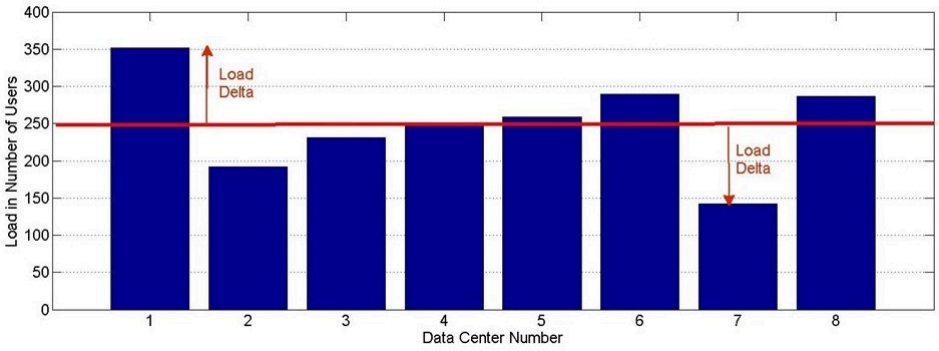

Figure 3.6: Load Balance Calculation

$$Avr.D = \sum_{u=1}^{U} Avr.D_u \tag{3.7}$$

Where:
D_{u,f_u}:Delay between the user (u) and the friend's (f_u) datacenter
T_{prop}: Signal Propagation Delay
$L_{u,i_{f_u}}$:The distance between user (u) location and the datacenter (i) of the friend (f_u)
V_{prop}: Signal propagation speed
T_{proc}: Server transaction processing delay
T_i: Server transaction processing time per user
U_i: Number of users in Datacenter (i)
T_t: Frame Transmission Delay
S: Frame Size (bits)
R: Transmission Rate (bit per second)
T_s: Switching Delay
F_u: Total number of friends of user (u)
$Avr.D_u$: Average delay across all friends of user (u)
$Avr.D$: Average delay for all users
U: Total number of users in the network

$$\Delta_{Li} = |U_i - U/I| \tag{3.8}$$

$$\Delta_{DCAvg} = \frac{\sum_{i=1}^{I} \Delta_{Li}}{I} \tag{3.9}$$

$$\Delta_{Avg} = \frac{\sum_{r=1}^{R} \Delta_{DCAvg}}{R} \tag{3.10}$$

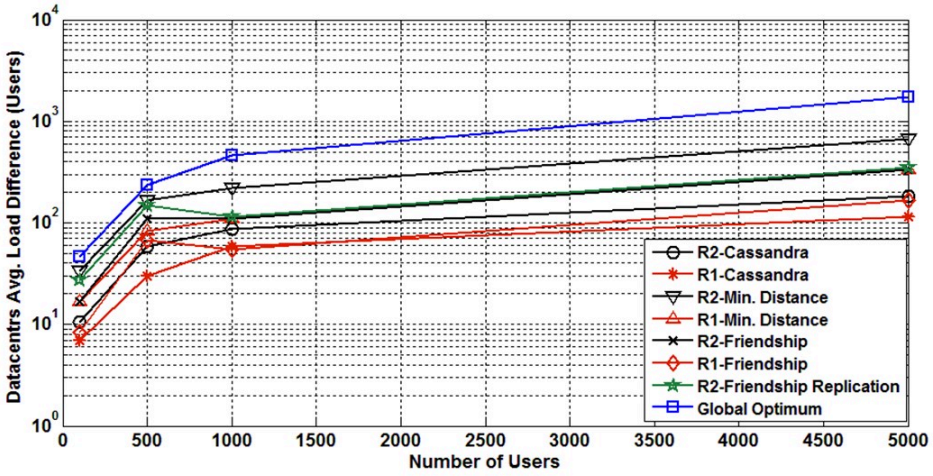

Figure 3.7: Load Balance Comparison (Users)

Where:
Δ_{Li}: Load Difference
U_i: Users in each DC
U: Total No. of users
I: No. of DCs
R: No. of runs

Fig. 3.5 shows the average users' access delay for the simulated OSN. The obtained results can be summarized as following: The integer optimization solution shows always the global optimum delay that is close to 1 msec value with different number of users. It is logic to have integer optimization as the best performance approach, however it is not a practical technique due to the problem complexity and processing delay; especially with a large number of users. The proposed friendship allocation with friendship based replication "R2-Friendship Replication" shows almost half the rack-unaware replication "R2-Friendship". Replication with Rack-unaware for the proposed friendship storage allocation of the primary copy has almost the same results like replication with Rack-Unaware for the Min-Distance Anycast based allocation. Rack-Unaware replication for Cassandra shows much higher delay compared to our proposed friendship based replication. For storage allocation, the proposed friendship based allocation saves almost 33% of the access delay by Min-Distance algorithm and has must less delay compared to Cassandra.

2. Load Balance Comparison: Because OSN has a huge load of read/write request, it is worth to consider the load balance across the OSN provider

cloud datacenters. So, load balance between the datacenters is important to avoid over utilization for a certain datacenters; which may lead to performance issues. In this section, we compare the proposed Friendship based storage with Min-Distance and Cassandra in term of the users' load balance with considering Rack-Unaware and friendship based replication. Firstly, we explain how the load balance is calculated in this simulation and then show the comparison results between the three algorithms.

As shown in Fig. 3.6, load balance is calculated by equations (3.8), (3.9) and (3.10). In equation (3.8), we calculate ; the modulus of the difference between the number of users in the datacenter and the perfect balance of all datacenters . In optimum load balance, should equal zero. In 3.10, the average load difference is calculated across all the simulation runs. Load balance simulation study shows the following result; as shown in Fig. 3.7: The global minimum by integer optimization has the worst load balance; which is logic because this solution assigns all the users to one or two datacenters only to minimize the delay between them.

- Cassandra shows much better load balance compared to Min-Distance algorithm.
- The proposed friendship aware data allocation and replication shows worse load balance than Cassandra, but better than Min-Distance allocation.
- With increasing the number of users, the load balance decreases across the datacenters.

Chapter 4

Survey on Live Migration Cost

Live migration is a key technology for data-centers and cloud computing environments, however there is a cost for the live migration process. The cost includes down time, migration time, network overhead, power consumption overhead, memory overhead and CPU utilization overhead.
There are many research articles that analyze and model live migration cost [69], [89], [116] and [129]. In addition, there are also many other papers that predict the migration cost for pre-copy migration in order to resolve the disadvantage of not having a predictable migration cost [124] - [51]. However, it is a challenge for readers of these papers to track the differences and map the suitable use-case for each model of these papers.
In this work, we review, compare, classify and summarize different up to date research articles in the area of live migration overhead modelling and prediction. Also, we cover in this project live migration overhead for servers with persistent memory inside; as an emerging memory technology recently used in modern datacenters. Finally, we discuss in this project the outstanding research topics in the area of live migration cost analysis and prediction. In more detail, this project contribution can be summarized in the following points:

1. Analyze the other related survey papers that cover pre-copy live migration topic and show their differentiation compared to this project.

2. Categorize and compare related live migration cost papers based on the papers focus (analysis or prediction), the cost parameters that are discussed and the hypervisors used.

3. Discuss live migration for modern datacenter servers with persistent memory technology.

4. Share the outstanding research problems in the area of live migration cost.

In this project, we focus on the pre-copy live migration cost as the mainly used technique by all virtualization hypervisors due to higher robustness; as mentioned in the previous section. In this section, we discuss in more details pre-copy live migration phases, the migration cost parameters considering their

root causes, and the different methods for migration cost analysis and prediction.
Optimization of live migration is proposed by many research articles with an objective to minimize one or many of the migration cost parameters. This survey paper [112] summarizes and classifies the different approaches in live migration optimization algorithms. As presented in [112], the optimization techniques can be based on compression, de-duplication, check-pointing and other optimization techniques. There is a major difference between the survey proposed in [112] and this survey paper. this project focuses on classification and comparison between the papers discuss live migration cost modelling and prediction techniques for the built-in live migration algorithms in different hypervisors. However in [112], the focus is to classify and summarize the papers that discuss the live migration optimization algorithms to minimize the migration cost.

4.1 Live Migration Cost Modeling

Pre-copy live migration cost is basically a result of the six phases of live migration that are mentioned in the previous section. In this section, we discuss in details the definition of each cost parameter, the root cause of it and its verified modeling for different hypervisors. The cost parameters are:

1. *Migration Time*: Migration time is the period between the VM migration request initialization and having the VM activated at the destination server. This time can take from seconds to minutes depending on the VM memory content size, the network transmission rate during the migration and the dirty pages rate.

2. *Down Time*: This is the time consumed in the stop and copy phase, when the VM stopping condition applies and the last iteration of the migration copy should start and then the VM networking being attached to the target server and until being activated. Down time should typically be in the range of milli-seconds and so the applications and the users do not feel interruption, however in some cases it takes several seconds [77].

3. *Network Throughput Overhead*: Network average rate is the average data rate at which data was transmitted from the physical host NIC card during the migration time interval. This represents the consumed bandwidth of the network in Bps for live migration process. Live migration process is managed by the cluster manager server which uses the Transmission Control Protocol/Internet Protocol (TCP/IP) in the networks layers 3 and 4 for the live migration management and the iterative copies of memory pages.

4. *Power Consumption Overhead*: Live migration process consumes CPU cycles from the source and the target servers [114]. This overhead parameter should not be ignored especially when live migration is used for data-centers power saving algorithms. Live migration transmission rate is the dominant parameter that controls the power consumption during the migration process [123].

5. *CPU Overhead*: VMs live migration consumes also from the source and target servers CPU resources due to handling the iterative copy phase; as a CPU intensive phase of the migration[133]. Meanwhile, the more available CPU resources, the less migration time in case of having available network bandwidth.

As shown; live migration time, down time, network, power and CPU are the main five cost parameters. In this thesis, we focus mainly on three of the five cost parameters; namely the migration cost, network overhead and power consumption. This is due to the challenges that we have faced to measure the CPU and down time practically. The challenges in CPU measurements are firstly the VMs vCPUs which are assigned to the physical cores and this assignment changes very frequently and so can not be monitored. Also, during the benchmarks, the CPU cores are instantaneously changing during the migration and so could not be monitored. For the down time measurement, the challenge is down time in not one of the performance parameters in VMware vCenter Server performance metrics and the this time is in the milliseconds range. So, it was hard to measure the down time practically.
Live migration cost is covered by different researchers, we list many of them in Table 4.1 and classify the articles based on research focus if it is cost prediction or just analysis, the validated hypervisors and the cost parameters that are considered.

We proposed empirical modeling techniques in [69] for VMs live migration in VMware environments to characterize live migration time, network rate and power consumption overhead. The proposed modeling is based on applying the regression techniques on the obtained test results to present a linear or non-linear regression based models for these migration cost parameters. In Reference [89], an analysis of live migration time and downtime is provided and then a comparison between Xen, KVM, Hyper-V and VMware vSphere hypervisors is presented in terms of storage migration and live migration time and downtime. A comparison between Xen and VMware live migration time and downtime is also presented in [116] with more investigation on the parameters that control the live migration time and downtime duration. The authors [56] show the impact of a VM live migration on the running applications performance from client side. The performance degradation of the application from client side was measured in operations per second. The impact of live migration on Internet Web 2.0 applications performance is discussed in [129]. This is important for environments with SLA requirements. For this purpose, a test-bed is built in [129] where the running Web 2.0 workload is Olio application, combined with Faban load generator that access the Apache 2.2.8 Web server with MySQL database. In [62] the authors propose a scheduling weighted based approach for Multi-VMs live migration requests in VMware. The objective of the proposed technique is to minimize the total migration time for Multi-VMs. The weight assigned to each request is based on the memory usage and the network bandwidth and the article shows the impact of scheduling the migration requests using this weight on the total migration time of the VMs. Article [104] studies the impact of virtualization technology and live migration on multi-tier workloads as well as the migration performance. Experimental tests show that virtualization technology does not have significant overhead on Multi-Tier applications, however

Table 4.1: Summary of Related Work

Paper	Research Scope	Mig. Time	Down Time	CPU	Network	Power	Testing Env.
[69]	Regression Modeling	X	-	-	X	X	VMware
[89]	Perf. Comparison	X	X	-	-	-	All
[116]	Analysis and Comparison	X	X	-	-	-	Xen VMware
[129]	Perf. Evaluation	X	X	-	-	-	Xen
[93]	Analysis on Apps Perf.	X	X	-	-	-	Xen
[62]	Multi-VMs Scheduling	X	-	-	-	-	VMware
[104]	Analytical & Regression based Modeling	X	X	-	X	X	Xen
[97]	Analysis and Model Checker	X	-	-	-	-	Xen
[59]	Analytical Modeling	X	X	-	-	-	All
[65]	Cost Analysis	X	-	-	-	X	KVM
[124]	Cost Prediction	X	X	-	-	X	Xen
[125]	Cost Prediction	X	-	-	-	X	KVM
[51]	Cost Prediction	-	-	X	X	X	KVM
[137]	Prediction	X	-	-	-	-	VMware but not vMotion
[55]	Prediction	-	-	X	X	X	Oracle Virtual Box
[94]	Prediction	X	X	X	X	-	All
[49]	Prediction	X	X	-	-	-	Xen
[117]	Prediction	X	X	-	-	-	VMware/ Xen/ KVM
[91]	Prediction	X	-	X	-	-	Xen
[78]	Markov Model Prediction	-	-	X	-	-	CloudSim
[113]	Prediction	X	-	-	-	-	Xen
[75]	ML based Prediction	X	-	-	X	X	VMware
[60]	Analytical Modeling	X	X	-	X	-	All
[90]	Cost modeling	-	-	-	-	X	Xen

live migration causes performance decrease due to the migration cost and down time. This performance degradation is more significant with memory intensive multi-tier workloads.
The authors in [53] use Probabilistic Model Checking (PMC) and Markov Decision Process (MDP) to study the impact of VM size, page size, dirty pages rate, network rate and pre-copy iterations threshold on the live migration time and down time. The proposed approach uses numerical analysis and the results should be valid for any pre-copy based live migration. In [97], the authors build a performance model for live migration using several migration tests in a Xen hypervisor based test bed and then use Probabilistic Symbolic Model Checker (PRISM) for modelling verification. The proposed approach is used to model live migration time for single and multiple concurrent VMs migration. In [59], analytical modeling is also used to formalize live migration time and down time for single and multiple VMs. Then a Markov model is build for inter-DC network to study the impact of network bandwidth, number of migration requests rate and the number of interconnected DCs on the blocking probability for migration requests.
In [65], the author studies the relationship between live migration cost parameters; namely the migration time, the network bandwidth, the power consumption and their correlation with the size of the VM memory. Testing results show that the migration time exponentially decreases as the network rate increases. The average power usage of the source as well as the destination server linearly increases as the network rate increases. The migration time and the energy consumption linearly increase as the size (memory content) of the virtual machine increases. The models presented in this project are experimental models that are obtained using KVM Hypervisor based test-bed.

4.2 Live Migration Cost Prediction

The other category of papers focus on live migration cost prediction. Classification of Live migration cost is provided in [124] with an explanation of the parameters that control migration time, downtime and energy consumption. Also, Mathematical models are proposed to estimate live migration time, downtime and energy consumption. Machine learning is used in [55] for VM placement elements predictive modeling like (CPU, memory, network and energy).The authors [49] analyze the parameters that control the migration time and the downtime and show the impact of the workload on the migration performance. Markov chains are used in [109] for hosts utilization prediction after live migration. The proposed Markov based prediction model is used for power saving algorithm that can achieve lower SLA violations, lower VM migrations as well as less power consumption [109]. Time series is used in [91] for time varying resources load prediction. The proposed model is used for power saving by minimizing the number of active physical machines with less live migration times and with satisfying the SLA requirements. The proposed technique is tested in a Xen cluster. A mathematical based prediction framework is also proposed in [51]. In [78], a Linear Regression based CPU Utilization Prediction (LiRCUP) method is proposed to determine the future CPU utilization. The objective is to minimize the power consumption and SLA violation level during

VMs live migration. This article compares four benchmark algorithms using CloudSim simulation tool. The first algorithm migrates a VM when the current CPU utilization exceeds a certain threshold. The second and third algorithms adjust the utilization threshold dynamically based on the median absolute deviation (MAD) and the inter Quartile range (IQR). The fourth method utilizes a local regression (LR) technique to predict the CPU utilization.
Authors in [113] evaluate the pre-copy migration in Xen hypervisor. The study objective is to minimize live migration duration and down time in Xen hypervisor by optimizing the total number of memory pages that should be transferred. The proposed approach achieves this target by combining two techniques. Firstly by avoiding repeated dirty pages based on using LRU (Least Recent Used) Stack Distance or using Probability Prediction that can predict the repeated pages. Secondly by using memory pages compression technique.
In [108] the authors propose a host CPU load detection algorithm called Median Absolute Deviation Markov Chain Host Detection algorithm (MADMCHD). The objective of the proposed algorithm is to minimize the SLA violation and to reduce the number of VMs migrations. This objective is achieved by using the past readings for hosts CPU utilization as input for Markov Chains that can therefore predict the future CPU states. Based on the servers CPU prediction, the proposed algorithm decides when to migrate VMs to consider SLAs and with minimizing the VMs migrations numbers. The authors of [68] propose an idea to enhance the compression for VMs memory pages before live migration. The idea is to characterize the memory pages and to identify the pseudo-stable pages. So these pages are predicted to change slowly during the iterations of live migration; which enhances the compression ratio for the pages before being migrated. The proposed algorithm is tested in a KVM lab that shows enhancement in the compression from 10% to 16%.
We have proposed in [75] our machine learning based approach to predict live migration cost in VMware environments. The proposed prediction approach uses regression techniques to predict the migration transmission rate, migration time and peak power increases given the VM active memory size.

4.3 Live Migration Cost Classification

In this section, we classify live migration cost related articles listed in Table 4.1 based on the the live migration cost. The objectives of this classification are to show:

- A simple presentation the different models for a cost parameter; which is useful for readers who search about specific live migration cost attribute modelling and the different testing environments used for modeling validation.

- How the same live migration cost parameter is modelled differently or similarly by different researchers with different tests and distinguished hypervisors. This should help the reader to identify for the same cost parameter how much modelling might change by different test-beds, or if it is a generic model that is independent on the testing environment.

- A comparison for prediction techniques of the same cost parameter in terms of prediction dependencies, accuracy and approach complexity.
- How the proposed classification list should help identifying the research area that are still open or need more research contributions for specific cost parameters modeling or prediction.

In the next subsections, we classify Table 4.1 papers in modeling and prediction summary for the different cost parameters that are listed in Table 4.1.

4.3.1 Migration Time Modeling and Prediction

As shown in Table 4.1, live migration time is studied by many articles from modeling and prediction points of view. In Table 4.2 more details about live migration time modeling and prediction are discussed.
Migration time is a critical cost parameter because this is the time consumed during the whole migration process. So in Table 4.2, there are many models with common parameters that control having long or migration duration.

Table 4.2: Migration Time Modeling and Prediction

Paper	Equation	Hyper.	Methodology
[69] & [75]	$\mathrm{T}_{mig} = a.(\frac{V_{mem}}{R_s}) + b$ T_{mig}: is the migration time duration in seconds. a and b are the equation constants R_s: is the source host network throughput V_{mem}: is the source host active memory size in kB at the time when the live migration should start.	VMware	Regression based approach is proposed to obtain the constants and predict the migration time
[104]	$\mathrm{T}_{mig} = \frac{V_{mem}}{R}.\frac{1-\lambda^{n+1}}{1-\lambda}$ T_{mig}: Total migration time. V_{mem}: Current size of VM memory during migration. R: Memory transmission rate during migration. D: Memory dirtying rate during migration. $\lambda = D/R$	Xen	Provides an analytical model for live migration time based on the analysis of pre-copy migration phases and iterative copy iterations
[97]	$y = \alpha + \beta * (VMsize(GB))*\mathrm{x}$ y: Total migration time (sec). α & β: Constants x: Concurrent migrating VMs	Xen	Uses empirical modeling and then probabilistic Symbolic Model Checker (PRISM) for model verification

[59]	$T^{(z)}_{mig,j} = \frac{V_z}{R_j}.\frac{1-\gamma_j^{n_j^{(z)}+1}}{1-\gamma_j}$ $n_j^{(z)} = min\{log_{\gamma_j}\frac{V_{th}}{V_z}, n_{max}\}$ $T_{mig,j}$: Total migration time of the j-th VM. V_z: Total memory content to be migrated for request z. R_j: transfer rate for the j-th VM in the requested set γ_j: ratio between the dirtying rate and the network bit rate	All	single and multiple VMs migration time analytical modeling. Then a Markov model is built for inter-DC network to study the impact of network bandwidth, number of migration requests rate and the number of interconnected DCs on the blocking probability for migration request.
[65]	$t = as + b$ t: Migration latency s: size of the VM memory	KVM	Experimental modeling using KVM platform
[124]	$(V_{mem}/b) \leq t_{mig} \leq ((m_{th}+1)V_{mem}/b)$ V_{mem}: Total Memory Size b: Network transfer rate m_{th}: Number of pre-copy iteration	Xen	Analytical modeling for migration cost based on related work survey
[125]	$t = A+(B*s)/b$ t: Migration time. s: VM memory size b: Network rate. A and B are constants	Xen	Experimental modeling to study VM size and network bandwidth on the migration cost
[49]	$Overheads + (VMSize/LinkSpeed) \leq TotalMigrationTime \leq Overheads+ (5*(VMSize-1)*$ page/LinkSpeed) $Overhead$: Pre- and post-migration overheads. $VMSize$: VM active memory size	Xen	studying link speed and page dirty rate impact on migration time
[117]	Mig. Time= $\min(t_{c1}, t_{c2}) + f(t_2)/r_u - t_0$ t_{cx}: The time when stop condition x is satisfied $f(t)$: Estimated number of memory pages that remain to be copied. r_u: Memory dirty pages migration rate. t_0: Start time of the live migration procedure	Xen, KVM and VMware	Migration cost prediction modelling given the live migration characteristic parameters

[91]	y= 0.0904 x+ 2.455 y: Migration time. x: VM memory size	Xen	Proposes a Prediction based Dynamic Resource Scheduling (PDRS) technique to minimize the number of active physical machines with satisfying the SLA

From Table 4.2 we get that the commonly used modeling and prediction formulas for migration time is directly proportional with the VM active memory size and inversely proportional with the network transmission rate, and this relationship is valid for all hypervisors. The next cost parameter to discuss is the migration down time; when the VM is handed over to the target host and not actually responding to the application requests.

4.3.2 Down Time Modeling and Prediction

Live migration down time is studied also by many articles as shown in Table 4.1. In Table 4.44.3, we show more details about live migration down time modeling and prediction.
Different analysis and prediction models of live migration down time as presented in Table 4.3. As shown, there is no common formula that is shared by the related papers. However, the proposed formulas show that the down time is directly proportional with the dirty pages rate and the page size; while the down time is inversely proportional with the transmission rate. In the next section, we discuss live migration CPU cost modeling and prediction techniques.

4.3.3 CPU Modeling and Prediction

In this subsection, we present live migration CPU overhead modeling and prediction formulation. Table 4.4 indicates that the contribution in live migration CPU cost is obviously lower than other cost parameters. This might be due to the modeling and prediction complexity for CPU overhead during the live migration process. So, we consider modeling and prediction of CPU overhead as one of the open research areas in pre-copy live migration cost. From Table 4.4, we see that the CPU cost of live migration has different formulations. In [51], the CPU overhead of the physical server is directly proportional with the number of vCPU and the estimated CPU utilization of the VM. In [91] and [78], the future physical host CPU overhead is obtained from the last CPU utilization. In the next subsection, we discuss the network cost of live migration.

4.3.4 Network Modeling and Prediction

In Fig. 2.5 , we explained the network setup of the VMware cluster as an example of virtualized platforms to show how live migration network is configured. In this subsection, we present the network overhead in kBps as a result of live migration of VMs.

Table 4.3: Down Time Modeling and Prediction

Paper	Equation	Hyper.	Methodology
[104]	$T_{down} = T_n + T_{resume}$ n=$\log_\lambda \frac{V_{thd}}{V_{mem}}$ T_n: Remaining dirty pages transferred during the stop-and-copy phase. T_{resume}: time spent on VM resuming at the destination host	Xen	Analytical model for live migration down time based on the analysis of pre-copy migration phases
[59]	$T^{(z)}_{s-down} = \frac{V_z}{b}\gamma^{n^z_s} + (M-1).\frac{V_z}{b}.\frac{1-\gamma^{n^{(z)}_s+1}}{1-\gamma}$ $+T_{res}$ $T^{(z)}_{p-down} = M\frac{V_z}{b}(M\gamma)^{n^z_p} + T_{res}$ T_{s-down}: Down time for request z of M VMs migration in series. $T^{(z)}_{p-down}$: Down time for request z of M VMs migration in parallel. V_z: Total memory content to be migrated for request z. b : Network pipe bandwidth. γ: ratio between the dirtying rate and the network bit rate.	All	single and multiple VMs down time analytical modeling. Then a Markov model is built for inter-DC network to study the impact of network bandwidth, number of migration requests rate and the number of connected DCs on the blocking probability of migration request.
[124]	t_{down}= $(d * l * t_n)$ / b t_{down}: Migration down time d: Dirty pages rate l: Page size b: Link speed t_n: Last pre-copy round n duration.	Xen	Analytical modeling for migration down time
[49]	$Post - migrationOverhead$ $\leq TotalDownTime \leq$ $Post - migrationOverhead+$ $((VMSize)/LinkSpeed)$ $Post - migrationOverhead$: Time consumed in commitment and activation phases of a VM migration.	Xen	studying link speed and page dirty rate impact on migration downtime
[117]	Down time= $f(t_2)/r_u$ $f(t_2)$: The number of remaining pages in the stop-and-copy phase. r_u: Transmission rate of non-empty pages.	Xen, KVM and VMware	$f(t)$ depends on the hypervisor, and in this project, different hypervisors are tested to measure $f(t)$ and so to predict the down time.

Table 4.4: CPU Overhead Modeling and Prediction

Paper	Equation	Hyper.	Methodology
[51]	$PMi_{Pred_U} = (\alpha * (\sum_{y=1}^{VM_{count}} (\text{VMx}_{ReqvCPUs} * \frac{VMx_{Pred_U}}{100})) + \beta) + (PMi_{Curr_U} - PMi_{Idle_U})$ PMi_{Pred_U} :The predicted PMi CPU utilization $\text{VMx}_{ReqvCPUs}$:Requested vCPU for each VM VMx_{Pred_U} :Predicted utilisation for each VM PMi_{Curr_U} :Current PMi utilisation PMi_{Idle_U} :Idle PMi utilisation	KVM	A linear regression model is applied to predict the PMs CPU utilisation based on the used ratio of the requested number of vCPU for the VMx with consideration of its current workload as the PMs
[91]	$R_{k+1}=\phi_0 R_k + .. + \phi_p R_{k-p+1} + \epsilon_{k+1} + \theta_0 \epsilon_k + ..\theta_{q-1}\epsilon_{k+1-q}$ R_{k+1}:Is the first step prediction for CPU resource usage R_k ϕ_i and θ_i :Constants estimated from the available date ϵ_k :Independent error terms	Xen	Auto-Regressive Integrated Moving Average (ARIMA) model is used as the basic prediction model to predict the time series of the CPU and memory usage at time k
[78]	$PredictUtil=\beta_0 + \beta_1 * CurrTotalUtil(h)$ $PredictUtil$: Host CPU utilization prediction β_0, β_1 :Regression coeff. parameters that estimate based on the last k CPU utilization of the host $CurrTotalUtil(h)$: Current CPU utilization of the host (h)	All	This technique used the last utilization over one hour ago and approximated a function. This function can forecast the short-term future utilization based on the current requested utilization in each host.

Table 4.5: Migration Network Modeling and Prediction

Paper	Equation	Hyper.	Methodology
[69] & [75]	$R_s = \alpha e^{V_{Mem}} + \beta$ R_s: is the source host network throughput. α and β are the equation constants V_{mem}: is the source host active memory size in kB at the time when the live migration should start.	VMware	Non-linear Regression approach is applied to model and predict the migration network rate given the V_{mem}
[104]	$R = D + \delta$ R: Memory transmission rate during migration. D: Memory dirtying rate during migration. δ :is a constant variable and its default value is empirically set as 100Mb/s.	Xen	Migration data transmission rate for each round is determined by adding a constant increment to the previous round's memory dirtying rate
[60]	$C_s^z = \alpha C_{s-net}^z + (1-\alpha) C_{s-app}^z$ $C_p^z = \alpha C_{p-net}^z + (1-\alpha) C_{p-app}^z$ $0 \leq \alpha \leq 1$ $C_{s-net}^z = \frac{1-\gamma^{n_s^z+1}}{(n_{max}+1)(1-\gamma)}$ $C_{p-net}^z = \frac{1-(M_z\gamma)^{n_p^z+1}}{(n_{max}+1)(1-M_z\gamma)}$ $C_{s-app}^z = \frac{V_z\left(\gamma^{n_s^z} + (M_z-1)\frac{1-\gamma^{n_s^z+1}}{1-\gamma}\right) + T_{res}b}{V_z(1+(M_z-1)(n_{max}+1)) + T_{res}b}$ $C_{p-app}^z = \frac{M_z V z (M_z\gamma)^{n_p^z} + T_{res}b}{M_z V_z + T_{res}b}$ M_z: Number of migrated VMs for request z V_z: Allocated memory for each VM γ: Ratio of the dirtying rate to the maximum transfer rate T_{res}: Time required to resume a VM at the destination host b: Network pipe max. transfer rate n_{max}: Maximum number of iterations that trigger the stop and-copy phase. n: Number of remote DCs in the cloud federation	All	Presents an analytical model for assessing inter-DC network performance in cloud federations, assuming that network load is caused by live migration of multiple VMs

Network cost requires also more contribution as obvious in Table 4.5; same as the CPU cost. So it is added also as one of the open research areas that requires more study in modeling and prediction. The formulas proposed in Table 4.5 show that the network transmission rate has an exponential relationship with the VM active memory size as in [69] and [75]. In [104], the network transmission rate has directly proportional relationship with the dirty pages rate during the migration process. In [60] the transmission rate is function of the number of VMs to be migrated, the dirty pages rate, the maximum bandwidth of the network pipe and the time required to resume the VM at the destination host. The last cost parameter table to discuss is for the power and energy modeling and prediction in Table 4.6.

4.3.5 Power and Energy Modeling and Prediction

In this subsection, we present the energy and power overhead modeling and prediction due to live migration

Table 4.6 lists different live migration energy cost models. The energy overhead in Joule is directly proportional with the memory content to be migrated as mentioned in [104], [124] and [125]. Another representation is shown in [69] and [75] for the power overhead in Watt as directly proportional with the transmission rate. Other models proposed in [51] and [90] say that the power consumption prediction in Watt is directly proportional with the CPU utilization.
Because live migration is basically a migration for the memory volume, so in the next section we discuss a new memory technology which is persistent memory that is recently provided by different servers manufactures and supported by many software vendors [29].

4.4 Open Research Areas

Based on this survey study, we can list the open research areas for pre-copy live migration cost in the below points:

1. To propose lightweight and practical prediction techniques for live migration cost that can be integrated with the hypervisors interfaces to be used by clusters' admins. Most of the proposed prediction techniques consume time and CPU intensively to train the models and predict live migration cost prediction [75]; which results in a challenge to have a technique that can be practically implemented.

2. Live migration cost aware load balance techniques for cloud environments. This is because load balance utilizes live migration in virtual and cloud computing environments [99]. So since live migration cost can not be ignored, it should be considered as an overhead for load balance techniques.

3. Live migration cost aware power saving techniques for cloud environments. Same as load balance, power saving also utilizes live migration [135], [81]

Table 4.6: Power and Energy Modeling and Prediction

Paper	Equation	Hyper.	Methodology
[69] & [75]	P_{mig}= c R_s P_{mig}: peak power overhead in Watt c: is constant. R_s: is the source host network throughput	VMware	Regression based approach is proposed to obtain the constants and predict the peak power overhead due to migration
[104]	$E_{mig} = (\alpha_s + \alpha_d)V_{mig} + (\beta_s + \beta_d)$ E_{mig}: Migration energy consumption overhead in Joules V_{mig}: Current size of memory content to be migrated in MB. $\alpha_s, \alpha_d, \beta_s$ and β_d : model parameters to be trained	Xen	Analytical model for live migration energy overhead at source and target stations. There is very little difference of energy consumption between source and target stations energy overhead
[124]	E_{mig}=$\alpha * V_{mig} + \beta$ E_{mig}: Energy overhead of VM migration. V_{mig}: Total Memory Size α and β: trained parameters in offline experiments	Xen	Analytical modeling for migration cost. The authors validated their approach based on five different benchmarks running on the migrated VM.
[125]	E_{ov}= A+B*s+C*b E_{ov}: VM migration energy overhead . s: VM memory size b: Network rate. A, B and C: are constants to be trained	Xen	Experimental modeling to propose a lightweight, linear model to estimate the energy cost of live migration of virtual machines
[51]	$PMi_{Pred_P} = (\alpha * \text{PMi}_{Pred_U} + \beta)$ PMi_{Pred_P} :The predicted PMi Power consumption for the source station in Watt PMi_{Pred_U} :The predicted PMi CPU utilization α and β : constant parameters	KVM	Propose steps to predict the PMs/VMs workload and power consumption, then estimate the total cost of the migrated VMs
[90]	P= a * $Util.$ + b P: The predicted Power consumption of the source server in Watt $Util.$: The predicted CPU utilization a and b: constant parameters	Xen	Proposes a practical experimental approach to evaluate the power consumption of VM migration. And then we quantify the power cost of VM migration both source and target servers

and [48]. So, it is important to consider live migration cost and especially the power overhead in power saving techniques for virtual and cloud computing environments.

4. Live migration cost modeling and prediction for WAN scale migration [127] and [132]. Most of the proposed modeling and prediction techniques were tested and proposed for LAN scale live migration; where the source and target hosts are within the same datacenter. So, it is an open research area to provide models and tests for WAN scale live migration.

5. Live migration cost analysis and prediction for VMware and Hyper-V. This is taking into account the fact that most of the proposed modeling and prediction techniques tests focus mainly on open source hypervisors like Xen and KVM. So more research work is needed toward commercially used hypervisors.

6. Live migration cost modeling and prediction with different in memory data analytics applications like Apache Spark [42] and in memory data base applications like Redis [41]. These in memory applications are memory intensive application and show example for modern applications trend that run on private and public cloud environments.

7. Referring to Table 4.4, more research work is still needed in live migration CPU cost modeling and prediction. Since few papers could be found that consider this cost parameter modeling and prediction.

8. Referring to Table 4.5, more research work is still needed in live migration network cost modeling and prediction. Since also few papers could be found that consider this migration network modeling and prediction.

9. From Tables 4.2 - 4.6, to compare between different prediction techniques for the same cost parameter. The comparison can be in terms of the prediction accuracy and the prediction CPU consumption overhead.

10. Live migration cost analysis and prediction for VMs with Persistent Memory (PMem) and to compare the cost versus the standard DRAM. This includes the different memory configurations of PMem as following:

    - VMs with 3D xPoint PMem in memory mode.
    - VMs with 3D xPoint PMem in App Direct mode.
    - VMs with NVDIMMs in memory mode.
    - VMs with NVDIMMs in App direct mode.

11. Consider variant memory capacities in the PMem test, in order to add contribution about very large memory VMs migrations; especially with 3D xPoint PMem as a high capacity/DIMM memory technology.

Chapter 5

Cost Modeling, Prediction and Timing Optimization

In this section, we discuss our main contribution of the thesis which is live migration cost modeling, prediction and timing optimization. The proposed models for these three techniques rely on using machine learning algorithms that can be implemented practically and show acceptable accuracy in order to help the IT admins to get an estimate cost of live migration before proceeding with that and also get a timing recommendation for the live migration event of a specific VM. In the next section, we discuss firstly how the live migration cost could be modeled.

5.1 Proposed Live Migration Cost Modeling

To obtain empirical models of live migration overhead, we used the following methodology. Firstly, we built a VMware testbed; with configuration details as in the next section. Then we run different live migration sessions for different number of parallel VMs and different memory size, because memory size is the dominant factor in live migration cost.
The running application is a main player in live migration overhead amount. In this study, we considered three different workloads; Linpack stress benchmark [9], network stress and idle VM. Linpack benchmark is a CPU and memory intensive benchmark which is the worst case scenario for an application from resource utilization perspective. Idle VM is the best case scenario for a VM utilization and finally network stress which represents virtual machines with web applications in cloud computing environment. This is because the VMs in the cloud environment are mainly accessed by remote users through the network and so the applications input/outputs should be transmitted through the network. By these three applications, we validate the obtained reading of live migration overhead performance metrics.
For each live migration session, we use VMware vCenter server performance tool to measure the migration time in seconds, the active memory utilization change in kB, the network throughput in kBps change, and the power consumption increase in Watt. From the obtained results, we get several distracted points that can be fitted to a defined mathematical relation. To find the best fit relation,

our target is to minimize the error between the obtained testing readings and the fitted curve line. This is done using MATLAB [13] to achieve the objective function of minimum error in equation:

$$\begin{aligned} \text{Min.} \qquad & E_{\theta 1,\theta 2} = \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)} - y^{(i)})^2 \\ & h_\theta(x^{(i)}) = \theta_1 + \theta_2 \\ & RMSE = \sqrt{E_{\theta 1,\theta 2}} \\ & Error\% = \frac{RMSE}{\frac{1}{m}\sum_{i=1}^{m} y_i}.100 \end{aligned} \tag{5.1}$$

Where:
i: test reading number
x: is the horizontal scale variable
$E_{\theta 1,\theta 2}$: Square Error as function of θ_1 and θ_1
$y^{(i)}$: testing reading value
m: Total number of readings
$RMSE$: The root mean square error

In the next section, we discuss the test-bed configuration details, experiment results, performance modelling and the theoretical validation of the obtained models

5.1.1 Testing Environment

To study live migration performance modeling, we have built a VMware lab setup with the following specifications: 2 physical hosts (Dell PowerEdge 2950). Each host has 8 CPU x 2.992 GHz Intel Xeon, 20GB RAM, 2 NICs and 1 HBA with 2 Fiber ports/ card. VMware vSphere 5.1 hypervisor is used vCenter server appliance for live migration and performance monitoring.

As shown in Fig. 5.1; both hosts are connected to a shared EMC VNX block storage [35] with 1TB LUN via FC-SAN. The SAN Switch is Cisco with 4Gbps ports. The Ethernet switch is Cisco with 1Gbps ports. Live migration process utilizes the Ethernet switch [34]. The two physical hosts are configured in as a cluster that is managed by VMware vCenter Server which manages the cluster resources and include vMotion feature [44]. Performance metrics are also gathered using vCenter Server performance monitoring interface. The VMs that are used in this migration are Linux Ubuntu 12.04 (32bit) with 4 vCPU. The testing benchmark is Linpack, network stress; Apache Bench (AB) and idle VM. Linpack is a CPU and RAM intensive benchmark [9]; which is the worst case scenario for a running application. The network stress application that we have used Apache Bench (AB). Apache Bench tool stresses the web servers with lots of requests to test the servers' response; which is also a stress on the network. Idle VM is simply an idle Ubuntu OS VM; with no running applications. So the workload comes only from the Ubuntu OS events.
The VM RAM size is the most effective parameter in the migration performance [124]. So we test the impact of live migration on the datacenter performance with different memory sizes to have different migration volumes. The VM RAM

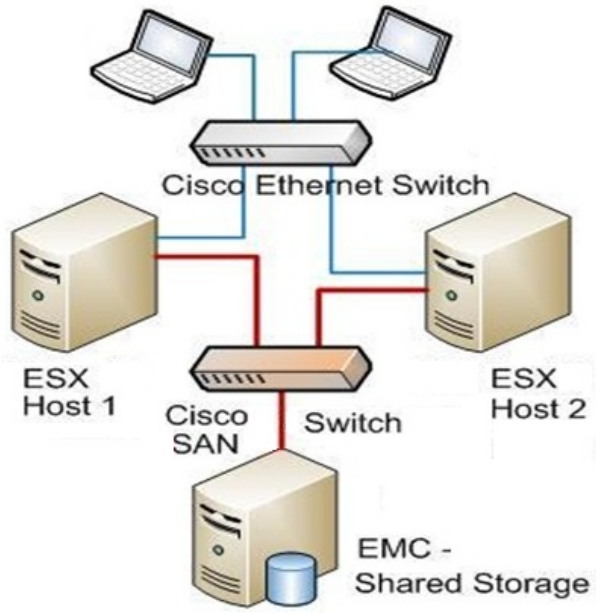

Figure 5.1: Testing Lab Network Diagram

size varies between 1GB, 2GB, 4GB and 8GB. The VM is powered on, the benchmark is run and after 5 minutes at least, the VM migration is started from one of the physical hosts to the other in order to distinguish between the benchmark impact and the migration impact on performance. After the migration is finished, the migration time is calculated and the impact on the target host performance is monitored. Finally the benchmark is stopped.
Based on the testbed configuration in section 3, we have run several testing scenarios in order to obtain empirical models for live migration performance that can be used for live migration cost prediction. Using the above infrastructure the testing sequence was as following; the migration was done 5 times for each memory configuration value of the same number of VMs running an application. For example in Linpack application, we run 1VM with 1GB RAM 5 times, 5 times for 2 GB RAM VMs, 5 times for 4GB RAM and 5 times for 8 GB RAM. The same steps are repeated for different number of VMs at the same time and for different applications.

5.1.2 Modeling Formulas

5.1.2.1 Migration Time Modelling

Observations Migration time is the period between the VM migration request initialization and having the VM activated at the destination server. In this experiment, we follow VMware vsphere client tasks list in order to know the VM migration start and end times. The difference between the two times is the VM migration time. Fig. 5.2 - 5.4 show the migration time testing results versus the active memory size over the network throughput due to migration. The results show the migration time duration for different number of VMs, with different memory sizes and using Linpack, network intensive and idle VM

applications. In Fig. 5.2, the migration time of Linpack benchmark is presented and from the obtained results, we see the following observations:

- With increasing the number of VMs in parallel migration, the migration time increases; so 1 VM migration has the least migration time and 4 VMs migration has longest migration time.
- The starting point of the line in the vertical axis has the lowest value for 1 VM migration and has the highest value for 4 VMs migration.
- The regression of the resultant testing points shows linear curve fitting; as represented in equation (1.6) which assures the proposed models in [69], [73] and [71].

$$T_{mig} = a.(\frac{V_{mem}}{R_s}) + b \tag{5.2}$$

  R_s: is the source host network throughput increase, V_{mem} is the source host active memory size before migration starts, T_{mig} is the migration duration time and a and b are constants that change with the datacenter hardware configurations.
- The linear relation slope increases with having more number of parallel VMs due to the greater impact on the migration time with having more VMs at the same value of the memory size over the rate.

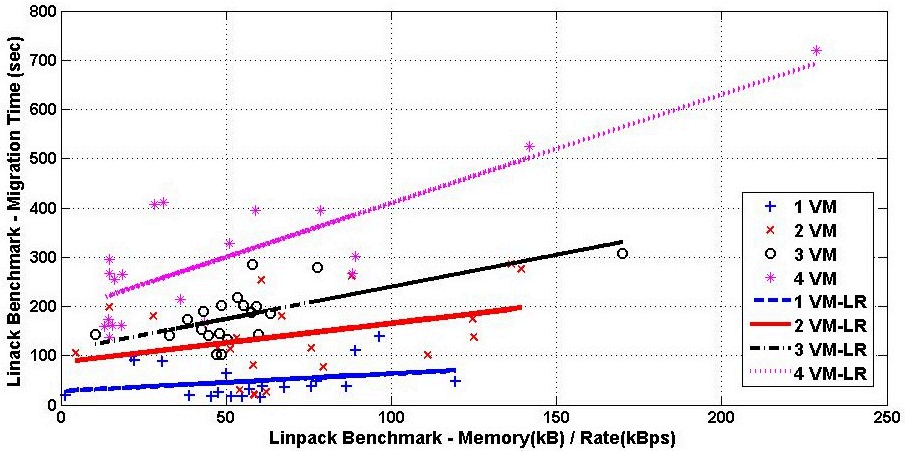

Figure 5.2: Avg. Migration Time – Linpack Benchmark

Theoretical Background – Migration Time The theoretical background of the migration time linear modelling in Fig. 5.2 - Fig. 5.4 can be explained as following. Equations set 5.3 represent the migration time analytical modelling proposed in [104]. D is the dirty pages rate in kBps, n is the number of live migration iterations and R_s is the transmission rate in kBps. For successful live migration λ should be less than 1; otherwise the migration copy iterations will exceed the stopping condition timeout and fail. Fig. 5.5 shows the relationship in the set of equation 5.3 with different values of n and λ. The normalized migration time can be fitted to a linear relation especially for small values of (n) or (λ). With higher values of (n) or (λ), the linear fitting error increases.

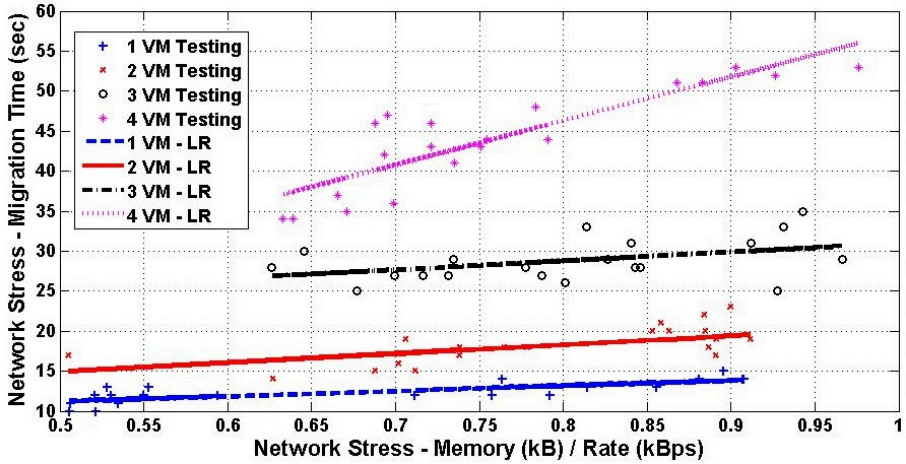

Figure 5.3: Avg. Migration Time – Network Stress Application

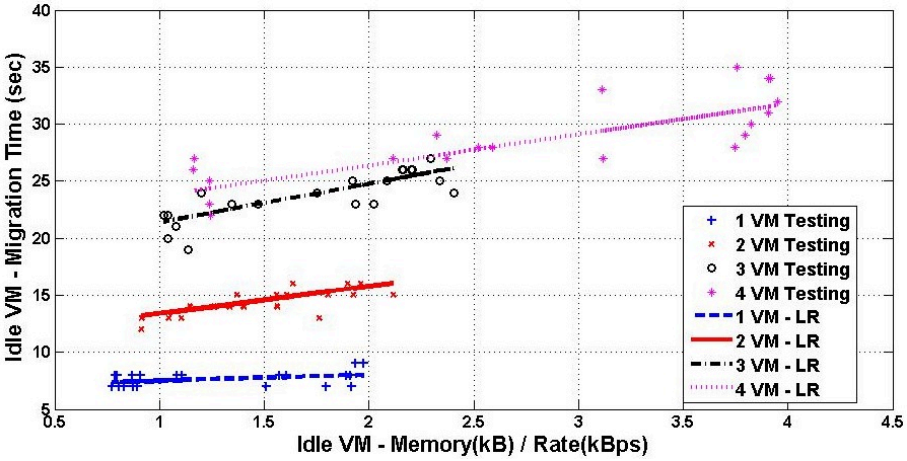

Figure 5.4: Avg. Migration Time – Idle VM

$$
\begin{aligned}
T_{mig} &= \sum_{i=0}^{n} T_i = \frac{V_{mem}}{R_s} \cdot \frac{1-\lambda^{n+1}}{1-\lambda} \\
\lambda &= \frac{D}{R_s} \\
T_{Norm} &= \frac{T_{mig}}{\frac{V_{mem}}{R_s}} = \frac{1-\lambda^{n+1}}{1-\lambda}
\end{aligned}
\tag{5.3}
$$

The modeling error of the migration time is presented in Table 5.1 based on the RMSE calculation and then the RMSE value division over the mean of the measured values. As shown in the table, the migration time modeling error does not exceed 20% which makes the proposed regression error acceptable.

5.1.2.2 Network Throughput Modelling

Observations Network average rate is the average throughput at which data was

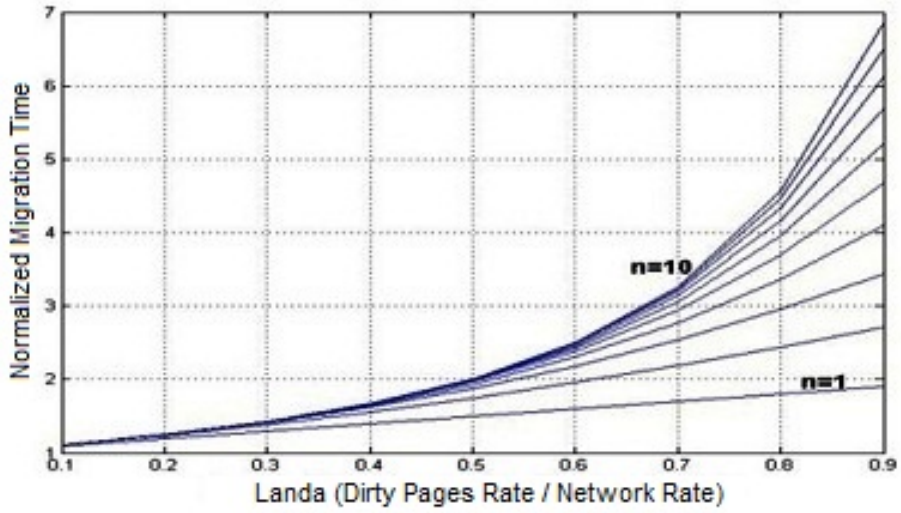

Figure 5.5: Migration time relation between λ and n

Table 5.1: Migration Time Regression Error

Model	**Linpack**	**Network Stress**	**Idle VM**
Avg. RMES% (Regression Error)	17%	14%	16%

transmitted from the physical host NIC card during the migration time interval. This represents the consumed bandwidth of the network in Bps for live migration process. Fig. 5.6, Fig. 5.7 and Fig. 5.8 show the relation between the network rate and active memory size of the source host. From these figures, we highlight the following observations:

- The relation between the network rate and the active memory size for the three curves can be modelled as an exponential relation.

$$R_s = \alpha e^{-V_{Mem}} + \beta \tag{5.4}$$

  Where: R_s: Source host throughput increase
  V_{Mem}: Source host active memory size before migration α and β are constants

- Values of the x and y axes increases with the number of parallel VMs to be migrated.

- 1 VM curve has the lowest start point in x and y axis in the three figures because it has the lowest active memory size and rate

- 4 VM curve has the highest network throughput in the three figures due to the largest active memory consumption.

- The curves may overlap at some points which is due to the change in the memory size with moving the x axis. So for example, 1 VM with 8GB memory will consume more throughput than 2 VMs with 2GB memory size.

Table 5.2 shows the modeling error of the live migration network cost. As listed in the table, the modelling error is less than 20% which presents an acceptable accuracy of the proposed modelling is acceptable.
We may notify here also that in the literature review other more accurate algorithms can be used, however the other proposed algorithms show more complexity and so more CPU consumption. This means that there is a trade-off between the modeling accuracy and complexity. For this modeling, we focus on having a model that is simple and with acceptable error. Such that this model can be used in real-time predictions; as we will discuss in the next section. Theoretical Background – Network Rate Live migration process is managed by VMware cluster vCenter server which uses the Transmission Control Protocol/Internet Protocol (TCP/IP) in the networks layers 3 and 4 for the live migration management and the iterative copies of memory pages. As mentioned in [4] and illustrated in Fig. 5.9, TCP congestion window is initially set to a maximum segment size. The sender then sends a burst of this number of bytes. If the burst is received (timer doesn't expire) then it doubles the congestion window and sends a burst of 2 * max segment size. It continues increasing the congestion window until a burst is lost (time-out). Now the sender knows how much the network can handle. This is called slow start. Slow start phase has almost an exponential relationship between the segment size and the number of rounds. When a timeout happens the threshold is set to one half the current congestion window, and increases in a saw-tooth shape [131].
Because live migration is running for short time; few minutes on average, we can say that it is mainly controlled by the slow start phase in TCP which has an exponential relationship between the number of round trips and the number of segments to be sent in each round trip. This represents also the relation between the active memory size and the network rate. Because in x-axis, the more active memory content, the more number of round trips and in y-axis the more number of segments to be sent at the same time, the higher network throughput.

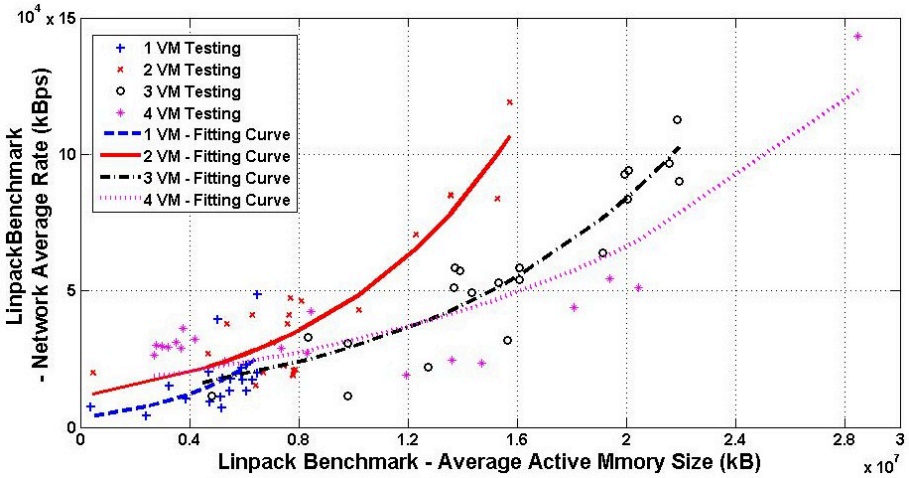

Figure 5.6: Avg. Network Rate – Linpack Benchmark

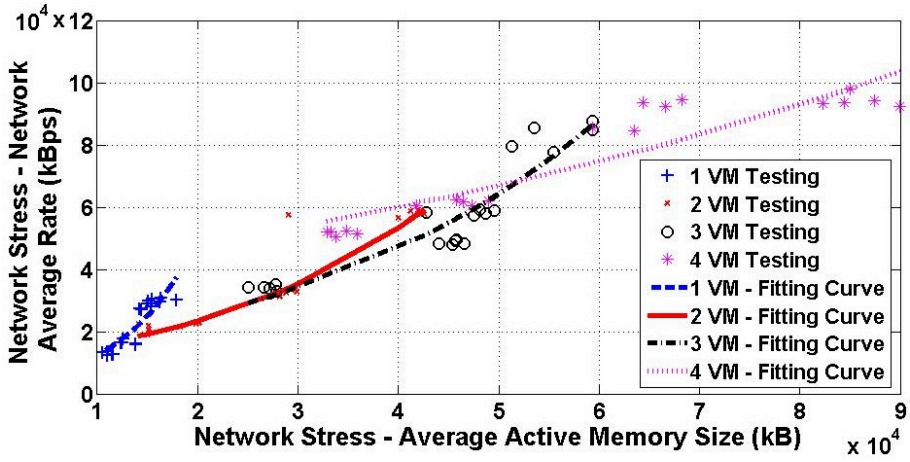

Figure 5.7: Avg. Network Rate – Network Stress Application

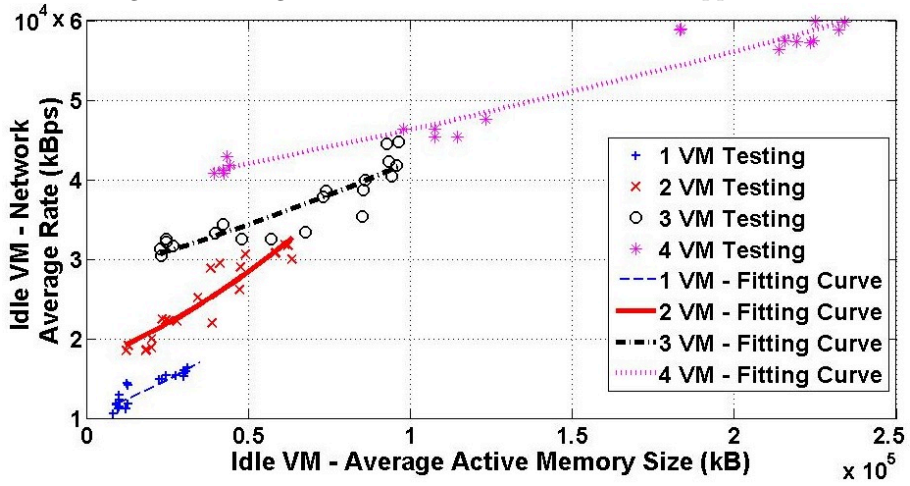

Figure 5.8: Avg. Network Rate – Idle VM

5.1.2.3 Power Consumption Modelling

Observations Power consumption is always a critical point in datacenters operations due to the AC power high running cost and impact on CO_2 emissions. One of the main benefits in resource virtualization is optimizing the power usage for IT systems. So we study in this sub-section, the power consumption overhead due to VMs live migration. In this experiment, we use VMware vCenter server performance tool to measure the peak increase in the power consumption in Watt of each physical host after each migration test and map this change with the VM memory size and number of VMs; as shown in Fig. 5.10 - 5.12. From the charts in the below power consumption figures, we note the following:

- Power consumption peak increase has linear relation with the transmission rate. This linear relation between the power consumption and the transmission rate is also obtained in [104], but for Xen environment. In this project, we prove the same relation also for VMware environment.

$$P_{mig} = \frac{dE_{mig}}{dt} = c\,\frac{dV_{mig}}{dt} = c\,R \tag{5.5}$$

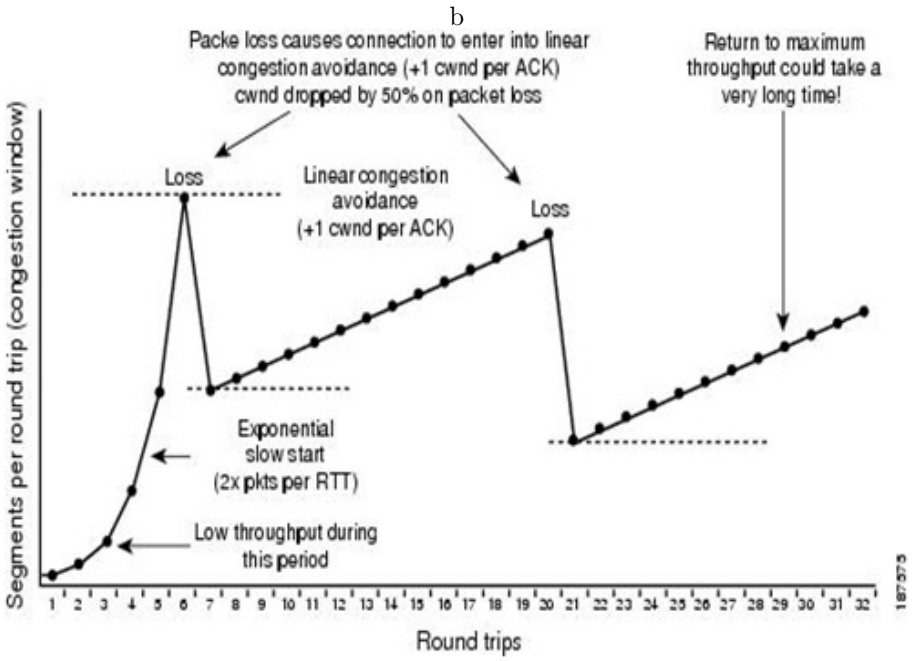

Figure 5.9: TCP congestion control [4]

where:
P_{mig}: Peak power increase after live migration
E_{mig}: Peak energy increase after live migration
V_{mig}: Migration volume
R: Migration rate

- Linpack application shows the highest peak power increase due to high network rate, in contrary the idle VM shows the lowest peak power consumption.
- The graph lines overlap at some points because high memory size for low number of VMs may consume higher power than low memory size for more number of VMs; for example in Fig. 5.11, 3 VMs with 8 GB memory consume more power than 4 VMs with 4GB memory.

Theoretical Background – Power Consumption Migration rate is the dominant factor for power consumption during the migration process; with higher network rate the migration duration becomes shorter and the power consumption increases [104]. Based on the proposed modelling for live migration energy

Table 5.2: Network Throughput Regression Error

Model	**Linpack**	**Network Stress**	**Idle VM**
Avg. RMES% (Regression Error)	18.5%	16.8%	15%

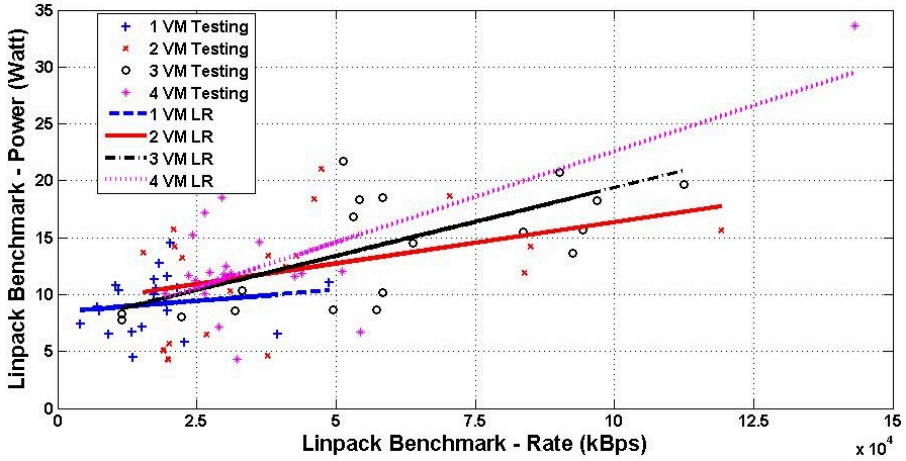

Figure 5.10: Power consumption – Linpack Benchmark

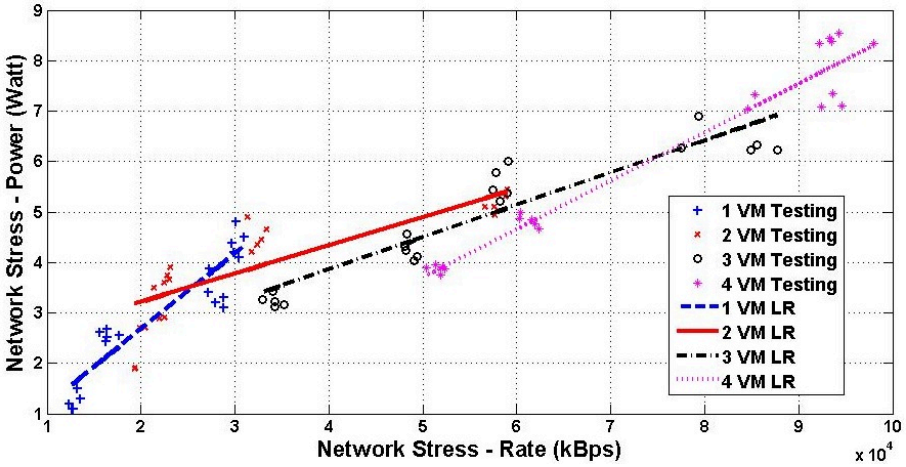

Figure 5.11: Power consumption – Network Stress Application

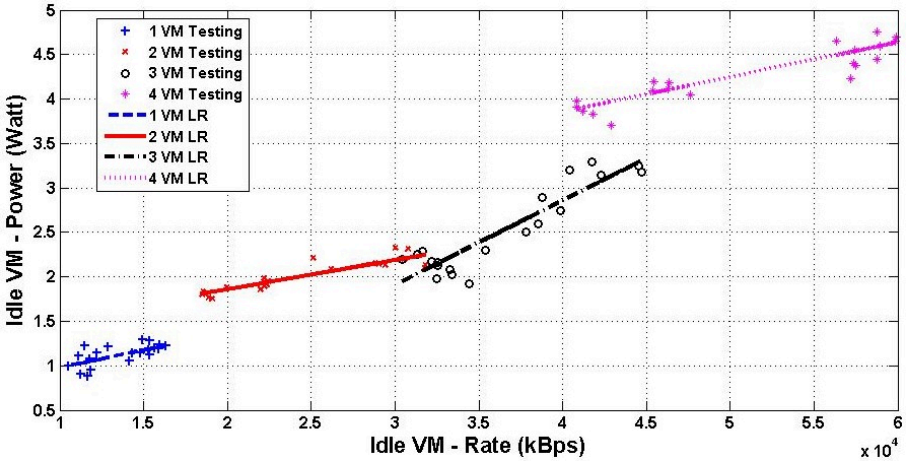

Figure 5.12: Power consumption – Idle VM

consumption in [104], there is a direct relation between the consumed energy

E_{mig} and the migration volume V_{mig}; equation 5.6 which represents also the direct relation between the consumed power and the migration rate:

$$E_{mig} = \alpha\, V_{mig} + \beta \tag{5.6}$$

5.2 Live Migration Cost Prediction

Machine learning is a segment of Artificial Intelligence (AI) where systems can learn for historical data, create patterns, predict results and take decisions [38]. Machine learning is commonly used in modern datacenters for different use-cases including medical, financial, transportation and retail industries. Linear Regression (LR) is one of the popular techniques in machine leaning. In LR, the relation between the input data and the prediction outcome can be represented in a linear relationship. If the relation is in a non-linear format, so non-linear regression technique can be used [38].
In this research, machine learning is used because the proposed models in equations (5.3) -(5.5) can not be used in live migration cost prediction. This is due to the constants included in the equations. These constants values depend on the cluster environment characteristics; like CPU, network and hypervisors versions configurations. So, machine learning is required to train the models in reference to equations (5.3) -(5.5) until the constants values are obtained for each cluster. Then, these equations can be used for cost prediction.

5.2.1 The Proposed Algorithm

In this section, we present the proposed machine learning based framework for live migration time, transfer rate and power consumption overhead prediction. As shown in the flow chart of Fig. 5.14, the proposed framework consists of two main phases, the training phase and the prediction phase.
The training phase starts when the VMware PowerCLI script connects to the cluster vCenter Server Appliance (vCSA). Then data collection starts with listing all the events happened in the cluster during the last 12 hours. This 12 hours cycle can be changed based on the cluster admin preference. From the collected events, vMotion events are filtered out. These vMotion events details like the source host, target host and time stamp are captured. The time stamp include the start time and the complete time of the vMotion event. Then the script calculates the complete and start time differences in order to get the migration time of each vMotion request. The performance logs of vCSA are collected at the start and the completion times at the vMotion events in order to get the active memory size of the migrated VMs in kB, the network overhead in kBps and the peak power change in Watt.
From the above data of each vMotion event, we use the regression models in equations (5.2, 5.4 and 5.5) to calculate the equations constants after doing several substitution and considering the minimum Root Mean Square Error (RMSE); equation 5.7.

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(d_i - f_i)^2} \tag{5.7}$$

Where N is the number of sample points collected during the last 12 hours. d_i is the measured performance value and f_i is the regression model equation value.
If the change in all the constants value became greater than 10% of the last 12 hours cycle, the script waits for more 12 hours and run again to continue in the training phase. If these changes became less than 10% of the last 12 hours, so we consider the the training phase of this cluster is finished, and the script then moves to the prediction phase. The time consumed until reaching this 10% convergence depends on the changes that happen in the VMs active memory size. These changes in the active memory depend on the running workload. This means that the more changes happen in the active memory size of the VMs, the less time required to reach the 10% convergence; this is because the regression relation curve will cover most of the values that the active memory size can take and so the regression relation model can be defined faster. This sequence of data collection and models training makes the algorithm can fit at any vCenter Server cluster and adapt its models based on the cluster configuration in order to provide cost prediction with high accuracy.
In the prediction phase when a vMotion request is sent by the cluster admin, the active memory size is captured by the script before proceeding with live migration. Once the active memory size is known, equation 5.4 is used to predict the source host network throughput. Then equation 5.2 is used to predict the migration time, and finally equation 5.5 is used to predict the peak power consumption. The prediction data is exported to a .csv file that the cluster admin can read, and then decide either to proceed with this migration or not.

5.2.2 Testing Environment

The testing environment is shown in Fig. 5.13; as shown it has a similar infrastructure to enterprise datacenters. It includes the following hardware setup; Three Hosts (Hewlett Packard DL980 G7) with 8 x Intel Xeon (Nehalem EX) X7560, 8GB RAM, 4 NICs, 2 HBA with 2 Fiber ports per card. The three hosts are connected to a shared storage EMC VNX5800; 1TB LUN via FC-SAN network.The Ethernet switch is Cisco with 1Gbps ports. From software prespective, VMware ESXi 6.5.0 Hypervisor is used with vCenter Server that manages both hosts and the VMs live migration. VMware PowerCLI 6.5.1 build 5377412 is connected to the vCenter Server to run the framework algorithm script.
In this set up we have created four Linux Ubuntu 12.04 VMs with 4 vCPU, and different RAM sizes (1GB, 2GB, 4GB and 8GB). The VMs have mainly 3 categories of workload:

- CPU and Memory intensive: This is considered as the worst case scenario for a running workload. The CPU intensive benchmark that we used is Linpack [9] and the memory stress is the Linux *Stress* Package [14].
- Network Intensive: The network stress benchmark that we have used is Apache Bench (AB). Apache Bench tool stresses the web servers with lots of requests through the network to test the servers response.
- Idle: VM is simply an idle Ubuntu OS VM; with no running applications.

From networking side, we have followed VMware best practice to isolate live migration traffic by creating a dedicated VMkernel vMotion port group [37].

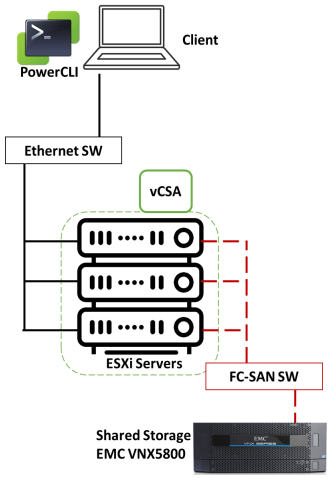

Figure 5.13: Testing Lab Layout

As a basic introduction to networking in VMware vCenter server, a virtual Distributed Switch (vDS) is created between the cluster physical hosts through the hosts physical network adapters. This vDS is connecting all the VMs insides these physical hosts; such that each VM is connected to this vDS using its virtual ports. A port group is an aggregation of multiple virtual ports to isolate their traffic using labeling. This is used basically to isolate management, kernel and data traffic from each other. vMotion has a specific port group that isolates its traffic from other kinds of traffic [37]. With this testing setup, we have run 12 testing scenarios; as a matrix of 3 workload categories and 4 different VM sizes. For each configuration, we have run live migration at least 10 times. So the resultant is 144 readings. For each run, we do live migration for a VM from one of the physical hosts to other of the other two hosts; without doing any storage migration [69] and [73]. So only the CPU state, memory and buffers contents are migrated. For data collection and models training, every 12 hours the script gathers all the live migration events that happened in the cluster and from the events timing details, the live migration can be obtained. Then the script uses $Get - Stat$ function to import the statistics beyond each live migration event; specifically the data rate change, the peak power increase and the active memory size of the migrated VMs. This data is used for models training for the cluster. For the prediction phase, we make use of the trained models to predict the future VM live migration cost when the admin sends a live migration request, and given the active memory in kB. The estimated cost is exported as csv file that the admin can check before proceeding with the live migration.

5.2.3 Results and Analysis

After testing the proposed approach in Fig. 5.14 on the test-bed of Fig. 5.13, we present in this section the prediction results for almost 144 readings. Before showing the prediction phase graphs, we start with the training phase. This is

to show how the models are trained until obtaining equations (5.2, 5.4 and 5.5) constants with at least 90 percent accuracy.

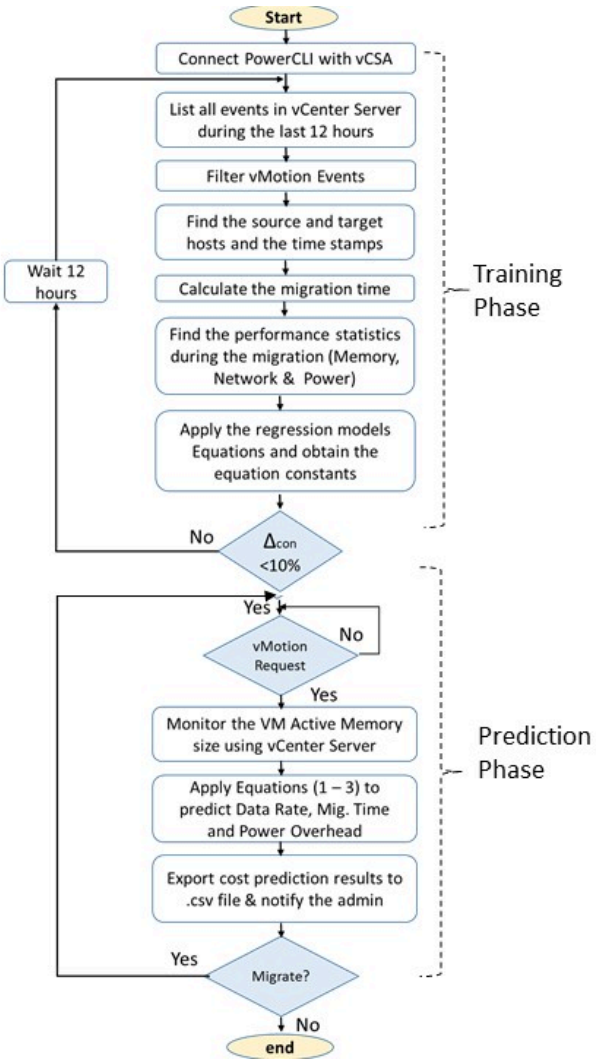

Figure 5.14: Proposed Prediction Framework

5.2.3.1 Training Phase

In this phase, the script collects the last 12 hours live migration events. Then the performance statistics of these live migrations are gathered including their time stamps. The migration time is calculated by the script; given the start and end time of the live migration event. The other gathered statistics include the active memory size, the source host transmission rate and the peak power change. All these details are used to train the models of equations (5.2, 5.4 and 5.5) and to obtain the constants of this cluster by solving several linear equations. For example in order to calculate a and b of equation (5.4), we use every two live

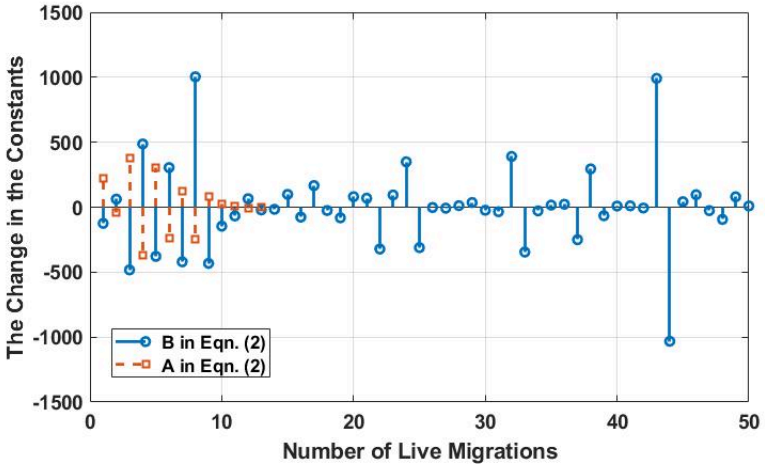

Figure 5.15: A and B Change until Saturation

migration events statistics to generate two equations in two unknowns. These unknowns are a and b in this example, because the migration time, the active memory size and the transmission rate are given. So, we gather every two live migration events statistics to solve for the constants of equation (5.4). The script keeps on solving for the values of a and b until finding the changes in the values of a and b are less than 10 percent compared to the calculated values of last equations solution. This means that these constants are at least 90 percent saturated. Fig. 5.15 shows the changes of a and b constants versus the number of live migration equations that were used until reaching the 90 percent saturation. As shown in Fig. 5.15; the difference in a is changing with the number of live migrations which represents solving more equations until the 90 percent saturation at difference equals 0.22 after 14 live migrations. At this point a=9.04. For b constant, the script has run 50 live migrations to reach the 90 percent saturation at difference equals 9.16. At this point b=21.04. This means that modeling with equation (5.4) could be used after 50 live migration runs for this cluster.

For equation (5.7), we could also solve every two equations of live migrations data as linearly to obtain the values of α and β. The is because the values of the active memory size and the migration time are given, so we can substitute with them and then solve two equations in two unknowns; α and β. Fig. 5.16 shows the differences happen in the values of α and β after each live migration until reaching the 90 percent saturation. As shown; the constant α could reach the saturation at difference equals 1850 after 54 live migrations. At this point α equals $2.02 * 10^4$. The value of β reaches the 90 percent saturation at difference equals 2225 also after 54 live migrations. At this point, β equals $2.33 * 10^4$. This means that modeling with equation 5.7 can be used after 54 live migration runs. Finally, equation (5.5), which has just one unknown; c and so it can be resolved given just one live migration statistics. So for each live migration run in the past 12 hours, we could read the transmission rate and the peak power overhead and then calculate the constant c. Fig. 5.17 shows the changes happen with each live migration calculation to the the constant c; as shown it 90 percent saturates

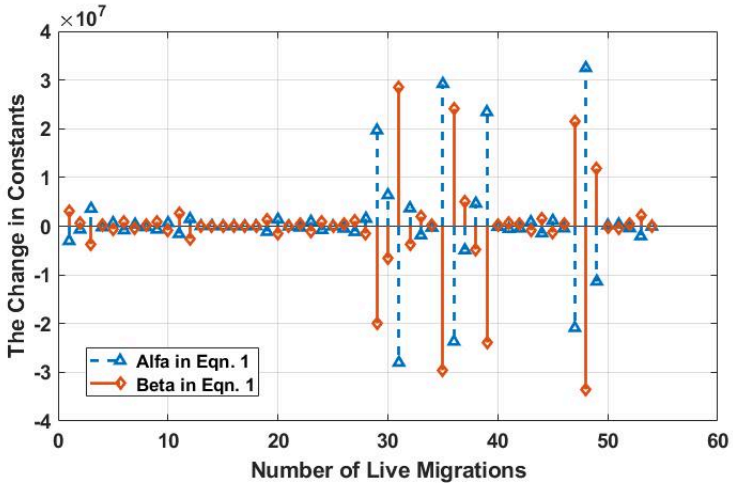

Figure 5.16: Alpha and Beta Change until Saturation

after just fours live migrations runs at difference of c value equals $0.6 * 10^{-5}$. At this point c equals $16 * 10^{-5}$.
From the above analysis, we find that it required 54 live migration runs to be able to train the models provided in equations (5.2, 5.4 and 5.5). In general, the required number of live migrations runs to finish the training phase depends on the error gap between the training data and the regression model. The closer gap between the training data set and the model, the lower number of live migration iterations required to reach the 90 percent saturation, and vice verse.

5.2.3.2 Prediction Phase

In this subsection, we build on the training phase that we have discussed above. Now, the regression models are trained for this cluster and ready to be used for future live migration cost prediction. The testing results in Fig. 5.18 - Fig. 5.20, show the regression models that are used and the actual measured data after migration.

The measurements in Fig. 5.18 - Fig. 5.20 are for VMs live migrations with different configurations including memory size of 1GB, 2GB, 4GB and 8GB VMs that utilize three different kinds of workloads. As discussed in section V, these workloads are CPU and memory intensive, network intensive and idle VMs. This results in 12 different VM configurations. Each configuration is tested 12 times; which represents the existing 144 measurement points in the following figure. The prediction starts with Fig. 5.18; so given the active memory size of the VM to be migrated, the source host transmission rate can be predicted. The VM active memory size can be measured before live migration. Fig. 5.18 shows the exponential relation as a valid regression model between the active memory size and the transmission rate. That is how the transmission rate can be predicted. Table 5.3 shows the RMSE of Fig. 5.18 in reference to equation 5.7. After obtaining the transmission rate from Fig. 5.18, we calculate now the active memory size over the transmission rate; which is the horizontal axis of Fig. 5.19. So the migration time can be predicted; using the linear regression

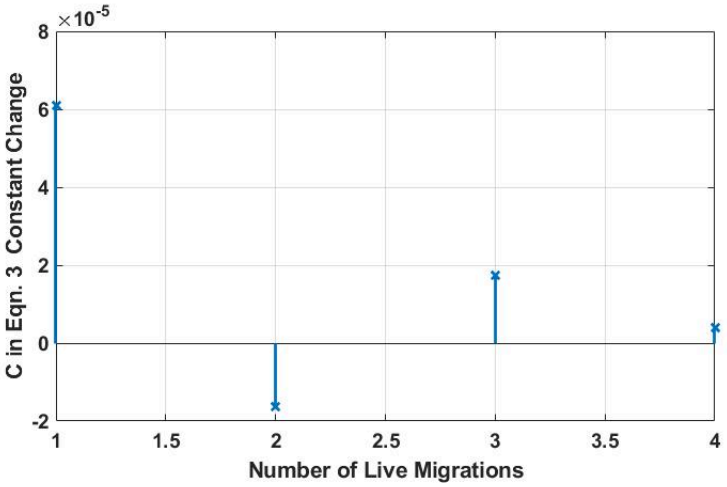

Figure 5.17: C Change until Saturation

Table 5.3: RMSE of the Regression Models

Model	Fig.	RMSE	Error %
Transmission Rate	Fig. 5.18	8187	18%
Migration Time	Fig. 5.19	15.5	16%
Peak Power	Fig. 5.20	1.7	15%

model of Fig. 5.19. The RMSE of the prediction in Fig. 5.19 is also listed in Table 5.3. Fig. 5.19 also shows that the migration time can consume several minutes in case of large memory and memory intensive VMs. The last model is for the source host peak power change; which is shown in Fig. 5.20. So given the source host transmission rate, the peak power change can be obtained using linear regression.

Table 5.3 shows the prediction error of the different cost prediction parameters. The Error% is calculated based on (5.1). The prediction of the transmission rate, migration time and peak power show a prediction error less than 20%; which is acceptable. As discussed before, the proposed prediction technique proposes a simple algorithm with acceptable error that can show prompt results to the IT admins. More accurate algorithms are proposed by other researchers, however more prediction accuracy requires more complexity and results in more CPU consumption. So there is a trade-off between accuracy and complexity. In the proposed prediction technique, we focus on presenting a simple technique showing real time results and acceptable prediction error.

All these predicted live migration cost parameters are exported to a .csv file that can be accessed by the cluster admin to check the estimated cost if he/she decides to do live migration to a certain VM. This help the admins to have better planning for live migrations, and avoid resource bottlenecks that lead to live migration failures and service quality degradation. This proposed framework script can adapt itself by changing the models constants using the training

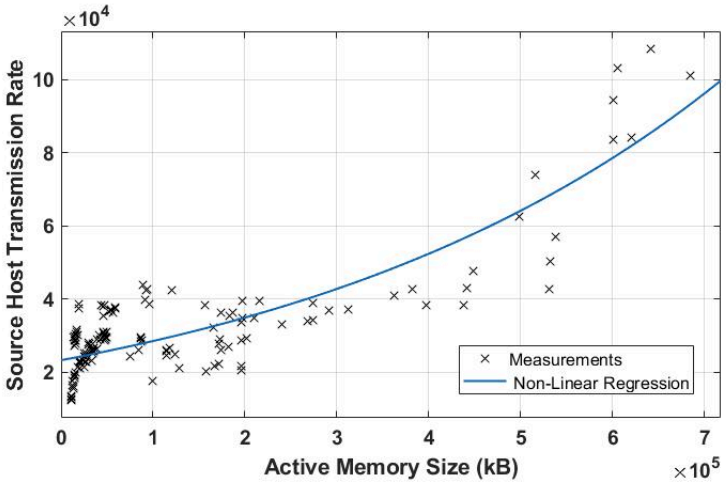

Figure 5.18: Rate vs Active Memory Size

phase; which make it flexible with any VMware cluster.

5.3 Timing Optimization for Live Migration

The objectives of our VMs live migration timing optimization proposal can be summarized in the following points:

- To propose a solution that can minimize the live migration cost in VMware environments by finding the optimal timing of the live migration process initiation and so to minimize the network contention for live migration traffic.
- To make use of the machine learning techniques there were used for live migration cost prediction and for network prediction techniques.
- The proposed approach should be a practical algorithm that can be implemented and integrated with cluster management portals.

5.3.1 Datacenter Network Utilization Prediction

Machine Learning (ML) has many applications that change our life and experience with lots of applications including healthcare, manufacturing, insurance, social networking and robotics industries. Using ML for datacenters optimization could resolve different challenges in modern datacenters infrastructure servers usage forcasting [121], networking [58], storage [120], security [52]and energy consumption [54].
In this section we focus on network traffic prediction using ML techniques. This is due to the fact that live migration has a massive impact on the datacenter networking. So from live migration cost parameters, networking overhead is the most impacted performance metric compared to other infrastructure performance metrics like CPU, memory and power overhead. On the other hand,

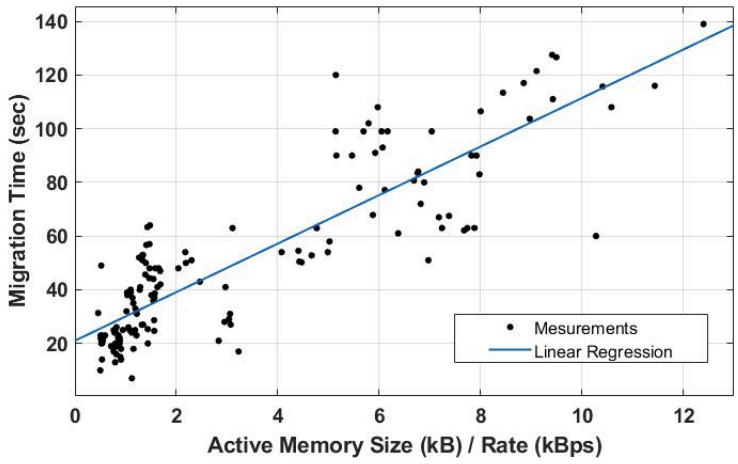

Figure 5.19: Migration Time vs Active Memory Size/Rate

in pre-copy migrations, iterative copy phase is the most time consuming phase and limitation in the network bandwidth can lead to copy process interruptions and so live migration failure.
In this work, we make use of the existing network prediction techniques proposed by other researcher to integrate it with the live migration cost prediction model that is proposed in related work to come up with a novel timing optimization for live migration in VMware environments. The proposed technique is published in [74] Using ML techniques for networking prediction is well covered in this survey paper [58] that cover ML applications for network traffic prediction, performance optimization and security. For network traffic prediction, this survey paper [58] has referred to four research articles. The first article [61] uses Artificial Neural Networks (ANN) technique with Multi-Layer perceptron (MLP) to analyze and estimate the Internet traffic over the IP network. In this proposed approach, model training is used with given inputs and outputs to optimize the weights of the neuron and minimize the error between the ANN output and the target output. For model training 750 points were used and for model testing other 250 independent points were used. Authors in [61] proved that Leven-berg-Marquardt (LM) and the Resilient back propagation (Rp) algorithms show highest precision compared to other training algorithms.

In the second article [102], the authors propose new ANN based prediction model for the inter-DC network traffic. In the model three inputs are collected for the ANN module; an elephant flow sample due to the massive amount of traffic, the total traffic and the traffic of the sublinks in both directions. The proposed model is applied at the largest DC backbone link in China that connects multiple datacenters with thousands of servers. Using this model could reduce the prediction error up to 30 percent and so the peak bandwidth can be reduced with 9 percent.
Authors in the third article [140] proposed a network traffic prediction model with high accuracy using Back Propagation Neural Network (BPNN) optimization and Particle Swarm Optimization - Artificial Bee Colony (PSO-ABC) al-

Table 5.4: Summary of Network Traffic Prediction Related Work

Paper	Technique	Dataset	Training	Output	Comp. Cost
(Chabaa et.)	MLP-ANN	1000 dataset	Past measurement	Expected Traffic Volume	High
(Li et.)	MLP-ANN	Every 30s traffic for 6 weeks	Time Series data decomposition using Db4	Next 30s expected traffic volume	High
(Zhu et.)	MLP-ANN	Hourly traffic for the past 2 weeks	PSO-ABC Algorithm	Next day hourly traffic	Lower
(Chen et.)	Hidden Markov model	Every 5 mins in 24 weeks Network Volume and flow count	KBR and RNN with LSTM unit	Traffic volume prediction	Lower

gorithm. BPNN is a supervised learning ANN based technique. In BPNN, the error between the desired output and the calculated output is back propagated to the ANN system input to minimize the error. PSO-ABC is used as an optimization algorithm that trains the ANN to minimize the prediction error and increase the performance stability [140].
In the forth paper [139], the authors propose network traffic prediction technique which is based on Hidden Markov Model that describes the relationship between the flow count and the flow volume and the dynamic behavior of both as a time invariant state-space model. The transition probability and the emission probability in the proposed Markov Model are unknown and so, packet traces are collected to learn the model and train the transition and emission probabilities. Then Kernel Bayes Rule will be used to obtain the estimation points for specific time interval with minimal error and computational overhead. Table 5.4 summarizes the comparison between the four papers that we discussed in this section. As shown in Table 5.4, we add a comparison column from CPU consumption overhead point of view for each technique. This is important for having an algorithm that can be applied in practical. So for our proposed timing optimization algorithm, we will make use of the network prediction algorithm proposed in [139]; since is networking prediction algorithm shows lower CPU consumption compared to the first two techniques. The forth technique shows also lower CPU consumption , however the output samples are hourly based; which is long period for our application. CPU consumption is critical for our application because, the computational delay for this network prediction process should be minimal in order to get back with a fast response to the network admin with the recommended migration timing when the live migration request is initiated.
this research contribution can be summarized in the following points:

1. We propose a solution that can minimize the migration time for single and multiple VMs migration in VMware environments. This solution is based on timing optimization for the live migration process initiation to minimize the network contention for live migration traffic.

2. The proposed timing optimization technique is a machine learning based approach that makes use of the machine learning techniques; mainly the previously studied machine learning based live migration cost prediction approach proposed in [75] and the machine learning based IP network prediction technique proposed in [139].

3. The proposed approach is a practical algorithm that is implemented as a VMware powerCLI script and integrate with VMware vCenter Server for cluster management.

4. To the best of our knowledge, studying live migration timing is not covered by any related work. We could only find some VMware articles that assures on using high speed GbE for vMotion to avoid network bottlenecks [23].

The proposed algorithm is presented in Fig. 5.21 flowchart that starts with connecting to VMware vCenter Server Appliance (vCSA) [44] using PowerCLI client [27] to run our PowerCLI script on the VMware cluster that is management by this vCSA. The next step is train the live migration cost prediction model from the past 12 hours events; as discussed in Fig. 5.14. Then network traffic prediction model is also trained using the Hidden Markov Model algorithm proposed in [139] and every 30 sec of the VMware VMkernel network traffic history of the past day; which means 2880 points as training dataset. As discussed in section 2.1, VMkernel network is the isolated network that includes vMotion and vSAN traffic. By finishing this step, the training phase can be considered as finished and the script is ready to predict.

When the network admin sends a vMotion request for a specific VM or for Multi-VMs migration, the VM live migration time and migration traffic rate is predicted by calling the prediction phase of the machine learning technique proposed in Fig. 5.14. By this step, the migration time and network rate are estimated. Then the prediction technique proposed in [139] is used to estimate the network traffic volume of the VMware cluster VMkernel network for every 30 sec during the next 1 hour. By finishing this step, the prediction phase of the network traffic volume, the live migration time and the migration transmission rate is finished and timing optimization check should start.
Timing optimization starts with a check if the current time; when the vMotion request is received is the a good time for initiating the vMotion process. To do this check, the script runs equation 5.8 that estimates the traffic rate during the estimated migration time interval.

$$\begin{aligned} R_{busy} &= \frac{\sum_{n=0}^{N_{mig}} V_n}{T_{mig}} \\ R_{Avail.} &= BW - R_{busy} \end{aligned} \tag{5.8}$$

R_{busy} is the estimated traffic volume in bps that will be utilized by other VMkernel network traffic, like vSAN, management,..etc. So, it is predicted to have the VMkernel network reserved with this rate during the migration time. n is the 30 sec based sample number in integer of the network traffic prediction technique N_{mig} is the last sample that approximately ends with the estimated migration time. V_n if the estimated traffic volume in bytes for each sample. $R_{Avail.}$ is the un-utilized traffic rate in bps that is available for vMotion traffic.

BW is the VMkernel network bandwidth in bps.
The succeed check point in the algorithm flow chart verifies simply the below condition in equation 5.9

$$R_{Avail.} > R_s * (1 + P_{Acc.}) \tag{5.9}$$

Where $P_{Acc.}$ is the prediction accuracy for live migration network rate. So in equation 5.9, the algorithm checks if the available network rate for VMkernel network $R_{Avail.}$ can afford the estimated migration transmission rate whilst considering the prediction accuracy that is mentioned in [75]. if this checkpoint result is (Yes), the live migration will start momentarily. If the result is (No), the algorithm moves to another phase which is finding the optimal time for initiating the VMs migration process.
When the momentarily migration check does not succeed, the algorithm checks for another better timing during the next hour from network availability point of view. So equation 5.9 is applied for the next hour prediction samples with 30 sec interval. If another optimal time is found, it will be shared with the network admin. If the admin accepts it, the VMs migration will be initiated at this time automatically. If the admin rejects this recommendation, the migration will be initiated momentarily. In case of not finding another optimal time during the next hour, the admin will be also alerted with this fact. In this case, the recommendation is to request the migration again after 1 hour. If the admin rejects that, the migration will be also initiated momentarily. If the admins accepts the recommendation, the algorithm stops.

5.3.2 Testing Environment

The testing environment is shown in Fig. 5.22; which has a similar infrastructure to enterprise datacenters that includes the following hardware setup; Three Hosts (Hewlett Packard DL980 G7) with 2x Intel Xeon (Nehalem EX) X7560, 8GB RAM, 2 NICs per server. The Ethernet switch is Cisco with 10 Gbps ports. The three hosts are connected to a 1 TB VMware vSAN datastore as a software defined storage platform. From software prospective, VMware ESXi 6.5.0 Hypervisor is used with vSAN 6.5 and vCenter Server that manages the hosts and the VMs live migration. VMware PowerCLI 6.5.1 build 5377412 is connected to the VMware PowerCLI [27] is used to run the algorithm flowchart script.
In this set up we have created four Linux Ubuntu 12.04 VMs with 4 vCPU, and different RAM sizes (1GB, 2GB, 4GB and 8GB). We focus on the RAM size change only becuase memory is a critical configuration parameter in defining live migration performance [75]. The VMs run a network intensive workload that represents web servers environment and memory intensive workload as worst case scenario for VMs migration. The network stress benchmark that we have used is Apache Bench (AB) [22]. Apache Bench tool stresses the web servers with lots of requests through the network to test the servers response. For memory stress, we have used AB for memory stress [14]. With this testing setup, we have run 16 testing scenarios per Workload for running single VM, 2 VMs, 3 VMs and 4 VMs migration in parallel. So the testing scenarios is a matrix of 4 different numbers of VMs and 4 different VM sizes. For each configuration scenario, we have run the migration at elast 5 times.

In our testing, we focus on studying the timing optimization impact on the VMs live migration time as a reflection for having less contention in the VMkernel network. Lower migration time for live migration requests, means faster migration with less interruptions and higher probability of migration success. This is basically due to avoiding the bottlenecks and the network peaks for initiating the migration process; specially for large memory VMs.

5.3.3 Results and Analysis

As presented in section 5.3, our proposed algorithm searches for the optimal timing for live migration requests during the next hour of the admin vMotion request using prediction techniques for the live migration time, network rate and the datacenter traffic volume prediction. We evaluate our proposed algorithm by showing the impact of timing optimization on the migration time. Lower migration time means less contention in the migration traffic within the VMkernel network; which represents higher transmission rate with less interruptions and higher probability of migration success in a shorter time. In section 5.3.2, we presented the testing setup and showed that for different VMs memory configurations and with different numbers of VMs, we test our algorithm using network stress and memory stress benchmarks. Fig. 5.23 and Fig. 5.24 show the proposed timing optimization algorithm testing results for memory stress and network stress workloads.
In Fig. 5.23, there are four charts that represent the obtained results for different number of VMs; as mentioned in the title on top of each chart. For each chart, we show the migration time consumed by different VMs memory configurations; 1GB, 2GB, 4GB and 8GB RAM VMs. For each configuration, There are three bars, the first solid black bar shows live migration time consumption in seconds with using the proposed timing optimization algorithm. The second dotted bar shows the average migration time for five times live migrations for the same VM without timing optimization. The third dashed bar shows the maximum observed migration time for the same VM from the five migrations happened without timing optimization. For the average and the maximum migration times bars, we add the difference in percentage on top of the bar versus the migration time achieved by using timing optimization. So for example in the 1 VM- 1GB Mem testing scenario, the migration time achieved with using timing optimization is 16 seconds and the average migration time is 1.1 percent higher. Also the maximum migration time observed is 1.25 percent higher than using timing optimization. The same explanation applies to all the charts in Fig. 5.23.

As shown in Fig. 5.23, the difference between the average and maximum

Table 5.5: Timing Optimization Performance

	Mem Stress	**Net Stress**
Average Mig. Time %	145	126
Max. Mig. Time %	205	136

migration times versus using timing optimization varies between different configuration scenarios and reaches large values in many cases; especially with multiple VMs migrations; for example on average 157% more time consumed versus migration timing optimization, and 221% more time for the maximum migration time case. These observed differences in Fig. 5.23 show the following:

- The performance of the proposed timing optimization techniques is compared versus not using it and proceeding with live migration randomly at any time. The performance metric is the migration time in sec of the VMs live migration.

- Table 4.3 shows the average of the percentages difference between the average migration time and the maximum observed migration time versus using timing optimization. As shown, for memory intensive workload, average migration time shows 145% more time than using the proposed timing optimization and the maximum migration time shows 205% more time. This means that the proposed timing optimization can save up to 50% of migration time and in average it saves 32% of the VMs migration time for memory intensive workloads.
  This enhancement in the migration time is basically due to the selection of an optimal migration timing based on the datacenter network utilization, such that live migration process can get higher network throughput. With higher migration transfer rate, live migration process can be accomplished in a shorter time and with higher success rate. Fig. 5.25 shows the difference between running vMotion with timing optimization versus without using our proposed timing optimization algorithm as an example for 4 VMs, 8GB memory for each VM and linpack benchmark workload. As shown; with timing optimization, live migration can be achieved with higher transfer rate and so the migration time becomes shorter. In this example, migration time consumes 26 time samples without timing optimization, however it consumes 16 samples with timing optimization. The sample is 20 sec.

- VMs with larger memory size consume significantly more migration time. This time is basically required for the memory content and dirty pages iterative copy migration phase. This assures the point that memory size is a significant parameter in live migration performance.

- Multiple VMs migration has also significant impact on live migration time. So the more number of VMs migrated in parallel, the more migration time required.

Fig. 5.24 has the same charts explanation like Fig. 5.23 and the difference is mainly in the results numbers. From the charts in Fig. 5.24, we share the following observations:

- Live migration time for network intensive workload shows lower values than memory intensive workloads. This is basically because the content and the dirty pages rate to be migrated is significantly bigger for memory intensive workloads.

- Table 4.3 shows also the average of the percentages numbers in Fig. 5.24; which shows 126% on average more migration time and 136% maximum

migration time versus the migration time achieved with using the proposed timing optimization approach. This means that the proposed approach can save up to 27% of the migration time and on average it saves 21% of the migration time for network intensive applications.

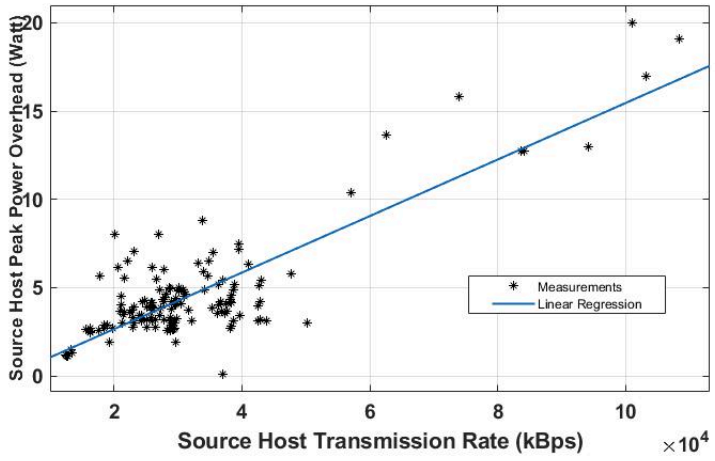

Figure 5.20: Peak Power Overhead vs Transmission Rate

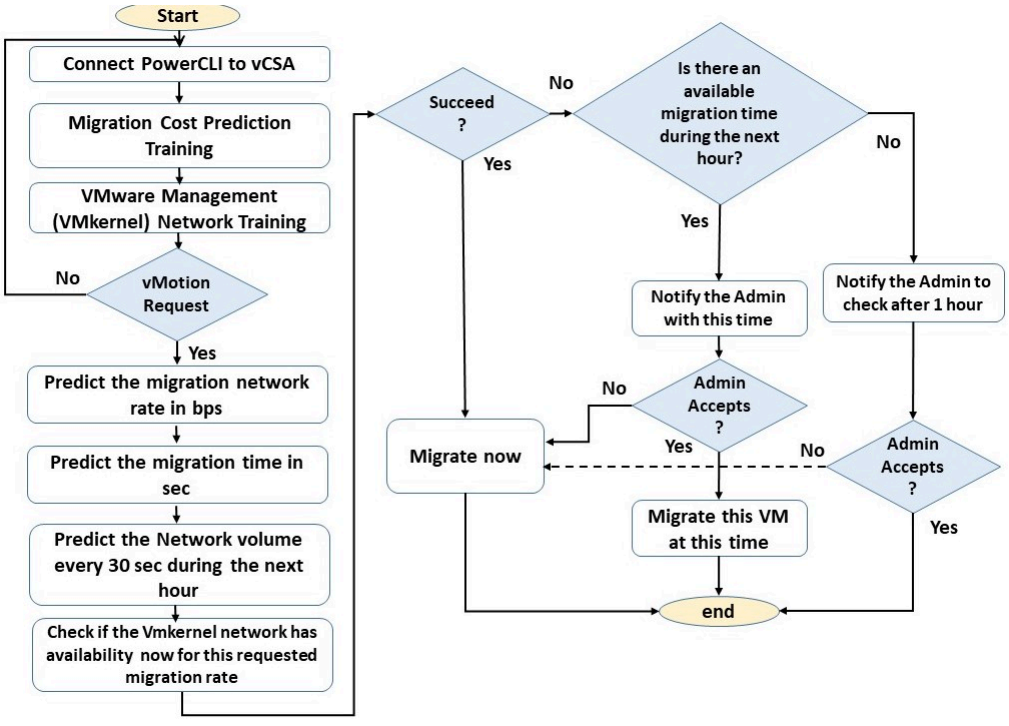

Figure 5.21: Proposed Timing Optimization Approach for VM Migration

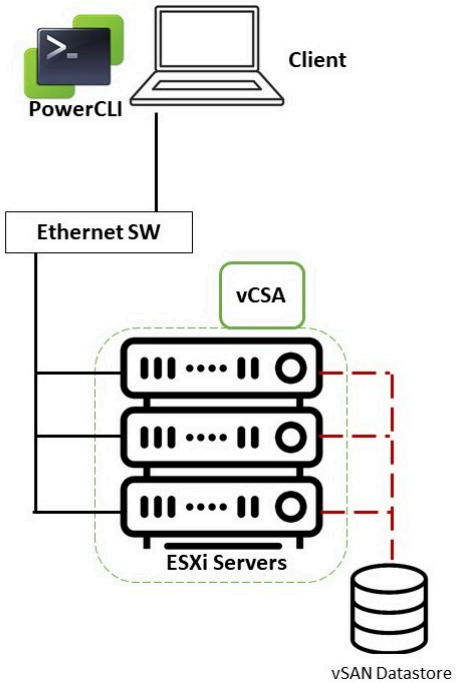

Figure 5.22: Testing Environment Infrastructure

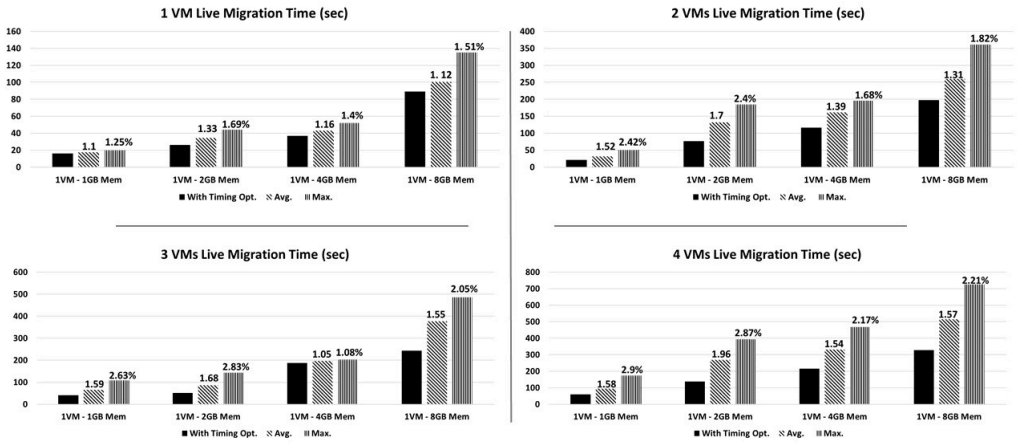

Figure 5.23: Memory Stress - Impact of Timing Optimization on Live Migration Time

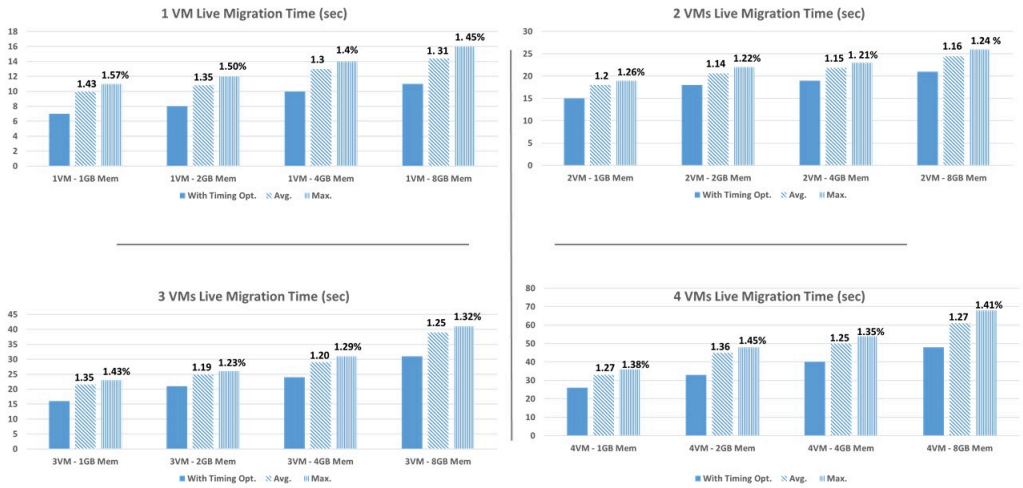

Figure 5.24: Network Stress - Impact of Timing Optimization on Live Migration Time

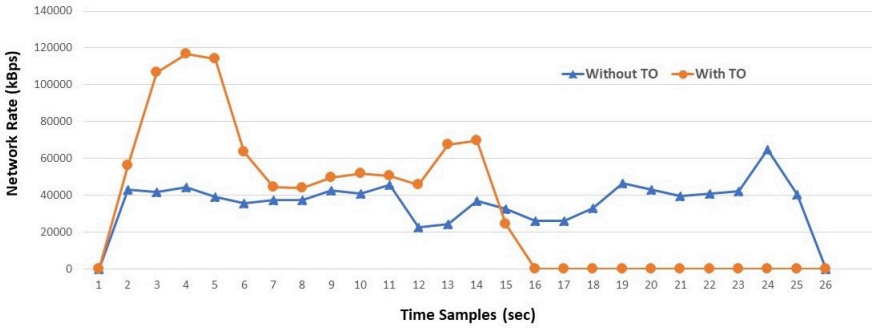

Figure 5.25: Live Migration without and with Timing Optimization

Chapter 6

Algorithms Integration with VMware

In this section, we prove that the proposed algorithms in section 5 can be practically implemented and integrated with cloud computing administration portals. VMware vCenter server is used in the chapter just as an example of a commonly used private cloud portals at different institutions and enterprises datacenters globally. We show in this chapter the different tools that we have used to integrate the proposed algorithms with VMware user interface.

6.1 Testing Environment

During this research study, we have used different lab clusters that consist of 2 VMware ESXi hosts that share the same storage and managed by VMware vCenter server. The hardware specifications and the version of VMware vSphere changed from time to time depending on the test time and the available lab resources.
For VMs live migration initiation, there are two possible methods; the first is built-in UI based live migration procedures; which is commonly used by the IT admins and VMware users. And the second method is used VMware PowerCLI shell commands. For live migration cost modeling, we used the UI. For live migration cost prediction and timing optimization, we used the VMware PowerCLI shell commands as part of the algorithm script. VMware vCenter server is used to gather the required performance parameters that were used for live migration network, power, time and memory consumption.

6.2 Integration with VMware vSphere UI

For resource management algorithms integration with VMware UI, we have used several software tools; that are presented in [76]. VMware provides a software development kit for building plugins for the VMware vSphere client. The structure of the plugin is shown in Fig. 6.1 and consists of the following components [76]:

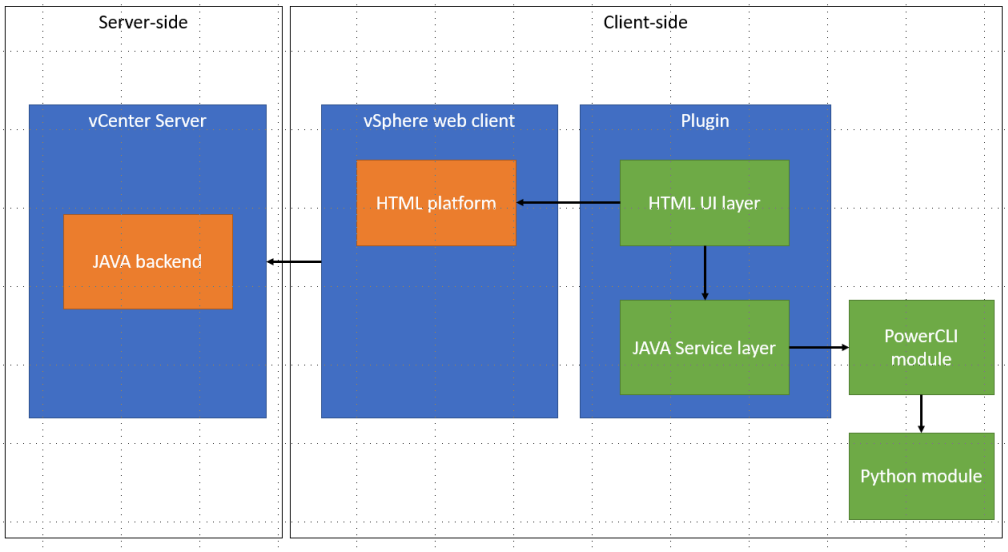

Figure 6.1: Integration with VMware Plug-in Components Structure

- HTML UI layer: this layer handles the way the plugin looks in the Web client. It allows adding menu options and navigation items.
- JAVA service layer: this layer is based on the Spring MVC and the OSGI framework, it represents the backend of the plugin. This layer communicates with the PowerCLI and Python modules to perform the coefficients calculation and provide the estimation.
- PowerCLI module: This component handles the live data collection through the PowerShell PowerCLI APIs. It collects data about the migrations that occurred in the last 12 hours. The collected data is then processed by the Python module and returned to the JAVA service layer.
- Python module: This component processes the data using the Algorithm described by this paper and outputs the value for the coefficients α, β, a, b, and c to predict the expected duration for the migration, the expected power consumption, and the expected network usage.

6.3 Solution Demonstration

In this section, we show and example of a resources management algorithm integration with VMware UI. We present a use-case for our proposed timing optimization algorithm published in [76]. This algorithm recommends the optimal timing for a VM migration based on the network utilization prediction and the VM migration network cost estimation. This technique could be integrated with VMware User Interface (UI) using the testing lab discussed in section 5.3.2. The integration with VMware UI as an extension work on [76] shows that the proposed timing optimization feature can be practically implemented and used by the IT admins.

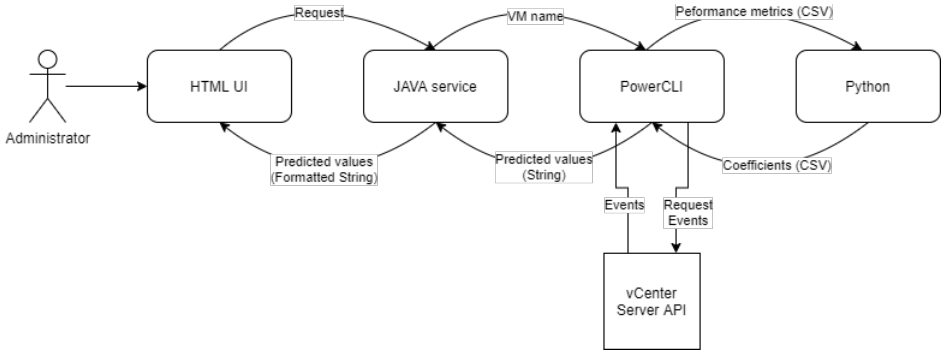

Figure 6.2: Integration with VMware Plugin Data-flow

Based on the integration of the above platforms and tools with VMware and the algorithms charts in Fig. 5.14 and Fig. 5.21, the steps we have used to integrate the proposed timing optimization algorithm with VMware vSphere are:

1. A shown in Fig. 6.2 The user interacts with the vSphere client UI, which contains the HTML layer added for the plugin, and requests the predicted migration overhead for a VM. This step is possible because the vSphere client software development kit allows developers to alter the user interface of the vSphere client to add new custom features.

2. The JAVA service layer receives the request from the UI layer and then forwards the request to the PowerCLI module.

3. The PowerCLI module gathers all the events run in the vCenter Server during the last 12 hours and filter the vMotion events. The script identifies the source and target hosts of each vMotion event and the start and end time stamp of each migration. Based on that, the performance statistics of each migration is collected and passed to the next step in CSV format.

4. The Python module then takes in the collected data set and for each pair of data items it calculates the coefficients. Once the script finds Two successive coefficients with less than 10% difference in value it stops the search and returns this value in a CSV format to the PowerCLI module.

5. The equations from (1), (2), (3) are then used to predict the duration, power consumption, and required network bandwidth of the migration.

6. These results are returned to the JAVA service layer as a string of characters and then returned to The HTML UI layer after formatting the string and shown to the user as in Fig. 6.3.

In the next section, we show the result of using these scripting tools and following the above steps to integrate the cost prediction and timing optimization algorithms with VMware vSphere UI.

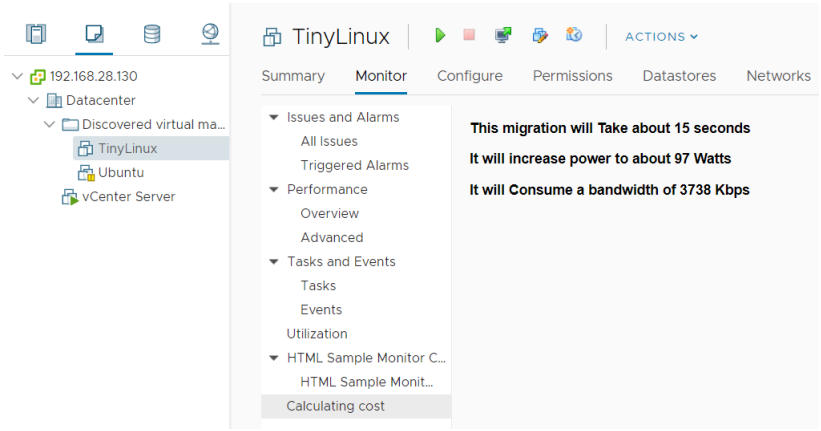

Figure 6.3: Added Icon: Cost Prediction Plug-in

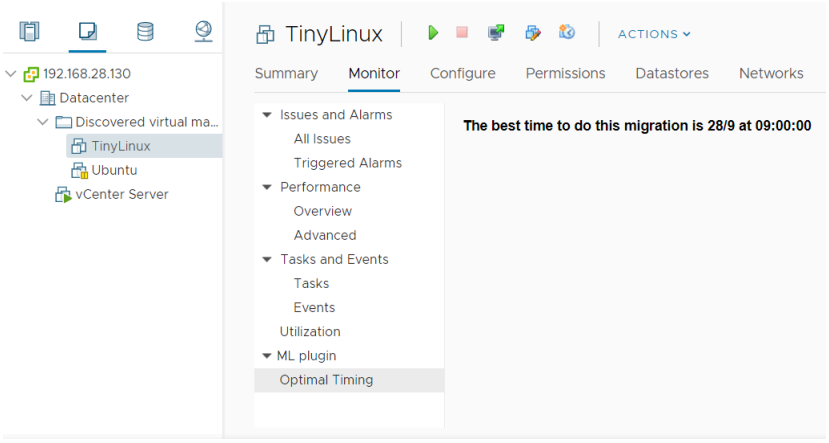

Figure 6.4: Added Icon: Optimal Timing Plug-in

6.4 Testing Results

The result of the integration with VMware vSphere UI is as shown in Fig. 6.3 and Fig. 6.4. As shown in Fig. 6.3, this newly added plugin in the UI of the VMware vSphere allow getting insights about the migration time, network rate and power consumption before initiating a VM migration and by using the data collected from the live migration history within the VMware cluster and with monitoring the VM active memory size. This allow the datacenter admins to make educated decisions about VMware vMotion events before committing the migration.

After determining the expected network rate of a VM live migration and using this integration method, it is also possible to create a more sophisticated migration system using the Hidden Markov model [139] in Fig. 5.21. Such a system would predict how the network bandwidth will change in the future and thus inform the datacenter administrator of the optimal time to do the migration in order to decrease the load on the network infrastructure. Fig. 6.4 shows how

Table 6.1: Timing Optimization Performance

	Mem Stress	**Net Stress**
Average Mig. Time %	145	126
Max. Mig. Time %	205	136

this system would look to the administrator. To implement such a system the following changes would be made:

- The PowerCLI module needs to periodically collect network data to be used by the Python module for training.
- the Hidden Markov model needs be implemented in the Python module. This model will be able to predict the network state in the future and thus guess the best time for a certain VM to migrate.
- When the user requests a migration, the UI will show the user the best predicted time to do the migration after running the model from the previous step.

As a result of using live migration cost prediction and timing optimization techniques, the live migration time of the VMs can be saved by just shifting the migration start time of the same VM to the recommended optimal time. Table 6.1 show average and the maximum migration time increase without using timing optimization as normalized values versus using the timing optimization techniques. As shown in Table 6.1, for memory stress benchmark the average increase in the migration time without timing optimization is 145% versus with using timing optimization. This average increase is 126% for network stress applications. From the peak increase in the migration time point of view, the maximum migration time increase for memory stress benchmark without timing optimization is 205% versus with using the timing optimization. This maximum increase is 136% for network stress benchmark. The results mentioned in Table 6.1 show how significantly the timing optimization can save the migration time cost for VMs live migrations.

Chapter 7

Live Migration Cost with Persistent Memory

Persistent memory is a recently developed technology that keeps the data stored at the memory tier even with system reboot or power off [6]. This technology is now supported by many hardware, operating systems, middle-ware and applications vendors[29].

7.1 Persistent Memory Technology Background

Memory intensive and in-memory applications are one of the main drives for persistent memory utilization. These applications use memory capacity intensively to gain high performance and low latency. However, the admins for these applications face challenges in large memory cost with DRAM and system reboot duration due to running a volatile memory. Persistent memory is a new tier of storage that is added between DRAM and standard flash SSDs; as shown in Fig. 7.1. This new byte-addressable tier of storage minimizes the bottlenecks in read and write operations between the DRAM and NAND SSDs due to the big gap in performance. So adding this new storage tier provides a more balanced data management for computer systems [6].

Persistent memory comes in two main silicon technologies; Non-Volatile DIMMs (NVDIMMs) or Non-Volatile RAM (NVRAM) and 3D xPoint Persistent Memory (PMem) which is 3D-xPoint based silicon technology released only by Intel [39] and Micron [15] in 2019. From the CPU prospective, Intel Xeon Cascade-lake CPU [3]; released in 2019 is the first and so far the only CPU that supports PMem; this is up to the date of writing this article.

7.1.1 Non Volatile DIMM (NVDIMMs)

NVRAM or NVDIMMs is based on the CMOS transistor that is used in the standard DDR4 RAM, however the memory architecture and power management ICs are modified to run with battery-backed power source that flushes the data from the DRAM to an internal NAND drive in order to have have data persistence during power failures or system restart [18], [19], [20] and [21]. So NVRAM has the same performance as the volatile DRAM but has the the

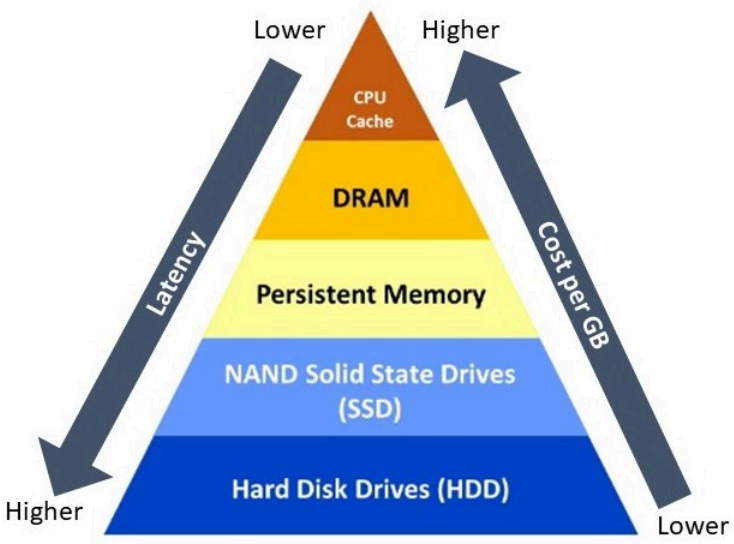

Figure 7.1: Storage Tiers Hierarchy with PMem [6]

endurance of NAND technology [101], [95] and [20]. NVDIMMs are supported by VMware vSphere 6.7 and above [17].

7.1.2 3D *x*Point PMem

3D xPoint is a new silicon technology developed by the memory vendors Intel and Micron [130]. Compared to NVRAM, 3D xPoint PMem has higher capacity density, lower cost per GB, higher durability and native persistent; without batteries [130]. From performance point of view, 3D xPoint PMem has comparable bandwidth and 10x more latency compared to NVRAM and DDR4 RAM. So, 3D xPoint PMem is not proposed as a replacement for DDR4 DRAM, but to be added as a new tier of memory between DRAM and the standard flash NAND SSDs as a higher capacity density and a more cost effective memory solution that provides also higher system availability[7].
Table 7.1 shows a comparison between DRAM, NVDIMMs and 3D xPoint PMem specifications. As shown; DRAM has advantages in latency and endurance, however the dis-advantages in DRAM are the low capacity per DIMM, volatile storage medium and the cost per GB. NVDIMMs has advantages in latency, and in data persistent, however the disadvantages in NVDIMMs are the low capacity per DIMM which is exactly as DRAM, the medium endurance and the cost per GB. The 3D xPoint PMem has advantage in the capacity per DIMM, the medium endurance, the data persistent and the lower cost per GB. However the disadvantages in 3D xPoint PMem is the higher latency compared to DRAM and NVDIMMs.

3D xPoint PMem has two modes of operations; Memory Mode (MM) and app direct mode (AppDir); as shown in Fig. 7.2. In MM, the PMem acts as a volatile memory and the only gains of using it are the capacity per DIMM and

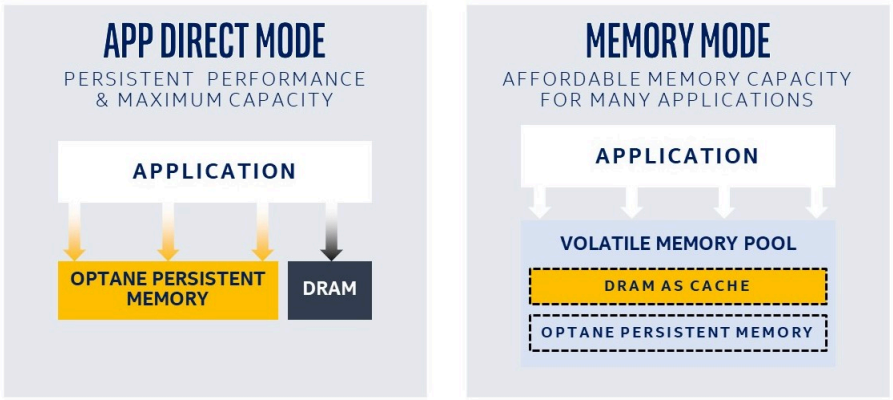

Figure 7.2: PMem AppDir Mode vs Memory Mode [25]

the endurance. In MM, there is no need to do any development in the hypervisor, the OS and the application to discover the PMem as the system discovers it as normal DRAM. In MM also, if there are PMem DIMMs as well as DDR4 DIMMs in the same server, the DDR4 DIMMs act as memory cache for the PMem DIMMs [20].

In AppDir mode, the PMem acts as persistent memory and in this case DDR4 and PMem are presented as two different types of memory to the hypervisors, the OS and the applications. So all these three software stacks should be developed to decide which part of the data should be stored in the PMem to utilize the larger capacity and persistent storage and which part should reside on the DDR4 to get the lower latency. PMem in both modes is supported by many applications (like SAP HANA, Apache Cassandra, Apache Spark SQL and Microsoft SQL server 2019), OSs vendors(like Windows Server 2019, RHEL 7.6, Ubuntu 18.10, CentOS* 7.6 and SLES* 12 SP4), hypervisors vendors (VMware* ESXi 6.7 EP 10 or later, Xen project and Microsoft Hyper-V 2019) and servers vendors (like Dell, Supermicro, Lenovo and HPE) [2] and [39].

For the AppDir mode presented in Fig. 7.2, we show how VMware vSphere discovers PMem in Fig. 7.3 as an example of a commonly used hypervisor that supports PMem starting with vSphere 6.7 release [47]. Fig. 7.3 is presented in [20] . As shown; for vSphere 6.7 and above, the hypervisor can actually recognize the PMem as a persistent memory device or as a very fast block storage disk; which is a memory class of storage. The decision of the PMem module representation is done using the server BIOS. For utilizing the PMem as a persistent memory, the logical persistent volume vNVDIMM should be used by the hypervisor to represent the PMem to the VM OS. For utilizing the PMem as a fast block storage, the logical volume vSCSI should be used to represent the PMem to the VM OS [2].

7.2 Live Migration with Persistent Memory

Live migration of VMs with PMem inside is supported by VMware vSphere, Microsoft Hyper-V and different Linux based hypervisors and operating systems

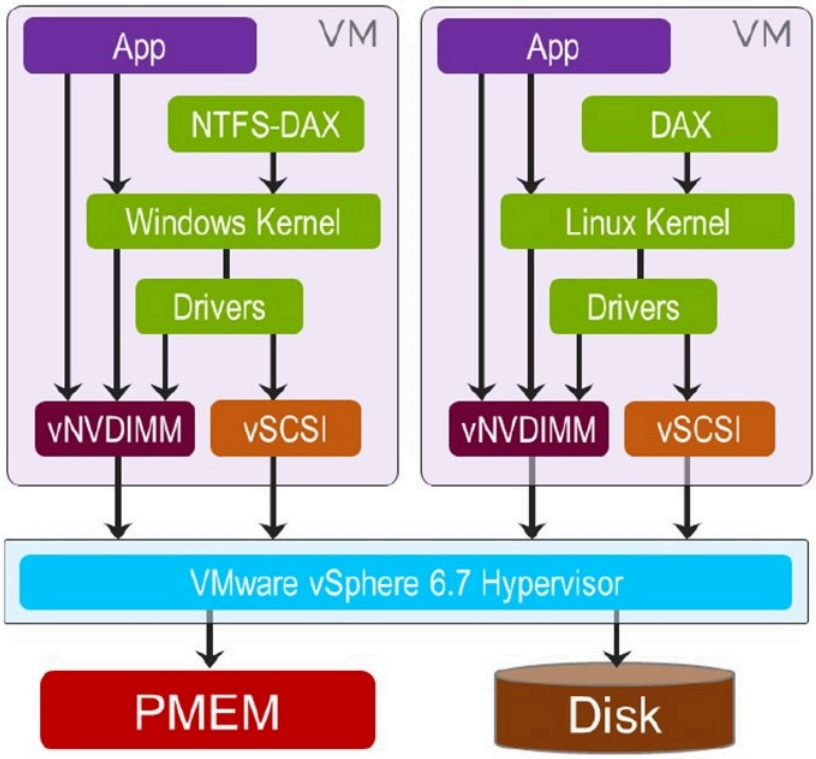

Figure 7.3: vSphere Configurations for PMem [20]

[24]. So for future research, it might be interesting to study if live migration cost modeling and cost prediction techniques there were studied on the DDR4 DRAM will be valid also for PMem as a new memory technology. In the this section, we focus on the live migration cost comparison between DRAM only servers and the PMem inside servers.
VMs can use PMem in a standalone server or in a cluster configuration. In App Direct Mode of Intel Optane or in NVM-DIMMs configuration, a local datastore is created to represent the PMem content. For each VM, the PMem required allocation is reserved whatever the VM is powered ON or Off. VM migration or deletion are the only situations when the PMem content of a VM is removed.
As shown in Fig. 7.3, as a persistent storage PMem can be represented to the OS as Virtual PMem (vPMEM) in byte addressable random access using the vNVDIMM driver. Or PMem can be represented as Virtual Disk (vPMemDisk) using the virtual SCSI (vSCSI) device. In both cases, the persistent memory is shown by VMware vSphere as a new datastore. This means that when a VM with vPMEM or vPMemDisk is live migrated to another host in a VMware cluster, both vMotion and storage vMotion requests should be initiated to consider the VM compute and persistent memory storage migration. An important consideration here is; a VM with vPMem can only be migrated to another host with vPMEM. However, a VM with vPMemDisk can be migrated to another host without a PMem as the local storage drive can takeover the vPMemDisk

data [26].
As shown in Fig. 7.2, PMem can be represented also as a volatile storage device; same as the standard DRAM. In this case, the DRAM DIMMs are handled by the system as memory cache for the PMem DIMMs and handling the cache read and write operation is managed by the CPU memory controller. This means that the write operations are always forwarded toward the DRAM DIMMs; which make the write operation has the performance as the DRAM only host. The read operation can be fetched from the DRAM DIMMs or from the PMem DIMMs depending on the cache hit or the cache miss operations. The memory controller firstly checks the DRAM Cache, if cache hit happens the response latency will be identical to DRAM. If cache miss happens, the read operation will face higher latency due to the read from the PMem DIMMs [10].
Live migration of VMs with PMem in memory mode is supported between hosts with and without PMem [84]. In this case, the memory content is read from the source host DRAM or PMem depending on the cache-hit or cache-miss operation and the migrated content is always written to the the target host DRAM.

Table 7.2 summarizes the supported live migration configurations between VMware cluster servers with DRAM, Intel Optane PMem in Memory Mode (MM) and in App Direct (AD) mode. As shown; live migration is always supported except from PMem - vNVDIMM in AD modes servers to PMem in MM or to DRAM only servers. Also, for AD mode, both compute and storage migrations should be requested to include the PMem content migration as well.

7.3 Related Work

To the best of our knowledge, live migration cost modeling for clusters with PMem is not covered yet by any of the research articles. Live migration with Non-volatile or persistent memory based clusters is basically covered in [64], [126] and [118] as following. In [64], [119], Microsoft research team provides a hardware architecture and develop a Byte-addressable Persistent File System (BPFS) that runs on a Phase Change Memory (PCM) which is a persistent memory technology. The proposed architecture is compared to NTFS file system on disks using four different benchmarks; Microbenchmarks, throughput benchmark, Apache benchmark and Patch benchmark. Results show that for the four benchmarks that were used consequently, BPFS outperforms NTFS-Disk systems in files creation time, read/write operations per second, directories tree build time and in file pattern re-write delay [64].
A new VM pre-copy live migration method is proposed in [126] for distributed edge servers that utilize DRAM and PM in the main memory. The main objective of the proposed technique is to decrease the memory pages sent and optimizes the page placement in destination hybrid memory systems by using memory pages classification to hot and cold pages. Cold pages are transferred to the persistent memory of the target server and the hot pages are saved more time in the source host transmission queues and finally transferred to the target host DRAM. The proposed technique results in less total transferred bytes, less number of migration copy iterations and so less total migration time [126]. Authors in [118] propose a multi-site synchronous VM replication technique that is is optimized for rack based memory centric architecture. In this architecture,

Table 7.1: Comparison between Non-Volatile Memory Technologies

Mem. Tech.	**Max. Cap. per DIMM**	**Latency (ns)**	**Endur.**	**Silicon Tech**	**Supplier**	**Cost /GB**
SDDRAM	128 GB	10's	CMOS	CMOS	Samsung	Medium
NV-DIMM	128 GB	10's	NAND	CMOS and NAND	Everspin	Highest
3D xPoint PMem	512 GB	100's	3D xPoint	3D xPoint	Intel/Micron	Lowest

Table 7.2: Supported Live Migration Configurations with PMem for VMware Platforms

Oper. Mode	**Source Host**	**Target Host**	**Valid?**	**Compute or storage Migration?**	**Notes**
MM	DRAM Only	PMem-MM	Yes	Compute	Reads from DRAM and writes to DRAM
MM	PMem-MM	DRAM Only	Yes	Compute	Reads from DRAM or PMem and writes to DRAM
MM	PMem-MM	PMem-AD	Yes	Compute	The Application decides which data should be stored at the target host DRAM and data at the PMem
MM	DRAM Only	PMem-AD	Yes	Compute	The VM does not use PMem-AD Mode as it is in MM Mode at the source host
MM	PMem-MM	PMem-MM	Yes	Compute	-
AD	PMem-AD	PMem-AD	Yes	Both	-
AD	PMem-AD	PMem-MM	No	Both	Supported only if the PMem is used as vPMemDisk
AD	PMem-AD	DRAM Only	No	Both	Supported only if the PMem is used as vPMemDisk

Remote Direct Memory Access (RDMA) is used to achieve a zero copy manner for VMs migration from a host to another; since they share the same centric memory. Compared to the common asynchronous VM replication, the proposed synchronous VMs replication in [118] increased the performance by 10% in a commodity DRAM-based system, and up to 27% for emulated prospective NV-RAM-based systems.
In [119] a distributed shared persistent memory system is proposed. The proposed architecture is called Hotpot that provides direct access to a shared persistent memory. The proposed architecture assures data durability, reliability and availability using two distributed data commit protocols with different consistency levels and corresponding recovery protocols. Two datacenters test-beds were used to demonstrate the proposed system performance and easy of implementation. Microbenchmark is used with five existing file systems and distributed memory systems. Testing results show that the proposed hotpot architecture achieves higher performance compared to other distributed persistent memory based systems like Octopus [105], Mojim [136], Persistent Memory File System (PMFS) [67] and temporary file system (tmpfs) [43].

7.4 Migration Cost of DRAM versus Persistent Memory

As discussed in the last section, live migration performance for VMs with PMem in MM depends on the read operations from the source host cache hit or cache miss. The data is always written to the target host DRAM as a cache tier of the PMem in MM. The read and write operation is running intensively during the iterative copy phase of the live migration process.
In this section, we compare the live migration cost between servers that have only DRAM versus the servers that have PMem in MM. The next section describes the testing environment.

7.4.1 Testing Environment

To test the VMs live migration cost with DRAM versus PMem in Memory Mode, we have built the test-bed in Fig. 7.4 with the below configuration:

- Four servers: two servers with DRAM and two other servers with PMem
- The four servers have: Dual socket Intel Xeon Gold 6252N CPU with 24 core/socket, two 25Gb NICs, One Intel Optane SSDs with 750 GB and two Intel SATA SSDs with 3.2 TB.
- For the two DRAM servers: each server has 2048 GB (32x 64GB DIMMs) of DDR4 DRAM
- For the two PMem servers: each server has 384 GB (6x 64GB DIMMs) of DDR4 DRAM and 1792 GB (14x 128 GB) of PMem in MM. The specifications difference between DDR4 DRAM DIMMs and Intel Optane PMem DIMMs that we have used in this test is summarized in Table 7.3 [92]
- The four servers are in the same vCenter Server cluster and share the same VMware vSAN data-store.

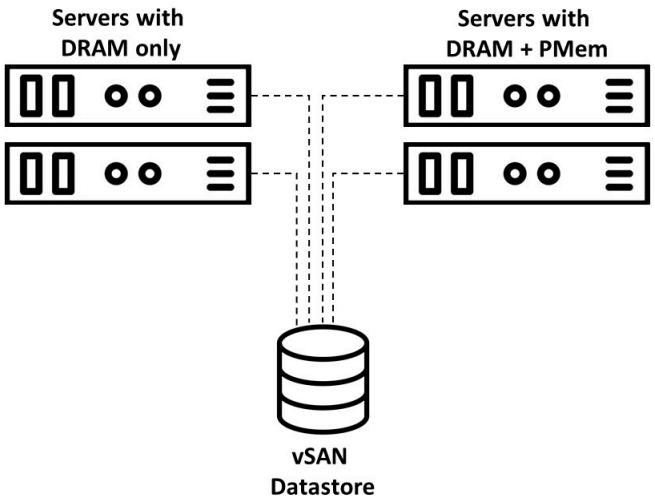

Figure 7.4: PMem Live Migration Cost Test-bed

Table 7.3: DDR4 DRAM vs PMem Specifications [92]

	DDR4 DRAM	Optane PMem
Module Capacity	64 GB	128 GB
Round-trip Latency	81 ns	305 ns
Read Bandwidth	110 GB/s	35 GB/s
Write Bandwidth	80 GB/s	10 GB/s

- VMs with different memory sizes are created to be live migrated between the different servers. The VMs configurations are 8GB, 16GB and 32GB. The workload used in the VMs migration testing is CPU and memory stress; which represents the worst case scenario for a running VM.
- Using these 4 servers, live migration of the VMs is running also with different servers types to show the migration cost difference based on the source and target servers memory types; as presented in Table 7.4.

7.4.2 Testing Results

Live migration testing with the configurations and scenarios that were discussed in the previous section have led to the results that we discuss in this section. The migration cost parameters that we evaluate are the migration time, the network overhead and the peak power cost.

Table 7.4: Source and Target Hosts Migration Scenarios

Scenario Number	Source Host	Target Host	Notes
1	DRAM Only	DRAM Only	-
2	DRAM Only	DRAM+PMem in MM	Target host writes to DRAM
3	DRAM+PMem in MM	DRAM+PMem in MM	Source host reads from DRAM or PMem but the target host writes to DRAM
4	DRAM+PMem in MM	DRAM Only	Source host reads from DRAM or PMem

Table 7.5: PMem in MM vs DRAM only - Migration Time Comparison

Scenario Number	4 GB	8 GB	16 GB	32 GB
1	6	13	21	42
2	5	14	19	41
3	9	17	34	60
4	8	16	33	59

7.4.2.1 Migration Time Cost

As discussed before, the VMs hosted at servers with PMem in MM has two tiers of memory. The DRAM tier acts as a memory cache for the PMem and the PMem tier that has the main system memory capacity. So, when the VMs are migrated from a server to another there are different migration scenarios depending on the source and target hosts memory configurations; as explained in Table 7.4. These different migration scenarios lead to different cost and overhead of these VMs live migration. Fig. 7.5 shows the migration time cost of the different migration scenarios mentioned in Table 7.4 and with different VMs memory capacities. Each VM is stressed by a CPU and memory stress benchmark using the stress benchmark of the Linux systems [14]. As shown in Fig. 7.5, the higher the VM memory size, the longer migration time consumed to move VMs from a host to another. It is also obvious from the results that scenarios 3 and 4 take longer migration time compared to scenarios 1 and 2. For the 4 different memory sizes of the VM, scenarios 1 and 2 show almost the same migration time and scenarios 3 and 4 have also almost the same migration time. The reason is in scenarios 1 and 2, the read and write operations of the live migration process are from the source host DRAM to the target host DRAM. However, in scenarios 3 and 4, the write operation is to the DRAM but the read operation might be from the PMem or from the DRAM; depending on the memory cache hit or cache miss events. Reading from the PMem has lower performance compared to reading from the DRAM due to the lower hardware specifications of the PMem compared to DRAM; as listed in Table 7.3. The gap in the migration time between scenarios 1 and 2 versus scenarios 3 and 4 depends on the VM memory size. Table 7.5 shows that for the same VM memory size, the gap between the average migration time with PMem can be increased with up to 73% if we consider the min. value of the DRAM only versus the min. of the PMem inside migration times.

Table 7.6: PMem vs DRAM only - Network Overhead Comparison (kBps)

Scenario Number	4 GB	8 GB	16 GB	32 GB
1	239213	466075	879171	999070
2	251352	460081	913160	1011184
3	227493	412096	658666	606394
4	234580	409406	691536	615810

7.4.2.2 Network Overhead

In this test a separate management network with 10 Gbps Ethernet network is used management and live migration network traffic. For VMware vSAN another 25 Gbps network is used. From memory overhead point of view, scenarios 1 and 2 could utilize higher memory throughput due to the higher read bandwidth using the DRAM at the source hosts. However in scenarios 3 and 4, PMem might be used in the read process in the event of memory cache miss. The higher bandwidth of the DRAM, leads to higher migration rate compared to using PMem as the source memory of the migration process. As presented in Fig 7.6, the network rate variations with the different scenarios is mapped with the migration times, such that the longer the migration time, the lower network rate.

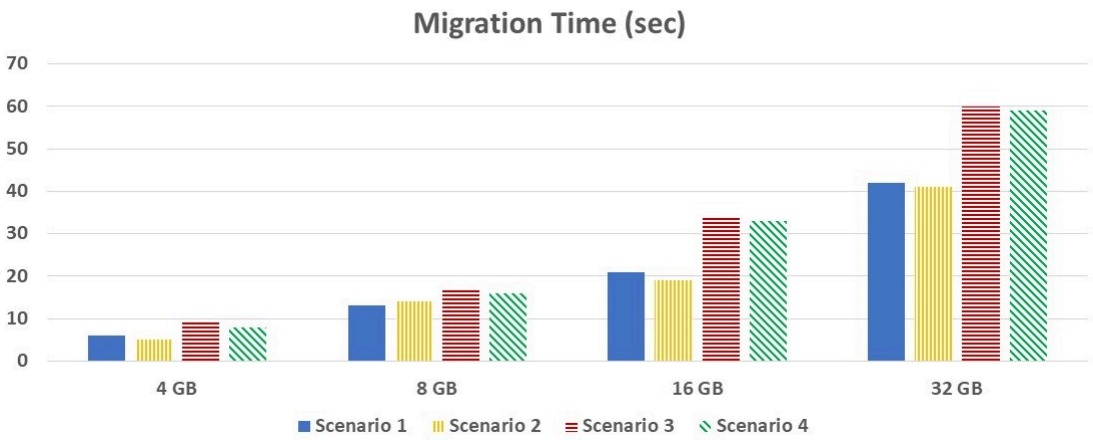

Figure 7.5: PMem vs DRAM Live Migration Time

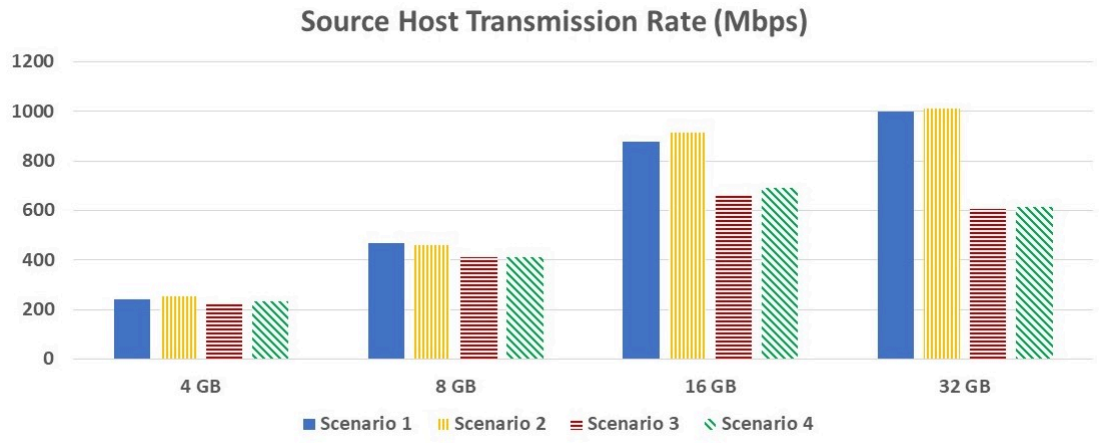

Figure 7.6: PMem vs DRAM Live Migration Network Overhead

Chapter 8

Conclusion

Live migration of virtual machines is an essential feature on virtual datacenters and cloud computing environments. Dynamic resource management, load balance, power saving and fault tolerance of the cloud VMs depends on live migration. The cost of live migration includes the migration time, down time, network overhead and power consumption increase. This cost can not be ignored and so in this research, we answer some of the open research questions about dynamic resource management in general and also about live migration cost modeling, prediction and live migration of VMs with persistent memory inside.

In this thesis, we start with a discussion about the concept of datacenters storage, network and compute resources dynamic management. Then we focus on live migration of virtual machines as the compute resources dynamic management in virtual datacenters and cloud computing environments. Because live migration cost is covered by many other research articles, we provide a detailed literature review for the related work that includes tens of the related papers. We could summarize the related word in a table that lists the contribution by each paper and the differences between these papers focus and methodology. We discussed also how the proposed work in this thesis is different compared to other research work; either in modeling, prediction, timing optimization or in the PMem migration test comparisons.

Using VMware test-beds, we show live migration cost modeling for VMware environment for single and multiple VMs migration. The empirical models that we propose make use of the regression techniques to formulate the relationship between the active memory of the VMs and the migration times of these VMs as well as the migration network throughput and the peak power consumption. Based on the proposed cost modeling formulas of live migration, we proposed also a prediction technique for live migration cost of VMs that helps the IT admins to get an estimation about the VMs live migration before proceeding with migration start. The proposed prediction algorithm shows a simple technique that has acceptable accuracy with maximum 13% error in prediction and agility to be integrated with VMware administration portals; which make it a practical solution for the IT admins to predict live migration cost. We show in this thesis, how the proposed prediction technique can be integrated with VMware cluster administration portal using HTML and PowerCLI scripting tool. The objective beyond adding the integration with VMware administration portal step is to

show to the readers the simplicity and agility to integrate the proposed algorithms with VMware portals and to practically use the proposed techniques.
In addition to that, we propose in this thesis a novel technique for live migration timing optimization that can be used to recommend a VM live migration optimal timing for the IT admins depending on the datacenter network prediction and using our proposed cost prediction algorithm. The proposed timing optimization technique results in higher migration network throughput which minimizes the migration time of the VMs. The proposed timing optimization technique saves up to 27% of the migration time and on average it saves 21% of the migration time for network intensive applications. For memory stress workload, the proposed technique saves up to 51% of the migration time and on average it saves 27% of the migration time.
During the last few years, persistent memory is released for production by different servers vendors and suppliers to offer four times higher memory density per DIMM as well as the data storage persistence. Persistent memory and high memory density should be useful for memory intensive applications like data analytics platforms and in-memory database. In this thesis, we discuss also the non-volatile and persist memory technology concept in details. We present the memory mode and the persistent mode configurations of the PMem and how virtual machines memory can have different scenarios of migration.
We have built a VMware cluster using four servers; two with DRAM plus PMem in memory mode and the other two with DRAM only for the purpose of live migration cost comparison. The results show that for the servers with PMem, always the write operations goes to the memory cache; which is the DRAM. However the read operations might be from the DRAM or from the PMem; depending on the cache hit or cache miss events. So, in the scenarios of migration from servers with PMem, the migration time looks higher than migration from DRAM only servers due to the lower bandwidth of the PMem DIMMs compared to the DDR4 DRAM DIMMs. The migration time gap depends also on the VM memory size, so the larger the VM size, the bigger migration time cost gap between PMem servers versus the DRAM only ones. Our test used VMs with 4, 8, 16 and 32 GB memory VMs, and the testing results with CPU and memory stress benchmark show that migration time cost can be increased with up to 73% more time compared to the same live migration from DRAM only servers. If the servers with PMem are the target servers, the write operation is always to the DRAM still since the DRAM acts as the server memory cache.

Chapter 9

Future Work

As an extension work to the research presented in this thesis, there are different topics of interest that we aim to be studied. Firstly, the live migration cost comparison between VMs with PMem in App Direct mode inside versus the VMs with PMem in Memory Mode and versus the VMs with DRAM only inside. This migration cost comparison between the different memory configurations should include also different other benchmarks with applications that are PMem in App Direct mode aware; such as Apache Spark, SQL and SAP HANA.
The second topic that can be studied in the future is the storage migration of VMs cost modeling, prediction and timing optimization; same as we did in this thesis with the compute migration of the VMs. The point is, in VMware when we migrate a VM the admin is asked firstly if it should be only a VM compute migration or a storage migration or both of them. In this thesis, we always use just compute migration. So as a second research topic in the future work, we talk about considering storage only migration and also both compute and storage migration of the VMs.
The third topic of interest also in the future work is to study the MAN/ WAN scale live migration cost modeling and prediction to see if the proposed LAN based live migration that we have studied is applicable also on the large MAN/WAN scale. This kind of large distance migration is very useful to study especially for public cloud customers who normally have their environment VMs distributed across different zones and regions across different locations in the globe. So it will be important for these cloud admins to know the cost of any VM that they plan to do before proceeding with this kind of large distance migration to avoid migration failures and minimize the live migration cost.

Bibliography

[1] AWS S3 Data Protection (Jan. 2021): https://docs.aws.amazon.com/AmazonS3/latest/dev/DataDurability.html.

[2] SNIA Persistent Memory Summit 2020 (April 2020): https://www.snia.org/pm-summit.

[3] Intel Cascadelake CPU (May 2019): www.intel.com/content/www/us/en/design/products-and-solutions/processors-and-chipsets/cascade-lake/2nd-gen-intel-xeon-scalable-processors.html.

[4] Cisco wan (oct. 2013): http://www.cisco.com/c/en/us/td/docs/nsite/enterprise/wan/wan_optimization/wan_opt_sg/chap06.html.

[5] Storage Virtualization in Microsoft Hyper-V (Oct. 2016): https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831656(v=ws.11).

[6] Programming with Intel Persistent Memory (August 2017): https://software.intel.com/content/www/us/en/develop/articles/introduction-to-programming-with-persistent-memory-from-intel.html, .

[7] Persistent Memory Technology value proposition for Database Applications (July 2019): www.intel.com/content/www/us/en/architecture-and-technology/optane-pmem-redis-enterprise-ra.html, .

[8] Hyper-V Live Migration (August 2016): https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/.

[9] Linpack Benchmark (Sept. 2010): www.netlib.org/linpack/.

[10] PMem in Memory Mode (June 2020): https://www.intel.com/content/dam/support/us/en/documents/memory-and-storage/data-center-persistent-mem/Intel-Optane-DC-Persistent-Memory-Quick-Start-Guide.pdf.

[11] Microsoft Storage Redundancy (Oct. 2021): https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy.

[12] SDN for Microsoft Hyper-V (Nov. 2017): https://docs.microsoft.com/en-us/system-center/vmm/deploy-sdn?view=sc-vmm-2019.

[13] Matlab (June 2014): https://www.mathworks.com/products/matlab.html.

[14] Memory Stress Benchmark (March 2014): https://launchpad.net/ubuntu/bionic/+package/stress/.

[15] Micron Persistent Memory Technology (May 2020): www.micron.com/campaigns/persistent-memory.

[16] Xen Live Migration (May 2021): https://docs.citrix.com/en-us/xenserver/7-0/downloads/administrators-guide.pdf.

[17] NVDIMMs Support for VMware vSphere (Sept.2020): https://core.vmware.com/resource/intel-optane-dc-persistent-memory-memory-mode-virtualized\-performance-study#_Toc29011, .

[18] Dell R740 server with NVDIMM (July 2017): https://www.delltechnologies.com/asset/en-us/products/servers/industry-market/direct_from_development_poweredge_r740_sql_server_performance_with_nvdimms.pdf, .

[19] Dell R940 server with NVDIMM (Feb. 2021): https://dl.dell.com/topicspdf/nvdimm_n_user_guide_en-us.pdf, .

[20] HPE NVDIMM (Jan. 2021): www.hpe.com/us/en/servers/persistent-memory.html, .

[21] NVDIMMs Support for HPE Servers (March 2018): https://support.hpe.com/hpesc/public/docDisplay?docId=a00038965en_us&docLocale=en_US, .

[22] Network Benchmark (May 2014): https://httpd.apache.org/docs/2.4/programs/ab.html, .

[23] Network Migration (March 2021): https://docs.vmware.com/en/VMware-NSX-T-Data-Center/3.1/migration/GUID-0161835B-EB06-4944-8CF4-46A94AE9E322.html, .

[24] PMem OS Support (July 2020): https://www.intel.com/content/www/us/en/support/articles/000032860/memory-and-storage/data-center-persistent-memory.html, .

[25] PMem Modes (Dec. 2019): https://downloads.dell.com/manuals/common/direct_from_development_-_intel_optane_dc_persistent_memory_module.pdf, .

[26] PMem Migrations (April 2021): https://docs.vmware.com/en/vmware-vsphere/7.0/com.vmware.vsphere.resmgmt.doc/guid-eb72d358-9c2c-4fbd-81a9-a145e155ce31.html.

[27] VMware PowerCLI 6.5 (March 2017):. https://code.vmware.com/fr_FR/tool/vmware-powercli/6.5.

[28] Redhat Migration (June 2021): https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.0/html/virtual_machine_management_guide/sect-migrating_virtual_machines_between_hosts.

[29] SNIA Persistent Memory Introduction (June 2019): https://www.snia.org/education/what-is-persistent-memory, .

[30] SNIA Virtual Storage (June 2011): https://www.snia.org/sites/default/education/tutorials/2011/fall/VirtualizationApplications/WaltHubis_Virtualization_I.pdf, .

[31] Storage Virtualization in VMware (May 2019): https://docs.vmware.com/en/VMware-vSphere/6.0/com.vmware.vsphere.storage.doc/GUID-05BA0A0B-DF24-4BF8-B375-FD6BDB9073CB.html.

[32] VMware VMkernel (Oct. 2020): https://kb.vmware.com/s/article/2054994.

[33] SDN for VMware vSphere (May 2021): https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmware-nsx-datasheet.pdf, .

[34] VMware vMotion (Nov. 2019): https://docs.vmware.com/en/vmware-vsphere/7.0/com.vmware.vsphere.vcenterhost.doc/guid-5211fd4b-256b-4d9f-b5a2-f697a814bf64.html, .

[35] EMC VNX (March 2016): https://www.delltechnologies.com/en-us/collaterals/unauth/white-papers/products/storage/h12197-vnx-storage-efficiency-wp.pdf.

[36] Xen Virtual Switch (May 2021): https://docs.citrix.com/en-us/citrix-hypervisor/technical-overview.html.

[37] VMware Network Guide (Feb. 2017): https://docs.vmware.com/en/vmware-vsphere/5.5/vsphere-esxi-vcenter-server-552-networking-guide.pdf.

[38] What is AI? (June 2020): https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

[39] Intel Persistent Memory Technology Partners (May 2021): http://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html.

[40] KVM Hypervisor (June 2021): https://libvirt.org/drvqemu.html.

[41] Redis Database (April 2021): https://redis.io/.

[42] Apache Spark (May 2021): https://spark.apache.org/.

[43] Active Memory (March 2021): https://www.kernel.org/doc/html/latest/filesystems/tmpfs.html.

[44] VMware vCenter Server: (April 2021), www.vmware.com/products/vcenter-server.html, note = Accessed: 2010-09-30.

[45] VMware virtual Distributed Switch (Feb. 2021): https://www.vmware.com/products/vsphere/distributed-switch.html.

[46] Hyper-V Switch (August 2020): https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v-virtual-switch/hyper-v-virtual-switch.

[47] vSphere Support for Persistent Memory (April 2020): https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.storage.doc/GUID-93E5390A-8FCF-4CE1-8927-9FC36E889D00.html.

[48] N. Akhter and M. Othman. Energy efficient virtual machine provisioning in cloud data centers. In *2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT)*, pages 330–334, 2014.

[49] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Predicting the performance of virtual machine migration. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '10, pages 37–46, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4197-6. doi: 10.1109/MASCOTS.2010.13. URL https://doi.org/10.1109/MASCOTS.2010.13.

[50] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Predicting the performance of virtual machine migration. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '10, pages 37–46, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4197-6.

[51] Mohammad Aldossary and Karim Djemame. Performance and energy-based cost prediction of virtual machines live migration in clouds. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*, pages 384–391, 2018. doi: 10.5220/0006682803840391. URL https://doi.org/10.5220/0006682803840391.

[52] S. Baek, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim. Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 205–210, June 2017. doi: 10.1109/CSCloud.2017.26.

[53] A. Bashar, N. Mohammad, and S. Muhammed. Modeling and evaluation of pre-copy live vm migration using probabilistic model checking. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–7, 2018.

[54] J. L. Berral, R. Gavaldà, and J. Torres. Power-aware multi-data center management using machine learning. In *2013 42nd International Conference on Parallel Processing*, pages 858–867, Oct 2013. doi: 10.1109/ICPP.2013.102.

[55] Josep Ll. Berral, Ricard Gavaldà, and Jordi Torres. Power-aware multi-data center management using machine learning. In *Proceedings of the 2013 42Nd International Conference on Parallel Processing*, ICPP '13, pages 858–867, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-5117-3. doi: 10.1109/ICPP.2013.102. URL `http://dx.doi.org/10.1109/ICPP.2013.102`.

[56] P. Bezerra, G. Martins, R. Gomes, F. Cavalcante, and A. Costa. Evaluating live virtual machine migration overhead on client's application perspective. In *2017 International Conference on Information Networking (ICOIN)*, pages 503–508, Jan 2017. doi: 10.1109/ICOIN.2017.7899536.

[57] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128, 2007. doi: 10.1109/INM.2007.374776.

[58] Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, Jun 2018. ISSN 1869-0238. doi: 10.1186/s13174-018-0087-2. URL `https://doi.org/10.1186/s13174-018-0087-2`.

[59] Walter Cerroni. Multiple virtual machine live migration in federated cloud systems. *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 25–30, 2014.

[60] Walter Cerroni. Network performance of multiple virtual machine live migration in cloud federations. *J. Internet Serv. Appl.*, 6(1):6:1–6:20, 2015. doi: 10.1186/s13174-015-0020-x. URL `https://doi.org/10.1186/s13174-015-0020-x`.

[61] Zeroual A Chabaa S and Antari J. Identification and prediction of internet traffic using artificial neural networks. In *Journal of Intelligent Learning Systems and Applications*, volume 2, pages 147–155, Nov 2010. doi: 10.4236/jilsa.2010.23018.

[62] Y. Chen, I. Liu, C. Chou, J. Li, and C. Liu. Multiple virtual machines live migration scheduling method study on vmware vmotion. In *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, pages 113–116, 2018.

[63] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1251203.1251223`.

[64] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. Better i/o through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS*

22nd Symposium on Operating Systems Principles, SOSP '09, page 133–146, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587523. doi: 10.1145/1629575.1629589. URL https://doi.org/10.1145/1629575.1629589.

[65] Waltenegus Dargie. Estimation of the cost of VM migration. In *23rd International Conference on Computer Communication and Networks, ICCCN 2014, Shanghai, China, August 4-7, 2014*, pages 1–8. IEEE, 2014. doi: 10.1109/ICCCN.2014.6911756. URL https://doi.org/10.1109/ICCCN.2014.6911756.

[66] Rui Ding, Ziyu Li, and Jie Yin. *A Replica Storage Optimal Method for HDFS: Based on SVM*, page 1–5. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450375382. URL https://doi.org/10.1145/3416921.3416923.

[67] Subramanya R. Dulloor, Sanjay Kumar, Anil Keshavamurthy, Philip Lantz, Dheeraj Reddy, Rajesh Sankaran, and Jeff Jackson. System software for persistent memory. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327046. doi: 10.1145/2592798.2592814. URL https://doi.org/10.1145/2592798.2592814.

[68] K. Elghamrawy, D. Franklin, and F. T. Chong. Predicting memory page stability and its application to memory deduplication and live migration. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 125–126, 2017.

[69] M. E. Elsaid and C. Meinel. Live migration impact on virtual datacenter performance: Vmware vmotion based study. In *2014 International Conference on Future Internet of Things and Cloud*, pages 216–221, Aug 2014. doi: 10.1109/FiCloud.2014.42.

[70] M. E. Elsaid and C. Meinel. Friendship based storage allocation for online social networks cloud computing. In *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, pages 1–6, 2015. doi: 10.1109/CloudTech.2015.7336970.

[71] M. E. Elsaid and C. Meinel. Multiple virtual machines live migration performance modelling – vmware vmotion based study. In *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 212–213, April 2016. doi: 10.1109/IC2E.2016.9.

[72] M. E. Elsaid, A. Shawish, and C. Meinel. A social aware approach for online social networks data allocation and replication. In *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, pages 266–271, 2017. doi: 10.1109/EECS.2017.56.

[73] M. E. Elsaid, A. Shawish, and C. Meinel. Enhanced cost analysis of multiple virtual machines live migration in vmware environments. In *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*, pages 16–23, 2018. doi: 10.1109/SC2.2018.00010.

[74] Mohamed Elsaid., Hazem Abbas., and Christoph Meinel. Live migration timing optimization for vmware environments using machine learning techniques. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 91–102. INSTICC, SciTePress, 2020. ISBN 978-989-758-424-4. doi: 10.5220/0009397300910102.

[75] Mohamed Esam Elsaid, Hazem M. Abbas, and Christoph Meinel. Machine learning approach for live migration cost prediction in vmware environments. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science, CLOSER 2019, Heraklion, Crete, Greece, May 2-4, 2019*, pages 456–463, 2019. doi: 10.5220/0007749204560463. URL `https://doi.org/10.5220/0007749204560463`.

[76] Mohamed Esam Elsaid, Mohamed Sameh, Hazem M. Abbas, and Christoph Meinel. Live migration timing optimization integration with vmware environments. In Donald Ferguson, Claus Pahl, and Markus Helfert, editors, *Cloud Computing and Services Science*, pages 133–152, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72369-9.

[77] P. Tröger F. Salfner and A. Polze. Downtime analysis of virtual machine live migration. In *The Fourth International Conference on Dependability (DEPEND 2011), ISBN: 978-1-61208-149-6, pages: 100-105, French Riviera, France.*

[78] Fahimeh Farahnakian, Pasi Liljeberg, and Juha Plosila. LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, sep 2013. doi: 10.1109/seaa.2013.23. URL `https://doi.org/10.1109%2Fseaa.2013.23`.

[79] D. Fernando, J. Terner, K. Gopalan, and P. Yang. Live migration ate my vm: Recovering a virtual machine after failure of post-copy live migration. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 343–351, April 2019. doi: 10.1109/INFOCOM.2019.8737452.

[80] Tiago C. Ferreto, Marco A. S. Netto, Rodrigo N. Calheiros, and César A. F. De Rose. Server consolidation with migration control for virtualized data centers. *Future Gener. Comput. Syst.*, 27(8):1027–1034, October 2011. ISSN 0167-739X. doi: 10.1016/j.future.2011.04.016. URL `https://doi.org/10.1016/j.future.2011.04.016`.

[81] Pablo Graubner, Matthias Schmidt, and Bernd Freisleben. Energy-efficient virtual machine consolidation. *IT Professional*, 15(2):28–34, March 2013. ISSN 1520-9202. doi: 10.1109/MITP.2012.48. URL `https://doi.org/10.1109/MITP.2012.48`.

[82] T. Gupta, J. Ganatra, and K. Samdani. A survey of emerging network virtualization frameworks and cloud computing. In *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 14–15, Jan 2018. doi: 10.1109/CONFLUENCE.2018.8442995.

[83] Koji Hasebe, Tatsuya Niwa, Akiyoshi Sugiki, and Kazuhiko Kato. Power-saving in large-scale storage systems with data migration. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM '10, page 266–273, USA, 2010. IEEE Computer Society. ISBN 9780769543024. doi: 10.1109/CloudCom.2010.105. URL `https://doi.org/10.1109/CloudCom.2010.105`.

[84] Mark Hildebrand, Julian T. Angeles, Jason Lowe-Power, and Venkatesh Akella. A case against hardware managed dram caches for nvram based systems. In *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 194–204, 2021. doi: 10.1109/ISPASS51385.2021.00036.

[85] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh, and S. Sekiguchi. A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 460–465, 2009. doi: 10.1109/CCGRID.2009.44.

[86] Y. Hsu, R. Irie, S. Murata, and M. Matsuoka. A novel automated cloud storage tiering system through hot-cold data classification. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 492–499, 2018. doi: 10.1109/CLOUD.2018.00069.

[87] Bolin Hu, Zhou Lei, Yu Lei, Dong Xu, and Jiandun Li. A time-series based precopy approach for live migration of virtual machines. In *Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems*, ICPADS '11, pages 947–952, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4576-9.

[88] Liang Hu, Jia Zhao, Gaochao Xu, Yan Ding, and Jianfeng Chu. Hmdc: Live virtual machine migration based on hybrid memory copy and delta compression, 2013.

[89] Wenjin Hu, Andrew Hicks, Long Zhang, Eli M. Dow, Vinay Soni, Hao Jiang, Ronny Bull, and Jeanna N. Matthews. A quantitative study of virtual machine live migration. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 11:1–11:10, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2172-3. doi: 10.1145/2494621.2494622. URL `http://doi.acm.org/10.1145/2494621.2494622`.

[90] Q. Huang, F. Gao, R. Wang, and Z. Qi. Power consumption of virtual machine live migration in clouds. In *2011 Third International Conference on Communications and Mobile Computing*, pages 122–125, 2011.

[91] Qingjia Huang, Kai Shuang, Peng Xu, Xu Liu, and Sen Su. Prediction-based dynamic resource scheduling for virtualized cloud systems. *JOURNAL OF NETWORKS*, 9:375–383, 2014.

[92] J. Izraelevitz, J. Yang, L. Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and S. Swanson. Basic performance measurements of the intel optane dc persistent memory module. *ArXiv*, abs/1903.05714, 2019.

[93] Xiaohong Jiang, Fengxi Yan, and Kejiang Ye. Performance influence of live migration on multi-tier workloads in virtualization environments. In *CLOUD 2012*, 2012.

[94] Changyeon Jo, Youngsu Cho, and Bernhard Egger. A machine learning approach to live migration modeling. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, pages 351–364, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5028-0. doi: 10.1145/3127479.3129262. URL `http://doi.acm.org/10.1145/3127479.3129262`.

[95] N. Katzburg, A. Golander, and S. Weiss. Nvdimm-n persistent memory and its impact on two relational databases. In *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, pages 1–5, 2018.

[96] Eric Keller, Soudeh Ghorbani, Matt Caesar, and Jennifer Rexford. Live migration of an entire network (and its hosts). In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, page 109–114, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450317764. doi: 10.1145/2390231.2390250. URL `https://doi.org/10.1145/2390231.2390250`.

[97] S. Kikuchi and Y. Matsumoto. Performance modeling of concurrent live migration operations in cloud computing systems using prism probabilistic model checker. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 49–56, 2011.

[98] Daniel King, Víctor Lopez, Oscar Gonzalez de Dios, Ramon Casellas, Nektarios Georgalas, and Adrian Farrel. *Application-Based Network Operations (ABNO)*, pages 245–267. Springer International Publishing, Cham, 2016. ISBN 978-3-319-30174-7. doi: 10.1007/978-3-319-30174-7_10. URL `https://doi.org/10.1007/978-3-319-30174-7_10`.

[99] Pawan Kumar and Rakesh Kumar. Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Comput. Surv.*, 51(6), February 2019. ISSN 0360-0300. doi: 10.1145/3281010. URL `https://doi.org/10.1145/3281010`.

[100] Tuan Le. A survey of live virtual machine migration techniques. *Computer Science Review*, 38:100304, 2020. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2020.100304. URL `http://www.sciencedirect.com/science/article/pii/S1574013720304044`.

[101] C. Lee, W. Shin, D. J. Kim, Y. Yu, S. Kim, T. Ko, D. Seo, J. Park, K. Lee, S. Choi, N. Kim, V. G, A. George, V. V, D. Lee, K. Choi, C. Song, D. Kim, I. Choi, I. Jung, Y. H. Song, and J. Han. Nvdimm-c: A byte-addressable non-volatile memory module for compatibility with standard ddr memory interfaces. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 502–514, 2020.

[102] Y. Li, H. Liu, W. Yang, D. Hu, X. Wang, and W. Xu. Predicting inter-data-center network traffic using elephant flow and sublink information. *IEEE Transactions on Network and Service Management*, 13(4):782–792, Dec 2016. ISSN 2373-7379. doi: 10.1109/TNSM.2016.2588500.

[103] H. Liu, S. Liu, X. Meng, C. Yang, and Y. Zhang. Lbvs: A load balancing strategy for virtual storage. In *2010 International Conference on Service Sciences*, pages 257–262, 2010. doi: 10.1109/ICSS.2010.27.

[104] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *HPDC*, 2011.

[105] Youyou Lu, Jiwu Shu, Youmin Chen, and Tao Li. Octopus: An rdma-enabled distributed persistent memory file system. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '17, page 773–785, USA, 2017. USENIX Association. ISBN 9781931971386.

[106] Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. Data storage management in cloud environments: Taxonomy, survey, and future directions. *ACM Comput. Surv.*, 50(6), December 2017. ISSN 0360-0300. doi: 10.1145/3136623. URL https://doi.org/10.1145/3136623.

[107] Violeta Medina and Juan Manuel García. A survey of migration mechanisms of virtual machines. *ACM Comput. Surv.*, 46(3), January 2014. ISSN 0360-0300. doi: 10.1145/2492705. URL https://doi.org/10.1145/2492705.

[108] S. B. Melhem, A. Agarwal, N. Goel, and M. Zaman. A markov-based prediction model for host load detection in live vm migration. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 32–38, 2017.

[109] Suhib Bani Melhem, Anjali Agarwal, Nishith Goel, and Marzia Zaman. Markov prediction model for host load detection and VM placement in live migration. *IEEE Access*, 6:7190–7205, 2018. doi: 10.1109/ACCESS.2017.2785280. URL https://doi.org/10.1109/ACCESS.2017.2785280.

[110] R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. De Turck, and S. Latré. Dynamic resource management in sdn-based virtualized networks. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 412–417, 2014. doi: 10.1109/CNSM.2014.7014204.

[111] Germán Moltó, Miguel Caballer, and Carlos de Alfonso. Automatic memory-based vertical elasticity and oversubscription on cloud platforms. *Future Gener. Comput. Syst.*, 56(C):1–10, March 2016. ISSN 0167-739X. doi: 10.1016/j.future.2015.10.002. URL https://doi.org/10.1016/j.future.2015.10.002.

[112] Mostafa Noshy, Abdelhameed Ibrahim, and Hesham Arafat Ali. Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications*, 110:1 – 10, 2018. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2018.03.002. URL http://www.sciencedirect.com/science/article/pii/S1084804518300833.

[113] M. Patel and S. Chaudhary. Survey on a combined approach using prediction and compression to improve pre-copy for efficient live memory migration on xen. In *2014 International Conference on Parallel, Distributed and Grid Computing*, pages 445–450, 2014.

[114] K. Rybina and A. Schill. Estimating energy consumption during live migration of virtual machines. In *2016 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–5, 2016.

[115] S. Sahni and V. Varma. A hybrid approach to live migration of virtual machines. In *2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–5, Oct 2012.

[116] Felix Salfner, Peter Tr oger, and Andreas Polze. Downtime Analysis of Virtual Machine Live Migration. In *The Fourth International Conference on Dependability*, pages 100–105. IARIA, IARIA, 2011. ISBN 978-1-61208-149-6.

[117] Felix Salfner, Peter Tr oger, and Matthias Richly. Dependable Estimation of Downtime for Virtual Machine Live Migration. *International Journal On Advances in Systems and Measurements*, 5:70–88, 2012. URL `http://www.iariajournals.org/systems_and_measurements/tocv5n12.html`.

[118] V. A. Sartakov and R. Kapitza. Multi-site synchronous vm replication for persistent systems with asymmetric read/write latencies. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 195–204, 2017. doi: 10.1109/PRDC.2017.33.

[119] Yizhou Shan, Shin-Yeh Tsai, and Yiying Zhang. Distributed shared persistent memory. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, page 323–337, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350280. doi: 10.1145/3127479.3128610. URL `https://doi.org/10.1145/3127479.3128610`.

[120] H. Shen and H. Zhou. Cstorage: An efficient classification-based image storage system in cloud datacenters. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 480–485, Dec 2017. doi: 10.1109/BigData.2017.8257961.

[121] N. Singh and S. Rao. Online ensemble learning approach for server workload prediction in large datacenters. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 68–71, Dec 2012. doi: 10.1109/ICMLA.2012.213.

[122] G. Somani and S. Chaudhary. Application performance isolation in virtualization. In *2009 IEEE International Conference on Cloud Computing*, pages 41–48, 2009. doi: 10.1109/CLOUD.2009.78.

[123] A. Strunk and W. Dargie. Does live migration of virtual machines cost energy? In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 514–521, 2013.

[124] Anja Strunk. Costs of virtual machine live migration: A survey. In *Proceedings of the 2012 IEEE Eighth World Congress on Services*, SERVICES '12, pages 323–329, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4756-5. doi: 10.1109/SERVICES.2012.23. URL http://dx.doi.org/10.1109/SERVICES.2012.23.

[125] Anja Strunk. A lightweight model for estimating energy cost of live migration of virtual machines. *2013 IEEE Sixth International Conference on Cloud Computing*, pages 510–517, 2013.

[126] D. Tan, F. Liu, N. Xiao, and Y. Xie. Optimizing virtual machine live migration in distributed edge servers based on hybrid memory. In *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD IS)*, pages 29–34, 2019. doi: 10.1109/HPBDIS.2019.8735491.

[127] Franco Travostino, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunath, and Phil Yonghui Wang. Seamless live migration of virtual machines over the man/wan. *Future Gener. Comput. Syst.*, 22(8):901–907, October 2006. ISSN 0167-739X. doi: 10.1016/j.future.2006.03.007. URL https://doi.org/10.1016/j.future.2006.03.007.

[128] H. Tseng, H. Lee, J. Hu, T. Liu, J. Chang, and W. Huang. Network virtualization with cloud virtual switch. In *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pages 998–1003, Dec 2011. doi: 10.1109/ICPADS.2011.159.

[129] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10664-4. doi: 10.1007/978-3-642-10665-1_23. URL http://dx.doi.org/10.1007/978-3-642-10665-1_23.

[130] Daniel Waddington, Mark Kunitomi, Clem Dickey, Samyukta Rao, Amir Abboud, and Jantz Tran. Evaluation of intel 3d-xpoint nvdimm technology for memory-intensive genomic workloads. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '19, page 277–287, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450372060. doi: 10.1145/3357526.3357528. URL https://doi.org/10.1145/3357526.3357528.

[131] Michael Welzl. *Network Congestion Control: Managing Internet Traffic (Wiley Series on Communications Networking amp; Distributed Systems)*. John Wiley amp; Sons, Inc., Hoboken, NJ, USA, 2005. ISBN 047002528X.

[132] Timothy Wood, K. K. Ramakrishnan, Prashant Shenoy, and Jacobus van der Merwe. Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '11, page 121–132, New York, NY, USA, 2011. Association

for Computing Machinery. ISBN 9781450306874. doi: 10.1145/1952682.1952699. URL https://doi.org/10.1145/1952682.1952699.

[133] Y. Wu and M. Zhao. Performance modeling of virtual machine live migration. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 492–499, 2011.

[134] S. Xi, C. Li, C. Lu, C. D. Gill, M. Xu, L. T. X. Phan, I. Lee, and O. Sokolsky. Rt-open stack: Cpu resource management for real-time cloud computing. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 179–186, 2015. doi: 10.1109/CLOUD.2015.33.

[135] R. Xie, X. Jia, K. Yang, and B. Zhang. Energy saving virtual machine allocation in cloud computing. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 132–137, 2013.

[136] Yiying Zhang, Jian Yang, Amirsaman Memaripour, and Steven Swanson. Mojim: A reliable and highly-available non-volatile memory system. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, page 3–18, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450328357. doi: 10.1145/2694344.2694370. URL https://doi.org/10.1145/2694344.2694370.

[137] Ming Zhao and Renato J. Figueiredo. Experimental study of virtual machine migration in support of reservation of cluster resources. In *Proceedings of the 2Nd International Workshop on Virtualization Technology in Distributed Computing*, VTDC '07, pages 5:1–5:8, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-897-8. doi: 10.1145/1408654.1408659. URL http://doi.acm.org/10.1145/1408654.1408659.

[138] Jie Zheng, Tze Sing Eugene Ng, and Kunwadee Sripanidkulchai. Workload-aware live storage migration for clouds. *SIGPLAN Not.*, 46(7): 133–144, March 2011. ISSN 0362-1340. doi: 10.1145/2007477.1952700. URL https://doi.org/10.1145/2007477.1952700.

[139] Zhitang Chen, Jiayao Wen, and Yanhui Geng. Predicting future traffic using hidden markov models. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–6, Nov 2016. doi: 10.1109/ICNP.2016.7785328.

[140] Yan Zhu and et al. Network traffic prediction based on particle swarm bp neural network. In *Journal of Networks*, volume 8, page 2685, Nov 2013. doi: 10.4236/jilsa.2010.23018.

Honesty Declaration

I hereby confirm that this thesis is the result of my own independent scholarly work, and that in all cases material from the work of others (in books, articles, essays, dissertations, and on the internet) is acknowledged, and quotations and paraphrases are clearly indicated. No material other than that listed has been used.