PREDICTION BASED DYNAMIC RESOURCE ALLOCATION IN VIRTUALIZED ENVIRONMENTS

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

BANE RAMAN RAGHUNATH

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE - 575025

APRIL 2019

Dedicated to my beloved family and teachers.

National Institute of Technology Karnataka, Surathkal

DECLARATION

by the Ph.D. Research Scholar

I hereby **declare** that the Research Thesis entitled **Prediction based Dynamic Resource Allocation in Virtualized Environments** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Computer Science and Engineering** is a **bonafide report of the research work carried out by me**. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

(**CS12F02, Bane Raman Raghunath**)
(Register Number, Name & Signature of Research Scholar)
Department of Computer Science and Engineering

Place: NITK, Surathkal.
Date: April 12, 2019.

National Institute of Technology Karnataka, Surathkal

CERTIFICATE

This is to *certify* that the Research Thesis entitled **Prediction based Dynamic Resource Allocation in Virtualized Environments** submitted by **Bane Raman Raghunath**, (Register Number: **CS12F02**) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Annappa B.
Research Guide
(Name and Signature with Date and Seal)

Chairman - DRPC
(Signature with Date and Seal)

Acknowledgments

In this incredible four years journey of my Ph.D., I had the fortune to meet and work with many outstanding people, without whom I never could have done it. Their encouragements have allowed me to push the edge of the envelope beyond what I originally thought to be feasible. For this reason, I would like to mention the individuals to whom I owe my genuine admiration and thankfulness.

I began this journey under the able guidance of **Dr. K. C. Shet**, who introduced me to the world of research by motivating me to work with enthusiasm and by putting me on the right track of research. Unfortunately, I could not remain under his guidance as he left this world for heavenly abode unexpectedly, leaving behind the legacy of his work and inspiration. I will always be indebted to him and keep up the good work he has motivated me to do. Thank you **Dr. K.C. Shet**.

I wouldn't know what a research is and its depth if I was not given an opportunity to do it. A great share of credit is due to my research supervisor, **Dr. Annappa** who gave me an opportunity to work in this relevantly significant field. There are absolutely no words to express how grateful I am to my guide who supported me throughout my journey. His constant support, guidance and supervision accorded me to focus on my research work. His unrelenting passion for quality and soundness in research, and his incredible professionalism, work ethic, and drive, have been a defining inspiration for my endeavors. Thank you sir for always leading me on the right path.

I am enormously thankful to the Research Progress Assessment Committee members **Dr. K Chandrasekaran** and **Dr. Vidya Shetty** for their insightful comments, critical questions, and valuable ideas. Their continuous interest about my research progress and their sharp and quick feedback in all matters has greatly helped me in achieving research related objectives.

I thank **Prof. K Chandrasekaran** for inspiring me in many ways during research tenure. His unparalleled commitment towards research and teaching has influenced me in the way that I cannot describe in words. I am very thankful to **Dr. Mohith Tahilani** for providing convincing solutions for the research and academics related matters. I am grateful to **Dr. Chandavarkar** and **Mrs. Saumya Hegde** for their cheering words. I humbly thank **Dr. Santhi Thilagam**, **Dr. Alwyn Roshan Pais**, HOD and Chairman(DRPC) and **Dr. Manu Basavaraju**, Secretary (DRPC) for helping me in research related aspects. I am thankful for the support and advices received from **Dr. Jeny Rajan**, **Mrs. Vani M** and **Dr. Basavaraj Talwar**.

My journey during Ph.D. wouldn't have been exciting without my friend **Mr. Patil Sachin** who will remain major part of my memories at NITK and has been a genuine pleasure working with him. My heartfelt gratitude to **Dr. Likewin Thomas** for thoroughly reading through my text and providing me with detailed comments and critical remarks. Feedbacks and bits of advice given by her helped me in improving the quality of my research related documents to many folds.

My cubicle/research lab was more exciting and fun with many colleague/friends around me. **Praveen** my late night lab mate, **Vishnu** who never fails to bring a smile on my face and such a great person to work with. **Manoj**, **Sumith**, **Kirthi**, **Vishal**, **Girish**, **Bhimappa**, **Khyamling**, **Manjunath Mulimani**, **Marimuttu**, **Raghavan**, and **Fathima** made the place so lively and educative. It has been a pleasure to share the lab with them and I will be ever grateful for their big or small contribution to this work.

I also thank **Shahaji** and **Amol** my friends who have been helpful in many aspects . **Mangesh Verlekar and Sudhir Nigudkar** are my soul mates and I am grateful to them for standing by me at all times.

I humbly thank the non-teaching staff of my department. **Mr. Dinesh Kamath**, **Mrs. Yashawanthi** and **Mr. Vairavanathan** were supportive in ensuring that the research-related seminars go well and uninterrupted. I am thankful to **Ms. Vanitha**, **Mrs. Seema Shivaram**, and **Mrs. Mohini** for acknowledging me through the academic work and helping me the all the time. **Ms. Krithi** and **Mr. Ravi** were wonderful in helping me for several times. I can't forget **Mr. Dayanand** for his help regarding academic and office related matters.

I deeply bow down to my parents **Mr. Bane Raghunath Arjun** and **Mrs. Bane Ranjana Raghunath** for being the main pillar of support through this journey. Their understanding and encouragement gave me the confidence to take up this research helped me in all my decisions. Without their support, this thesis would not have been possible. I am very thankful to Prof. Mahesh Satam and his family for showing me the approach to understanding and experiencing life and this world. I want to thank my mother-in-law **Sushila Sawant** and brother-in-law **Mandar Sawant** for giving me support and confidence to complete this work. I am extremely grateful to my amazing wife **Megha** for her constant and unconditional love and support. She stood by me during the good times and the tough times and made my journey to the finish line much more enjoyable. Even though I was away from her and my childrens for a long duration, she prayed for me and my success during the stay in NITK. Without her constant support, this journey would have been incomplete. My special thanks to my dear daughters **Swara** and **Arya** who have been wonderfully obedient and disciplined in my absence. I once again, would like to thank all of my family members for all the support and encouragement I have always got from them.

In the end, all that remains is to thank you, dear reader. If you have found at least a small part of this thesis useful or interesting, you have made all my work worthwhile.

Finally my inmost gratitude towards almighty for blessing and helping me get through this journey successfully.

Place: Surathkal **Bane Raman Raghunath**

Date: April 12, 2019.

Abstract

Cloud computing has become more popular in recent years. Information Technology industries and individual users are attracted towards cloud computing as they can get required number of resources from it. Cloud computing basically provides Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS). Companies such as Google, Microsoft and Amazon, host large datacenters networked with high end computer systems and made available to users on rent. These users may be an individual researcher, an organization or a company. As datacenters are heavily used by many clients and workloads are of different types varies in length and consumption of the resources , allocation of underlying resources is the most important issue for its efficient utilization. In most of the large-scale datacenters virtualization is the technology used for resource sharing among different applications running on Virtual Machines (VMs) created on the same Physical Machine (PM). Virtual Machine Monitor (VMM) provides resource isolation among co-located VMs. However, this resource isolation does not provide performance isolation between VMs. Resource scaling is the important property of virtualization. Elastic auto-scaling is the need of the day and studies show that most of the existing data center infrastructure resulted in either over-provisioning or under-provisioning. It necessitates, on-demand resource allocation to individual VMs from the physically shared pool of resources as per their dynamic requirements to satisfy the Service Level Agreements (SLA) between the customer and cloud provider.

Hence, it is necessary to predict the resource requirements periodically and well in advance. Most of the prediction techniques presented in the literature are useful with a particular type of workload. Hence, it is necessary to analyze which one should be used depending on the type of workload. Most of the studies are concentrated on local resource allocation. When resource deficiency is present we can think of remote

allocation as most of the VMMs provide live VM migration facility. As VM migration process itself is a resource consuming process, its effects on other running VMs have to be studied and the VM for migration has to be selected accordingly.

This thesis presents an architecture for dynamic on-demand resource allocation using statistical machine learning techniques. The resource allocation controller allocates the resources locally on the same PM or remotely through a live VM migration on another PM. The need for migration is determined in advance so that it triggers the migration when, sufficient number of resources are available. The migration manager selects VM for migration which produce less interference to other running VMs at a less migration cost. This migration is done without affecting the performance of the applications running on migrating VM. The prevalent approaches are manual or automatic and all of these are reactive approaches where action will be taken after specific situation is detected. Hence it experiences the unavailability of required number of resources until the action is taken. The proposed approach is proactive, hence the sufficient number of resources are available even at peak time.

Experiments are carried out with synthetic and real application workloads. Prediction of future requirement is done with fuzzy prediction system and Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM). The workload prediction accuracy is received with Mean Absolute Error (MAE) of 0.056. The type of the workload is identified with the help of chaos indicator designed to decide which particular prediction technique is used. Scaling of the CPU and network resources is done automatically in accordance with the dynamically changing workload at a minimum granularity of 2 seconds with savings in the resources as compared to static allocation. It has been found that the proposed scheme allocates resources as per their dynamic requirements with minimum difference between actual requirements and allocation. The resource saving with proposed method is around 30-50% as compared to static allocations. The resource underestimation errors due to spikes in the workload are minimized. The performance improvement in terms of response time of an application is around 15-20% as compared to other methods because of proper selection of VM for migration by the migration manager.

Table of Contents

List of Figures

List of Tables

Abbreviations and Nomenclature

Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
AR	Auto-Regressive
ARMA	Auto-Regressive Moving Average
BPNN	Back Propogation Neural Network
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
DNS	Domain Name Service
EN	Ensemble
FFT	Fast Fourier Transform
HPC	High Performance Computing
HTTP	Hyper Text Transfer Protocol
I/O	Input Output
IEEE	Institute of Electrical and Electronics Engineers
KBC	Kalman Basic Controller
KVM	Kilobyte Virtual Machine
LAN	Local Area Network

LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MIMO	Multi Input Multi Output
NFS	Network File System
NIMO	NonInvasive Modeling for Optimization
NN	Neural Network
OLTP	Online Transaction Processing
OS	Operating System
PM	Physical Machine
PRESS	PRedictive Elastic reSource Scaling
QoS	Quality of Service
RAM	Random Access Memory
RNN	Recurrent Neural Network
SISO	Single Input Single Output
SLA	Service Level Agreement
SLO	Service Level Objectives
VFR	Virtual Firewall Interface
VM	Virtual Machine
VMM	Virtual Machine Monitor
WME	Weighted Majority of Experts
WWS	Writable Working Set
WWW	World Wide Web

Nomenclature

α	Resource Pressure
$C_{i,t}$	CPU resource usage at time t of VM i
$C_{i,t}$	CPU usage on VM i at time t
C_{M_d}	CPU Resource Contention due to Migration at Destination P_d
C_{M_i}	Interference due to CPU usage of VM v_i on PM P_i
C_{R_i}	CPU Co-resident interference caused by migrated VM v_i on the destination PM
$CapTotal$	Total Calacity of the Resource
$CapVal$	Capacity allocated to any particular resource
cpu	CPU Utilization
D_i	Number ofunique pages dirtied in an iteration tine T_i
D_i	Unique Dirtied Pages in iteration i
E_i	Number of Eligible pages to Skip in an iteration E_i
I_{M_d}	Migration Interference at P_d
I_{M_i}	Total Migration Interfrence due to VM v_i
I_{R_i}	Total Co-resident interference caused by migrated VM v_i on the destination PM
M_{wws}	Size of Writable Working Set
mem	Memory Utilization
N_{M_i}	Network I/O interference due migration VM v_i on PM P_i
N_{R_i}	Network I/O co-resident interference caused by migrated VM v_i on the destination PM

nw	Network Utilization
P	Average Number of Unique Dirtied Pages per time interval j
P	Number of Memory Pages (unique) Dirtied during Migration
P_d	Destination PM for migration
P_{new}	Number of First Time Dirtied Pages in an Bitmap Collection Interval Time t
P_{new}	Number of New Dirtied Pages per time interval j
R_d	Memory Dirtying Rate
R_{page}	Memory Page Transfer Rate
S_i	Number of Memory Pages Skipped in iteration i
T_B	Bit Map Collection Period
T_i	Total Time Duration of Interval i
T_{down}	Total Down Time
T_{I_i}	Total Interference due to Migration of VM v_i to destination PM
T_{mig}	Total Migration Time
V_{mig}	Network Traffic Generated during VM live Migration
V_{size}	VM memory size

Chapter 1

Introduction

1.1 Virtualized Environments

Cloud computing has become a major platform of technology adoption in recent years. Its users vary from individual users to large organizations. Users utilize the term "cloud" very commonly without knowing the configurations of computing machines they are accessing, their placement in the data center, etc. Cloud Computing is an example of distributed computing where applications run over multiple computing systems connected by a specific network. This network is referred to as *data center* network which has a tree-like topology, high transmission rates with low latencies. Each physical machine (PM) in this network hosts many virtual machines (VMs) which share the resources of the PM. Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. Fig.1.1 depicts the typical data center environment.

As per NIST (National Institute of Standards and Technology) definition, "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Typically, a cloud is owned and managed by one large organization; companies such as Google, Amazon, Microsoft, etc. The cloud infrastructure is used by a variety of customers ranging from individual user such as common man using Gmail service; students or researchers from university running their program codes on virtual machines; large organizations deploying their

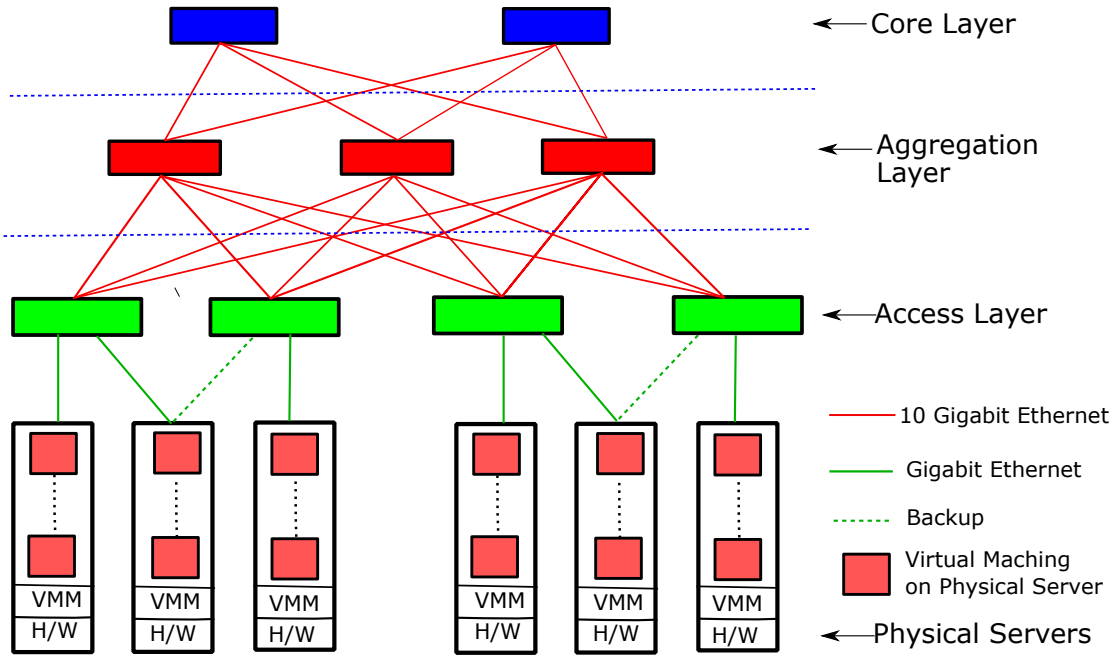

Figure 1.1: Typical Data Center

services across thousands of virtual machines (AWS, 2012a) (AWS, 2012b). These users benefit from the cloud, as they do not have to purchase or even manage their own computing infrastructure of physical machines.

1.1.1 Cloud Computing Deployment

Cloud computing can be broadly deployed in three ways: public clouds, private clouds, community and hybrid clouds.

Public clouds are the most commonly used cloud computing deployment methods. The third party cloud service provider owns the cloud resource infrastructure like servers and storage. These resources are delivered to remote users called customers over the Internet. These resources are shared among the customers of different organizations. Cloud providers use pay-as-you-Go model and customers pay for the cloud resources as per their use. Public clouds examples include Amazon EC2 (EC2, 2012), Google Compute Engine (Google, 2012), HP Cloud (HP, 2012), Rackspace (Rackspace, 2012), and Windows Azure (Azure, 2012).

Advantages of the public clouds:

1. Pay-as-you-Go: Pay only for the subscribed resources. No need to purchase hardware or software resources.

2. No maintenance: Service Providers takes care of cloud maintenance.

3. Near-unlimited scalability: Resources can be allocated and de-allocated on demand. Hence elastic resource scaling is provided.

4. High reliability: As data centers contain thousands of servers it ensures availability of service 24 X 7 throughout the year, using replication.

In contrast, the private clouds consists of computing resource infrastructure used exclusively by the organization for its specific business. The customer and the provider are typically associated with the same organization, sharing the same goals. The infrastructure is solely dedicated to the organization and services are provided on the private network of the same organization or these services can be accessed from outside using public network by the employees of the organizations. Thus organizations can customize its resources to satisfy any specific IT resource requirements using private clouds. Private clouds are often used by financial institutions (banks), government agencies, and IT organizations with the specific business goals and control over the infrastructure. Many companies maintain their own hardware infrastructure called private cloud, which is used by different departments of the companies where the company itself is the provider of the cloud services and its employees are the customers of the cloud services. Similarly, many universities maintain private cloud where users are its faculty members and students.
Advantages of a private clouds:

1. More flexibility: Organizations establish their own infrastructure of resources as a cloud and customize it to meet specific business requirements.

2. Improved security: Cloud resources are not shared with other organization or public, hence higher levels of control and security can be established.

3. High scalability: Although private, these clouds still provide scalability as and when required.

Hybrid clouds combine private and public clouds to get the benefit of both. Organizations sometimes need some of their information to be private while the rest can be on public cloud or private cloud. In this situation hybrid cloud is used. The

organizations get the flexibility of moving its data and applications between private and public cloud as per the business needs. For example, public cloud can be used for high-volume but lower-security needs applications such as email service and the private cloud can be used for confidential, sensitive and business-critical functions . Hybrid cloud supports "cloud bursting". This application or service runs in the private cloud until sufficient resources are present, but when the resource demands are not satisfied through private cloud the services can be moved to public cloud.

Hybrid clouds provides following advantages:

1. Control: Organizations can store sensitive information in private infrastructure and use public cloud for other services.

2. Flexibility: Organizations can take benefit of additional resources from the public cloud when they are in need.

3. Cost effectiveness: When additional resources are needed than privately available for smaller time period, organizations can scale to public cloud instead of purchasing the additional infrastructure.

4. Ease: Moving towards the cloud can be done gradually in phase.

Researchers at Berkeley in 2009 (Armbrust et al., 2010) outlined the problems with cloud computing. For example, services or applications may not run at expected speed. In public clouds, customers subscribe for some amount of resources and run their service. But customers do not understand the exact resource requirement to be subscribed to run the service at an expected speed. The cloud provider also does not guide in this regard. The workloads/ applications of other customers in the cloud also affect the performance of the application.

1.1.2 Cloud Computing Delivery Models

Cloud Services can be generally delivered as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

IaaS means housing dedicated hardware that is purchased or leased for running specific application, and providing basic services necessary to run a cloud. Typical IaaS system provides services like scalability, pay as you go and best-of-breed

technology and resources. In PaaS, the vendor offers a development platform and solution stacks to application developers, who develop applications and offer those services through the provider's platform. PaaS is a variation of SaaS whereby the development environment is offered as a service. PaaS platforms provide multi-tenant development tools, multi-tenant deployment architecture, integrated management, integrated billing mechanisms. Software as a Service (SaaS) is an application software distribution framework in which services or application programs are hosted by a service provider and these are made available to customers over a private or public network. The customer does not purchase the software, but rents it for use on a subscription or pay-per-use model

1.2 Issues in IaaS

IaaS provides: (1) access to the shared infrastructure resources whenever required, with access and location transparency (2) information about details of server images, if demanded, storage and other resource information (3) flexible control of physical resource infrastructure, virtual machines instances and running applications on it. IaaS faces following issues (1) support multiple tenants while supporting virtualization, (2) managing of various resources and network infrastructure, data storage , (3) support for APIs , etc.

Different issues are discussed below.

(i) Support Virtualization with multiple tenants: Virtulization is the heart of cloud computing which hides the internal hardware infrastructure complexity from the user and enables enhanced flexibility (through routing, aggregation, routing , etc). In a multi-tenant environment, multiple users share the same application on the same hardware, with the same data-storage mechanism. The separation among the users is taken care in the application design itself. Virtualization abstracts the physical resources of the machine into separate virtual machines on which different customers applications are running.

Multi-tenancy brings in security issues in cloud computing like data protection from multiple users in cloud.

Hardware with virtualization layer has many benefits but it does not provide

scalability to offer cost effective solution to very large number of customers. Multi-tenancy with virtualization removes this

(ii) Dynamic Resource management: As the application workloads are fluctuating, whenever there is a need to allocate additional resources, they have to be allocated effectively in such a way that Service Level Agreements (SLA) have to be satisfied. The CPU and network resources are very limited as compared to others, and therefore should be efficiently shared among all the applications running on the VMs. Hence there are many issues in resource management such as resource allocation, resource provisioning, resource mapping, and resource adaptation. A fair amount of resource management related work has been done by Urgaonkar et al. (2008) and Vaquero et al. (2008) to examine present and future challenges of providers and users of cloud. Chase et al. (2001) have presented the energy-efficient way of homogeneous resource management in Internet hosting data centers. Here the main problem is to estimate the current resource request demand of an application and allocate these many in an efficient way. Metering of resource utilization and billing is needed for providing elasticity in cloud computing .

(iii) Network infrastructure management: Administrators face problems in managing huge network components such as switches, hubs, routers, bridges, etc. It adds so much administrative cost. Hence there is need for automated management system to monitor very huge amount of data size. Gupta and Singh (2003) suggested to put resources like network interfaces, switches, links and routers into sleep mode when they are in the idle condition. Chiaraviglio and Matta (2010) have proposed cooperation between content providers and ISPs which efficient allocates computing resources and network routes which reduces energy consumption.

(iv)Security and Privacy: These are obviously required in all the systems which deals with confidential and sensitive data and code. In order to add security feature in cloud computing, various security features such as authentication, confidentiality, integrity, and non-repudiation need to be provided.

(v) Storage/ Data management : This is the main work in cloud computing, specifically for storage clouds which deliver storage as a service. Here data is distributed across multiple resources at different locations. The consistency is highly required in replicated resources. It is necessary to know the data location when data is repli-

cated across different data centers. The latencies between different locations and their workload also needs to be considered.

(vi) Cloud platform features can be exploited by using cloud APIs and/or enhancement in programming. Programming models for cloud requires that the scalability in terms of resource allocation must be automatic and developer must take a note of such things. The user must get only this feature through programming model.

1.2.1 IaaS Resource Management

The different types of resource managment in IaaS cloud are given in Table 1.1.

1.3 Motivation

Virtualization (Barham et al., 2003) is the heart of cloud computing where resources are provisioned to the hosted VMs as per their dynamic resource requirement requests. It tries to deliver the necessary performance to the application with isolation and security among different VMs. It also provides an interface to create, configure and manage virtual machines. It allocates and deallocates the resources in an elastic manner. Hence it can deliver pay-as-you-go service to the customers of the data center. It monitors the applications and as per requirement it allocates the resources to the VMs.

The reactive provisioning of resources sometimes degrades the performance of applications as it involves initial delays in allocation. Hence, a proactive scheme which can allocate the resources at proper time is required so that performance will not be degraded. Data centers usually run business applications which require Quality-of-Service requirements. While allocating enough resources to achieve performance guarantees it is necessary to avoid over-provisioning of resources for making it cost effective and allowing many concurrent applications that run on single physical server in the data center. It is also required to resolve the situations where required number of resources are not available for allocation. The following section presents research objectives. The users may be an individual researcher, an organization or a company. Their workload varies in the length and consumption of resources.

As datacenters are heavily used by many clients and workloads are of different

Table 1.1: Resource allocation schemes

Schemes	Definition
Resource Provisioning	Allocating cloud provider's resources to a customer
Resource Allocation	It Allocates cloud infrastructure resources effectively among all VMs such that it should be economical and the desired SLA is satisfied.
Resource Adaptation	Adjusting the cloud resources as per the dynamic need to fulfill defined SLA of the user
Resource Mapping	It Maps between required resources by the customer and available resources with the cloud provider.
Resource Modeling	Model that describes the important resource management properties to predict resource need in the future.
Resource Estimation	Estimation of the resource requirement of an application,to carry out the work with desired throughput.
Resource Discovery and Selection	Identifying available resources to which the job can be submitted and choosing the best among these.
Resource Brokering	Resources are negotiated to ensure that the sufficient resources are available at the time of need
Resource Scheduling	It Schedules different events with their required resources. It determines when an event should start and stop.

types, allocation of the underlying resources is the most important issue for its efficient utilization. In most of the large-scale data centers virtualization is the technology used for resource sharing among different applications running on virtual machines (VMs) created on the same physical machine (PM). The Virtual Machine Monitor (VMM) provides resource isolation among co-located VMs. However, this resource isolation does not provide performance isolation between VMs. Resource scaling is the important property of the virtualization. Elastic auto-scaling is the need of the day

as the studies shows that most of the existing data center infrastructure has resulted in either over-provisioning or under-provisioning. It necessitates on-demand resource allocation from the physically shared pool of resources to individual VM, as per their dynamic requirements to satisfy the Service Level Agreements (SLA) between the customer and cloud provider.

To allocate the resources as per the dynamic needs, it is necessary to predict the resource requirements periodically and well in advance. Most of the prediction techniques presented in the literature are useful with particular type of workload. Hence it is necessary to analyze which one should be used depending on the type of workload. Most of the studies have concentrated on local resource allocation. When resource deficiency is present we can think of remote allocation as most of the VMMs provide live VM migration facility. As VM migration process itself is a resource consuming process, its effects on other running VMs has to be studied and accordingly select the most appropriate VM for migration.

1.3.1 The Problem

Cloud Computing service is also facing problems which were addressed by different researchers at Berkeley in 2009 . The applications running in the data centers do not always run as per the expectations of the customers. Customers may not be knowing the exact amount of resources needed by the application in advance. Also provider is unaware of the application needs. Hence initially while subscribing the VMs for running applications, the VMs are either over-provisioned or under-provisioned. Over-provisioning leads to under-utilization of resources and under-provisioning leads to application performance degradation. This lack of knowledge leads to non efficient utilization of cloud resources' performance degradation. Cloud provider also can not give performance guarantees to their customers.

We will show in this thesis that by predicting the requirements of applications running on VMs we can allocate the resources as per the real requirements. If demanded resources are not available then the VM can be replaced at some different place where sufficient number of resources are available. By doing this there can be efficient utilization of resources in addition to the performance improvements in running application and performance guarantees given by the provider.

1.3.2 Philosophy

The aim is to improve cloud performance and resource utilization such that, one benefits from knowing the history of the application run; what is presently happening with the application and what may happen in the near future. Instead of changing the network topology or routing algorithms, the solution depends on simply knowing the future workload or resource requirements of the application running on VM. Hence this work focuses on resource measurement. What has happened in the past, what is happening currently needs to be looked into, so that the future requirements can be predicted.

It has been observed that, these measurements can be carried out in a fast and light-weight manner, so that it can not affect the other running application's performance. As seen, the workloads are of different types, hence there must be some indicator which can identify this and apply appropriate prediction technique. It is also believed that there can be an error in the prediction process, hence prediction error correction can be done after predicting the future requirements.

1.3.3 Content of the Thesis

This thesis focuses on design and analysis of a dynamic resource allocation technique in virtualized environments. This solves two problems-

1. Cloud provider can determine the dynamic requirements of the running application by making prediction about future resource requirements of application and thus dynamically allocate the resources as per the requirement well in advance.

2. Determine the need for application VM migration in future and place the VM on other PM, where enough resources are available.

1.3.4 Challenges

Application workload changes with time. These variations depend upon the running applications. Thus customers are not expected to know the resource demand of application in future. Hence the first problem is workload or resource demand prediction. In order to solve this problem the system must overcome the following challenges-

- To make predictions based on history, resource monitoring system which must not incur overhead on the system is needed.

- While making predictions, the resource allocation system must first identify whether the workload pattern is linear or non-linear, in variations with respect to time and accordingly apply the prediction technique.

- The predictions may be wrong sometimes and underestimation errors are not allowed. Hence the system must identify and make under-estimation error corrections.

- The resource allocation system must be able to predict future resource demand with a smaller amount of history data, as it is unlikely that a customer will wait for a long time to make the first prediction. It is also important to make predictions at multiple time steps ahead.

The second problem of placing the VM on other PM must overcome the following challenges.

- If the predicted demand is not available for the application on PM then it or any other VM must be placed on other PMs, in a way that migration overhead and its interference on other running VMs is less.

- The VM live migration process consumes significant amount of resources, like CPU and Network bandwidth for iteratively copying work of the VM live image. Hence the system must make predictions of resource unavailability and hence the need for migration before it actually happens.

- The selection of VM for migration and the destination PM needs knowledge about workloads of all VMs and also total workloads of all the PMs. Hence there must be a procedure to gather this knowledge at the resource allocation controller.

In addition to all these challenges, the system should address the issues without making modifications in the resource infrastructure side and without any manual interference. Thus the system has to be autonomous.

1.3.5 Limitations of Prevalent Approaches

- The prior approaches considered specific type of workloads assuming the prior knowledge about the characteristics of an application for future prediction.
- Few efforts are made to predict the overload at the time of migration which is a transient problem due to migration process itself. The real application overload need to be addressed.
- The prior approaches are reactive in nature which takes actions after the overload is detected and hence it may results into SLA violations.
- The prediction error handling is not considered in prior approaches.
- The prior systems are not fully autonomous. Manual intervention is required to monitor and take the action accordingly for resource allocation.

1.4 Problem Statement

Design a prediction based architecture for dynamic allocation of resources in virtualized environments to improve resource utilization.

1.5 Research Objectives

1. Estimate the resource requirement of an application running on a VM by observing the past resource usage and accordingly allocate the resources nearer to real dynamic needs to make resource allocation efficient. Handle under-estimation errors.
2. Model VM migration process to calculate the performance and cost metrics. Empirically study the effects of live VM migration on other VMs running on source and destination PMs. Model these interference effects.
3. Make efficient migration strategy by leveraging above used prediction schemes to predict the real application workload for early detection of overloads and then trigger the migration using the parameters determined in objective 2.

1.6 Research Contributions

1. Presents the resource allocation controller which resides in every VM. It determines the future resource needs by observing past resource usage, using fuzzy prediction system or prediction with weighted majority of experts. It uses Kalman filter based prediction when workload is very non-linear. Adaptive padding scheme is applied on predicted values to remove under-estimation errors. If under-estimation errors are still detected, then these are corrected by immediately raising resource cap with some value. This value is estimated by observing the pressure on the resource at that time.

2. A live migration procedure model is presented by considering all the parameters which are responsible for performance of live migration. The interference effects caused by the migration procedure itself on other running VMs on PM are studied and a model is presented to calculate CPU and Network interference effects.

3. Presents the migration manager which detects when to trigger migration, select the VM for migration so that interference effects are reduced and select the destination PM where selected VM is to be migrated.

   The proposed architecture is shown in Fig.1.2. The resource management architecture comprises of fuzzy logic based prediction system per VM, to predict the future resource needs by observing past resource usage. The prediction system contains resource monitoring module which records the resource usages seen in the past. The predicted resource values are given to the resource allocator of the PM. The resource allocator decides whether the required resources can be allocated locally or migration is required. It can also predict the need for migration in the future time. If migration is required then it signals this to the migration manager. The migration manager selects the best VM for migration and the best PM as a destination for the migrating VM.

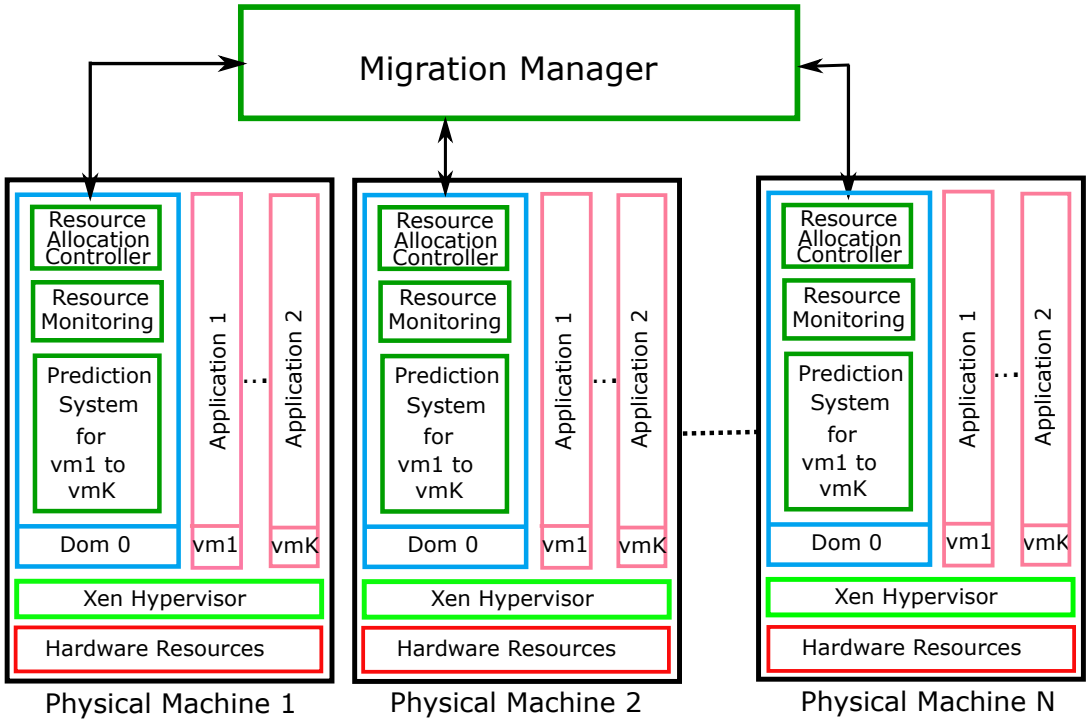

Figure 1.2: Dynamic Resource Allocation System

1.7 Outline of the Thesis

Chapter 2 describes the preliminaries required to understand further chapters. Chapter 3 describes the literature survey done. Further it summarizes and defines the problem statement of the research work. It describes the research objectives. Chapter 4 describes the fuzzy prediction system and Kalman Filter. Further it shows how to handle under-estimation errors. Chapter 5 describes the model for estimating performance parameters of live VM migration method. It also describes a model to estimate interference effects of a migrating VM on other VMs. Chapter 6 describes the resource allocation controller and migration manager.

Chapter 2

Preliminaries

2.1 Types of Resources

Any physical or virtual component with limited availability is a resource in a cloud system. The main computing resource is the Central Processing Unit (CPU), communication resource is the network bandwidth; task execution is the memory resource. The Operating System (OS) is also one of the important resources which is required to run the application on the hardware. These resources can be broadly categorized as physical resources and logical resources. Different cloud infrastructure resources and their impact on its performance is discussed below.

2.1.1 Cloud Physical Resources

The Cloud data center physical resources mainly includes CPU, memory, storage and peripheral devices.

(i) CPU : It executes instructions of the program and thus does most of the work inside a computer system. Its utilization is very important as many clients work parallel of the cloud infrastructure. CPU utilization is the usage of the CPU to carry out execution of any application on a computer system. In cloud CPU utilization depends on how many applications are running and the type of applications. Certain applications are CPU intensive, while others are not because of non-CPU resource requirements. Hence required amount of CPU allocation as per the application requirement is necessary in cloud for efficient utilization of CPU resource in cloud.

(ii) Memory: In a cloud computing environment, different types of applications are running static memory allocation are not useful. Rather dynamic memory allocation

is strongly required. Memory resource in the form of virtual entities is required in the cloud architecture as the cloud data center servers consist of many number of CPU cores and since the virtualization platform requests huge amount of memory resource.

(iii) Storage: In a cloud environment storage generally means, storing the data to some data server which may belong to a third-party. It is a remote database server, called storage server. Using Internet, this data can be accessed. Cloud computing storage service includes hundreds of data/storage servers. Many a times, data is replicated at many locations with redundancy. This cloud storage provides reliability and security. Clients may not trust storing their data in these data storages without a guarantee that they can access their data whenever they need it and no one else has an access to this data. Thus cloud storage provides Stoarage-as-a-Service.

(iv) Workstations: Powerful workstations with large CPU, RAM and network resources are treated as workstations. These machines are mainly used to do a lot more local processing. But if more resources are required to carry out the processing and the required resources are not available, then help is taken from cloud computing technology. These workstations can be used to generate High Performance Computing (HPC) facilities for the clients connected through Internet . Then user can run his application on local workstation and shift it on the cloud resources when there is insufficient number of resources in order to get the work done. This must be done automatically and hence resource usage monitoring is required.

(v) Network elements: Administrators face a lot of problems in managing millions of network elements such as switches, hubs, bridges, routers, etc. The administrative cost is very high, hence an automated method, for management of these network elements, is required. This automated method should be capable of handling sizes of monitored data which is several orders of magnitude than the traditional systems. Thus cloud provides communication-as-a-service.

2.1.2 Cloud Logical Resources

These are the abstractions of cloud physical resources which have temporary control over it. These are useful in carrying out execution of an application over a cloud system and also in communication protocols required for carrying out communication between the client and the cloud provider. The importance of cloud logical resources

are described as follows.

(i) Operating system: Operating System provides an environment to the user to execute an application on the hardware. It is also possible for the user to manage physical resources and it gives mechanisms to control of resources. Usually operating systems do all the management work like resource management, file management, performance management with efficient utilization of resources, security, device management, etc.

(ii) Energy: In cloud, the main aim is to minimize the energy consumed by resources. For this, the server consolidation approach is applied, where the entire workload is concentrated on the minimum number of physical servers, by shifting the workloads to these servers and switching off the idle physical servers. Thus energy consumption is reduced, but it is energy/performance trade-off, as throughput of an application may degrade due to this.

(iii) Network throughput: Higher network throughput is desired and hence there should be mechanisms to control the network bandwidth allocated to a particular application. Congestion is prevented using network bandwidth management protocols. The main issue here is the bandwidth allocation problem which depends on integration network link capacities through different services.

2.2 Resource Management Techniques

Cloud resource management consists of provisioning of a cloud provider's resources to a customer. When customers request a cloud provider for computer resources, the cloud provider has to create suitable number of VMs with proper allocation of resources to them. This resource provisioning can be of different types: advance provisioning, dynamic provisioning and user self-provisioning. In advance provisioning, the customer requests for the service and the cloud provider provision the requested resources in advance to the user. Cloud provider charge customer for all the resources provisioned at once or on monthly basis. In dynamic provisioning, the provider allocates required resources as per the need. Whenever customer wants more resources than provisioned, provider will provide it and when customer don't want the resources at that time these will be taken back. Hence customer will be billed as per the use

of the resources. In user self-provisioning, the customer purchases resources from the cloud provider using Internet for a specific time period. Customer creates an account with provider and subscribe the resources as per the need. These resources customer can use for the specified time. Customer will be billed as per the charges for the used resources for that time period.

Live migration of virtual machine is useful tool for hot-spot mitigation, server consolidation and load balancing. It avoids difficulties faced related to dependencies with operating system by process level migration as it migrate entire OS and all its applications running on it to the destination physical machine. It is extremely powerful tool for cloud administrators, allowing server consolidation, load balancing, improving cloud resource usage, taking the server for maintenance on which VM is running. Thus In combination of resource virtualization and VM migration significantly improves manageability of the cloud.

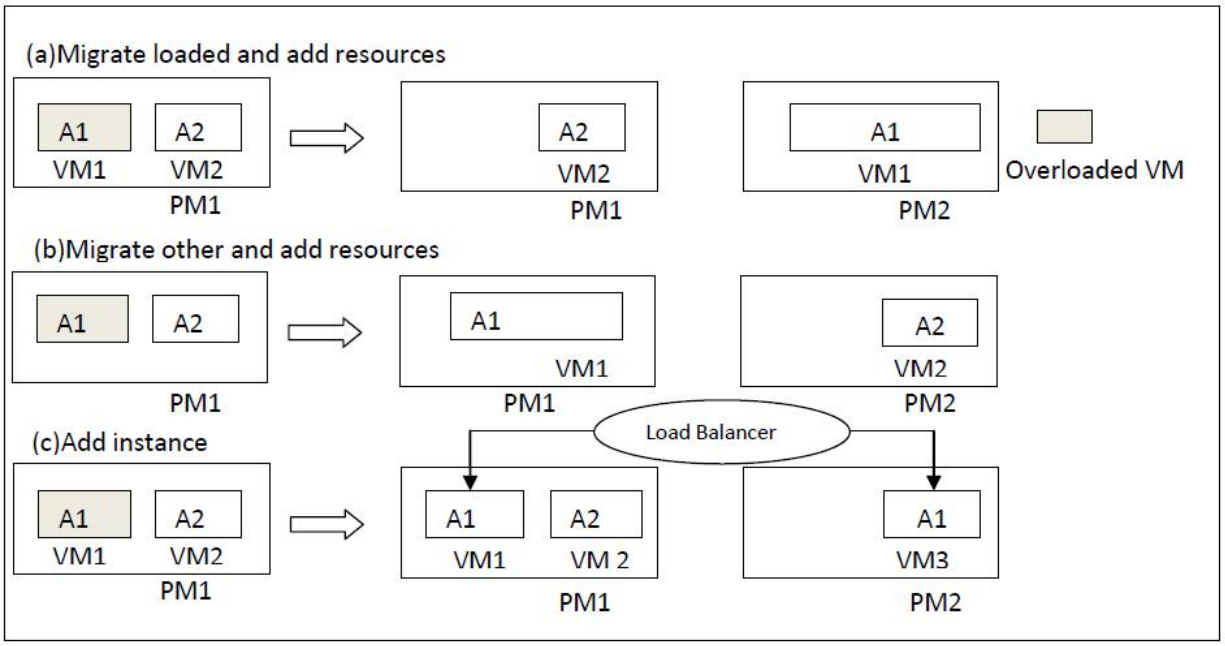

Figure 2.1: Resource hot-spot mitigation using vertical and horizontal scaling up operation

2.3 Dynamic Resource Provisioning

Dynamic resource provisioning of physical machine resources to virtual machines in virtualized data center can be achieved by two ways (i) vertical scaling i.e. adding/re-

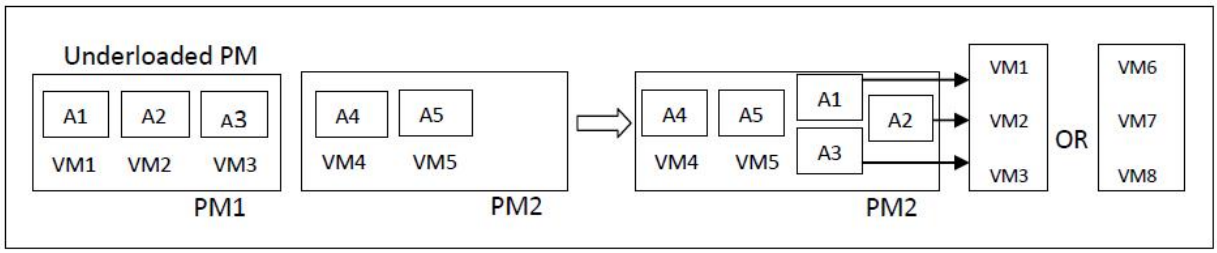

Figure 2.2: A Server Consolidation Example

moving attached resources from existing virtual machine and (ii) horizontal scaling i.e. adding a new virtual machine with additional resources. There are two general scenarios that appears in data centers i.e. Hot-spot-when a virtual machine load level increases, it requires additional amount of resources, additional allocation should be done to satisfy the required performance level and server consolidation- when a virtual machine load level decreases, it results in under-utilization of allocated resources. Hence, the extra resources can be removed and VMs can be consolidated onto a fewer number of physical machines to reduce power usage and maintenance cost, if possible. Resource hot-spot mitigation: When a virtual machine faces a resource hot-spot (due to increased load level of an application instance hosted on this virtual machine), it can be resolved using the following two techniques-

1. Vertical Scaling Up: add more resources to the VM facing the hot-spot.

2. Horizontal Scaling Up: host an additional instance of the application on a new VM with required resources. It results in multiple VMs executing the same application, with a load balancer used to distribute the incoming workload proportionally to each VM (assuming that one virtual machine can host only one instance of an application).

2.3.1 Hot-spot Mitigation

If the physical machine has adequate free resources to allocate to an overloaded virtual machine hosted on it, the resource hot-spot can be mitigated instantaneously and efficiently by using vertical scaling-up tools such as xm schedcredit or cpulimit provided by existing virtualization software, (Barham et al., 2003). We term this operation as add-resource with the help of a. If the physical machine does not have enough free resources, hot-spot can be mitigated by employing vertical scaling-up

process with the aid of virtual machine live migration (Clark et al., 2005), which is a process of moving an executing virtual machine from one physical machine to another with minimal downtime of application. Here, one way to perform add-resource is by migrating either the overloaded virtual machine to a physical machine where adequate free resources are available, i.e., migrate-loaded & add-resource, or by migrating other virtual machines to make free resources such that the resource hot-spot can be mitigated, i.e. migrate-other & allocate-resource. Another way to resolve this hot-spot situation by employing horizontal scaling process in which we add one more virtual machine hosting the same application with additional required resources. We call this operation as add- instance. When the resource requirement of an application is more than the physical machine resource capacity, the only way to solve this resource hot-spot is by employing horizontal scaling operation on another PM, i.e., add-instance. An example of resource hot-spot mitigation is given in Fig. 2.1. Two unrelated applications, A1 and A2, are hosted on a physical machine PM1. Suppose, virtual machine VM1 hosting application A1 requires additional amount of physical machine resources. However, PM1 does not have adequate free resources to resolve the resource hot-spot. This resource hot-spot can be mitigated in two ways. One way is to migrate out either VM1 or VM2 to physical machine say PM2 and allocate additional resources to VM1. Another way is to create a new virtual machine instance VM3 on another PM where sufficient resources required by VM1 are free and host the application A1 along with a load balancer as shown in part (c) of Fig. 2.1.

2.3.2 Server Consolidation

When multiple physical machines are underutilized due to inactive or underloaded virtual machines, consolidation of these virtual machines onto a fewer number of physical machines is better idea in conserving energy and in turn reduce operational expenditure. This can be achieved by reducing the allocated resources and migrating all VMs hosted on a underutilized physical machine to other physical machines (i.e., remove-resource & migrate operations). This can also be achieved by creating new virtual machines on other physical machines, for each VM hosted on the underutilized PM, and hosting the respective application (i.e., add-instance operation). Once the old virtual machine processes all its on-going requests, it can be shutdown. We term

this operation as add-instance & remove-instance. An example of server consolidation is given in Fig 2.2., where the physical machine PM1 is underutilized as all three applications A1, A2 and A3 are under-utilizing the resources allocated to them. The consolidation can be achieved by removing unused resources from virtual machines VM1,VM2 and VM3, and migrating these VMs to physical machine PM2 where sufficient capacity is available to host VM1, VM2 and VM3. Once the applications hosted on virtual machines VM1, VM2 and VM3 process all their on-going requests, these virtual machines can be removed.

2.3.3 Load Balancing

Applications running on VMs changes their resource requirements during their runtime. This dynamic change imbalance the resource utilization pf PMs. This results in some PMs become heavily loaded and some becomes underloaded. For efficient utilization of the data center the workload should be distributed equally among all the running PMs i.e. balance the load. This can be achieved using live VM migration facility of VMMs. This is shown in Fig. 2.3.

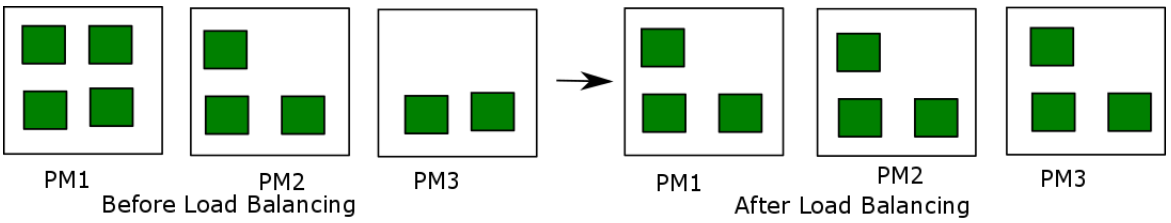

Figure 2.3: Load Balancing

2.4 VMware Distributed Resource Management System

Distributed Resource Scheduler (DRS) is a resource management System in WMware vCenter Server Management software for the cloud environment. It applies the resource management policies to utilize the resources in the cloud efficiently. It allocates physical to the VMs running on the cluster of VMware ESX servers as shown in Fig.2.4. DRS defines a resource model that consists of resource controls, hierarchically organized resource pool and resource pool divvying process. It defines three resource

controls, Reservation, Limit and Shares. Reservation specifies minimum guaranteed amount of a certain resource. Limit on resource specifies the upper limit on its consumption. Shares specify that, a VM is allowed to utilize the resources proportional to its shares. DRS manages the cluster of ESX servers by carrying out resource divvying. DRS performance follows major resource management operations-

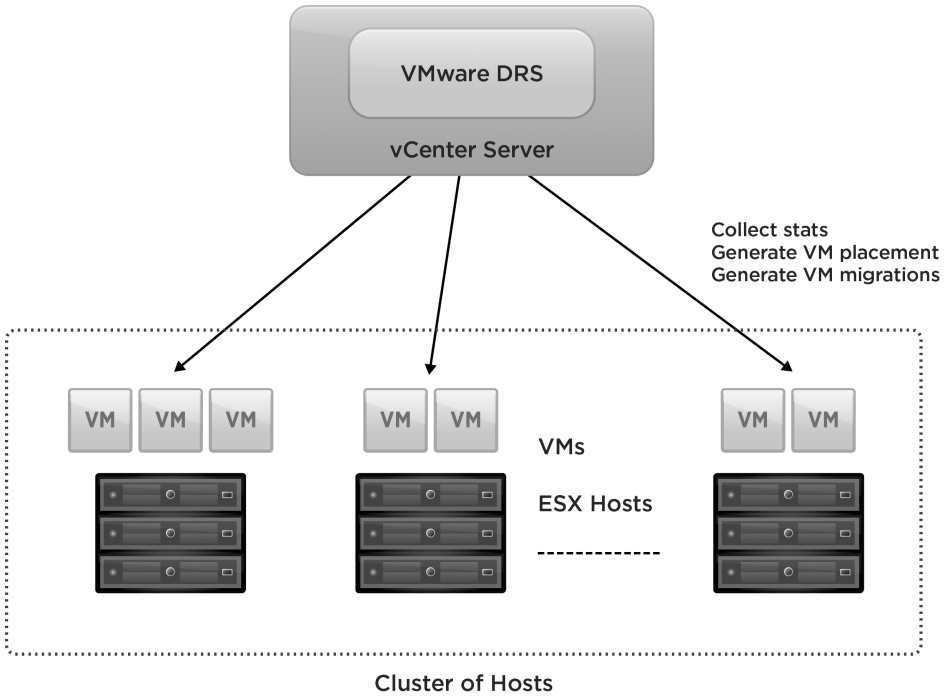

Figure 2.4: VMware DRS Overview

1. Computing the runtime resource demands of each VM running in the cluster.
2. Computing each VM's physical resource entitlement based on runtime demand, reservation, limit and share values.
3. Load balancing using live VM migration (vMotion)
4. Initial placement of VMs on the cluster.
5. Reduces power consumption by extending DRS with VMware Distributed Power Management(DPM).

DRS is a reactive resource management system. The proposed system is proactive. Although the reactive resource management scheme is very useful in minimizing the

operating cost of the cloud environment, it adds an additional responsibility on system administrator or cloud provider to ensure that spikes in the resource demand can be handled in a timely manner. On the other hand this system proactively handles the spikes in resource demand by predicting the future resource usage and allocating the required resources or carrying out the required migrations before hand.

2.5 Live VM Migration over LAN

Live Migration can be over LAN which is most popular or over WAN across different geographical locations (Wood et al., 2015). In LAN based migrations the disk image of the VMs are stored on some shared storage like NFS for which disks never need relocation in the course of VM movement. These involve transfer of memory state of the VM alone. Also since, the VM is moved on a new destination physical machine(PM) on the same LAN network reconfigurations are not needed. But network configurations and disk transfer are both needed in for WAN based migrations as these are across widespread regions making it more challenging.

Live VM migration process can consist of following three phases (Clark et al., 2005).

1. Push Phase : The memory pages of the VM to be migrated are transferred to the destination PM in multiple iterations. In first iteration all the memory pages are transferred and in later iterations the memory pages which are dirtied in the previous iteration are transferred. This is required as application on the VM is running, hence memory pages getting dirtied again and again. The VM to be migrated continues to run on the source PM during this phase.

2. Stop-and-copy phase: When one of the parameters crosses the defined threshold (given in section 5.1) the VM to be migrated is suspended on the source PM. The remaining memory pages and the hardware state is copied at once to the destination PM. and the VM is resumed on the

3. Pull phase: The VM to be migrated is resumed on the destination PM. If the page accessed by the application on that VM caused page fault, then this page is taken from the source PM using the network.

In general live VM migration strategies consists of either one or two of the above described phases. Xen's pre-copy mechanism combines push with stop-and-copy phases, the post-copy mechanism combines pull with stop-and-copy phases.

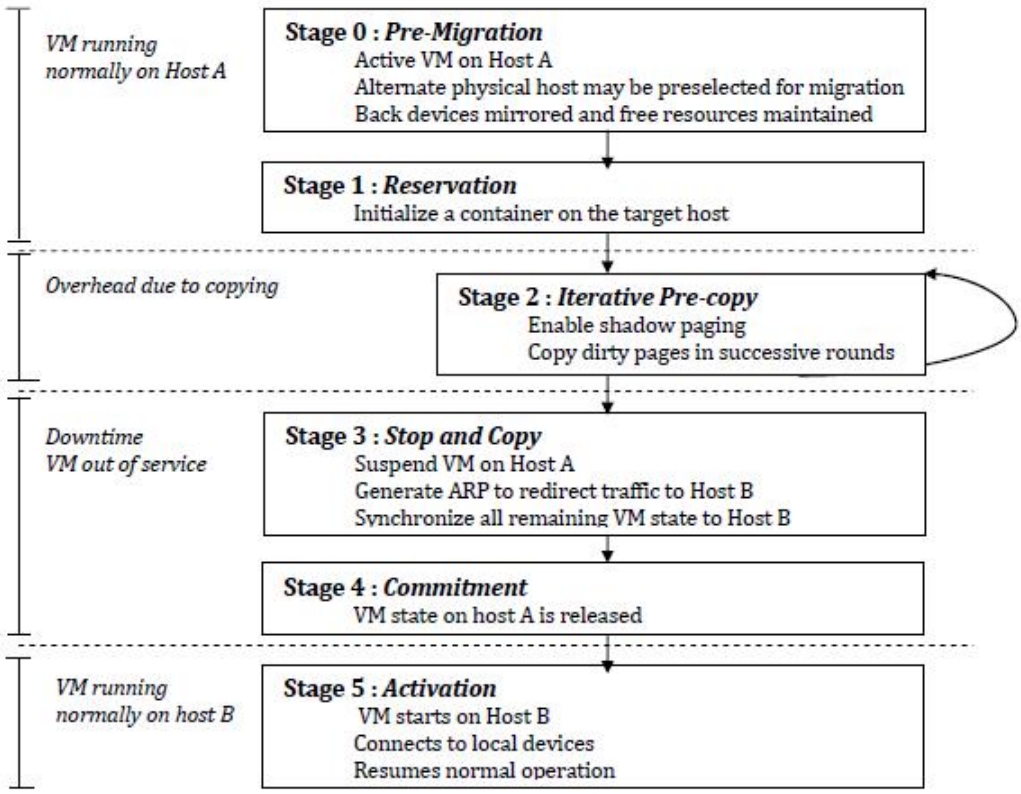

Figure 2.5: Live Virtual Machine Migration- Precopy

2.6 Live Migration Strategies

Following are the different Migration Strategies.

1. Pre-copy-

   It uses Push phase and Stop-and-copy phase. The detailed steps involved in pre-copy migration are shown in Fig. 2.5. It consist of five stages- Pre-migration, Reservation, Iterative Pre-copy, Stop and Copy, Commitment and Activation. It iteratively transfers memory pages of a VM as shown in Fig. 2.6 . The first iteration transfer all memory pages, and the next iterations transfers pages which are dirtied during the previous iteration. The VM execution is suspended

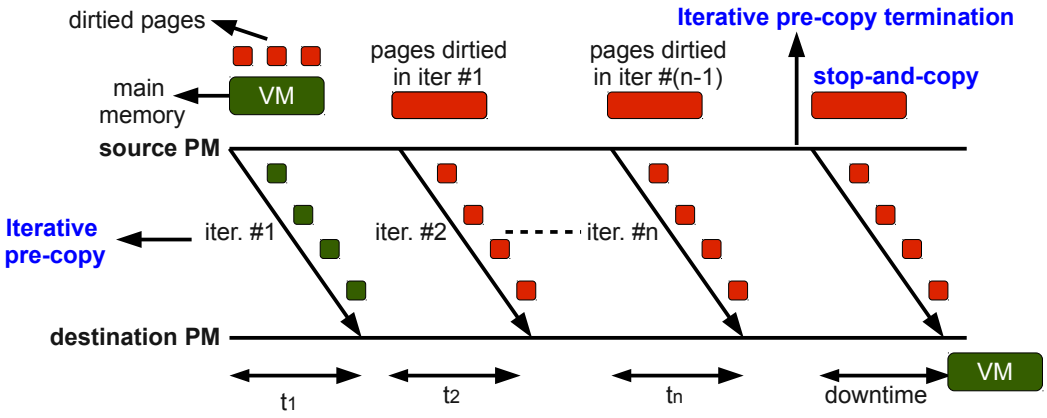

Figure 2.6: Iterative transfer of memory pages in Precopy

when one of the parameters crosses the threshold. At this time, the hardware state and remaining memory pages and the are transfered to the destination machine. This technique is used in Xen for live VM migration. An additional optimization technique is implemented in Xen, called page skip, which minimizes the amount of transfer.

2. Post-Copy-

   It uses pull phase for memory page transfer. It copies the VM's hardware state to the destination PM and VM is resumed there. The request to the application are handled by the migrated VM at the destination. It fetches the memory pages from source PM as and when required. There are many variants present for fetching the VM's memory from the source PM. Here the memory pages are transferred once hence the avoiding duplicate transmission.

Post-Copy has three different ways of memory page transfer.

(1)Post-Copy via Demand Paging: The target VM is resumed at the destination PM and page faults are handled by demanding the page from source PM over the network. But the total migration time is unacceptable and performance of application running on it will be degraded as every time the page faults are handled over the network.

(2)Post-Copy via Active Pushing: This is proactively pushing the memory pages from source PM to destination PM while the target VM is resumed and running at destination PM. If page faults occurs it will be solved by considering this demand transfer from source PM on high priority. This gives better performance than previous

one.

(3)Post-Copy via Pre-Paging: This is an extension to active pushing. It estimates the memory access pattern to determine the locality of the memory pages to be accessed, thus eliminating page faults that are going to occur otherwise. This gives better performance than other two.

2.7 Issues and Tradeoffs

Issues and tradeoffs involved in live LAN migration are-

1. Minimize total migration time
2. Minimize downtime during which services are totally unavailable to the customer.
3. Ensure that migration process does not unnecessarily disrupt active services through resource contention (e.g. CPU, n/w bandwidth) with migrating operating system.

The workload increase can be handled by increasing the resources available on the physical server, otherwise simply by migrating the virtual machine to a less loaded physical server. The resource requirements of VM changes dynamically. The wrong placement of VM may lead to performance degradation and hence SLO violation, hence it is necessary to carry out proper placement of VM. It is necessary to find out when to trigger the migration, which VM should be selected for migration and choose the destination PM for this migrating VM.

The parameters for evaluating the performance of live virtual machine migration process are as follows-

1. Total migration time - It is the time between initiation of successful completion of migration process, and
2. Downtime- the time during which the services provided by the VM are not available, i.e. time taken by stop and copy phase of memory transfer.

Performance of the migration process is evaluated based on following parameters. These are the resources which are consumed during the migration process.

1. VM memory size.

2. CPU availability at source and destination PM.

3. Network bandwidth availability

4. The application page dirty rate.

The live virtual machine migration cost evaluated as :

1. CPU utilization for sending memory pages at source PM and CPU utilization for receiving the same at destination PM,

2. Network traffic generated during migration,

3. Migration rate at which pages are transferred.

The strengths and weaknesses of the proposed system are-

1. The resource monitoring system is implemented in Dom-0 of the PM and hence not incurs costs on individual VM.

2. It identifies the workload pattern is linear or non-linear with the help of chaos indicator. It should also handle the non-stationarity of workloads.

3. It detects and handles underestimation errors, hence avoids SLA violations during the peaks in workload usage.

4. Apart from finding VM for migration it identifies the need for migration in advance.

5. It does not discuss the security implications of usage monitoring, which will be addressed in the future work.

6. The system is fully autonomous and needs no manual intervention.

2.8 Effects of Live VM Migration on Application Performance Running on Different VMs on the same Physical Machine

The effects of a live VM migration in cloud environment, on the performance of applications running on same VM as well as other VMs on the same physical machine are discussed here. The performance is observed in terms of response time. The response time is observed during migration of one Apache Server VM. The experiment setup contains three PMs and five VMs (first two VMs on first PM, later two on second PM and last one on third PM). All VMs run Apache Webserver application serving dynamic PHP web pages. The request to webserver are generated using "httperf". This mimics the requests from 400 different clients. The PHP scripts written are CPU intensive. This will create heavy CPU load on the server PM without much memory and network utilization while serving the requests.

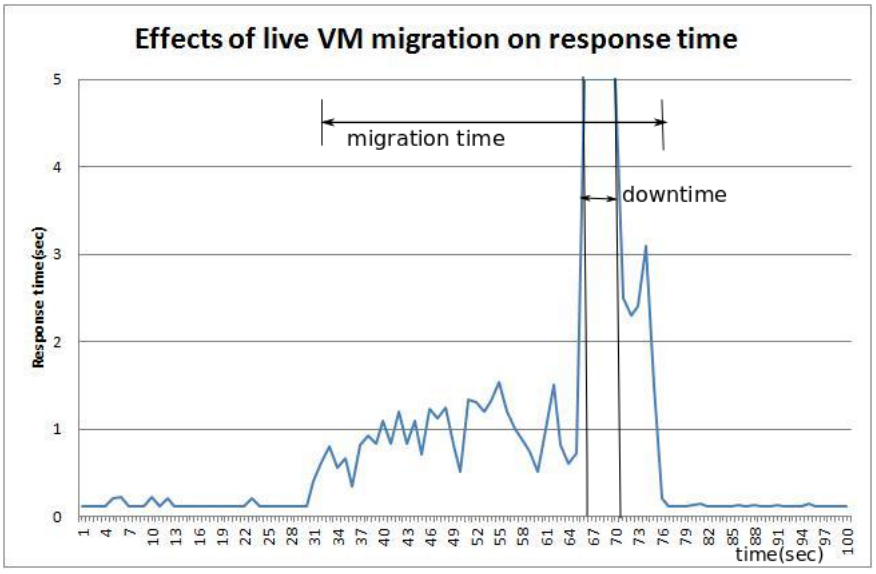

Figure 2.7: Effects on response time of an application running on migrating VM due to migration

The effect of VM mkigration process on response time of an application running on VM to be migrated is shown in Fig. 2.7 and the effects on respose time of other application running on other VM is shown in Fig. 2.8. The readings are taken every 4 sec interval. The total migration time required for migration was 45 Sec. and downtime experience was 2 Sec. Response time shows highest peak immediately after

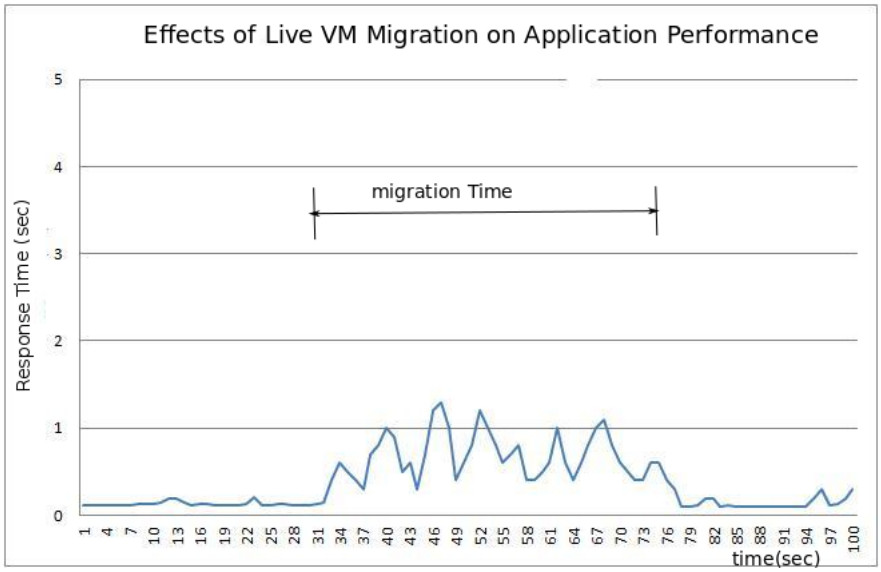

Figure 2.8: Effects on response time of an application running on different VM due to migration

the migrating VM resumed at destination PM as all the requests are queued and remain to be served due to downtime. Then the response time gradually decreases as the requests gets served from the VM.

Dynamic resource provisioning in cloud creates following challenges -

1. How to make resource allocation to the VM elastic so that resource scaling for dynamic need of application can be economical and the defined SLA can be satisfied.

2. How do predictive models to estimate the future resource demands for proactively allocating resources so that the applications performance will not be degraded at the peak demands.

3. How to provision resources to the application mix in the data center in a way with which SLA of all such applications will be met.

2.9 Summary

This Chapter explained the preliminaries required to understand the domain of the work, followed by challenges in dynamic resource provisioning on which the thesis work is carried out.

Chapter 3

Literature Survey

3.1 Resource Allocation Schemes

The resource allocation schemes can be broadly classified into categories like off-line/ on-line profiling, rule based, model Driven, control theory based, trace-driven, reinforcement learning based and fuzzy logic based.

3.1.1 Offline/ Online profiling

Initial work related to resource management used offline and online profiling to determine application requirements using real application workloads or benchmarks. This takes longer time to determine the resources and needs extra computing machines as well. Hence it takes longer time. Urgaonkar et al. (2002) derived accurate estimate of application resource needs by profiling applications on dedicated nodes and then used these profiles to place the applications on shared nodes. Govindan et al. (2009) used measurement driven profiling and prediction framework to characterize key statistical properties of the power needs of hosted workloads and their aggregates. (Wood et al., 2008) used set of microbenchmarks on a given platform to profile the different types of virtualization overhead. He also used a regression-based model that maps the native system usage profile into a virtualized one. In (Zheng et al., 2009) a sandboxed environment created by the the infrastructure where experiments can be run on a very small number of machines using real system state and real workloads. Automated management systems or the system administrator exploits the infrastructure parameters to perform management tasks on the on-line system.

3.1.2 Rule based Resource Allocation

This scheme uses a set of rules, expressed as "event-condition-action" rules which are taken into action when given condition is satisfied. These rules are defined by system experts. For example, the HP-UXWorkload Manager (HPUX, 2012) allows relatively controlled CPU utilization within the user specified limits. Rolia et al. (2006b) monitors resource utilization of an application on VM and uses some predefined thresholds to evaluate the current allocation is suitable for the workload at hand. But due to the mix of applications and the complexity of the cloud, it is difficult to decide these thresholds and take appropriate corrective actions for all possible states of the system.

3.1.3 Model Driven Resource Allocation

There were attempts to propose model-driven resource allocation schemes. These schemes mostly use statistical leaning methods or queueing theory to build models to evaluate the impact of different resource allocation schemes on the application performance running on VM. Shivam et al. (2006b) proposed an machine learning approach which analyses frequently used application performance histories to build predictive performance models which can evaluate their future performance. These models are used to decide the task placement. The histories collected consist of readings from previous runs which include the readings of cpu, network, storage resources. It used linear regression to predict application performance and the completion time with allocated resources. Shivam et al. (2006a) presented NIMO (NonInvasive Modeling for Optimization) that automatically learns cost models for predicting task execution time on a heterogeneous environment such as computational grids consisting of large scale networked utilities. It does active sampling of resource assignments. To build accurate cost models NIMO generates appropriate training samples by monitoring the application under varying conditions and used statistical learning techniques for learning. Accurate cost models are required for selecting efficient plans to carry out execution of these applications on the grid.

Stewart et al. (2008) presented an approach which is driven by a performance model and used for guiding cross-platform management for real-world Internet services. It predicts the application performance when there is a change in resource al-

location. The model is created with several empirically observed sub-models. It does the platform-aware load balancing in heterogeneous cluster in the cloud. Ganapathi et al. (2009) designed a system using machine learning which predicts the performance metrics such as execution time of the database queries. Thus it gives the performance of the query before it is actually executed in the environment. This helps to schedule it at appropriate time for execution and capacity planning. For training and testing, several queries are ran on an HP Neoview four processor system and multiple configurations of a Neoview 32 processor system. It actually finds multivariate correlations among the query properties and query performance metrics. It then uses this relationship to predict performance metrics of queries varying in execution time. Doyle et al. (2003) also used model based approach to predict the amount of resources(memory and storage) to be allocated under changing load. It creates a models capturing application workload and its running behavior with which the system is able to predict the effects of changes to the workload intensity or resource allotment.

Chandra et al. (2003) captured a transient behaviour of the application workloads. The model is created which relates dynamically changing workload characteristics to their application resource requirements. The parameters of the model are continuously updated by continuous monitoring and learning. The expected workload characteristics are predicted with measures system metrics. Prediction is carried out with time series analysis. Then the server resources allocated based on estimated application requirements. This is basically designed for allocating resources to web services running on the web-server.

In Sha et al. (2002) timing performance of a network server(Apache Server) is controlled and kept close to the service level specification using queueing model based feedback control. Combining feedback control with queueing models gave better tracking of quality of service specifications than with alone. Urgaonkar et al. (2005a) analytically modeled the behaviour of multi-tier applications. He developed a model using network of queues. The queues are the representations of different tier of an application. It captures the behaviours of the tiers of application with significantly different performance characteristics. It can also capture session based workloads and also used for tier replication ,identification of load imbalances across replicas. Xu et al. (2006) presented predictive controller model using three different prediction

algorithms. Machine learning models used in these are autoregressive(AR), a combined ANOVA(Analysis of Variance)-AR. He compared adaptive control models with predictive models for managing resource utilization. These control models are evaluated with CPU traces in hypothetical virtual server environment. N. Bennani and A. Menasce (2005) proposed a solution for dynamically redeploying servers among various applications environments in the data center where workload intensity varies widely and unpredictably. The solution addressed the scalability limitations of previous approaches. His solution is based on the use of multiclass open queues which forms analytical queuing network models combined with combinatorial search techniques. Liu et al. (2005)

All these resource allocation models incurs overhead and needs to prior knowledge about the characteristics of the virtualization platform and the applications running on it. (e.g., cpu speed, input data size) which is practically not possible in cloud environment. On the other hand, our proposed system completely free from such requirement i.e. platform and application agnostic.

3.1.4 Control Theory based Resource Allocation

Zhu et al. (2008) designed dynamic resource management architecture which can satisfy SLA in changing data center conditions. It hides the complexity between application owner and data center operator. Thus application owner can concentrate on the SLA for their applications. Control theory based resource controller is designed. Individual interfaces are given to coordinate individual controller so that their cannot be duplicate policy among controllers. Advantage is taken by combining these three controllers.

Kalyvianaki et al. (2009) presented a resource allocation scheme that combines Kalman Filters and feedback controller. Kalman Filter is used for state estimation to track CPU utilizations and update the resource allocations accordingly, thus guiding the resource allocation.

Padala et al. (2007) designed feedback-driven resource control system that dynamically allocates the CPU resource shares to individual tiers of an application running on VMs. It allocates the CPU resources in order to meet the respective application level QoS and there will be efficient CPU resource allocation. Parekh et al. (2002)

used statistical ARIMA model to fit historical measurements of the target being controlled. The controller will achieve service level objectives. Padala et al. (2009) presented *AutoControl*, a resource controller which adapts to the dynamic resource requirement changes to met the defined SLA . It is a combination of a multi-output (MIMO) resource controller and online model estimator. The model estimator finds out the relation between resource allocation and application performance. The required amount of required resources are allocated by MIMO controller to met the SLAs.

Thus previous work has applied control theory for fine grained resource allocations based on SLA feedbacks. However, these approaches need their parameters to be specified and tuned before hand. In contrast, our system are application and platform agnostic and prior tuning is not required.

3.1.5 Trace Driven Resource Allocation

Rolia et al. (2006a) build the workload manager which monitors its workload demand and accordingly adjust the CPU allocation to the workloads with a aim of allocating each with the capacity it needs. It performs this dynamic allocation using *burst factor* times the most recent estimated resource demands. The *burst factor* is calculated off-line based on different QoS levels. Chandra et al. (2003) captured a transient behaviour of the application workloads. The model is created which extracts the relationship in application resource requirements and workload change. Online monitoring is used to update the model parameters. The expected workload characteristics are predicted with measures system metrics. Prediction is carried out with time series analysis using auto-regression and histogram based methods. Then the server resources allocated based on estimated application requirements. This is basically designed for allocating resources to web services running on the web-server. These methods are time consuming.

Gmach et al. (2007) gave a trace driven approach to capacity management that relies on auto-correlation and Fourier transform to perform offline extraction of cyclic patterns in the workload. Our work does not assume whether the workload is cyclic or acyclic. It predicts the resource demands for both cyclic and acyclic patterns. This approach is suitable for long periodic intervals and assumes that repeating periods

are known in advance. AR requires more computation time hence it is not suitable for short-term resource scaling.

Gmach et al. (2008) proposed an integrated approach for VM placement using fuzzy feedback controller and peak demand prediction. Two separate fuzzy logic controllers are build to identify overloaded and underloaded situations.

Gong et al. (2010a) proposed PRESS that identifies dynamic patterns in application resource demands and allocates the resource accordingly. It used Fourier Transform and Markov Chains to achieve this. It handles both cyclic and non-cyclic types of workloads. In contrast, our approach efficiently handles under-estimation errors specifically due to spikes in the workload and concurrent scaling conflicts due to insufficient resources.

3.1.6 Reinforcement Learning based Resource Allocation

Tesauro (2005) used reinforcement learning for automatic allocation of resources. Reinforcement learning generally prepares a table to store the information it receives through training and used it to look into. Its size grow rapidly as the number of state parameters increases. Reinforcement learning consumes more time for training as it does not have the domain knowledge base. Whereas, fuzzy prediction system stores the knowledge base in terms of fuzzy rules which requires less space. Fuzzy prediction system training takes less time.In Tesauro et al. (2007), proposed a combination reinforcement learning and queuing models, where reinforcement learning trains on data collected offline and a queuing model based policy control mechanism improves training time.

3.1.7 Fuzzy Control based Resource Allocation

Diao et al. (2002) presented feedback control mechanism based on profit to improve SLA attainments in webserver system. It uses fuzzy control atomizing admission control decisions so that it balances loss due to rejected work and penalties due to more response time.

3.1.8 Other Methods for Resource Allocation

Bonvin et al. (2011) proposed adaptive adjustment of cloud resource allocation in

order to satisfy application response time and SLA. This adaptive adjustment is achieved through detection and removal of stale cloud resources, component replication and migration for accommodating load variations and to support load balancing. It does cost-effective resource allocation and component placement for minimizing operational cost of cloud application. The load prediction is absent here. Sladescu et al. (2012) explained how workload burst are associated with important events. He proposed event aware prediction that can leverage this association. But basically it needs the knowledge of upcoming events.

Islam et al. (2012) used neural networks and linear regression model with sliding window techniques for workload prediction. But linear regression is not suitable for all types of workload. Generally for workloads which varies non-linearly this technique is not useful.

Roy et al. (2011)and Calheiros et al. (2015) used Auto-Regressive Moving Average method (ARMA) for workload prediction. Hu et al. (2016) Kalman Filter model for prediction purpose. Yang et al. (2013) proposed prediction using linear regression model and ARMA model. Auto-scaling methods are divided into three categories-self-healing scaling, resource-level scaling and VM-level scaling. First two are vertical scaling and later is horizontal scaling.

Morais et al. (2013) proposed multilayer controller and each layer use different predictors like auto-correlation (AC), Linear Regression (LR), Auto-Regression (AR), Auto-Regression with Moving Average (ARMA) and Ensemble (EN). Reig and Guitart (2012) proposed a method to anticipate CPU demands for web application. This combines statistical and machine learning techniques to predict CPU requirements.

Bankole and Ajila (2013), Nikravesh et al. (2015) used Support Vector Machine, Neural Networks and Linear Regression methods for workload predictions. SVM found suitable for periodic and growing workloads. NN found suitable for forecasting unpredicted workload patterns. Li et al. (2012) identified daily workload patterns by applying Predictive Bayesian Network Model. He has given algorithm for consolidating heterogeneous applications into smaller number of servers. Jheng et al. (2014) proposed Grey Interval Forecasting based on Grey System Theory. proposed load prediction algorithm by combining linear programming with improved Knuth-Morris-Pratt(KMP) string matching model. The system was able to predict the uncertain

facts, estimates/judges the future tendency of undecided case in the system. Liu et al. (2016) proposed novel Ensemble Workload Prediction system for predicting Job Submission Number in Google Cluster Trace.

Automatic resource management in virtualized environments applied control theory like fuzzy logic to deal with time-varying workloads the build models are automatically updated to the current changes done. Our works adds Kalman Filter based on Chaos indicator by analyzing the pattern in workload. PRedictive Elastic Resource Scaling (PRESS) by Gong et al. (2010b) for cloud systems achieved online prediction. Signal processing techniques are used to discover signature. We have used Hurst Exponent based chaos indicator to decide a specific controller to be used. In CloudScale (Shen et al., 2011) predicts online resource demands. It also handles prediction errors although it does not assume any prior knowledge about application. Rolia et al. (2006b) derives a burst factor offline and multiplies it with estimated resource usage to dynamically decide resource allocation. Ganapathi (2009) used queueing theory and statistical methods for building models of prediction. But these methods assume prior knowledge about applications.

Xu et al. (2016) proposed a double auction-based model for resource allocation algorithm in cloud environment and a model for pricing mechanism. They considered the profit of the middle auctioneer. The model adds reliability by determining the the price for the CSPs based on the price of the resource and the reliability index.

Bose et al. (2011) proposed VM image replication mechanism and scheduling for optimizing live VM migration over wide area networks. Replication of VM image is based on the cost of storage and computation units at different cloud apart from the latency requirement of the end user. It chooses the replica of a VM image as a primary copy and propagate any changes in the primary to the other replicas. They proposed to reduce the extra storage requirements due to replication by finding out common features in VMs using de-duplication techniques.

Pillai and Rao (2016) used principles of coalition formation and the uncertainty principle of game theory and proposed a resource allocation mechanism in cloud environments. The knowledge of type of requesting VM is used to form coalitions of machines. According to this coalition it is hosting the VMs besides having availability of actual request. It gives higher payoffs to that machines that are close in proximity

while forming coalition. It is more advantages in running hadoop applications. It gives better resource utilization with higher request satisfaction.

Ma et al. (2014) presented five major topics in resource allocation and scheduling, namely locality-aware task scheduling; reliability-aware scheduling; energy-aware resource allocation and scheduling (RAS); Software as a Service (SaaS) layer for resource allocation and scheduling; and workflow scheduling.

Ganesh S et al. (2016) presented challenges in implementing proactive resource management in data centers. It suggested analytics engines to collect resource usage readings, apply statistical learning methods for future resource demand prediction. This work mainly presented the issue of dealing with mis predictions. It also suggested load balancing in the data center.

Hoyer et al. (2011a) addressed the challenges of estimating the resource future demands of VM in order to reduce delays in VM migration and need of the scheduling algorithm based on current distribution of VMs to server and provides guarantee of finding steps to resolve the resource shortage if the actual resource demand of the VMs meets the expected one.

Mahdhi and Mezni (2018) proposed a VM consolidation approach based on future estimation of requested resources and VM migration traffic. He used Kernel Density Estimation technique (KDE) as a powerful mean to forecast the future resource usage of each VM. Migration traffic between PMs is modeled with a weighted-graph representation

Dabbagh et al. (2015) proposed framework: i) predicts the number of virtual machine (VM) requests, to be arriving at cloud data centers in the near future, along with the amount of CPU and memory resources associated with each of these requests, ii) provides accurate estimations of the number of physical machines (PMs) that cloud data centers need in order to serve their clients, and iii) reduces energy consumption of cloud data centers by putting to sleep unneeded PMs.

Chou et al. (2018) presented the dynamic power-saving resource allocation (DPRA) mechanism based on a particle swarm optimization algorithm. It used the least squares regression method for predicting PMâĂŹs resource utilization for allocating VM and avoiding VM migrations.

Adami et al. (2015) proposed two Fuzzy Inference Systems for data center resource

allocation. These are based on Mamdani and Sugeno inference processes. It has taken the advantage of heuristic rules for efficient virtual machines allocation.

Li et al. (2017) proposed monitoring data in the data center. It used deep reinforcement learning (DRL) framework to design cooling control policy into an energy cost minimization problem with temperature constraints. They proposed an end-to-end cooling control algorithm (CCA).

Tang et al. (2019) proposed the dynamic resource allocation scheme for cloud environment which consists of the resource scheduling algorithm and the resource matching algorithm. In resource scheduling algorithm following points are considered- penalty of scheduling contents, the value of scheduling contents and the transmission cost of scheduling contents. In the resource matching algorithm following points are considered- the resource location, the task priorities and the network transmission cost. For optimal matching problem complete bipartite graph is used.

Tseng et al. (2018) proposed a multiobjective genetic algorithm (GA) to predict the future resource utilization and energy consumption in cloud data center. This is designed as a multiobjective optimization problem of resource allocation. It considers CPU andd memory resource utilization of PM and VMs on it. It also consider the energy consumption of data center.

Yang et al. (2016) modeled the problem of minimizing energy consumption as a Stackelberg game using game theoretic approaches. The profit can be maximized by adjusting resource provisioning. They gave a model to minimize average response time of tasks as a non-cooperative game among agents.

Table 3.1: Resource allocation schemes

Name of the Scheme	Functioning
Offline / Online Profiling Urgaonkar et al. (2002),Govindan et al. (2009) Wood et al. (2008),Zheng et al. (2009)	Experimentally derived application resource requirements using benchmark or real application workloads. However, profiling needs extra machines and may take a long time to derive resource requirements.
Rule based Resource Allocation HPUX (2012) Rolia et al. (2006a) Adami et al. (2015) Xu et al. (2008a)	This approach uses a set of event-condition- action rules which are provide by the system that are triggered when some precondition is satisfied (when some metrics exceed a predefined threshold)
Model Driven Resource Allocation Xu et al. (2016) Mahdhi and Mezni (2018) Ganapathi et al. (2009)Mukherjee et al. (2010) Ganesh S et al. (2016)	These approaches use queuing theory or statistical learning methods to build models that allow the system to predict the impact of different resource allocation policies on the application performance. The model needs to be calibrated in advance with prior knowledge
Energy Efficient Resource Allocation Dabbagh et al. (2015) Chou et al. (2018) Al-Qawasmeh et al. (2015)	Approaches based on minimizing the energy consumption of the data center provides accurate estimation of the resources.
Game Theory based Resource Allocation Pillai and Rao (2016)Xu et al. (2016) Yang et al. (2016)	Used principles of coalition formation and uncertainty principle of game theory, minimizing energy consumption as a Stackelberg game resource demands to guide resource management decisions
Proactive Resource Allocation Ganesh S et al. (2016) Hoyer et al. (2011b) Mahdhi and Mezni (2018)	Proactive resource allocation look into the future and take the action beforehand of any spike coming in the workload,

The proposed work contrasts with the earlier work described in the literature survey.

(1)Resource allocation controller is present in every PM and central migration manager is present to take the decision about resource scaling using VM live migration This makes the design more flexible. The central migration manager takes the resource utilization information from all other and take the decision of migration. (2) Fuzzy-prediction system used in our approach represents the relationship between system variables so that multi-step ahead prediction can be achieved . (3) It is not assuming any type of prior knowledge about characteristics of the application. It learns the relationship between input-output parameters very fast. Hence this can be applicable to any type of application hosted on virtual machines. Fuzzy prediction system can effectively model the non-linearity with dynamically changing workload. (3) It is fully automatic as the fuzzy rules are generated automatically from the data monitored from the system. Rule base is continuously updated when the new relationship found.

Virtual machine migration (Clark et al., 2005) is widely used for dynamic resource provisioning. Sandpiper (Wood et al., 2007) system automates the process of monitoring resource demands and detecting hot-spots, decides the new placement for the VM to resolve it. This new placement decision is achieved using live VM migration. It uses the VM resource usage readings (blackbox approach) and the readings from the operating system and DomU (graybox approach) to decide the resource allocation decisions. The migration is initiated when certain load exceeds threshold for a sustained predefined time and new predicted value of the resources also exceeds the threshold. The proposed system does the long-term prediction to predict the future requirement of VM migration and does this proactively before the actual need. Also it handles underestimation errors.

3.2 Modeling Live VM Migration

Data-center management includes several several management works such as mitigating resource hot-spots (Wood et al., 2007), load balancing in the data center Gong et al. (2010a), consolidation of VMs for saving resources and intern energy Varasteh and Goudarzi (2017), moving virtual machines across cloud locations for cloud burst-

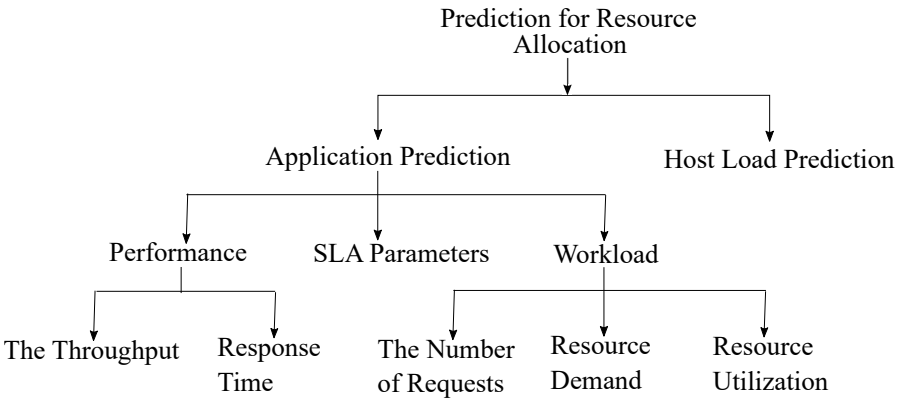

Figure 3.1: Different levels of prediction for Resource Allocation

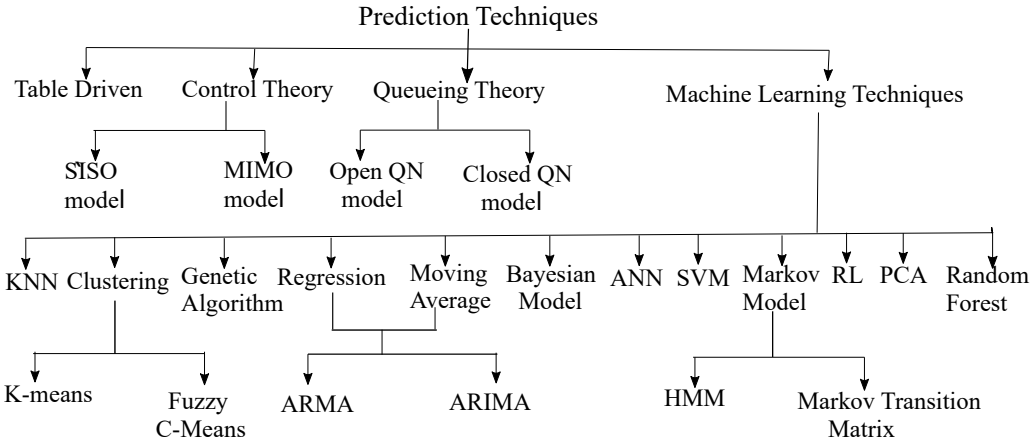

Figure 3.2: Taxonomy of prediction methods

ing (Guo et al., 2014) and moving the portions of the data center to other location because of maintenance purpose (Singh et al., 2013). These tasks involve migration of VMs from one PM to another. They live migration technique explained in (Clark et al., 2005) performs this live VM migration. An optimized technique called the page skip (Nathan et al., 2013), skips transfer of pages which are dirtied frequently. This avoids unnecessary data transfer. There are many models exists for calculating performance and cost of VM migration which can be used to effectively manage the data center. There are several models defined to predict VM migration time for KVM (Aldhalaan and Menascé, 2013) (Deng et al., 2013) (Li et al., 2014) (Liu and He, 2015a) (Mann et al., 2012) (Nguyen et al., 2013) (Xu et al., 2014) (Zhang et al., 2014) (Zheng et al., 2013) and Xen (Akoush et al., 2010) (Aldhalaan and Menascé, 2013) (Liu et al., 2011) (Wu and Zhao, 2011). In all these proposed models of VM live migration, some or all the performance(total migration time, downtime) and cost parameters(amount of network traffic generated) are taken into consideration, but the

cost of interference with other running VMs is not considered. The addition of cpu and network interference during the migration process itself is considered in the our proposed model.

Migration time estimation model for cloud bursting are given by (Guo et al., 2014) Williams et al. (2012) Liu and He (2015b) Shrivastava et al. (2011) Sonnek et al. (2010) Sudevalayam and Kulkarni (2013)

Hence we would like to propose an accurate model for estimating VM migration time.

3.3 Resource Allocation using live VM Migration

Dynamic resource provisioning techniques (Bila et al., 2012) Choi et al. (2008) (Das et al., 2010) (Heo et al., 2009) (Jeong et al., 2013) (Kumar et al., 2009) (Mishra et al., 2012) (Padala et al., 2009) (Salomie et al., 2013) (Williams et al., 2011) (Wood et al., 2007) (Xu et al., 2008b) allocate resources dynamically to a VM based on changing load levels.

3.4 Summary of Literature Survey

It has been found that the prediction methods are used to determine future resource demand but type of workload need to be considered while applying the technique. Also if migration is triggered then its interference is need to be considered which will affect the performance of other running VMs. The proposed work uses fuzzy rule based prediction and Kalman Filter which can effectively model nonlinear system with dynamically changing workload. It takes the decision of applying fuzzy or kalman filter base prediction based on Chaotic indicator determined using hurst exponent which represents the index of dependence of CPU variations. It additionally does underestimation error corrections which improves accuracy of the system. Most of the performance and cost parameters are considered while selecting VM for migration. The CPU and Network interference effects are evaluated and taken into the decision of migration. The need for migration is predicted beforehand and accordingly resources are allocated hence it improves the efficiency of the proposed system. The prediction depends on incoming online workload which makes system fully autonomous.

Chapter 4

Virtual Machine Resource Demand Prediction

Tremendous increase in web services and on-line information exchange has increased the number of enterprise data centers. Virtualization in data centers provides resource sharing among many applications. Applications of different vendors are running on virtual machines. Data centers acting as resource provider for the applications to run on VMs, should provide performance guarantees as per SLA, while optimizing utilization of cloud resource to reduce the cost. Resource requirement of the applications running in cloud is changing dynamically. Hence it necessitates dynamic resource allocation to the running applications so as to achieve desirable performance with efficient utilization of resources. To allocate the resources to the application as per their dynamic needs, it is necessary to determine the amount of resources needed by the application and accordingly request for these resources. If future needs are predicted based on recent resource utilization, the resource allocation can be prompt and effective. This chapter proposes local resource allocation controller which dynamically allocates system resources to the VM as per the dynamically changing requirement of the application to run. It uses fuzzy prediction system to determine future resource requirements of an application base on the observation of recent resource utilization.

4.1 Introduction

The first component of the resource allocation controller is the application workload prediction module. Data centers host many different types of application. Sometimes the workload is measured in terms of request rate. But it is very difficult to define

workload of an application as it does not give information about how much resources each request is consuming. Hence, in general, it is better to directly measure the resource consumption by monitoring system level metrics which gives exact utilization of hardware resources like cpu, network, and memory. The basic idea is to predict future resource needs by observing past resource usage. The prediction approach does not make any assumption about the application behavior or it does not have any prior knowledge about the same. The contributions in chapter are Monitoring system, prediction algorithm, and a trace-based analyses of the proposed approach to show how well it works.

The monitoring system monitors three different resources viz. CPU, Network, and Memory. The fuzzy system is designed to predict the future resource requirements based on the observations of the past history resource usage. Later the algorithm is analyzed using real workload traces to show how efficient the the predicted results are.

4.2 Resource Monitoring

The management domain, i.e. Dom-0 of every Xen hypervisor, contains the resource monitoring system, which captures CPU, Memory and Network bandwidth utilized. This system collects and sends reports to Resource Usage Collector in the time interval of 10 Sec. In the proposed work 300 such values of every resources have been stored.

4.2.1 CPU Monitoring

Dom-0 of Xen Virtual Machine Monitor (VMM) captures the CPU scheduling events. It tracks the time, when the VM is scheduled to get CPU and when it is released from the CPU. With this, the CPU resource utilization of all the VMs running on the physical machine is captured.

Dom-0 processes the I/O and Networking requests on behalf of VMs. The CPU requirement for this is charged on individual VM depending on how many requests are served for the particular VM. This CPU monitoring is done with a 10sec time interval. Xenmon tool, which comes along with Xen, works on the above specified principal. Xenmon is modified to get the required results.

4.2.2 Network Monitoring

Dom-0 in Xen implements Virtual Firewall Interface (VFR) (Clark et al., 2005) interface, which other domains access through clean device abstraction. Each VM attaches one or more virtual interfaces to VFR. The networking statistics is gathered in, /proc/net/dev file which can be accessed and the network utilization is captured in the specified time interval which is configured to 10sec in proposed case.

4.2.3 Memory Monitoring

Xen maintains shadow page tables (Clark et al., 2005) to track memory accesses of the guest VMs. It is translated from the guest page table, on demand. It is specifically used by virtual machine migration algorithm to determine which pages are dirtied during migration. But trapping each memory access adds significant overhead. Hence, the memory usage is inferred as memory pressure exists or does not exists, by tracking number of read/write to swap partitions on NFS disk. This work has used xentop tool to get the memory usage at 10sec interval.

These readings are stored in two ways. First is the time series of individual resource usage of individual VM; collective load of individual resource of all VMs on one PM. Second is the probability distribution of usage, in which histogram of all observed usage of individual resource within a specific interval, is computed and normalized to get this distribution.

4.3 Fuzzy Prediction System

Fuzzy logic based prediction system gives a basic way of representing the relationship between different variables of the system. This only relates the variables without need for the knowledge of application characteristics. The system is shown in Fig. 4.1. The fuzzy rules are generated by observing the monitored data. The monitored data is divided into input space and output space. The fuzzy rules draw the mapping from input space to output space.

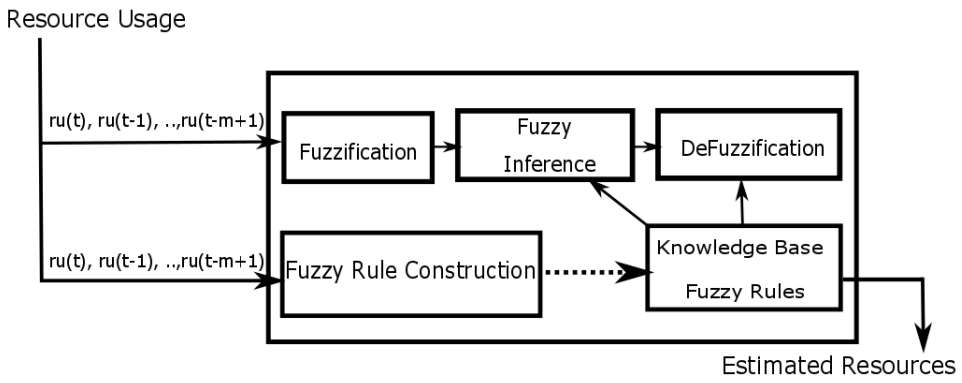

Figure 4.1: Fuzzy Prediction System

4.3.1 Fuzzy Rule Construction

The important block in the system is fuzzy rule construction. The fuzzy rule is created on-line using monitored data. The input-output data pair is extracted from the monitored data. One component is input sequence of monitored data pair followed by output subsequence as shown in Fig. 4.2. Let $C_{i,t}$ (where t = 1,2,3,....) be the

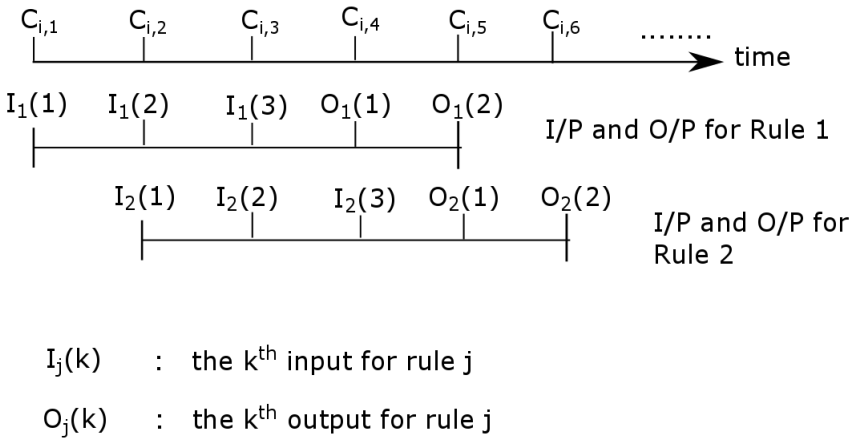

Figure 4.2: Fuzzy rule construction with 3-input, 2-output data sequence

CPU resource usage at time t of VM i. The input to the fuzzy prediction system is the recent m CPU usage measured $C_{i,t}$, $C_{i,t-1}$,....,C_{t-m+1}. Then fuzzy prediction system predicts future resource needs as $C_{i,t+1}$,$C_{i,t+2}$,.....,$C_{i,t+n}$, where m are the number of

inputs and n are number of outputs of a fuzzy rule. A fuzzy rule is created using input-output data pair. Fig. 4.2 shows fuzzy rules with three input and two output. Assuming that the cpu usage readings are normalized between 0 to 1, each space is divided into $2N+1$ fuzzy sets, denoted by $F1$,$F2$, ...,F_{2N+1} where N is total number of inputs and outputs. Each fuzzy set is assigned with fuzzy membership function with N=5 where the membership function is the triangular membership function. The given CPU reading data points are mapped to fuzzy set with highest membership degree. e.g. input I_1 is in fuzzy set F_6 and output O_1 is fuzzy sete F_10. The fuzzy rule is constructed from input-output data. The rule i represented as :

IF I_1 is F_{I1} and I_2 is F_{I2},.....,I_m is F_{Im}

THEN O_1 is F_{O1} and O_2 is F_{O2},.....,O_m is F_{Om}

At every sampling instance of monitoring a new rule is constructed with m input and n output. It is compared with the already stored rules. If it matches with anyone, its count is incremented, otherwise stored in the rule base. If newly constructed rule has the same IF part but different THEN part, then the rule, whose count is greater, is activated, the others are then deactivated in the rule base. Thus the total number of rules are limited by the number of fuzzy sets. The first rule is constructed after initial $m + n$ resource usage measurements.

The fuzzy inference block of the system takes latest resource measurements as shown in Fig. 4.1; looks into the rule base and gives output for future resource usage. In fuzzification block, the input data points are mapped to fuzzy sets with fuzzy membership functions. The defuzzification block aggregates the fuzzy outputs and gives a numeric value.

4.3.2 Updating Fuzzy Rule

At every sampling time a fuzzy rule is generated and if each of these is stored in the rule base, the memory required for storage is more. The rules have m inputs and n outputs. There may me conflicting rules generated. The conflicting rule have the same IF part and different THEN part. The input-output space is partitioned into (m+n) domains and there is, at most one rule for every domain. Hence the total number of rules in the rule base does not cross the total number of domains available.

To handle the conflicting rule problem, the a reliability index is defined for each

rule denoted as R_i = the number of occurrences of rule i. When a rule is generated by the fuzzy system, it is compared with existing rules. If it matches with the existing one, its reliability index is incremented by one, else, a new generated rule is added in the rule base with reliability index one. If conflicting rule exist in the rule base then the rule with a higher reliability index is taken for action. If conflicting rules have same reliability, then the most recent one is taken for action.

4.3.3 Fuzzy Inference Engine

When latest resource usage is given as an input the fuzzy inference engine looks into the rule base to activate a matching fuzzy rule. Output of the activated rules determines the output, which is a future resource demand. Initially the rule base is empty. The first rule is generated after the first (m+n) resource measurements and subsequently a rule is generated at every sampling point. The decision of keeping the rule 'in rule base' is taken based on the above criteria. Thus rule update procedure is automated and self learning.

4.4 Kalman Filter based Prediction

If a very noisy and chaotic signal is seen as indicated by an chaos indicator, it switches to Kalman Filter based prediction, otherwise switches to Proactive Resource Demand Prediction. The SISO Kalman Basic Controller (KBC) as described in Gong et al. (2010b) is implemented in this work. This controller is based on Kalman filtering technique as shown in Fig. 4.3. Kalman filter is used as utilization tracking controller and not for estimating the parameters of the application performance model.

4.5 RNN with LSTM based Prediction

Over the past few years deep learning has become very prevalent and it found to be applicable in a wide range of areas such as, speech processing, computer vision, natural language processing, etc. A specific type of neural network called Recurrent Neural Networks (RNNs) is found to be applicable in the areas where there is need to deal with sequences. Sequences in NLP are sentences made up of words; sequences in speech processing are sequence of phonemes; video is the sequence of images. When

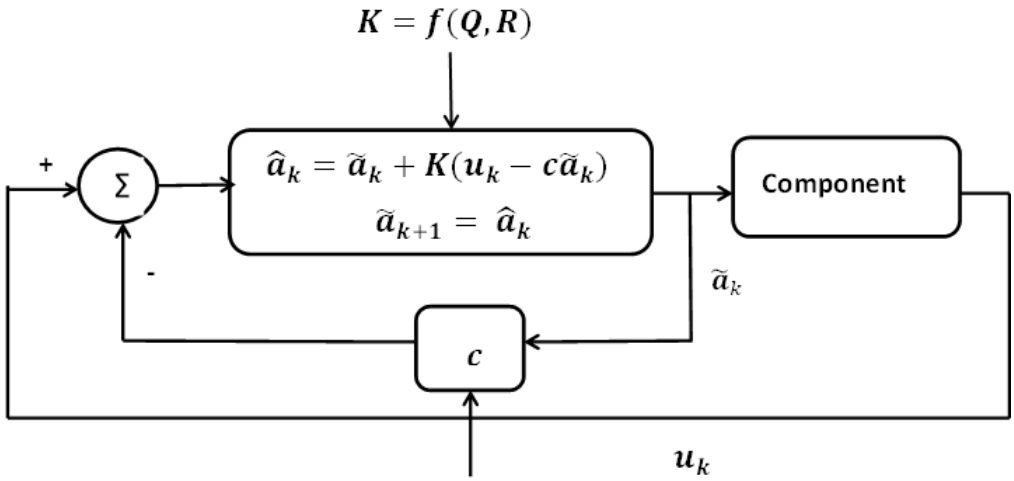

Figure 4.3: Basic Kalman Controller

there is need to deal with sequential data and to do various things on it such as classification, sequence prediction the recurrent neural network (RNN) is required. The resource usage prediction in this work is also a problem related to sequence of resource usage. Hence RNN can be utilized for future resource prediction. To overcome the challenges faced during training of RNN, a special kind of RNN, which is Long Short Term Memory networks (LSTM) and Gated Recurrent Units(GRUs) is used. The inspiration to use RNN-LSTM network is to predict the future resource usage in the data center environment.

4.5.1 The Model : LSTM Network

The LSTM network model consist of several memory cells which include internal state variables and internal gates. The gates operates on internal state to do selective read, selective write and selective forget operations. The basic schematic digram of LSTM unit is shown in Fig. 4.4. The state S_i of RNN record information from all previous time steps.The state variables of previous cell are transformed to state variables of next cell by using gate operations. There are three operation gates.

(1) Forget Gate:

It does the linear transformation of the previous output i.e. h_{t-1} at time $t-1$ and current input (x_t) at time t with the learned parameters W_f, U_f and b_f of this gate. Then sigmoid activation function is applied on it to calculate the forget gate output. This gate value is applied on the previous state (S_{t-1}) at time $t-1$ to calculate the

state S_t at time t. This operation is given by-

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4.1}$$

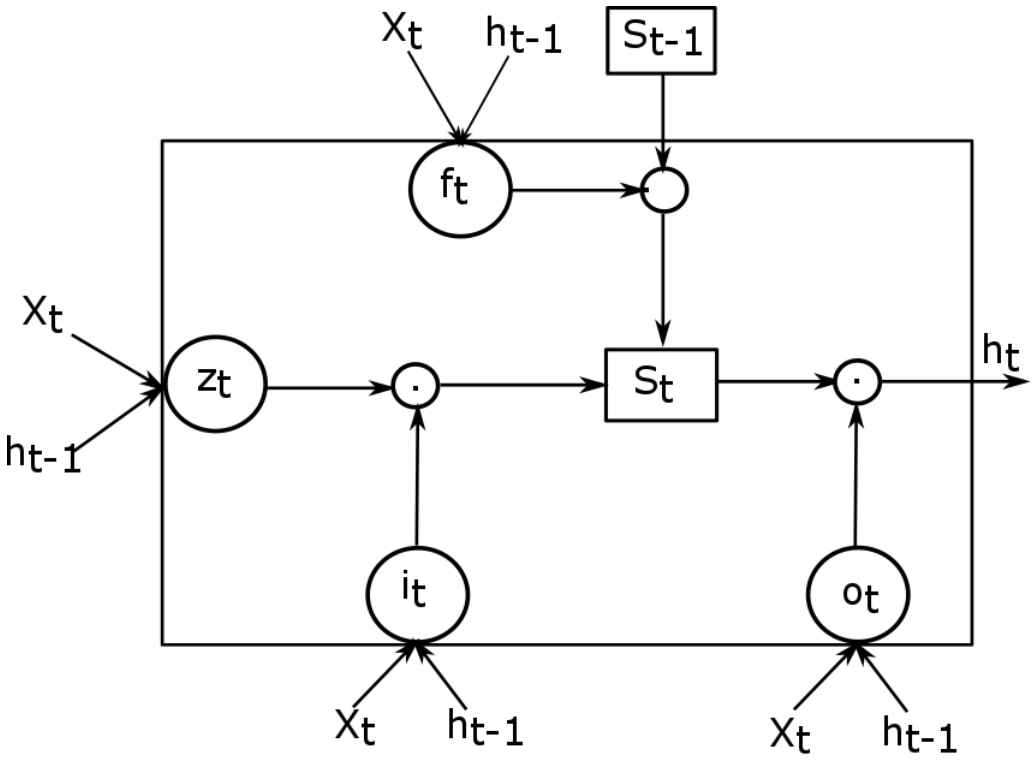

Figure 4.4: LSTM Cell Structure

(2) Input Gate:

It does the linear transformation of the previous output i.e. h_{t-1} at time $t-1$ and current input (x_t) at time t with the learned parameters W_i, U_i and b_i of this gate. Then sigmoid activation function is applied on it to calculate the input gate output. The value of the input gate is given by

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4.2}$$

The intermediate cell output is multiplied with this gate output to estimate the final state at time t. The intermediate cell output at time t is given by-

$$z_t = g(W_z x_t + U_z h_{t-1} + b_z) \tag{4.3}$$

The internal state of the cell at time t is calculated as:

$$s_t = i_t \odot z_t + f_t \odot s_{t-1} \tag{4.4}$$

(3) Output Gate:

Output gates decides how much of the internal state is passed to the output of the cell. It does the linear transformation of the previous output i.e. h_{t-1} at time $t-1$ and current input (x_t) at time t with the learned parameters W_o, U_o and b_o of this gate. Then sigmoid activation function is applied on it to calculate the input gate output. It is given by equation-

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4.5}$$

The final output state is calculated as follows-

$$h_t = o_t \odot h(s_t) \tag{4.6}$$

Here h_{t-1} and s_{t-1} are the output vector and cell state at time $t-1$. W_i, W_f,W_o, W_z, U_i, U_f,U_o, U_z,b_i, b_o,b_f,b_z are learning parameters weights and biases at the respective gates. Thus RNN learns with LSTM, how much the past output to be considered, how much of the input to passed and how much from the previous state to allowed to the next state after forgetting. The $\odot$ mark means pointwise multiplication. The $\sigma(x)$, $g(x)$, and $h(x)$ functions are the activation functions of every part in LSTM, which determine the amount of information that can be passed. We used sigmoid as the activation function of three gates ($\sigma(x)$ in the formulas) and $g(x)$. The rectified linear units (ReLUs) function is used as the function for $h(x)$. ReLU function is a very popular new nonlinear activation function and it is defined as follows:

$$h(x) = max(0, x) \tag{4.7}$$

The ReLU activation function can make the training faster than using equivalents with saturating neurons like *tanh* and *sigmoid*. It is hard to train only RNN because of problem of exploding and vanishing gradient. With addition of LSTM networks it solves the problem of exploding and vanishing gradient. Hence RNN with LSTM can able to learn long-term dependencies faster and very well. This avoids making

hand-generated features.

LSTM is used to capture the sequential information of resource usage. LSTM layers generate highly abstract features which then used to predict the future resource usage.

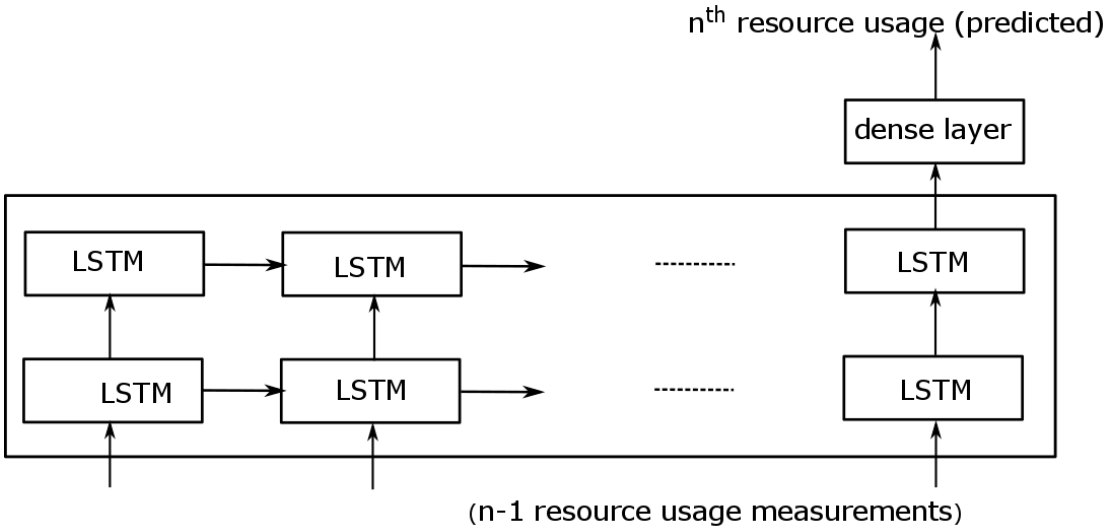

Figure 4.5: RNN with LSTM Prediction System

Model for Resource Prediction

The RNN-LSTM network model for future resource usage prediction is shown in Fig. 4.5. It consist of two layers of RNN-LSTM, which can capture the features of the physical resource usage sequence. The input to the LSTM network is last $n-1$ resource usage readings. The output is the n^{th} predicted value. Then the input window is shifted by one position to the right to form second training example. The generation of input and output training vectors is shown in Fig. 4.6.

A dense layer which is a linear function is used as a output layer because it is a many-to-one mapping. The loss function used for training is *squared error loss function* between actual and predicted values. If the predicted value is represented by $\hat{y}_i$ and the true value is represented as y_i, then the loss function is given by

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.8}$$

where θ are the parameters of the network.

For multi-step ahead prediction recursive version of the above model is used to predict the next values, when previously predicted value is taken as input. The other method is also applied to do multi-step ahead prediction. The input and the output is formed in terms of window of sequence values. Input window size of m and output window size of n is used as training vector. First row(vector) in the data set is first m data points in the sequence as a input and next n data points are used as a output. Thus total window size for creating training vectors is $m+n$. The next training vector is formed by moving the window of size $m+n$ to the right, and so on. Fig. 4.7 shows the generation of training vectors with $m = 7$ and $n = 3$.

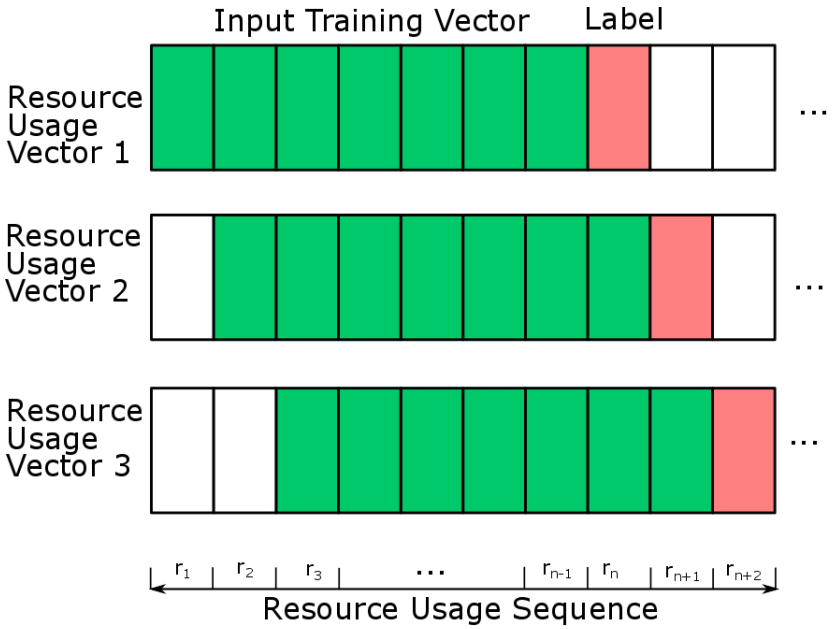

Figure 4.6: Generation of Training Vectors

The RNN-LSTM prediction model is tested with three different workload viz. Google Cluster Trace, OLTP, and Hadoop Workload Trace. Fig. 4.8 4.9 and 4.10 shows prediction error comparison with fuzzy prediction system. The prediction accuracy increased 10%- 20% when compared with our Fuzzy Prediction System.

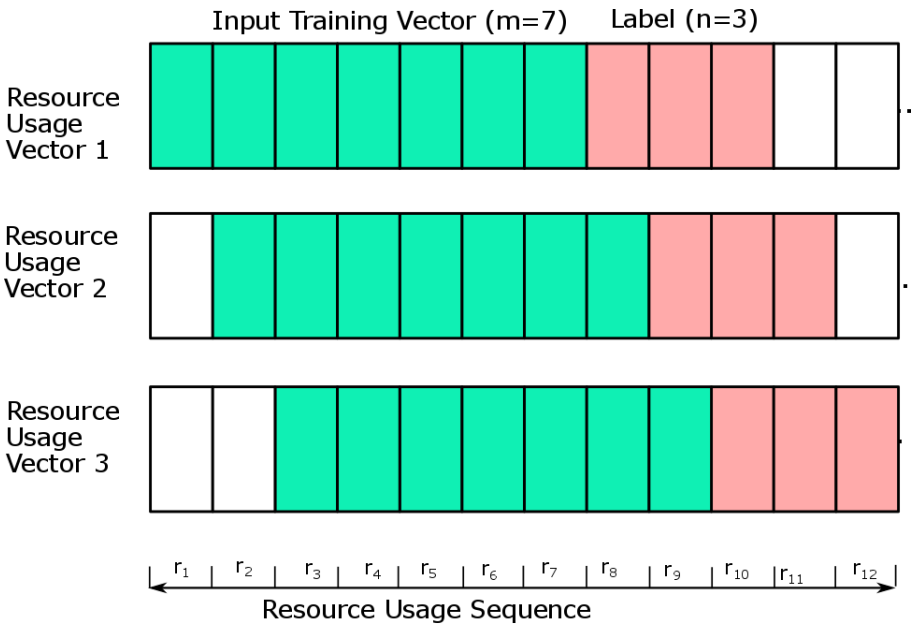

Figure 4.7: Generation of Training Vectors for Multi-step Ahead Output

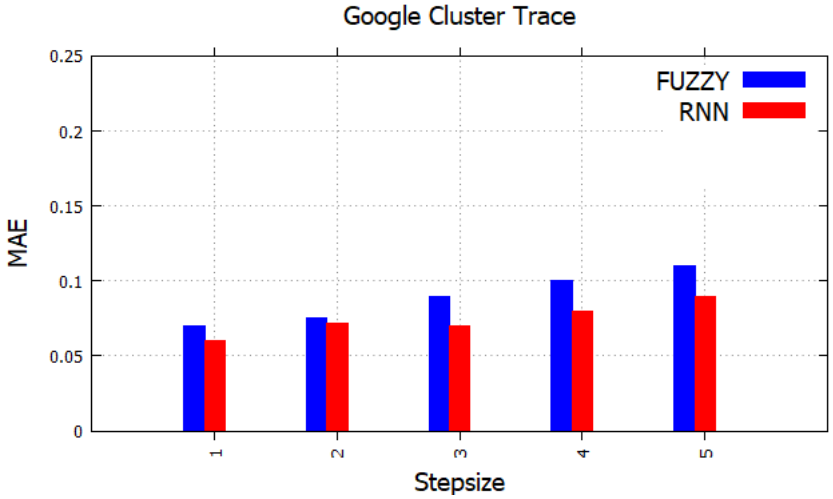

Figure 4.8: Google Cluster Trace

4.6 Resource Allocation using Combined Approach (Local Allocation)

4.6.1 Chaos Indicator

This module uses the Hurst exponent (Exponent, 2019) which represents the index of the dependence of the CPU variations in the measured time series of CPU usage. It extracts the relative behavior in a time series data. The value of H in the range 0.5 $<$ H $<$ 1 indicates a time series with long-term positive autocorrelation. This means the probability that a high value may be followed by a high value, is very high and

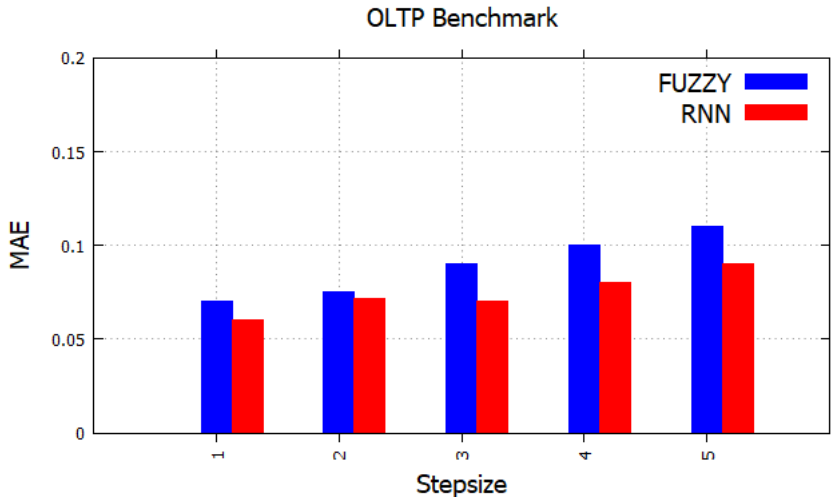

Figure 4.9: OLTP Trace

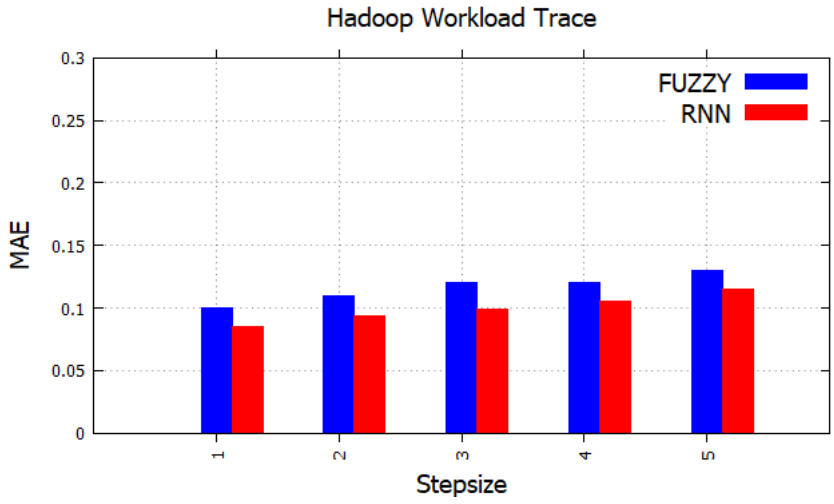

Figure 4.10: Hadoop Workload Trace

this may continue to be high for a long time. This represents patterns in the time series. The value of H; the range $0 < H < 0.5$ indicates a time series, with long-term switching between high and low values in adjacent pairs. This means that a single high value is probably be followed by a low value and that the values after that tend to be high, with the tendency to switch between high and low values lasting a long time into the future. A value of H=0.5 can indicate a completely uncorrelated series. But in fact it is the value applicable to the series for which the autocorrelations at small time lags can be positive or negative, but instead the absolute values of the autocorrelations decay exponentially to zero.

4.6.2 Combined Approach with Fuzzy Prediction and Kalman Filter based Prediction

The CPU Usage prediction system has been improvised using the findings in the recent literature survey. The system now consist of two modules: one including the base allocation prediction system and the reactive allocator which is used when a dominant pattern is seen in the workload data of the server, the other with Kalman filter controller. There is a chaos indicator which indicates the non linearity of the pattern and switches between the former module and the latter depending on whether the usage data has a pattern or is chaotic respectively. This improved controller is shown in Fig. 4.11. Related algorithm is shown in Algorithm 1. Line 1 determines the Hurst Exponent. Line 2 takes the decision of applying Kalman Filter or directly fuzzy prediction system based on the value of H. Line 3 predicts the future at various steps.

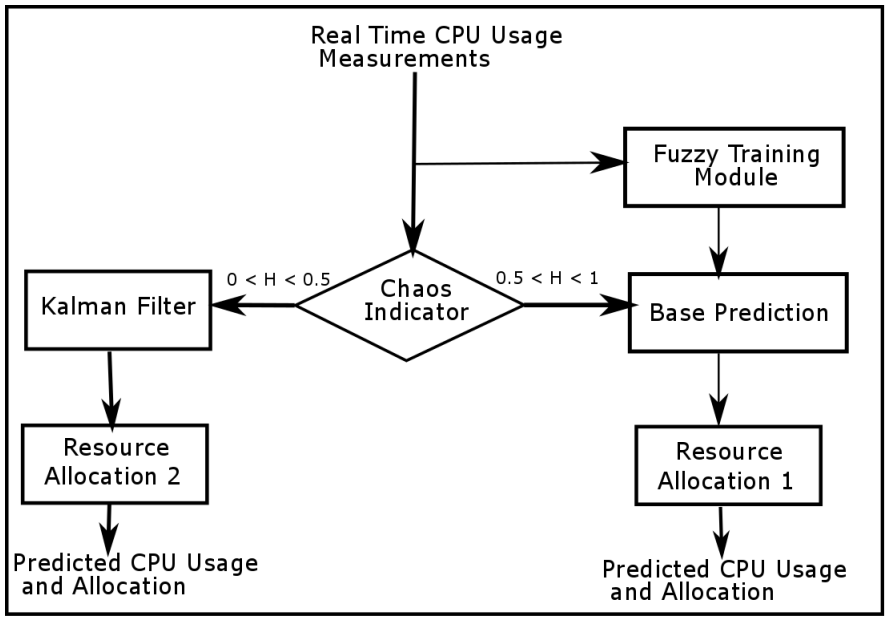

Figure 4.11: Combined Controller Approach with Fuzzy Logic and Kalman Filter

Various time series graphs have been analysed and found a threshold to switch between the two modules namely the Kalman filter and proactive module. A value higher than this indicates high autocorrelation between the values in the time-series which means a more linear-pattern can be seen in the time series and we can hence use regression and other such pattern finding proactive measure(moving average pre-

Algorithm 1: Prediction System for every VM

Input: Observed resource usage time series from a VM,
Output: Predicted Resource needs of a VM

```
Determine the Hurst Exponent (Chaos Indicator) from the time series data;
if H < 0.5 then
    Apply Kalman Filter based prediction;
end
else
    Apply Fuzzy prediction;
end
Predict one step, two step and three step ahead prediction;
```

diction system in our case) for good predictions of base pattern of workload. A lower value would imply that the time series is more chaotic/non-linear and hence a noise reducing base pattern estimator such as Kalman filter (used in our case) can be used. If the resource readings are beyond some range, then it may affect the leaning. Hence such things are also detected and corrected to maximum limit of that particular resource.

4.7 Other Prediction Methods used for Hot/Spot Detection

Time series prediction (Sapankevych and Sankar, 2009) can be mathematically stated as -

$$\hat{x}(t + \Delta t) = f(x(t - a), x(t - b), x(t - c), ...) \tag{4.9}$$

where $x(t+\Delta t)$ is predicted value of discrete tie series x value. The objective of time series prediction is to find a function $f(x)$ such that the predicted value at a future point at time t is unbiased and consistent. By using the regression analysis following equations defines prediction function for linear and non-linear data.

$$f(x) = (w.x) + b \tag{4.10}$$

$$f(x) = (w.\phi(x)) + b. \tag{4.11}$$

If the data is non-linear in its input space, it is necessary to map the data $x(t)$ to a higher dimension "feature" space via this kernel function given in equation 3.3 and then perform a linear regression in high dimensional mapped values. The goal is to

find optimal weights w and threshold b.

4.7.1 Autoregressive Modelling and Prediction

Mathematically the *Auto Regressive* $AR(p)$ model (Sapankevych and Sankar, 2009) can be expressed as-

$$x(n) = \sum_{i=1}^{p} a(i) * x(n-i) + e(n) \tag{4.12}$$

where x(n) is predicted linearly from its past p values x(n-1), x(n-2),.....,x(n-p). The variable p is called as order of prediction and model is called as *p-order AR* model. Prediction coefficients or model parameters $a(i)s$ are calculated with the help of Yule-Walker method in (Sapankevych and Sankar, 2009) where $e(n)$ is the mean error term.

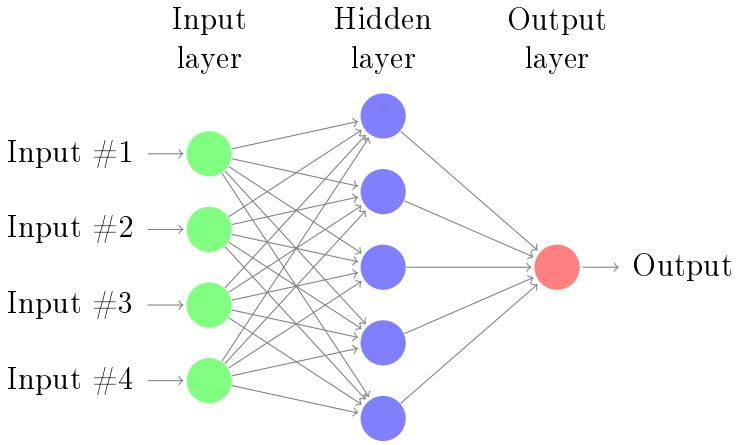

Figure 4.12: Artificial Neural Network Architecture

4.7.2 Artificial Neural Network

ANN is a powerful tool for self-learning, and it can generalize the characteristics of load by proper training. It is inherently a distributed architecture with high robustness and has been used in resources state prediction in the past. The structure of a standard multilayer feedforward neural network is in Fig. 4.12. It consists of an input layer with input neurons $[x_{t-p}\ ,\ x_{t-p+1}\ ,\ .\ .\ .\ ,\ x_{t-1}]$, a hidden layer with hidden neurons $[h_1\ ,\ h_2\ ,\ .\ .\ .\ ,\ h_k]$, and an output layer with one output neuron $\hat{x}_t$. Every node in a layer is connected to every other node in the neighboring layer. These connections are known as synapses. Each synapse is associated with a weight which is to be

determined during training. During the training phase, the network is fed with input vectors, and random weights are assigned to the synapses. After presentation of each input vector, the network generates a predicted output $\hat{x_t}$. The generated output is then compared with the actual output $\hat{x_t}$. The difference between the two is known as the error term.

The Back Propagation Neural Network (BPNN) algorithm is the most popular and the oldest supervised learning feed forward neural network algorithm proposed (Rumelhart et al., 1986). The Back Propogation Neural Network (BPNN) learns by calculating the errors of the output layer and back propogate the errors to the hidden layers. This algorithms is very suitable where the relationship between input and output does not exist. It is flexible and capable of learning nonlinear things. It has been used for the comparison with the presently proposed system. It is empirically configured it with six input neurons, one hidden layer with ten hidden neurons and two output neurons. The inputs are mean CPU request at time t, Mean memory request at time t, Mean CPU usage, Mean memory usage, CPU capacity and Memory Capacity at time t. The outputs are Mean CPU usage and Memory usage at time $t + stepsize$.

The system needs to learn the weights $(w_{ij})^l$ to minimize

$$J(W) = \sum_{s=1}^{N} \frac{1}{2} (\sum_{j=1}^{n_L} (y_j^L(X^s, W) - d_s^j)^2) \tag{4.13}$$

where, $(X^s, d^s), s = 1, ..., N$ is the training set. $(w_{ij})^l$ is the weight on the edge that connects node i in layer l to the node j in layer $l + 1$. y_j^L is the j^{th} node output of the network. n_L are number of nodes in the output layer. The iterative Gradient Descent method is used for minimization.

Computation of Network is as follows-

- For i/p layer $y_i^1 = x_i^S$, i= 1,.....,n_1
- For l=2,....,L compute,

$$n_j^l = \sum_{i=1}^{n_{l-1}} w_{ij}^{l-1} y_i^{l-1} \tag{4.14}$$

$$y_j^l = f(n_j^l) \tag{4.15}$$

- Once the output of the network is obtained, errors δ_j^l need to be computed.

- At the o/p layer

$$\delta_j^L = (y_j^L - d_j^S) f^{'}(n_j^L) \tag{4.16}$$

- Now for the layers l = (L-1),.....,2 compute

$$\delta_j^l = (\sum_{s=1}^{n_l+1} \delta_s^{l+1} w_{ij}^l) f^{'}(n_j^l) \tag{4.17}$$

- Once all δ_j^l are available, the weight is updated by

$$w_{ij}^l(t-1) = w_{ij}(t) - \lambda \delta_j^{l+1} y_i^l \tag{4.18}$$

This is training the neural network by minimization of empirical risk under squared error loss. The network is empirically configured with six input neurons, one hidden layer with 10 hidden neurons and two output neurons. The inputs are mean CPU request at time t, Mean memory request at time t, Mean CPU usage, Mean memory usage, CPU and Memory capacity at time t. The outputs are Mean CPU usage and Memory usage at time t. The outputs are Mean CPU usage and memory usage at time $t + stepsize$. $stepsize$ is taken as 60 sec.

4.7.3 Prediction with Weighted Majority of Experts

The previously observed resource usage is called as experts (Haider et al., 2009). To predict the resource requirements at time $t+1$ these experts are used at time t i.e. usage readings from time t to $t-N$ assuming the use of N experts. Following Algorithm 2 is used to determine resource requirement at time t.

Algorithm 2: Prediction with Weighted Majority of Experts

Input: C_1, C_2,.....,C_t be the experts at time t
w_1, w_2,.....,w_N be the respective weights of the N experts
Output: Resource Requirements at time $t+1$

1 **for** $i \leftarrow 1$ **to** N **do**
2 $\quad w_i = w_i.exp(-\eta.l(C_i, C_t))$
3 **end**
4 $\hat{C}_{t+1} = \sum_{i=1}^{N} w_i.C_t \ / \ \sum_{i=1}^{N} w_i$

Here the goal is to minimize the regret which is the difference between cumulative loss of player and cumulative minimum loss of expert. The initial weights are given

such that sum of all the weights is zero. The prediction is given by following equation

$$\hat{C_t} = \frac{\sum_{i=1}^{N} w_i.C_{t-1}}{\sum_{i=1}^{N} w_i} \tag{4.19}$$

The predicted value is calculated in line 4. Once the true output is known and the losses $l(C_i, C_t)$ of every expert are calculated. Weights are updated as follows-

$$w_i = w_i.exp(-\eta.l(C_i, C_t)) \tag{4.20}$$

where η is the learning rate used which is shown in lines from 1 to 3 of the Algorithm 2.

A multi-step-ahead future prediction is implemented by exploiting one-step-ahead prediction iteratively. Suppose time series data with m step ahead prediction is represented as $\{....,x_{t-2},\ x_{t-1},\ \hat{x_t},\ \hat{x}_{t+1},\hat{x}_{t+2},.....,\hat{x}_{t+m-1}\}$. The prediction of $\hat{x}_{t+m-1}$ is based on the series $\{....,x_{t-2},\ x_{t-1},\ \hat{x_t},\ \hat{x}_{t+1},\hat{x}_{t+2},.....,\hat{x}_{t+m-2}\}$ where only $\{....,x_{t-2},\ x_{t-1}\ \}$ are the measured data and $\{\ \hat{x_t},\hat{x}_{t+1},\hat{x}_{t+2},...\ ....,\hat{x}_{t+m-2}\}$ are all estimated with prediction. This is based on the assumption that the predicted data is the same as the measured data. But, due to dynamic nature of the cloud, this assumption is very difficult to hold. Hence each one step ahead prediction, may be with an error. This error is propagated in the further predictions and errors are accumulated next.

It is observed that, during multi-step-ahead load prediction of cloud, the importance of input data series gradually increases and then decreases. The differentiating point is between last measured data and first predicted data. This importance of the training data points can be implemented by placing different weight on the ϵ - insensitive errors based on the importance of the training data. So, an importance coefficient imp_i is added to regularization constant. Then the risk function is translated to-

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} imp_i(\xi_i + \xi_i^*) \tag{4.21}$$

under constraints specified previously and imp_i is the increasing function for measured data and decreasing for predicted data. ξ_i and ξ_i^* are the slack variables.

4.8 Handling Errors in Predictions

Better performance and high availability are the two primary goals of organizations running Web Services. Once this web service is deployed on cloud environment it has to be monitored by the data center administrator. If any performance issues are found while monitoring, then it must be quickly resolved. Service unavailability may cause losses in millions of dollars to the companies. Hence operators must respond quickly to such issues.

To maintain a better performance and high availability, dynamic resource provisioning through workload prediction has been looked at. But it has been observed that these prediction results are incorrect during the spikes in the workload. Due to these spikes, the underestimation errors exists in predicted values. Hence there are performance degradation and violation of application SLAs. Hence it is required to work on the identification of these spikes and accordingly allocate the amount of resources required.

The data center operators need to understand the patterns in the workload and long-term trends. The data center operators must handle the unexpected spikes in the workload. Most web services are highly predictable, as they show daily, weekly or yearly changing patterns. This can be used for long-term and short-term scheduling. Sometimes, millions of users can create unexpected workload spikes in the data center and creates hotspots. e.g. During Michael Jackson's death the aggregate workload increase on *wikipedia.org* was 5%, but 15% of the total requests had been for Jackson's document. Hence it is necessary to understand properties of spikes. Hence workload modeling and synthesis is important to stress-test the application.

In the case of stateless webservers the spikes can be resolved by adding more servers, switches, DNS servers, etc. But in the case of data-intensive social networking sites such as Google, Amazon, Wikipedia, etc. sudden increase in specific resource demand must be dealt with, properly. Due to the dynamic nature of the data center environment, sudden spikes may arise, which can affect the predicted value. The predicted value may not be correct one. Hence errors due to sudden spikes needs to be handled correctly.

4.8.1 Underestimation Error Correction

Algorithm 3 explains the calculation of padding values. As todays application workload is very dynamic, data usage sequence may contain spikes due to burst in the workload traffic. Hence there may me under-estimation error to occur. To avoid such errors it is better to pad the predicted values with some small amount of resource. If it is a fixed value, it may lead to over or under provisioning. Hence proper padding has to be added to predicted values. Otherwise such underestimation errors can be corrected immediately by raising CPU cap.

4.8.2 Padding

Algorithm 3 gives the procedure to calculate padding values. A window $W = [C_{t-l},.....,C_{t-1}]$ of resource usage time series is observed at time t. A signal processing technique specifically Fast Fourier Transform (FFT) is applied on the observed window W and it is stated in line 2 of the Algorithm . It gives the coefficients, which represents the amplitude of each frequency component. Higher frequency components show burst pattern in the traffic. An inverse FFT is applied on the top higher frequencies in the spectrum to synthesize the burst pattern to values which is shown in line 3 and 4 of the Algorithm. Out of these values only positive values are considered so as to avoid underestimation errors. The count of such positive values gives the frequency of occurrence of the burst. If for extracted burst pattern, this count is high (more than 50%), then maximum count value among all the observed burst patterns is set as padding value. Otherwise a smaller count value is set as a padding value. This is stated in lines 5 to 10 of the Algorithm 3.

The system observes the prediction errors. Let e_1, e_2,....,e_k be all prediction errors. The weighted average of all negative errors is calculated. The system sets a larger value out of this average and previously decided padding value as a padding value. These steps are shown in lines 13-17 of the Algorithm.

4.8.3 Immediately Rising Resource Caps

The detected underestimation errors are corrected by raising the CPU cap value. The CPU cap value are multiplied by some ratio α, until these errors vanish or there is no

more remaining cpu capacity of the physical server.

If CPU cap of the current step is x, then CPU cap after k steps is given by $x * \alpha^k$. If no more CPU capacity exists, then migration is triggered.

To determine the value of α the resource pressure is considered. The resource pressure is the ratio of resource usage to the resource cap. It is denoted by P varies in between 0 and 1. The value of α is determined on this pressure value as shown below.

$$\alpha = \frac{P - P_{under}}{1 - P_{under}} \tag{4.22}$$

where P_{under} is the thresholds set for the resource pressure e.g. raise resource cap when P_{under} crosses 0.9.

Algorithm 3: Function for evaluating the padding value

Input: $W = \{C_{t-l},.....,C_{t-1}\}$, P = $\{P_{t-l},.....,P_{t-1}\}$
Output: Padding value at time t

Function EvaluatePadVal(W, P);
A = FFT(W);
`// Consider only high frequency components`
A = Select top 80% from A;
B = ReverseFFT(A);
f = Percentage of +ve values in B;
If $f > 50\%$ **then** ;
`// workload contains bursts`
    b_{temp1} = Maximum value from B;
else ;
    b_{temp1} = 80^{th} percentile value from B;
endif;
$\{e_{t-l},.....,e_{t-1}\}$ = P - A;
`// Take weighted moving average of all the errors`
b_{temp2} = $\|$ WMA($\{e_{t-l},.....,e_{t-1}\}$) $\|$;
If $b_{temp1} > b_{temp2}$ **then** ;
    $PadVal = b_{temp1}$;
else ;
    $PadVal = b_{temp2}$;
endif;
Return $PadVal$;

4.9 Resource Allocation Controller of each PM

The working of Resource allocation controller for each PM is shown in Algorithm 4. Once the predicted resource needs of every VM is received by PM then it estimates

Algorithm 4: Resource Allocation Controller of each Physical Machine

Input: $W = \{C_{t-l},.....,C_{t-1}\}$, P = $W/CapVal$, $CapTotal$
Output: Padding values, value of α to raise the CPU cap, Allocation of additional resources as per the predicted needs,

1 $PadVal$ = EvaluatePadVal(W, P) ;
2 $\{e_{t-l},.....,e_{t-1}\}$ = P - A;
3 **if** *Any of the e component is negative* **then**
4 $\alpha = (P\text{-}P_{under})/(1\text{-}P_{under})$;
5 **end**
6 **repeat**
7 $CapVal = CapVal * \alpha^k$;
// k^{th} `step`
8 $\{e_{t-l},.....,e_{t-1}\}$ = P - A;
9 $CapTotal = CapTotal$ - $CapVal$;
10 **until** *All the e's become positive OR* $CapVal < CapTotal$;
11 **if** $CapVal > CapTotal$ **then**
12 Trigger migration ;
13 **end**

the padding values for the predicted resource demands. This is shown in line 1 of an Algorithm. The system observes the prediction errors. Let e_1, e_2,....,e_k be all prediction errors. These are calculated in line 2 of an Algorithm. The weighted average of all negative errors is calculated. Prediction errors are calculated as shown in line 2of the algorithm. If underestimation errors are observed then the value of α is calculated as per the resource pressure P by using Equation 4.22 as shown in line 3 and 4 of the algorithm. The resource *cap* value is raised as described in section 4.8.1 which is shown in step 7 of the algorithm. The total capacity of individual resource is updated in line 9 of an Algorithm. This continues until underestimation errors are resolved or no resource is available. If no resources are available with PM, then migration is triggered and shown in steps 6 to 13 of the algorithm.

4.10 Results and Discussion

Experiments have been conducted with benchmark workloads of the datacenters. A mix of workloads, such as network intensive, CPU intensive and memory intensive workloads on VMs are used.

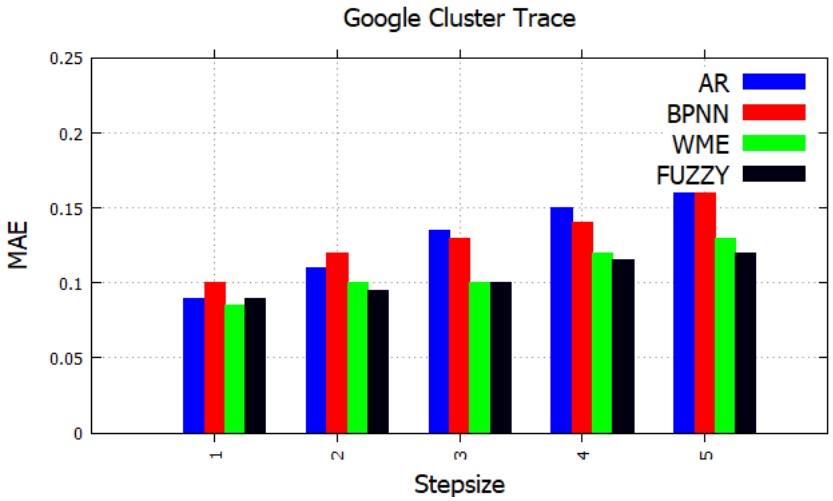

Figure 4.13: Google Cluster Trace

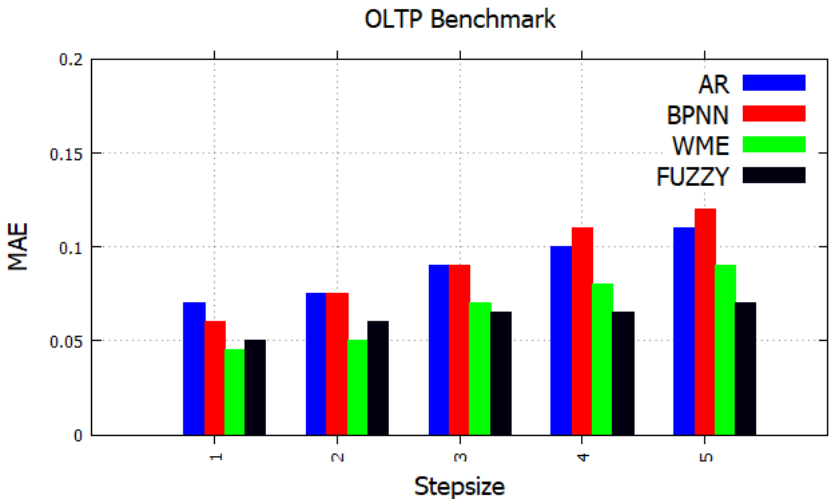

Figure 4.14: OLTP Benchmark

4.10.1 Actual resource usage and Predicted requirements with different workloads

Time series prediction is done with both AR-p model prediction algorithm, Back Propagation Neural Network (BPNN) model and prediction with Weighted Majority of Experts (WME). The implementation of these is carried out with machine leaning Python Library (scikit-learn). The time series values for all resources are given to the prediction module and future values, at an interval of 60,120,180,240 and 300 sec, are predicted. Three real time workload traces are used for experimentation viz. Google Cluster Trace (Wilkes, 2011), OLTP database System trace (OLTP, 2008) and Hadoop Map Reduce (hadoop, 2012). The actual values and predicted values

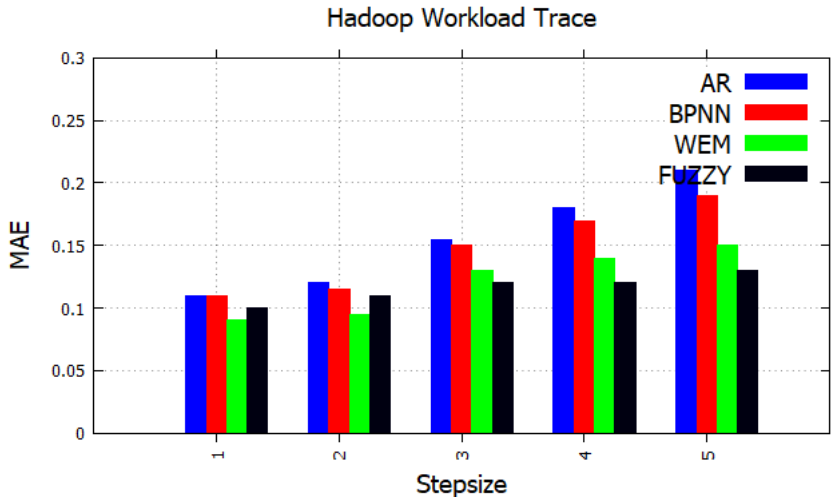

Figure 4.15: Hadoop Workload Trace

are compared. The prediction performance for these intervals is found suitable with WME and Fuzzy Prediction System. The Figure 4.6 shows the MAE values for AR, ANN, WME and Fuzzy prediction algorithms at different time intervals. The stepsize of prediction is 60 Sec. The values shows that it has got decent average MAE of 0.0658 for Google trace data, 0.053 for OLTP data and 0.9987 for Hadoop Map Reduce data. The prediction accuracy is shown upto 300 Sec which is found to be suitable for the proposed work.

Fuzzy prediction system gives accuracy on average as MAE of 0.056. Figures 4.13 to 4.15 specifically shows the MAE of different machine learning algorithms on different workload traces taken from Internet.

4.10.2 Comparison with Kalman Filter Prediction

When the workload is very non-linear in nature, the prediction accuracy with Fuzzy as well as Weighted Majority of Experts is less. Hence, the proposed has applied Kalman filter under such situations. This has resulted in better prediction accuracy with Kalman, as it is best suited for non-linear data. The errors received with very non-lineaar data traces, like ClarkNet data trace (ClarkNet, 2012), CAIDA data trace (CAIDA, 2016) and synthetic workload created with RuBiS (RUBiS, 2012) , are represented with MAE in the Figures 4.16 to 4.18. Kalman filtering gives more accuracy compared to all other prediction techniques. It filters out random variations and noise to find dominant patterns in the time. Hence accuracy received with Kalman is

comparatively well than others.

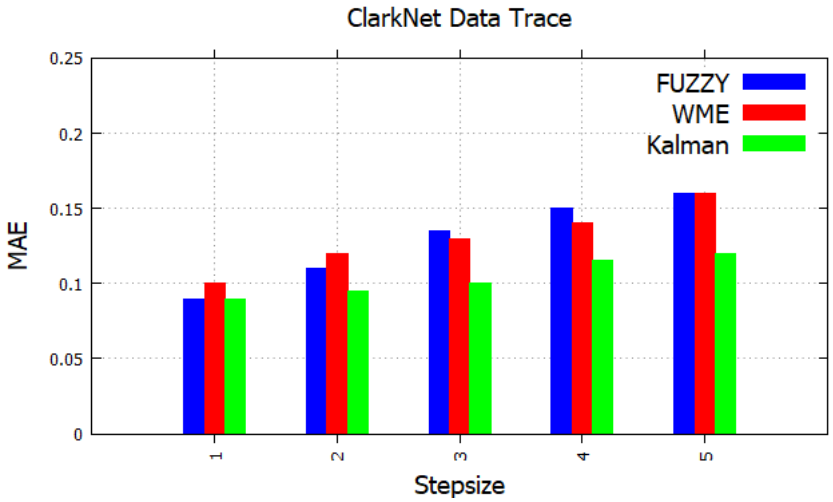

Figure 4.16: ClarkNet Workload Trace

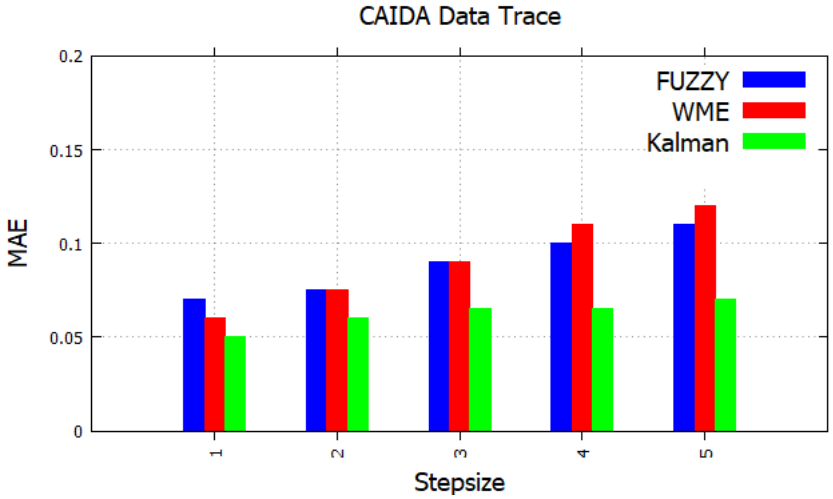

Figure 4.17: CAIDA Workload Trace

4.10.3 Analysis of Resource Allocation Controller

The resource allocation with the proposed method is validated by experimenting with real time workloads and synthetic workloads. Wikimedia http trace has been simulated using *logreplayer* (Logreplayer, 2011) on VM1 and ClarkNet (ClarkNet, 2012) HTTP trace has been simulated on VM2 using two *httperf* (Httperf, 2015) clients. RUBiS (RUBiS, 2012) is used to implement e-commerce site on both VMs to which http requests can be issued. It is found that the allocations done by our proposed

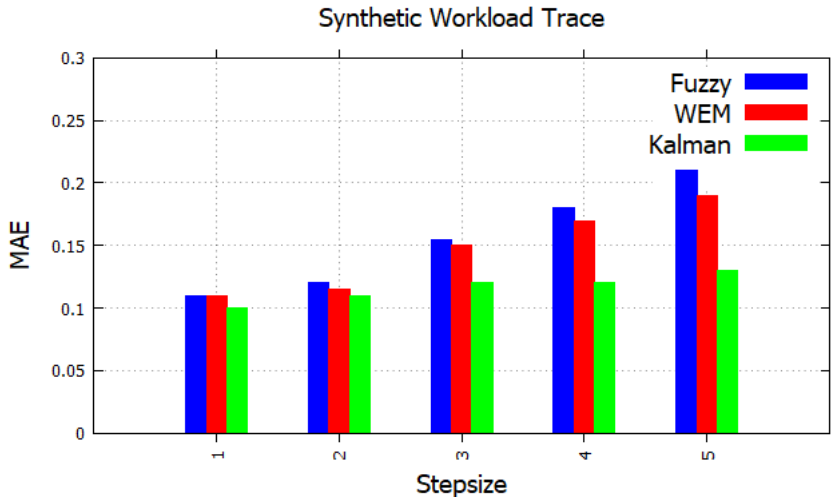

Figure 4.18: Synthetic Workload Trace

method is in line with actual resource (CPU) usage. The CPU usage and allocations from our proposed method are shown in Fig. 4.19 and Fig. 4.20. The difference between allocation and actual usage is less than 1% on average. At spikes, the allocations are slightly more but below 5-10%, which is better than other methods in the literature. This is due to the underestimation error handling. Without underestimation error handling, the cpu allocation at spikes are below actual requirement as shown on Fig. 4.21 and Fig. 4.22.

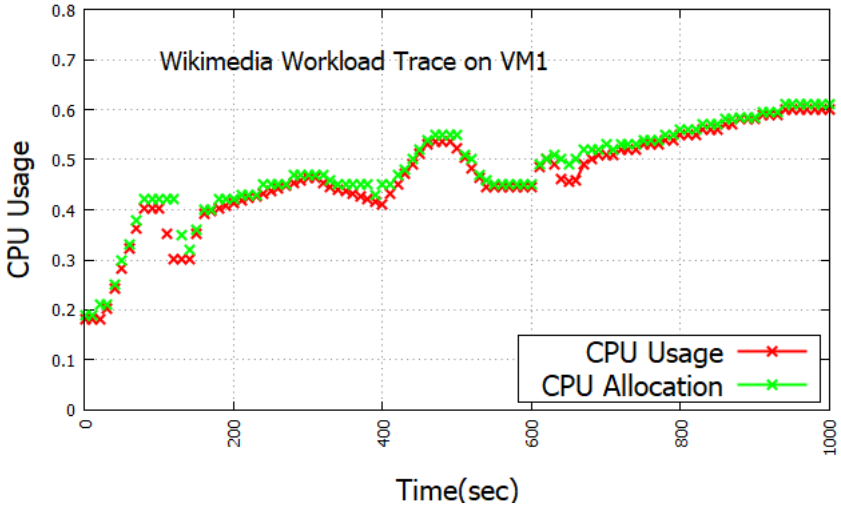

Figure 4.19: CPU Usage Prediction on VM1 with Under Estimation Error Correction

Thus under-estimation error handling is very important for dynamic resource management with predictions. The proposed techniques use the dynamics of the system

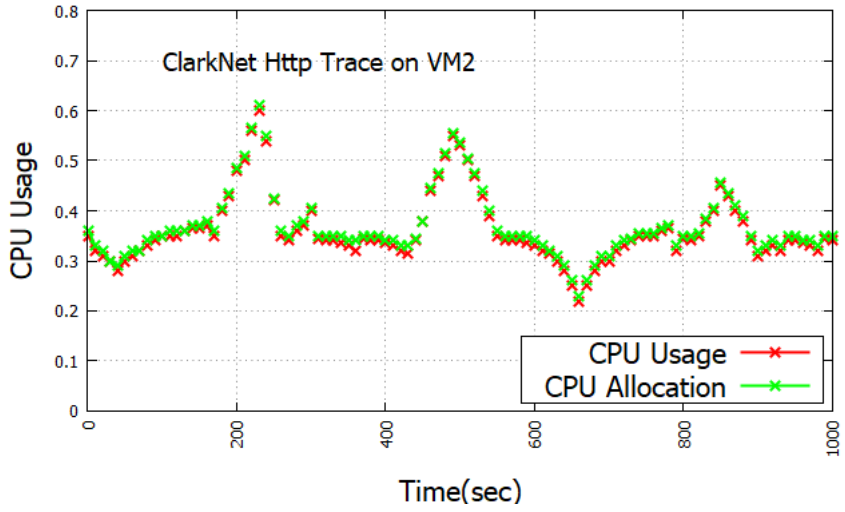

Figure 4.20: CPU Usage Prediction on VM2 with Under Estimation Error Correction

for validating the next observation. The results shows that it dynamically adapts to the variations in the workload and is able to differentiate between important workload changes or small variations.

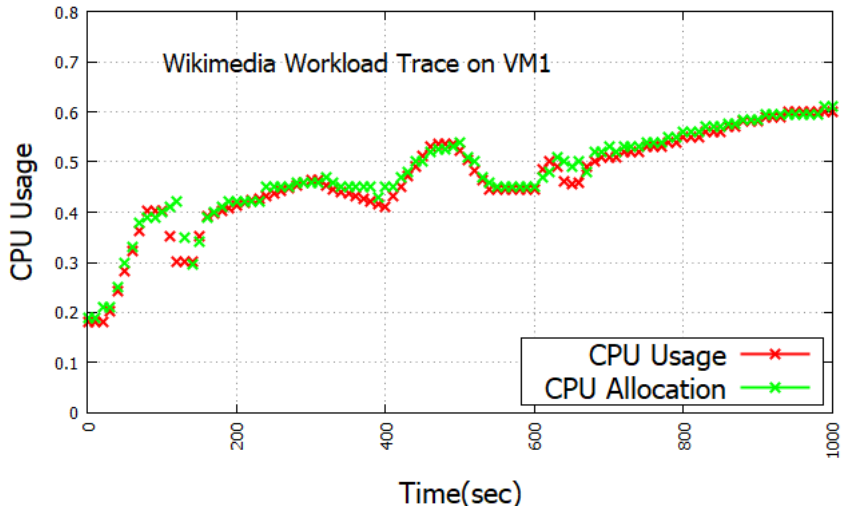

Figure 4.21: CPU Usage Prediction on VM1 Without Under Estimation Error Correction

4.11 Summary

This chapter presents the resource allocation controller which resides in every VM. It determines the future resource needs by observing past resource usage using fuzzy prediction system or prediction with weighted majority of experts. It uses Kalman

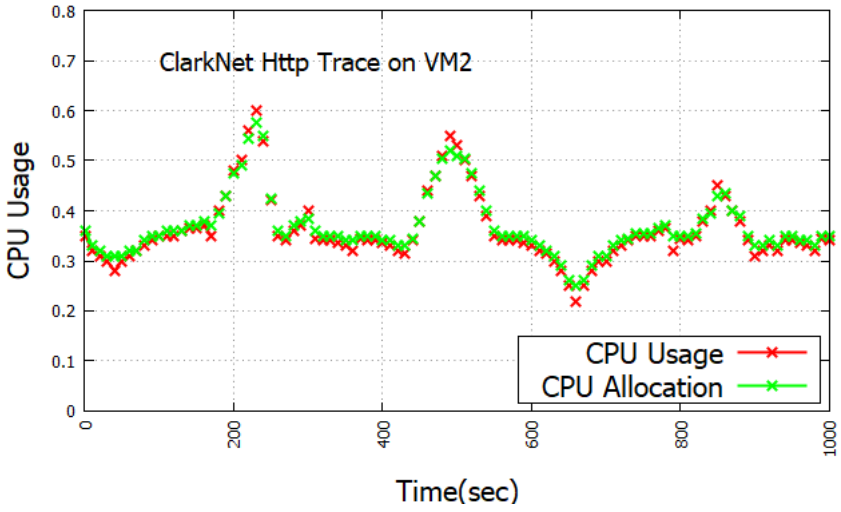

Figure 4.22: CPU Usage Prediction on VM2 Without Under Estimation Error Correction

filter based prediction when workload is very non-linear. Adaptive padding scheme is applied on predicted values to remove under-estimation errors. If under-estimation errors are still detected, they are corrected by immediately raising resource cap with some value. This value is estimated by observing the pressure on the resource at that time. Finally it is observed that resource allocation with our scheme is parallel to the actual usage. Also, at the time of workload spikes also the allocation is free from underestimation errors. When the workload is very non-linear, the Kalman filter suits well. Hence combined approach has given satisfactory results.

Chapter 5

Live VM Migration Performance Modeling

Data-center management consist of several tasks like load balancing, hot-spot mitigation, server consolidation, moving virtual machines across different physical machines of different data centers during cloud bursting, etc. Many of these task require movement of VM from one server in the data center to another. It consists of transfer of hardware state and memory pages on the VM. Live migration of virtual machine can be carried out by pre-copy and post-copy method as described in chapter 2. The pre-copy or post-copy live migration technique performs this migration when VM is running. As the VM is in running state, memory pages get dirtied multiple times, which requires iterative transfer of these; the condition for stopping VM and finally transferring remaining memory pages.

5.1 VM Live Migration using Pre-Copy Technique

Clark et al. (2005) have defined pre-copy mechanism for iterative transfer of memory pages. All the memory pages are transferred during the first iteration and subsequent iterations transfers the memory pages which are getting dirtied during the previous one. When one of the stop-and-copy conditions given by Nathan et al. (2013) is met, the execution of VM is suspended and all the remaining memory pages, on the source and the hardware state are transferred to destination in one final iteration. Xen and KVM use this iterative pre-copy technique. Xen optimizes this method by skipping the memory pages which are frequently dirtying, to reduce the amount of data transfer. This optimization technique is called *pageskip*. In this research work, Xen is used

as a virtualization platform and hence the pre-copy live migration technique. The *pageskip* technique is explained below.

The page skip technique : It identifies the dirtied pages during the transfer of every k pages in an iteration. It skips transferring these dirtied pages in the current iteration, in view of getting dirtied again in the current iteration. In pre-copy mechanism the pages which get modified in the prior iteration, are selected for transferring to destination in present iteration. Thus, the pages which are skipped are those which are dirtied in current as well as previous iteration. In case of the first iteration, all the pages which are dirtied are eligible for skipping.

5.2 Parameters Affecting the Performance

The performance of the live VM migration depends on total migration time and downtime. These performance parameters are affected by (i) memory size - V_{size}, (ii) transfer rate of memory page - R_{page}, (iii) count of unique pages dirtied - D_i, and (iv) count skipped pages - S_i.

The first two parameters are configured by the user and others depends on the characteristics of the application running on the VM. It is clear that the memory page dirtying characteristics of an application gives the unique number of pages dirtied in one iteration. Similarly the pages which are to be skipped in the current iteration also affects the migration time. Hence these are two important parameters in estimating VM total migration time and it's downtime.

5.3 Proposed Model for Live VM Migration Performance

The performance of live VM migration could be estimated by total migration time , and downtime during which the VM's execution on the source PM is suspended. Similarly, the cost of VM migration can be estimated by the network traffic generated during migration. The total migration time is calculated by adding iterative pre-copy time and suspended VM's downtime. The generated network traffic is the volume of data transferred between source and destination PM during the iterative transfer. The Algorithm 5 thus presented is for modeling the migration process and to calculate

parameters affecting the performance of migration. As per the live migration procedure described above, line 2 to 9 show the iterative page transfer. All the memory pages of the VM are transferred in first iteration. In subsequent iterations, the pages (unique) dirtied during the previous iteration are selected for transfer in the current iteration. The skip technique selects the pages which are dirtied in both previous and present iteration for skipping. The skipped pages are subtracted from the number of pages selected for the transfer in current iteration and shown in line 3. The iterative pre- copy mechanism may not be complete and it can go on continuously, as some of the memory pages get dirty during the execution of VM. Hence some conditions are defined to stop this and are called as $stop-and-copy$ conditions. If any threshold defined on different parameters is crossed, then the VM is suspended and all remaining pages are copied from source to destination.

1. Threshold on maximum number of iterations are completed.

2. Threshold on maximum traffic is generated.

3. Threshold on maximum pages dirtied in an iteration

4. Threshold on memory dirtying rate of an application threshold defined

5.3.1 Model for Number of Memory Pages which may Get Dirtied

Algorithm 5 gives the model for VM live migration process. Line 1 initializes the variables. The algorithm iterates through lines 2 to 9 until any of the above defined threshold is reached. Line 3 calculates the migration time required for the current iteration. Line 4 accumulates the total migration time. Let P be number of average unique memory pages dirtied in a specific time interval t in an iteration and P_{new} be the number of new memory pages dirtied in time t. Thus the number of unique pages dirtied which is represented by D_i is calculated as shown in line 6 in the Algorithm 5. Line 7 takes these dirtied pages for the transfer in the next iteration. Line 8 increment the iteration count. Line 10 calculates the total down time. Line 11 returns the total migration time, total downtime and the amount of volume generated for transfer.

Algorithm 5: Model of VM Migration Process

Input: V_{size} (VM Memory size), R_{page} (Page transfer rate), $Thresholds$ on parameters for $stop-and-copy$ phase.

Output: T_{mig} (Total migration time), T_{down} (Downtime), V_{mig} (Amount of volume generated).

1 Let $V_1 = V_{size}$, $i = 1$, $T_{mig} = 0$, $V_{mig} = 0$;
// i denotes interation number
2 **repeat**
3 $T_i = (V_i$ - $S_i)$ / R_{page};
// $SKIP_i$ the pages skipped in iteration i
4 $T_{mig} = T_{mig} + T_i$;
5 $V_{mig} = V_{mig} + V_i$;
6 $D_i = min([P + (C-1) * P_{new}], M_{wws})$;
// C are the number of partitions in time T_i
7 $V_{i+1} = D_i$;
8 $i = i + 1$;
9 **until** *any one of the defined threshold reached*;
10 $T_{down} = V_i$ / R_{page};
11 **return** T_{mig}, T_{down}, V_{mig}

The number of average unique pages dirtied depends on the maximum size of writable working set of an application which is represented by (M_{wws}).The procedure for computing P, P_{new} are derived form the dirty bitmap collected at every time interval. M_{wws} can be estimated with the help of average number of memory pages skipped in all the iterations during migration. Algorithm 6 estimates the number of memory pages which may get dirtied during next iteration and these steps are shown in line 4 to 8 of the Algorithm 6. Line 2 calculates the number of intervals in an iteration. Line 3 calculate newly dirtied pages where P_{new}^{T} represents first time dirtied per time t in a time period between 0 to T seconds. Line 4 to 8 is the calculation of number of pages to be dirtied depending on the number of intervals. The minimum of total newly dirtied pages in all intervals in an iteration and Writable Working Set size is selected as the number of pages going to be dirtied in an iteration.

5.3.2 Computing P and P_{new}

Whenever there is a need for predicting the performance of VM migration, the dirty bitmaps are collected at a specific bitmap-collection interval t for a specific time period. From this dirty bitmaps page dirtying information is known, from which P and P_{new} can be computed.

Algorithm 6: Estimate the Number of Pages to be Dirtied

Input: T_i Iteration Time, P Number of unique memory pages dirtied during the migration, P^t_{new} Newly dirtied pages during time t, M_{wws} Writable working set, j interval time.

Output: Number of memory pages to be dirtied.

1 **Function** NDP(T_i, P, P^t_{new}, M_{wws}, j);
2 $C = T_i/j$;
`// C denotes number of intervals`
3 P_{new} <— $P^{round(T_i)}_{new}$;
4 **If** C $<$ 1 **then** ;
5 **return** min([P + (C-1) X P_{new}], M_{wws});
6 **else** ;
7 **return** min([P x C], M_{wws});
8 **endif**

Computing P : It is the number of memory pages (unique) dirtied during migration in a bit map collection time interval. Xen hypervisor set the bitmap bit for the dirtied pages. P is estimated by taking an average of the number of set bits in all collected bitmaps.

Computing P_{new} : These are the number of first time dirtied pages in an bitmap collection interval time t. It is estimated by taking the average over such first time dirtied pages in all the bitmaps collected for the specific time interval T seconds. P^T_{new} represents first time dirtied per time t in a time period between 0 to T seconds. This is shown in line 3 of the Algorithm.

5.3.3 Computing Maximum Writable Working Set M_{WWS}

Every workload has a set of hot pages which are dirtied again and again and are skipped (S_i) during the iterative pre-copying procedure. This set of pages is called as Writable Working Set (M_{wws}).

The memory dirtying rate can also be calculated with dirty bitmap using shadow page tables (Clark et al., 2005). Generally writable working set of most of the applications is approximately proportional to the pages dirtied in each precopying round. Hence,

$$M_{wws} = \eta.V_i \qquad and \qquad \eta = \alpha.T_{i-1} + \beta.R_d + \gamma \tag{5.1}$$

where η is a variable, correlating time duration of each iteration T_i and memory dirtying rate R_d.

For a particular application multiple observations are taken, for the number of pages dirtied in each iteration and time duration of respective iteration, to calculate η. By solving these simultaneous linear equations, model parameters α, β and γ are calculated.

5.3.4 Computing Number of Memory Pages to Skip S_i

The memory pages to be skipped in an iteration are the number of pages which are dirtied in both present and previous iterations of memory transfer. Hence the number of skipped memory pages as per $Skip - technique$ of Xen, can be computed by subtracting the memory pages dirtied (represented by U) in both the iterations (T_i + T_{i-1}), from the addition of memory pages dirtied in the two individual iterations. If memory pages dirtied in present and previous iterations are estimated as D_i and D_{i-1}, are the memory pages to be skipped in current iteration i can be given by-

$$S_i = .(D_{i-1} + D_i) - U \tag{5.2}$$

These can be put in Algorithm 5 and updated algorithm is shown in Algorithm 7. Lines 3-11 shows the iterative page transfer as in Algorithm 5. Line 4 calculates the the number of pages to be skipped. Line 5 calculates the current iteration time. Line 6 and 7 combines the total migration time and total traffic generated. Line 8 gives the number of dirtied pages to be considered for transfer in the next iteration. Line 9 takes the dirtied pages to transfer in next iteration. Line 10 increments iteration count. Line 12 gives the downtime in the migration process and line 13 returns the total migration time, downtime and total traffic generated during the migration.

5.4 Study of Interference due to Live VM Migration

To study the effects of live VM migration on other VMs hosted on source and destination physical machine, the VMs are migrated between two physical machines. A Xen virtualization platform is installed on two PMs. Two VM instances have been created and started, denoted by $vm1$, $vm2$ on one PM and $vm3$ on another PM. These VM instances are large enough to represent production standard environment, having 2 VCPUs , 8 GB memory and cap value of 400. Instance $vm2$ is made idle. Instance

Algorithm 7: Model of VM migration

Function LVM(VM_{size}, R_{page}, P, P^t_{new}, M_{wws}, BT);
i<–1; V_i<–VM_{size};
repeat
    $S_i = (\alpha * R_{page}) + (\beta * E_i) + \gamma$;
    $T_i = (V_i - S_i) / R_{page}$;
    $T_{mig} = T_{mig} + T_i$;
    $V_{mig} = V_{mig} + V_i$;
    $D_i = \text{NDP}(T_i, P, P^t_{new}, M_{wws}, j, BT)$;
    $V_{i+1} = D_i$;
    $i = i + 1$;
until *any one defined threshold crossed*;
$T_{down} = V_i / R_{page}$;
return T_{mig}, T_{down}, V_{mig}

$vm1$ is run with network intensive workload $netperf$ (Netperf, 2015) application and transmits TCP streams to instance $vm3$ running on second PM. The migration of $vm2$ is started from $pm1$ to $pm2$. The network output throughput readings of $vm1$ at source PM and network input throughput of $vm3$ at the destination PM are plotted. It has been observed that network output throughput of $vm1$ and network input throughput of $vm03$ has dropped from 950 Mbps to 610 Mbps when $vm2$ is migrating, as shown in Fig. 5.1.

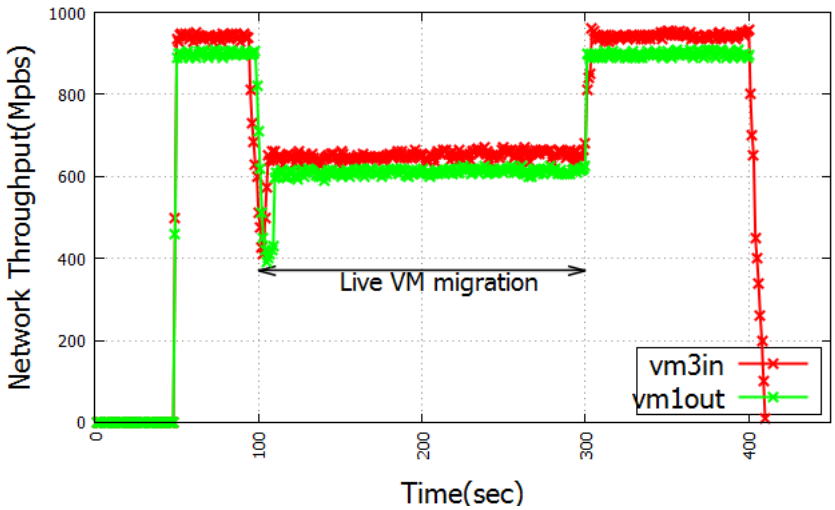

Figure 5.1: netperf TCP

The interference due to VM migration with CPU and memory-intensive workloads is further observed. We started mcf application to run on $vm1$, $vm2$ and $vm3$. The arrangement of two separate VCPUs are made for both the instances, on the source,

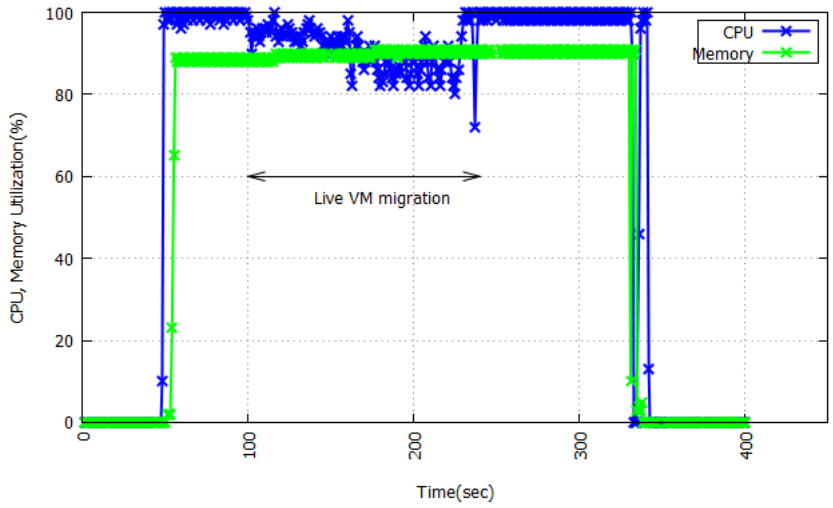

Figure 5.2: mcf Application

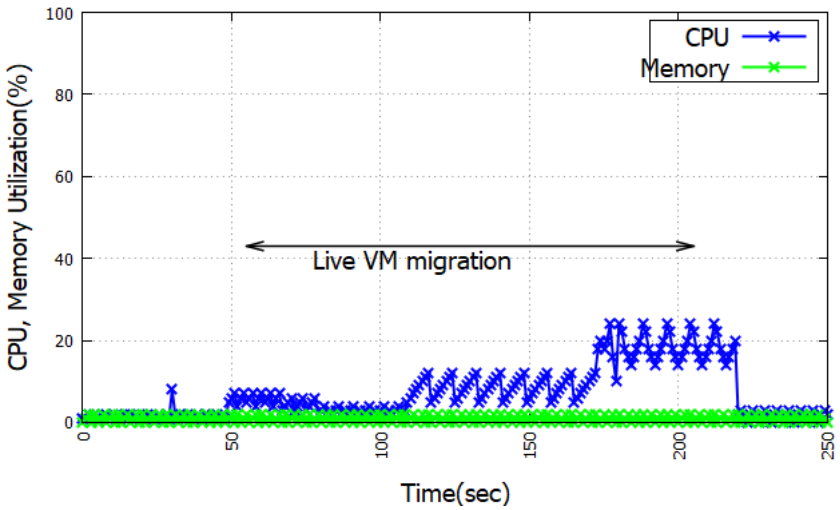

Figure 5.3: CPU and Memory Usage

as well as on destination PM . $vm2$ from $vm1$ is migrated to $vm3$. Experimental result shows that the execution time taken by this application on $vm1$ and $vm3$ is increased by 40-50 seconds during migration of $vm2$. From Fig. 5.2 shows that the CPU of $vm1$ is moderately affected while the memory of $vm1$ remains unchanged during the migration process. Hence it is clear that due to VM migration there is performance degradation in other VMs.

While the above experiments show the VM migration interference which is collected from guest VMs, additional experiments have been carried out to see the performance interference on Domain-0 of source PM. *mcf* application has been installed on $vm1$, $vm2$ on PM1 and $vm3$ on PM2. $vm2$ migrated from PM1 to PM2 and the

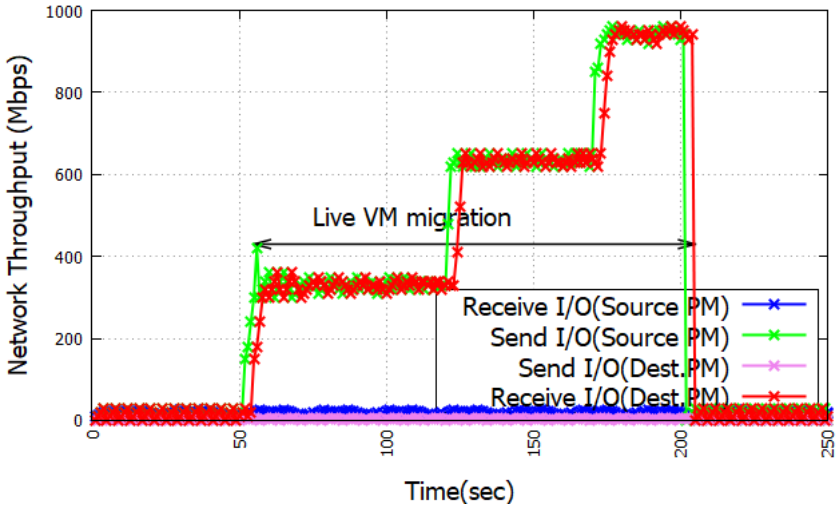

Figure 5.4: Network Throughput in Dom-0 of Source PM and Destination PM

Domain-0 parameters has been observed. It has been observed that CPU usage of domain-0 has increased from 6% to 25% while memory usage remains low, as shown in Fig. 5.3. Network output throughput varies from 30%-90%, in Dom-0 of source PM. Similarly, the network input throughput of domain-0 at destination PM varies from 30%-90% as shown in Fig. 5.4. CPU and memory readings are almost the same as that of source PM.

Apart from VM migration interference, when the migrated VM is resumed at the destination PM, its co-resident interference on other running VMs on destination PM has to be measured and considered in VM selection strategy. This VM co-resident interference helps to decide potential migration destinations for the VMs which need to be migrated. Hence, VM co-resident interference on the destination PM is studied next. VM co-resident interference degrade the performance of the running VMs, hence a degradation in performance can be defined as the ratio of performance in degradation to the original performance. The degraded performance can be measured as decrease in network throughput or increase in execution time.

The performance degradation with two running applications *netperf* and *mcf* is shown in Fig. 5.5 with varying co-resident VMs. It shows that VM co-resident interference increases with the number of co-resident VMs on PM. Performance degradation becomes worse in case of *mcf* application, when number of co-resident VMs goes beyond 8. In case of *netperf* application it becomes worse when number of co-resident VMs goes beyond 5. This is because the *netperf* application is network intensive and

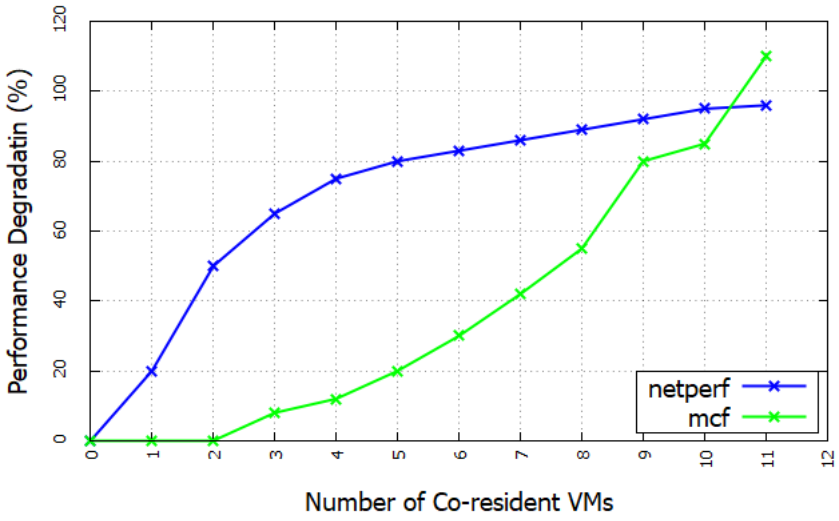

Figure 5.5: The Performance Degradation

VM live migration is itself very network intensive. As the *mcf* application is CPU and network intensive the degradation effect is initially less as compared to *netperf* application.

5.5 Modeling Interference

The migrating VM interferes with the performance of other running VMs on the physical machine. This VM migration interference can be calculated as network I/O contention and CPU resource contention on both migration source and destination PMs. The interference due to network I/O as variations in the network throughput of VMs is modeled. This distribution of variation can be observed as ratio of standard deviation of the network throughputs and mean of network throughputs collected of different VMs. For this the network throughput of VM v_i during some time interval of τ sec has been collected. Then mean μ_i and standard deviation σ_i of the samples that have collected during this time interval of τ seconds are calculated for every VM v_i . Hence interference due to network I/O of Migration process on VM v_i on PM P_i is given by Equation 5.3, where α_i are the number of VMs hosted on PM P_i.

$$N_{M_i} = \sum_{j=1}^{\alpha_i} \frac{\sigma_j}{\mu_j} \tag{5.3}$$

Similarly interference due to CPU usage on VM v_i on PM P_i is given by Equation

5.4.

$$C_{M_i} = \frac{\sum_{j=1}^{\alpha_i}(CPU_j).N_j}{N_i} \tag{5.4}$$

where CPU_j is the CPU utilization of VCPU given to VM v_j and N_i is the number of PCPU hosted by P_i. Hence the total migration interference incurred can be calculated as

$$I_{M_i} = a_1.N_{M_i} + a_2.C_{M_i} \tag{5.5}$$

where a_1 and a_2 are parameters that are used to control the N_{M_i} and C_{M_i} values with respect to their minimum and maximum values.

The main reason of VM co-resident interference is contention at the shared and limited network bandwidth and CPU resources at the destination PM. The interference due to co-resident contention and migration interference during migration, at destination PM, gives the total network I/O interference. This total I/O interference N_{R_i} due to migrating VM v_i on the destination PM P_d is given by

$$N_{R_i} = \frac{\mu_i}{\mu_d} + I_{M_d} \tag{5.6}$$

where μ_i is the mean network requests/interrupt value inside the V_i which can be taken as request demand of network I/O and μ_d is the capacity of destination PM P_d for serving the request demands. The migration interference at P_d is I_{M_d}.

Similarly CPU resource contention due to migrated VM v_i at the destination PM P_d is given by

$$C_{R_i} = \frac{C_i}{N_d} + C_{M_d} \tag{5.7}$$

where C_i is the CPU resource demand of v_i and N_d is the number of physical CPUs available with P_d and C_{M_d} is the CPU resource contention due to migration at destination PM.

The total co-resident interference due to physical resource contention is given by-

$$I_{R_i} = b_1.N_{R_i} + b_2.C_{R_i} \tag{5.8}$$

where b_1 and b_2 are the values that control N_{R_i} and C_{R_i} with respect to their minimum and maximum values.

The total VM performance interference T_{I_i} is the addition of these two migration

interference and co-resident interference. T_{I_i} can be calculated as

$$T_{I_i} = c_1.I_{M_i} + c_2.I_{R_i} \tag{5.9}$$

where c_1 and c_2 are the values used to control I_{M_i} and I_{R_i} with respect to minimum and maximum values.

5.6 Selecting VM for Migration

When there is a need to migrate virtual machine due to insufficient resources for allocation or any other reason like load balancing, power saving, server consolidation in the data center, a subset of VMs, which have least migration interference given by Equation 5.5, is selected. Similarly, potential migration destination PMs are selected by estimating VM's co-resident interference given by Equation 5.8. For every VM in C, a destination PM is selected based on resource requirements and minimum co-location interference. After iterating such pairs of VM and its destination PM, a pair with minimum overall performance interference is selected to carry out migration of that particular VM. This process is represented in the Algorithm 8. Line 1 selects candidate VMs which gives minimum migration interference on source PM of migration and minimum collocation interference on the destination PM. This becomes set C. For every VM in set C, step to identifies a destination PM in such a way that combined VM migration and co-resident interference at the destination PM is less which is shown in line 2. Finally line 3 selects a pair with minimum overall performance interference to carry out migration of that particular VM.

Algorithm 8: Selection of VM for migration

Input: Set V of possible VMs selected for migration , Set P PMs
Output: Selection of VM to be migrated, Selection of destination PM where seleced VM will be migrated

1 Select candidate VMs which gives minimum migration interference on source PM of migration. This set is C.
2 For every VM in set C, identify a destination PM in such a way that combined VM migration and co-resident interference at the destination PM is less.
3 Select a pair with minimum overall performance interference to carry out migration of that particular VM.

5.7 Validating Performance Model of Live VM Migration

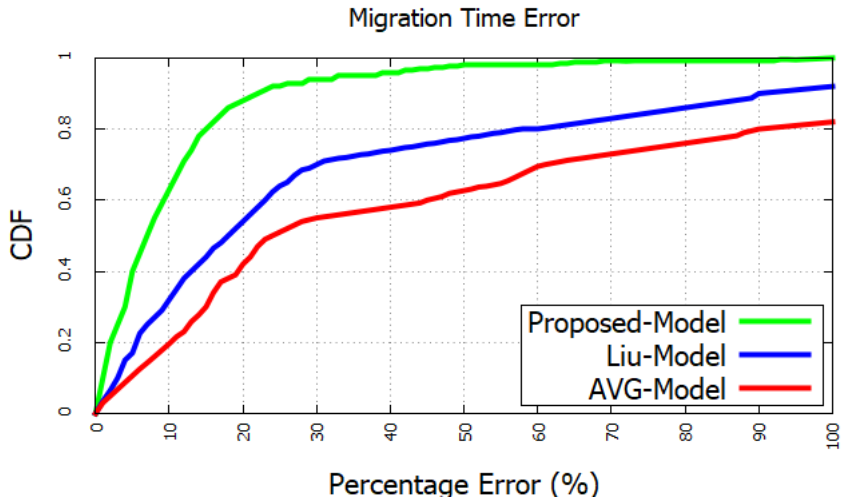

Figure 5.6: Migration Time Error (Percentage)

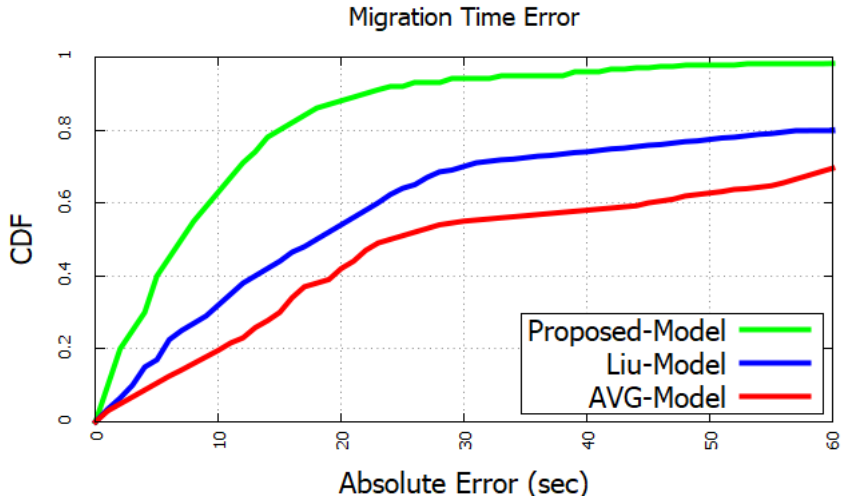

Figure 5.7: Migration Time Error(Absolute Error)

The model of estimating performance of live VM migration has been compared with other models with different techniques of estimating number of skip pages. The other two models, Liu-Model and AVG-Model which are compared have been equipped with *prop-skip* and *hot-skip* techniques respectively as described before. The errors between the values that are received with the proposed model and with actually implementing the migration techniques are compared. The percentage errors and

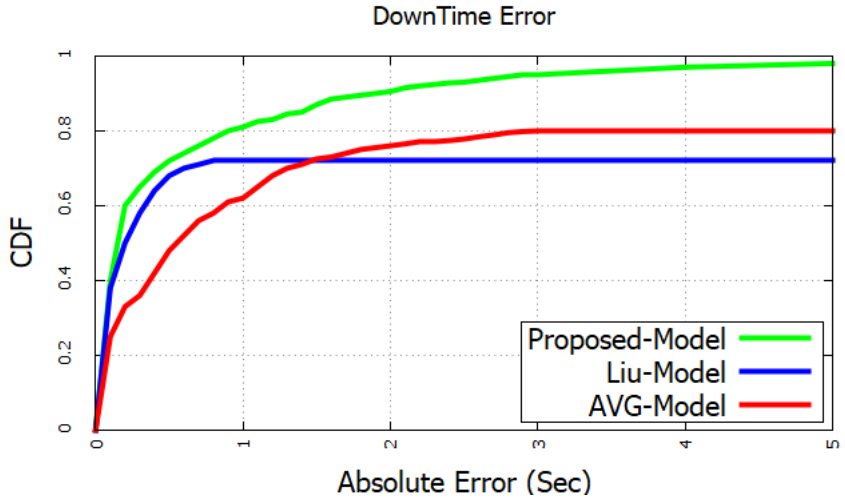

Figure 5.8: DownTime Error

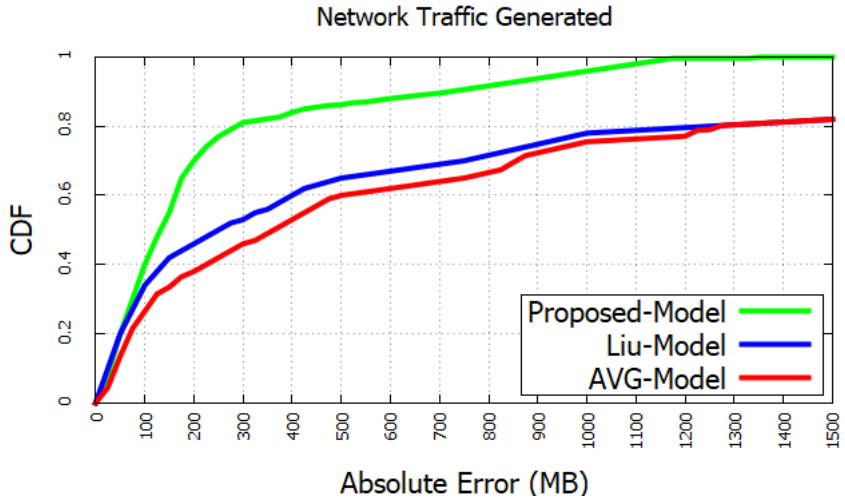

Figure 5.9: Network Traffic Error

absolute errors that we have received with these different techniques are drawn as Cumulative Distribution Function (CDF), as shown in Fig. 5.6 and Fig. 5.7 for migration time. The CDF for downtime and network traffic generated errors are shown in Fig. 5.8 and Fig. 5.9. In all comparisons the proposed model gives minimum errors for the maximum, i.e.80-90%, readings from the model.

5.8 Validating VM Interference Models

We first validate interference due to VM migration which is give by Equation 5.5. For this three VMs have been launched on one PM. Each VM runs the web service. Using *netperf* network performance application network I/O demands are generated

to these web servers on VMs. As these are three different network I/O, demands on three VMs there is network I/O contention. The network throughput of $vm1$ and $vm2$ is fixed to 200Mbps and 450 Mbps and varied on $vm3$ from 100Mbps to 1000 Mbps. Thus the network I/O contention scenario is created, instead actually migrating the VM. The actual network I/O throughput is recorded for 10 seconds. The network I/O interference is calculated by using Equation 5.3. The CPU utilization is recorded to calculate the value of CM_i given by Equation5.4. We calculated the migration interference given by Equation 5.5. The overall degradation in network I/O throughput according to actual network throughput received is calculated. It is observed that calculated network I/O interference values follows the trend of network throughput degradation of the three VMs as shown in Fig.5.10. Next, the estimation

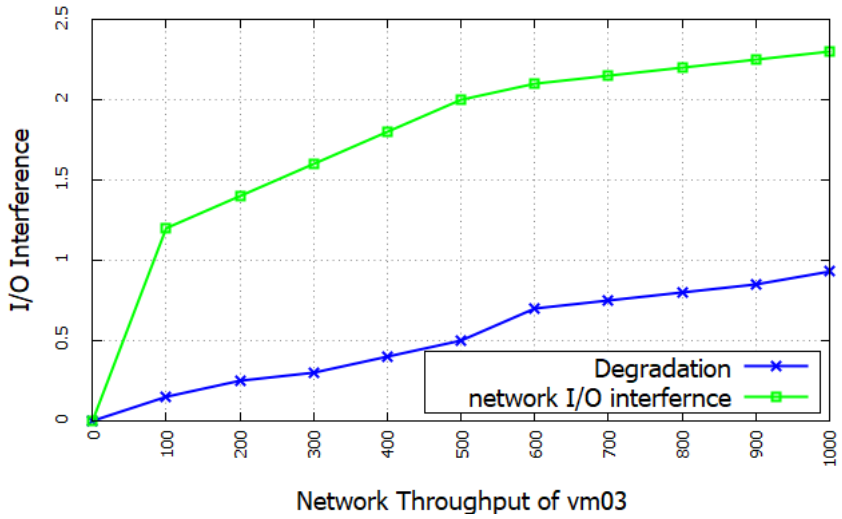

Figure 5.10: Validation of Migration Interference Estimation Model

model for co-resident interference due to migrated VM at destination PM which is given by Equation 5.8 is validated. For this we have taken two PMs are taken, $pm1$ as a source PM and $pm2$ as a destination PM in VM migration. Source PM $pm01$ is running with one VM, $vm01$, which is to be migrated to destination PM $pm02$. We vary the number of VMs on $PM2$ from 1 to 10 and run the experiments separately and taken the reading. Each VM is running with mcf application of SPECCPU2006. The network interrupt readings are collected from the linux $vmstat$ tool and also collected CPU resource utilization for each separate migration with varying number of VMs on the destination PM. The VM co-resident interference RI is calculated

which is given by Equation 5.8. As shown in Fig. 5.11, we observe that, as there is an increase on number of VMs on $PM2$, migrating $vm1$ from $PM2$ to $PM2$ would result in substantially more co-resident interference. This is the same trend as we received performance degradation of mcf application shown in Fig. 5.5.

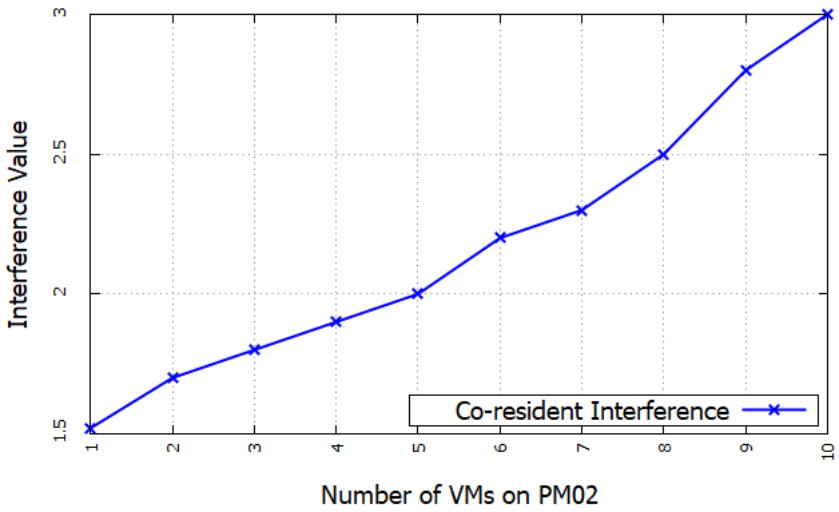

Figure 5.11: Validation of Co-resident Interference Estimation Model

5.9 Performance Comparison in Selecting VM for Migration

In an experimental setup 10 physical machines have been taken. Each PM is installed with Xen virtualization platform, on which small and large type of VM instances are created. The VM types and parameters are taken as small and large instances of VMs. Small instance contains 1 VCPU, 1 GB memory, weight value 256 and cap value 100. While large instance consist of 4 VCPU, 8 GB RAM, weight 256 and cap 400. These represent industry standard VM instances. Typical data center workload is created with different applications running on VMs. Various applications such mcf from SPECCPU2006, $netperf$, $terasoft$ of Hadoop.

The VMs are set on PM_1 as the potential candidates VMs for migration and PMs P_2,P_3,P_4,P_5,P_6,P_7,P_8,P_9,P_{10} are the available destination PM for migration. The physical machine P_1 runs with one large instance and five small instances. Compared to this P_1, others are running with small instances. It is observed that the VM selection for migration and destination PM selection with Sandpiper (Urgaonkar et al.,

2005b) and the proposed method for comparison. Sandpiper strategy selects the large VM instance for migration from PM p_1 and select PM P_5as a destination PM. Sandpiper policy is to select heavily loaded VM for migration to the lightly loaded PM. Hence Sandpiper strategy selects large VM instance for migration towards lightly loaded PM, which is P_5. But the interference strategy selects an other VM which has less migration interference from PM P_1 to destination PM P_8 has the least estimated VM co-resident interference.

5.10 Summary

A live migration procedure model is presented by considering all the parameters which are responsible for performance of live migration. The interference effects caused by the migration procedure itself on other running VMs on PM are studied and the model is presented to calculate CPU and Network interference effects. Next, live migration interference with other running VM on source and destination PM is studied and modeled to capture actual interference. The migration interference and co-resident interference are estimated with the defined model. Migration interference estimation is considered when selecting the candidate VMs for migration, whereas co-resident interference is considered in the decision of selecting destination PM.

Chapter 6

Live VM Migration Manager

In chapter 4, the discussion is on how to predict an application workload and dynamically allocate the resources to the VMs as per their dynamically changing resource requirements. If the demanded requirement cannot be satisfied through the locally available resources on the PM, then resource allocation has to be done using live migration of the VM which is demanding additional resources, to the PM where sufficient amount of demanded resource are available. The alternative solution will be to move any other VM so that sufficient resources will be released and these can be allocated to the demanding VM. During this dynamic allocation of resources using live VM migration, the ultimate aim is not to degrade the performance of the other running VMs and the migrating VM. Hence it is very important to select such a VM for migration such that there will be very minor performance degradation of the applications running on the different VMs on the PM. In Chapter 5, we modeled a live VM migration pre-copy mechanism, which is used in Xen virtualization platform is modeled. Performance and cost parameters of live VM migration can be determined in advance which can be used in deciding which VM has to be selected for migration so that there will be less performance degradation with very little cost of migration. While modeling the pre-copy mechanism we determined performance parameters such as migration time and downtime is determined. The cost parameter like network traffic generated is determined. These parameters are given individual performance and cost for migration. To understand the effects of migration interference on other running VMs on the source and destination PM, migration interference is empirically studied with realistic experiments with benchmark workloads. VM migration performance interference is measured as the addition of CPU and network interference on

PM from where VM is migrating. In selecting the VM for migration this interference cost is also considered. The migration manager which determines the interference cost in addition to migration performance and cost parameters values to decide which VM should be selected for migration.

6.1 Allocating resources using Live VM Migration

6.1.1 Resource Allocation Controller

The fuzzy prediction system is installed for each virtual machine (VM) which predicts the future resource usage. If predicted resource needs can be satisfied from the local physical server on which VM is running these are allocated or deallocated at this machine itself. If predicted resource needs can not be satisfied from the local machine then the VM should be migrated to another physical server in the data center. It is better to predict such situations beforehand where demand is more than supply of resources at the physical machine. Resource Allocation controller of every PM does this work. If migration is required, then this message is given to migration manager. Migration manager selects the VM to be migrated and the destination PM for this migration. This Resource allocation controller is given by Algorithm 8. The load on any VM can be given by following equation.

$$load = \frac{1}{1-cpu} * \frac{1}{1-nw} * \frac{1}{1-mem} \tag{6.1}$$

where *cpu*, *nw* and *mem* are utilization readings of CPU, network and memory normalized by maximum number of CPUs, maximum network interfaces and maximum amount of memory allocated to VM respectively.

6.1.2 Prediction of Need for Migration

Using the same resource usage prediction system the future time resource needs are predicted for every VM. Let there are K number of VMs running on a physical server. These VMs are denoted by vm_1,vm_2,.....,vm_K. Let $[C_{i,t+1}, ..., C_{i,t+L}]$ denotes future resource needs predicted at time t for future times $t+1$ to $t+L$. The total demand

Algorithm 9: Controller of each Physical Machine

Input: $W = \{C_{t-l}, ..., C_{t-1}\}$, P = $W/CapVal$

Output: Padding values, value of α to raise the CPU cap, A : Allocation of additional resources as per the predicted needs,

1 $PadVal$ = EvaluatePadVal(W, P);
2 $Load$ = (1/(1-cpu)) * (1/(1-nw)) * (1/(1-mem));
3 $\{e_{t-l}, ..., e_{t-1}\}$ = P - A;
4 $N_{M_i} = \sum_{j=1}^{\alpha_i} \frac{\rho}{\mu_j}$;
5 $C_{M_i} = \frac{\sum_{j=1}^{\alpha_i} (CPU_j).N_j}{N_i}$;
6 $I_{M_i} = a_1.N_{M_i} + a_2.C_{M_i}$;
7 **if** *Any of the e component is negative* **then**
8 α = (P-P_{under})/(1-P_{under});
9 **end**
10 **repeat**
11 $CapVal = CapVal * \alpha^k$;
12 **until** *All the e's become positive OR* $CapVal < CapTotal$;
13 **if** $CapVal < CapTotal$ **then**
14 Trigger migration ;
15 **end**
16 **if** *Need for migration is estimated at time* t_m **then**
17 Send the signal to migration manager;
18 **end**

on the physical server is given by-

$$[\sum_{i=1}^{K} C_{i,t+1}, ..., \sum_{i=1}^{K} C_{i,t+L}] \tag{6.2}$$

We can estimate when there is need for migration by comparing the calculated total demand with its capacity.

$$\sum_{i=1}^{K} C_{i,t_m} > C \tag{6.3}$$

The selection of VM for migration and its destination PM is decided by following Algorithm 9. Line 1 calculates the padding value as described in Algorithm 3. Line 2 combines the load using Equation (6.1). Line 3 calculates underestimations errors observed. Line 4 calculates the interference due to network I/O of migration process of every VM on every PM using the Equation (5.3). Line 5 calculates the interference due to CPU usage for every VM on every PM using the Equation (5.4). Line 6 combines the interference of network I/O and CPU contention as total interference due to migration. Lines 7 to 9 calculates the ratio required to raise the cap value of

the resource if any underestimations errors are present. Line 11 raise the cap value of the resource. Line 10 to 12 is repeated until all underestimation errors got resolved. If raised cap value is not accommodated in the present PM then migration is triggered as shown in lines 13 to 15. Lines 16 to 18 also identifies is there any need for migration by leveraging previous prediction methods. If there is need for migration in future is predicted then a signal is given to the migration manager.

6.1.3 Migration Manager

The migration manager observes all PMs in the data center with the resource monitoring system described in previous chapter. It collects the resource usage information from all VMs on all PMs. The manager is shown in Algorithm 10. Each PM arrange all the VMs in the decreasing order of the $load\ /\ VM_{size}$ ratio. The load is considered with respect to the size of the VM. Line 1 represents this step. If there is a signal for migration from any PM this algorithms selects a VM for migration and the destination for the same. All these steps are shown in lines 2 to 9 of the Algorithm 10. Line 3 select the VM from PM having highest $load/VM_{size}$ value and minimum V_{mig} and T_{mig} values and minimum total T_{I_i} value given by Equation (5.9). Lines 4 to 6 select a destination PM for a selected VM. Such PM should be with least loaded and with sufficient resources availability. If so such PM found then select a VM with second highest $load/VM_{size}$ value as shown in line 8 to 10 for migration. Repeat the steps in line 7 to 11 until suitable PM is found.

6.2 Performance Verification

We studied the performance of proposed migration manager in the cloud environment were mixed types of workloads are running. We set up a cluster of eight physical servers and installed xen virtualization platform on every server. Our proposed system is added on above this. A mix of different workload VMs are created and running on the cluster. Each type of workload is running in nine different VMs, thus in all (4*9) =36 VMs are running in the cluster. Out of these VMs, few VMs are with larger memory and others are with small. The VMs initial placement is shown in Fig. 6.1. The following notations are used to denote the running workloads and VM with

Algorithm 10: Migration Manager

Input: Load on individual VM on every PM, Total load on PM
Output: VM to be migrated, Destination PM where seleced VM will be migrated

Within each PM arrange the VMs in the decreasing order of $load\ /\ VM_{size}$;
if *Trigger for Migration from any PM* **then**
    Select the VM from PM having highest $load/VM_{size}$ value and minimum V_{mig} and T_{mig} values and minimum total T_{I_i} value given by Equation (5.9);
    repeat
        Select the next least loaded PM as a destination for candidate VM;
    until *Least loaded server found with sufficient resources available*;
    repeat
        if *No PM can satisfy the candidate VM* **then**
            Select the VM from highest loaded server having next highest $load/VM_{size}$ value and minimum V_{mig} and T_{mig} values and minimum T_{I_i} value given by Equation (5.9);
        end
    until *until a suitable PM is identified*;
end

specific workload.

1. $W1$: SPECCPU2006
2. $W2$: netperf v.2.4.0
3. $W3$: Hadoop v.0.20.2
4. $W4$: SPECweb2005 Release 1.30
5. VM_{Wn_L} : Large VM running Wn type of benchmark workload, where n in 1..4
6. VM_{Wn} : Small VM running Wn type of benchmark workload, where $n \in 1..4$
7. $VM_{Wn(m)}$: m number of VMs running Wn type of benchmark workload, where n in 1..4

To verify the performance of the proposed system we set the VMs on PM_1 in such a way that these will be selected for migration and PMs (P_1,P_2,P_3,....,P_n) are the destinations for the migration. The VM migrations decisions taken by the different schemes are shown in Fig. 6.2. We compared our proposed system with two others, *Sandpiper* and $First - FitDecreasing(FFD)$. The *Sandpiper* is for load balancing

given by and the FDD is power and migration cost aware application placement technique given by. It is observed that both *Sandpiper* and FFD selects VM_{W1_L} on PM1 for migration, whereas proposed system selects VM_{W3} for migration. The destination chosen by *Sandpiper*, FFD and proposed system are PM_7, PM_4 and PM_8 respectively.

The reasons behind selecting VM_{W1_L} for migration by *Sandpiper* and FFD are - (1) *Sandpiper* select the most loaded VM for migration and the destination is the least loaded PM. The load in *Sandpiper* is calculated as $\frac{1}{1-cpu} * \frac{1}{1-nw} * \frac{1}{1-mem}$. According to this formula VM_{W1_L} is the most loaded as CPU usage is almost 100% and memory allocated is 90%. Hence it is selected for migration. PM_7 is the least loaded PM, hence it is selected as destination PM. (2) FFD selects largest CPU and memory size VM for migration and destination will be selected such that power consumption will be the least. Accordingly, FFD selects VM_{W1_L} for migration as it is consumed with more memory and CPU. The destination PM will be PM_4.

The proposed system selects small VM_{W3} for migration and the destination PM is PM_8. The reason behind this is- (1) The least estimated interference due to VM migration on source PM is given by VM_{W3} and (2) PM_8 is with least estimated VM co-resident interference.

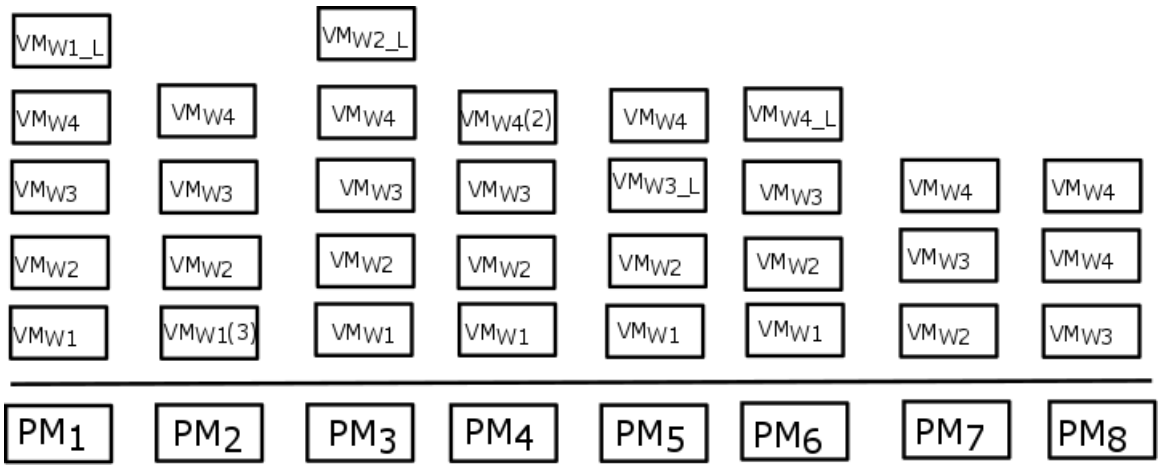

Figure 6.1: VM's Initial Placement

The performance of the different workloads is measured as application execution time, network throughput. The performance measurements given here are the average across multiple runs on different VMs in the cluster. In case of SPECWeb and Hadoop

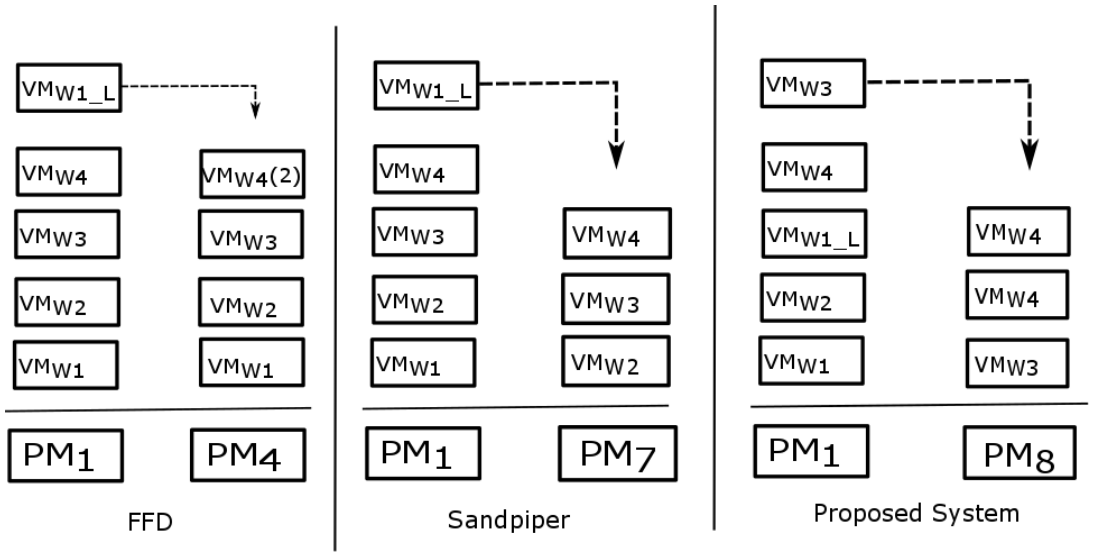

Figure 6.2: Migration Decisions under Different Schemes

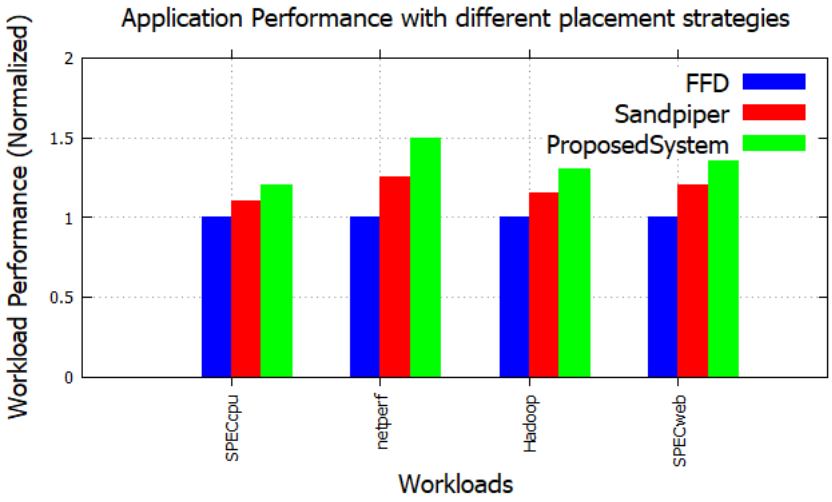

Figure 6.3: Performance of Applications under Different Schemes

performance is retrieved form their "master" instances which co-ordinate with "slave" to complete the job. Here the performance is number of successfully completed jobs against submitted. Fig. 6.3 compares the performances of different workloads during migration. The performance measurements are normalized to their maximum performances when no any other interference exists during migration. The measurements shows the proposed migration manager increase the system performance by around 45%-50% and improvement of 25%-30% for cpu and memory intensive workloads as compared with FFD. As compared with Sandpiper for network intensive workload the improvement is 35%-40% and for cpu and memory intensive workload the improve-

ment is 15%-20%. Sandpiper selects the VM with maximum *load* to *memory_size* ratio. Hence it will select the heavily loaded VM with less memory size. But as the VM is heavily loaded it consumes high network bandwidth and high CPU. Hence its interference will be more when taken for migration. In our algorithm the VM incurs less interference cost is selected for migration. Hence the improvement in the throughput.

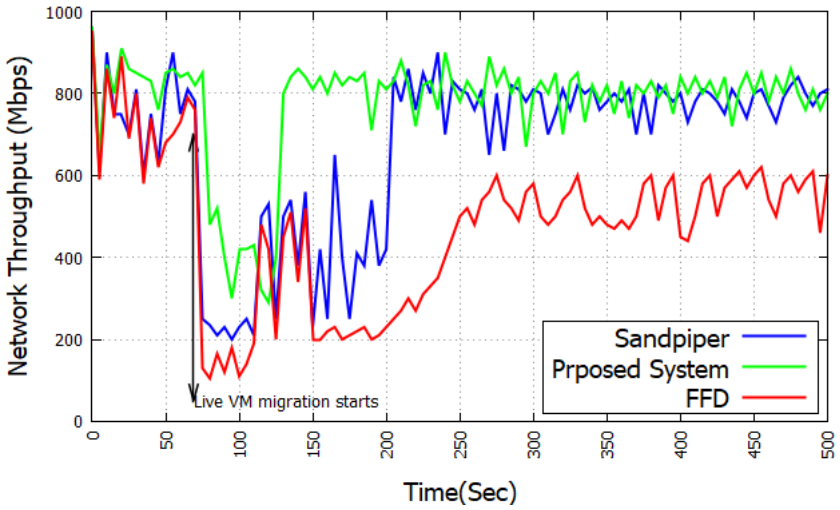

Figure 6.4: Performance of Applications under Different Schemes

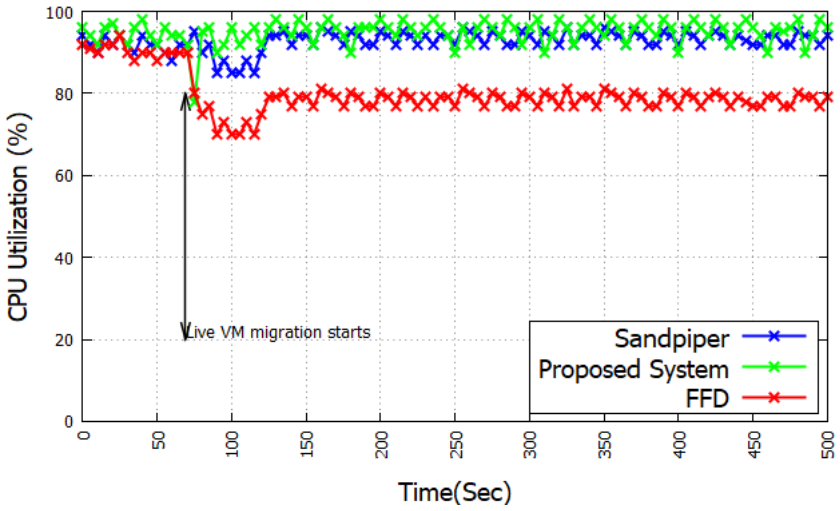

Figure 6.5: Performance of Applications under Different Schemes

It is also observed that the performance specifically during the migration time and after VM migration. It is shown in Fig. 6.4 and Fig. 6.5. It is found that both Sandppiper and FDD severly degrade the network throughput and moderately

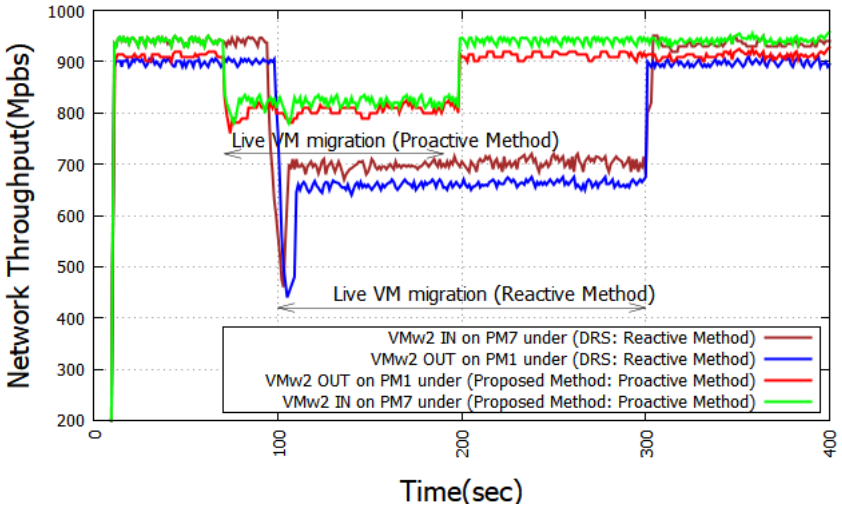

Figure 6.6: Network Throughput Variation during Migration

degrade CPU utilization of VMs compare to proposed system. The VMs selected by Sandpiper and FFD are with heavily loaded with large VM size and heavy resource consumption. Destination selected by Sandpiper and FFD during migration have less number of resources which are in contention with others. Thus these migration decisions results into more CPU, network and memory interference at source and destination PMs. This ultimately results into poor performance. In contrast proposed method takes care of migration and co-resident interference during migration, hence the VM selected for migration and the destination PM will incur less interference. Hence performance is so much degraded during and after migration. Thus the proposed system not only consider the migration time, downtime, and traffic generated during the migration, but it also looks into the interference effects on the same VM and other running VMs.

The proposed system is proactive in nature. It is compared with the reactive system VMware DRS and network throughput is observed before, during and after the live migration of VM. The proposed system gives better network throughput all the time as shown in Fig. 6.6. The improvement in the results in the proposed system is due to proactive handling of resource needs and migration decisions. The resources are allocated on the same PM or the migration is done if sufficient resources are not available on the same PM before the actual peak requirement came into existence. This need is detected prior to the actual happening and accordingly resource allocation is done. This all observed during the live VM migration time in the Fig. 6.6.

6.3 Service Level Agreements (SLA) Violation Study

A small datacenter testbed is created on two Intel Xeon Servers with hyperthreaded dual Intel Xeon 3.2 GHz CPUs and 4 GB RAM. Virtualization layer is created with Xen hypervisor. Virtual machines are created on these servers machines with different initial minimum requirements to run OS, Local Controller and Web Server like application. Local controller is deployed on every virtual machine. A Global Controller is deployed on separate Desktop Machine with Intel i5 processor and 4GB RAM. It collects the CPU requirements from Local Controllers and requests the required amount of CPU resource to the Xen resource pool. All these servers and desktop machine are connected with Gigabit Ethernet. HTTP Workload is generated from real time HTTP workload traces using real-time Logreplayer (2011). RUBiS (2012) benchmark is used to create a WebStore which represents typical e-business application. Httperf is used to generate http requests which represents http workload. Different real-time traces like (Wikimedia, 2013), CAIDA (CAIDA, 2016) , TPC-W (TPC-W, 2005) are used for experimentation and evaluation of the proposed system. Analysis with all these workload traces are shown in following subsections. All the controller approaches described above viz. Proactive Moving Average Predictive Controller, Fuzzy Logic Controller, Kalman Filter Controller and combined Fuzzy Logic Prediction and Kalman Filter Controller are tested with different real-time workload traces and analyzed.

These traces contain two weeks worth of all HTTP requests to the ClarkNet web server. ClarkNet is a full Internet access provider for the Metro Baltimore-Washington DC area. The workload trace was transformed to a time series of web requests per minute and used for testing the various resource management systems that have been developed. The RUBiS benchmark is used to represent a real world e-business application. RUBiS is an auction site prototype modeled after eBay.com. HTTP workload that represent the ClarkNet trace is generated with various client requests issued by $httperf$. The estimation of resource required to satisfy the incoming workload is done by local controller. It is implemented using the Fuzzy Prediction approach described earlier. This fuzzy prediction based local controller is compared with other techniques like Proactive Moving Average Predictive Controller, Fuzzy Logic Controller, Kalman

Filter Controller and Combined approach with Kalman and Fuzzy prediction. The results are shown in Fig. 6.7. The RNN with LSTM proposed method for prediction gave less number of violation in all.

The datacenter cluster set of 29 days provided by Google is also tested with respect to CPU usage. The First twenty days data is used for modeling and prediction. The CPU usage requirements for day 21,22 and 23 are calculated and counted the number of underestimates/ SLA violations. The results shows that the number of SLA violations are less with proposed method. Here also Kalman Filter gives the noise reducing advantage. The results are shown in Fig.6.8 and found that the proposed method gives less number of SLA violations. Due to advantage of LSTM, RNN gave less number of SLA violation.

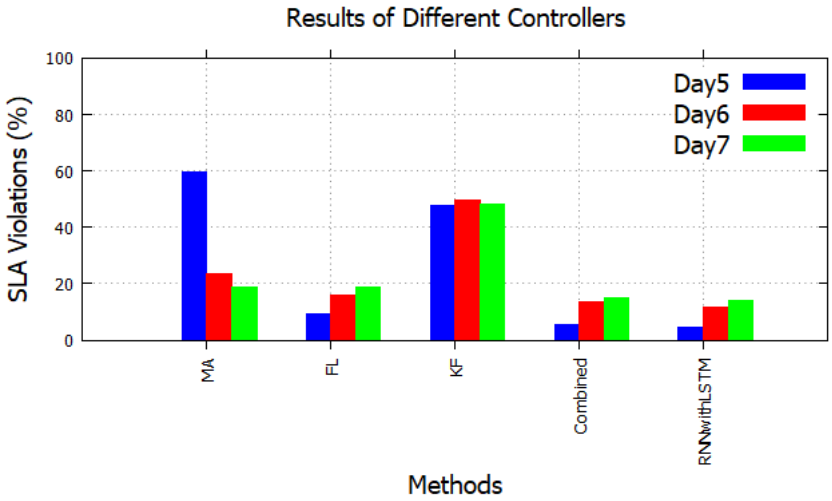

Figure 6.7: ClarkNet Data Trace

The system is also evaluated with three recent real world traces. The first includes traces of requests to the web server from the Wikimedia Foundation collected August 2016. These includes HTTP requests requesting for static web pages and images aggregated in one hour intervals. The second one includes traces from CAIDA (Center for Applied Internet Data Analysis) internet traces collected in April 2016. Web Store is implemented using RUBiS benchmark. The http requests representing these workloads are issued to Web Store. The CPU resource requirement is predicted for the next hour for allocating resources in both the cases and given it to global controller for allocation. The third data set used is TPC-W benchmark for the specification of

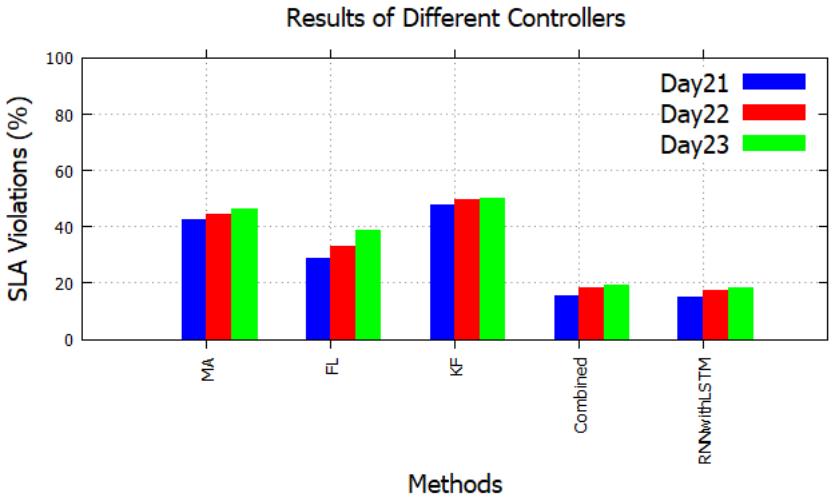

Figure 6.8: Google Cluster Trace

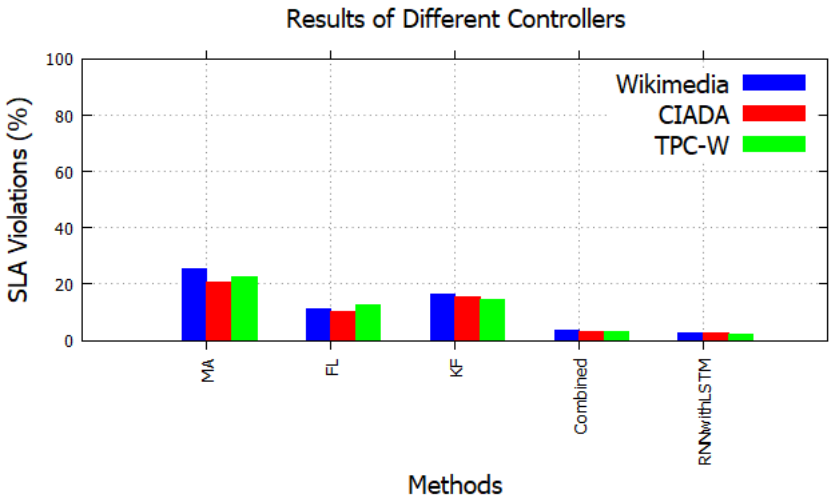

Figure 6.9: Wikimedia, CAIDA & TPC-W

e-commerce applications. The http requests are issued as per the benchmark specification. The CPU resource is predicted for next one hour. The percentage SLA violations with these three data sets are given in Table 6.1 and shown in Fig.6.9. The proposed approach produce less number of SLA violations.

The prediction accuracy is validated using Wikimedia, CAIDA and TPC-W workloads. The measurement parameter MAE (Mean Absolute Error) is calculated with various methods as shown in Fig.6.9. The prediction accuracy with average MAE of 0.08 which is pretty good.

Workload Trace	Moving Average	Fuzzy Logic	Kalman Filter	Combined
Wikimedia	25.34	11.25	16.20	3.80
CAIDA	20.65	10.40	15.35	3.16
TPC-W	22.55	12.40	14.35	3.30

Table 6.1: Percentage SLA Violations

6.3.1 Resource Allocations through VM Live Migrations

The need for migration is predicted by leveraging the same fuzzy prediction based technique. The results are shown in the following figures. When CPU resource goes beyond 50%, system predicts the future load and it concludes that further load is increasing drastically at time t=630 sec as shown in Fig.6.10 when total CPU usage goes beyond total capacity. Hence CPU availability is more for contention.

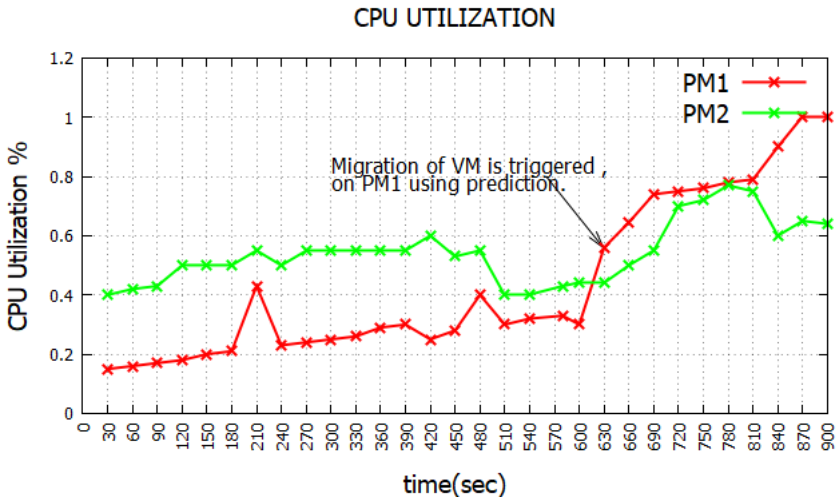

Figure 6.10: CPU Utilization of PM1 and PM2

In case of Network Utilization as shown in Fig.6.11, when the network load goes beyond 60%, it is predicted that the network load will increase from time 450 sec hence at this point migration is triggered.

The response time for the web application running on a VM to be migrated as shown in Fig.6.12 is calculated for every 4sec interval during migration. The RUBiS workload with 100 clients is generated and response time is measured at the interval of 4sec. It is observed that there is delay in response time during the period of migration process. If it is experimented without proposed system the delay in response time is

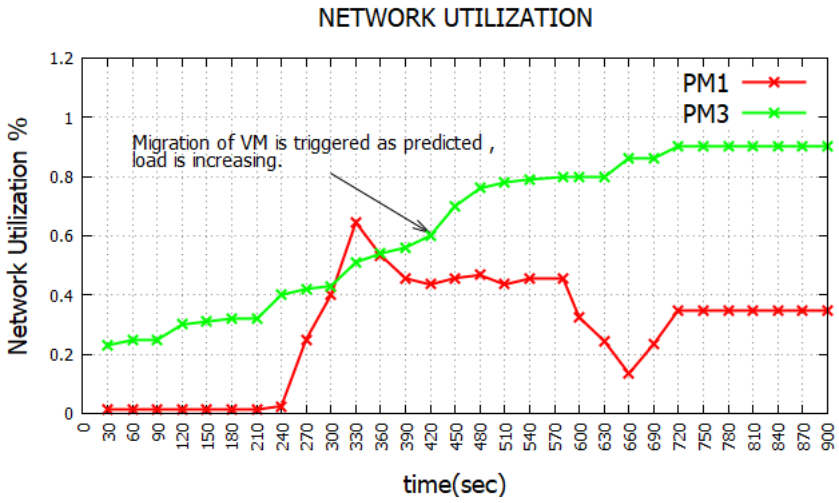

Figure 6.11: Network Bandwidth Utilization of PM1 and PM3

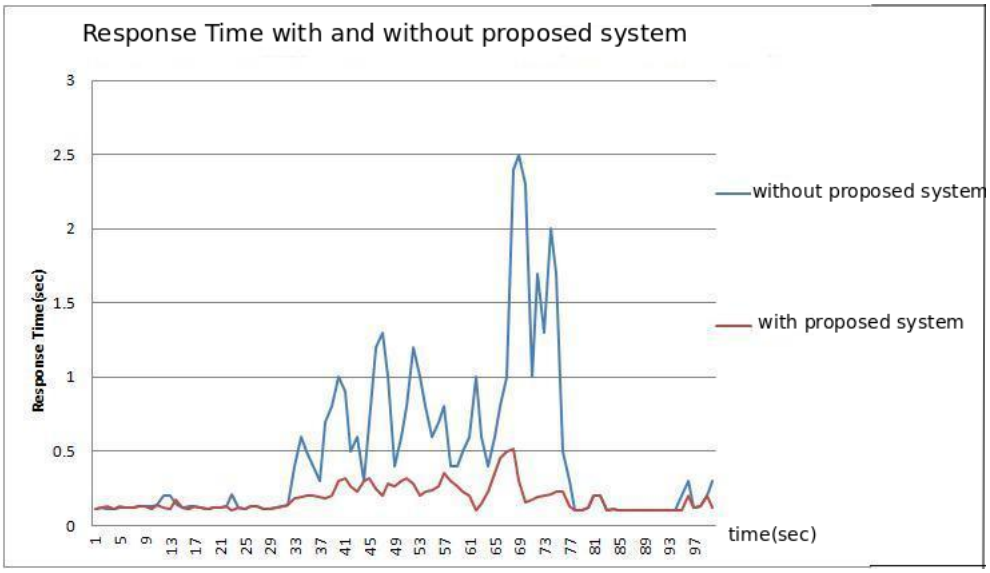

Figure 6.12: Response Time

more and it is successful in processing 148 requests . With the proposed system the delay is less and response time is quick, hence 198 requests were served. It is observed that there is an average 50% decrease in the response time.

6.4 Summary

The migration manager presented in this chapter detects when to trigger migration, select the VM for migration so that interference effects is reduced and select the destination PM where selected VM is to be migrated.

Chapter 7

Conclusions and Future Work

This work presents an intelligent system which predicts future resource demand of the applications on VMs and allocates the resources in line with actual requirements to run with the desired SLA. It combines the fuzzy prediction system and prediction based on Kalman filter depending upon the workload type and time needed for prediction. The combined approach gives better results as compared to others used in the system. The Chaos indicator is found to be a very efficient module to check how chaotic the trend of CPU usages have been, and this is also able to distinguish noisy signals from clean signals which follows a deterministic pattern. If a very noisy and chaotic signal is seen, it switches to use Kalman filter for base prediction otherwise it switches to fuzzy prediction system. It thus optimizes on base allocations in both scenarios. Kalman Filter also was found to perform well and is able to generate base patterns on the fly as required. When the incoming workload is noisy, we conclude that it provides superior performance compared to general proactive system. Finally the prediction model with RNN and LSTM gives better performance than all other methods implemented above. The RNN-LSTM prediction model is tested with three different workload.The prediction accuracy increased 10%- 20% when compared with our Fuzzy Prediction System.

To avoid underestimation prediction errors due to spikes in the workload, the predicted values are padded with proper value. If such errors are detected again then resource caps have to be raised immediately resource caps are raised. The resource conflict (when enough resources are not available) situations are resolved with live VM migration. Migration manager takes care of properly selecting VM for migration and destination PM so as there is not be any performance loss. The migration manager

triggers the migration with the help of performance, cost and interference parameters generated from the live migration model. It improves the performance of the system.

The scaling of the CPU resource is automatically done in accordance with dynamically changing workload at a minimum granularity of 2 seconds. The resource saving with proposed method is around 30-50% as compared to static allocations. The performance improvement in terms of response time of an application is around 15-20% as compared to other methods because of proper selection of VM for migration by the migration manager.

In this work, the workload type is studied with respect to finding pattern in it or not. Other characteristics of the data may be useful to improve the prediction performance. Hence in future we plan to study the characteristics of the data and to find out whether it is useful for improving prediction accuracy. Most of the parameters affecting live VM migration are considered, but parallel migrations of VMs if required in the scenario is not studied. This will be explored in the future. The applications workloads may be non-stationarity in nature which will be addressed in the future work. The storage of the resource usage also may reveal the working style of an application and some important behavior. Hence these should be addressed.

References

Adami, D., Gabbrielli, A., Giordano, S., Pagano, M. and Portaluri, G. (2015). "A Fuzzy Logic Approach for Resources Allocation in Cloud Data Center." *2015 IEEE Globecom Workshops (GC Wkshps)*. 1–6.

Akoush, S., Sohan, R., Rice, A., Moore, A. W. and Hopper, A. (2010). "Predicting the Performance of Virtual Machine Migration." *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. 37–46.

Al-Qawasmeh, A. M., Pasricha, S., Maciejewski, A. A. and Siegel, H. J. (2015). "Power and Thermal-Aware Workload Allocation in Heterogeneous Data Centers." *IEEE Transactions on Computers*, 64(2), 477–491.

Aldhalaan, A. and Menascé, D. A. (2013). *Analytic Performance Modeling and Optimization of Live VM Migration*. Springer Berlin Heidelberg, Berlin, Heidelberg, 28–42.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2010). "A View of Cloud Computing." *Commun. ACM*, 53(4), 50–58.

AWS (2012a). "AWS Case Study: NASA/JPL's Desert Research and Training Studies." `http://aws.amazon.com/solutions/case-studies/nasa-jpl/`. (Dec. 5, 2016).

AWS (2012b). "AWS Case Study: Netflix." `http://aws.amazon.com/solutions/case-studies/netflix`. (Dec. 5, 2016).

Azure (2012). "Windows Azure." `http://windowsazure.com`. (Jan.25, 2017).

Bankole, A. A. and Ajila, S. A. (2013). "Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment." *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. 156–161.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A. (2003). "Xen and the Art of Virtualization." *SIGOPS Oper. Syst. Rev.*, 37(5), 164–177.

Bila, N., de Lara, E., Joshi, K., Lagar-Cavilla, H. A., Hiltunen, M. and Satyanarayanan, M. (2012). "Jettison: Efficient Idle Desktop Consolidation with Partial VM Migration." *Proceedings of the 7th ACM European Conference on Computer Systems*. EuroSys '12, ACM, New York, NY, USA, 211–224.

Bonvin, N., Papaioannou, T. G. and Aberer, K. (2011). "Autonomic SLA-Driven Provisioning for Cloud Applications." *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 434–443.

Bose, S. K., Brock, S., Skeoch, R. and Rao, S. (2011). "CloudSpider: Combining Replication with Scheduling for Optimizing Live Migration of Virtual Machines across Wide Area Networks." *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 13–22.

CAIDA (2016). "CAIDA Data Set." `https://www.caida.org/data/` (Oct. 15, 2016).

Calheiros, R. N., Masoumi, E., Ranjan, R. and Buyya, R. (2015). "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications 2019; QoS." *IEEE Transactions on Cloud Computing*, 3(4), 449–458.

Chandra, A., Gong, W. and Shenoy, P. (2003). "Dynamic Resource Allocation for Shared Data Centers Using Online Measurements." *SIGMETRICS Perform. Eval. Rev.*, 31(1), 300–301.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M. and Doyle, R. P. (2001). "Managing Energy and Server Resources in Hosting Centers." *SIGOPS Oper. Syst. Rev.*, 35(5), 103–116.

Chiaraviglio, L. and Matta, I. (2010). "GreenCoop: Cooperative Green Routing with Energy-efficient Servers." *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. e-Energy '10, ACM, New York, NY, USA, 191–194.

Choi, H. W., Kwak, H., Sohn, A. and Chung, K. (2008). "Autonomous Learning for Efficient Resource Utilization of Dynamic VM Migration." *Proceedings of the 22Nd Annual International Conference on Supercomputing*. ICS '08, ACM, New York, NY, USA, 185–194.

Chou, L., Chen, H., Tseng, F., Chao, H. and Chang, Y. (2018). "DPRA: Dynamic Power-Saving Resource Allocation for Cloud Data Center Using Particle Swarm Optimization." *IEEE Systems Journal*, 12(2), 1554–1565.

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I. and Warfield, A. (2005). "Live Migration of Virtual Machines." *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*. NSDI'05, USENIX Association, Berkeley, CA, USA, 273–286.

ClarkNet (2012). "ClarkNet Data Set." `ftp://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html` (Oct. 20, 2016).

Dabbagh, M., Hamdaoui, B., Guizani, M. and Rayes, A. (2015). "Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers." *IEEE Transactions on Network and Service Management*, 12(3), 377–391.

Das, T., Padala, P., Padmanabhan, V., Ramjee, R. and Shin, K. G. (2010). "LiteGreen: Saving Energy in Networked Desktops Using Virtualization." *USENIX Annual Technical Conference [BEST PAPER]*. USENIX.

Deng, L., Jin, H., Chen, H. and Wu, S. (2013). "Migration Cost Aware Mitigating Hot Nodes in the Cloud." *2013 International Conference on Cloud Computing and Big Data*. 197–204.

Diao, Y., Hellerstein, J. L. and Parekh, S. (2002). "Using fuzzy control to maximize profits in service level management." *IBM Systems Journal*, 41(3), 403–420.

Doyle, R. P., Chase, J. S., Asad, O. M., Jin, W. and Vahdat, A. M. (2003). "Model-based Resource Provisioning in a Web Service Utility." *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*. USITS'03, USENIX Association, Berkeley, CA, USA, 5–5.

EC2 (2012). "Amazon EC2." `http://aws.amazon.com/ec2/.` (April. 10, 2017).

Exponent, H. (2019). "Hurst Exponent." `https://en.wikipedia.org/wiki/Hurst_exponent` (April. 15, 2016).

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M. and Patterson, D. (2009). "Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning." *Proceedings of the 2009 IEEE International Conference on Data Engineering*. ICDE '09, IEEE Computer Society, Washington, DC, USA, 592–603.

Ganapathi, A. S. (2009). *Predicting and Optimizing System Utilization and Performance via Statistical Machine Learning*. Ph.D. thesis, EECS Department, University of California, Berkeley.

Ganesh S, G. P., Batista, D. M. and da Fonseca, N. L. (2016). "Towards Proactive Resource Management in Virtualized Datacenters." *VMware Technical Paper*, 4(1), 3.

Gmach, D., Rolia, J., Cherkasova, L., Belrose, G., Turicchi, T. and Kemper, A. (2008). "An integrated approach to resource pool management: Policies, efficiency and quality metrics." *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. 326–335.

Gmach, D., Rolia, J., Cherkasova, L. and Kemper, A. (2007). "Capacity Management and Demand Prediction for Next Generation Data Centers." *IEEE International Conference on Web Services (ICWS 2007)*. 43–50.

Gong, Z., Gu, X. and Wilkes, J. (2010a). "PRESS: PRedictive Elastic ReSource Scaling for cloud systems." *2010 International Conference on Network and Service Management*. 9–16.

Gong, Z., Gu, X. and Wilkes, J. (2010b). "PRESS: PRedictive Elastic ReSource Scaling for cloud systems." *2010 International Conference on Network and Service Management*. 9–16.

Google (2012). "Google Compute Engine." `https://cloud.google.com/products/compute-engine/`. (Dec. 15, 2015).

Govindan, S., Choi, J., Urgaonkar, B., Sivasubramaniam, A. and Baldini, A. (2009). "Statistical Profiling-based Techniques for Effective Power Provisioning in Data Centers." *Proceedings of the 4th ACM European Conference on Computer Systems*. EuroSys '09, ACM, New York, NY, USA, 317–330.

Guo, T., Sharma, U., Shenoy, P., Wood, T. and Sahu, S. (2014). "Cost-Aware Cloud Bursting for Enterprise Applications." *ACM Trans. Internet Technol.*, 13(3), 10:1–10:24.

Gupta, M. and Singh, S. (2003). "Greening of the Internet." *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '03, ACM, New York, NY, USA, 19–26.

hadoop (2012). "Apache Hadoop Workload." `http://hadoop.apache.org/` (Sept. 23, 2014).

Haider, A., Potter, R. and Nakao, A. (2009). "Challenges in resource allocation in network virtualization." *20th ITC Specialist Seminar, 18.-20. May 2009, Hoi An, Vietnam*. ITC 2009, ACM, New York, NY, USA, 317–330.

Heo, J., Zhu, X., Padala, P. and Wang, Z. (2009). "Memory overbooking and dynamic control of Xen virtual machines in consolidated environments." *2009 IFIP/IEEE International Symposium on Integrated Network Management*. 630–637.

Hoyer, M., Schröder, K., Schlitt, D. and Nebel, W. (2011a). "Proactive Dynamic Resource Management in Virtualized Data Centers." *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*. e-Energy '11, ACM, New York, NY, USA, 11–20.

Hoyer, M., Schröder, K., Schlitt, D. and Nebel, W. (2011b). "Proactive Dynamic Resource Management in Virtualized Data Centers." *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*. e-Energy '11, ACM, New York, NY, USA, 11–20.

HP (2012). "HP Cloud Services." `http://hpcloud.com/`. (Nov. 10, 2016).

HPUX (2012). "HP-UX Workload Manager." `http://docs.hp.com/en/5990-8153/ch05s12.html`. (May. 15, 2015).

Httperf (2015). "HTTP Workload Generator." `https://github.com/httperf/httperf` (Jan. 13, 2015).

Hu, Y., Deng, B., Peng, F. and Wang, D. (2016). "Workload prediction for cloud computing elasticity mechanism." *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. 244–249.

Islam, S., Keung, J., Lee, K. and Liu, A. (2012). "Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud." *Future Gener. Comput. Syst.*, 28(1), 155–162.

Jeong, J., Kim, S.-H., Kim, H., Lee, J. and Seo, E. (2013). "Analysis of Virtual Machine Live-migration As a Method for Power-capping." *J. Supercomput.*, 66(3), 1629–1655.

Jheng, J.-J., Tseng, F.-H., Chao, H.-C. and Chou, L.-D. (2014). "A novel VM workload prediction using Grey Forecasting model in cloud data center." *The International Conference on Information Networking 2014 (ICOIN2014)*. 40–45.

Kalyvianaki, E., Charalambous, T. and Hand, S. (2009). "Self-adaptive and Self-configured CPU Resource Provisioning for Virtualized Servers Using Kalman Filters." *Proceedings of the 6th International Conference on Autonomic Computing*. ICAC '09, ACM, New York, NY, USA, 117–126.

Kumar, S., Talwar, V., Kumar, V., Ranganathan, P. and Schwan, K. (2009). "vManage: Loosely Coupled Platform and Virtualization Management in Data Centers."

Proceedings of the 6th International Conference on Autonomic Computing. ICAC '09, ACM, New York, NY, USA, 127–136.

Li, J., Shuang, K., Su, S., Huang, Q., Xu, P., Cheng, X. and Wang, J. (2012). "Reducing Operational Costs Through Consolidation with Resource Prediction in the Cloud." *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*. CCGRID '12, IEEE Computer Society, Washington, DC, USA, 793–798.

Li, J., Zhao, J., Li, Y., Cui, L., Li, B., Liu, L. and Panneerselvam, J. (2014). "iMIG: Toward an adaptive live migration method for KVM virtual machines." *The Computer Journal*, 58(6), 1227–1242.

Li, Y., Wen, Y., Guan, K. and Tao, D. (2017). "Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning." *CoRR*, abs/1709.05077.

Liu, B., Lin, Y. and Chen, Y. (2016). "Quantitative workload analysis and prediction using Google cluster traces." *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 935–940.

Liu, H. and He, B. (2015a). "VMbuddies: Coordinating Live Migration of Multi-Tier Applications in Cloud Environments." *IEEE Transactions on Parallel and Distributed Systems*, 26(4), 1192–1205.

Liu, H. and He, B. (2015b). "VMbuddies: Coordinating Live Migration of Multi-Tier Applications in Cloud Environments." *IEEE Transactions on Parallel and Distributed Systems*, 26(4), 1192–1205.

Liu, H., Xu, C.-Z., Jin, H., Gong, J. and Liao, X. (2011). "Performance and Energy Modeling for Live Migration of Virtual Machines." *Proceedings of the 20th International Symposium on High Performance Distributed Computing*. HPDC '11, ACM, New York, NY, USA, 171–182.

Liu, X., Zhu, X., Singhal, S. and Arlitt, M. (2005). "Adaptive entitlement control of resource containers on shared servers." *2005 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005. IM 2005.* 163–176.

Logreplayer (2011). "Logreplayer." `http://www.cs.viginia.edu/~rz5b/software/software.htm` (Feb. 25, 2015).

Ma, T., Chu, Y., Zhao, L. and Ankhbayar, O. (2014). "Resource Allocation and Scheduling in Cloud Computing: Policy and Algorithm." *IETE Technical Review*, 31(1), 4–16.

Mahdhi, T. and Mezni, H. (2018). "A prediction-Based VM consolidation approach in IaaS Cloud Data Centers." *Journal of Systems and Software*, 146, 263 – 285.

Mann, V., Gupta, A., Dutta, P., Vishnoi, A., Bhattacharya, P., Poddar, R. and Iyer, A. (2012). "Remedy: Network-aware Steady State VM Management for Data Centers." *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*. IFIP'12, Springer-Verlag, Berlin, Heidelberg, 190–204.

Mishra, M., Das, A., Kulkarni, P. and Sahoo, A. (2012). "Dynamic resource management using virtual machine migrations." *IEEE Communications Magazine*, 50(9), 34–40.

Morais, F. J. A., Brasileiro, F. V., Lopes, R. V., Santos, R. A., Satterfield, W. and Rosa, L. (2013). "Autoflex: Service Agnostic Auto-scaling Framework for IaaS Deployment Models." *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. 42–49.

Mukherjee, T., Banerjee, A., Varsamopoulos, G. and Gupta, S. K. S. (2010). "Model-driven Coordinated Management of Data Centers." *Comput. Netw.*, 54(16), 2869–2886.

N. Bennani, M. and A. Menasce, D. (2005). "Resource Allocation for Autonomic Data Centers Using Analytic Performance Models." *Proceedings of the Second International Conference on Automatic Computing*. ICAC '05, IEEE Computer Society, Washington, DC, USA, 229–240.

Nathan, S., Kulkarni, P. and Bellur, U. (2013). "Resource Availability Based Performance Benchmarking of Virtual Machine Migrations." *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ICPE '13, ACM, New York, NY, USA, 387–398.

Netperf (2015). "Networking Performance Benchmark." `http://www.netperf.org/` (June. 25, 2015).

Nguyen, H., Shen, Z., Gu, X., Subbiah, S. and Wilkes, J. (2013). "AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service." *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. USENIX, San Jose, CA, 69–82.

Nikravesh, A. Y., Ajila, S. A. and Lung, C.-H. (2015). "Towards an Autonomic Auto-scaling Prediction System for Cloud Resource Provisioning." *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '15, IEEE Press, Piscataway, NJ, USA, 35–45.

OLTP (2008). "OLTP Benchmark." `https://github.com/oltpbenchmark/oltpbench` (Jan. 5, 2016).

Padala, P., Hou, K.-Y., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S. and Merchant, A. (2009). "Automated Control of Multiple Virtualized Resources." *Proceedings of the 4th ACM European Conference on Computer Systems*. EuroSys '09, ACM, New York, NY, USA, 13–26.

Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A. and Salem, K. (2007). "Adaptive Control of Virtualized Resources in Utility Computing Environments." *SIGOPS Oper. Syst. Rev.*, 41(3), 289–302.

Parekh, S., Gandhi, N., Hellerstein, J., Tilbury, D., Jayram, T. and Bigus, J. (2002). "Using Control Theory to Achieve Service Level Objectives In Performance Management." *Real-Time Syst.*, 23(1/2), 127–141.

Pillai, P. S. and Rao, S. (2016). "Resource Allocation in Cloud Computing Using the Uncertainty Principle of Game Theory." *IEEE Systems Journal*, 10(2), 637–648.

Rackspace (2012). "Rackspace." `http://rackspace.com/.`

Reig, G. and Guitart, J. (2012). "On the Anticipation of Resource Demands to Fulfill the QoS of SaaS Web Applications." *2012 ACM/IEEE 13th International Conference on Grid Computing*. 147–154.

Rolia, J., Cherkasova, L., Arlitt, M. and Machiraju, V. (2006a). "Supporting Application Quality of Service in Shared Resource Pools." *Commun. ACM*, 49(3), 55–60.

Rolia, J., Cherkasova, L. and McCarthy, C. (2006b). "Configuring Workload Manager Control Parameters for Resource Pools." *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*. 127–137.

Roy, N., Dubey, A. and Gokhale, A. (2011). "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting." *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*. CLOUD '11, IEEE Computer Society, Washington, DC, USA, 500–507.

RUBiS (2012). "Rice University Bidding System." `http://rubis.ow2.org` (Dec. 15, 2014).

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1." chapter Learning Internal Representations by Error Propagation. MIT Press, Cambridge, MA, USA, 318–362.

Salomie, T.-I., Alonso, G., Roscoe, T. and Elphinstone, K. (2013). "Application Level Ballooning for Efficient Server Consolidation." *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys '13, ACM, New York, NY, USA, 337–350.

Sapankevych, N. I. and Sankar, R. (2009). "Time Series Prediction Using Support Vector Machines: A Survey." *Comp. Intell. Mag.*, 4(2), 24–38.

Sha, L., Liu, X., Lu, Y. and Abdelzaher, T. (2002). "Queueing model based network server performance control." *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.* 81–90.

Shen, Z., Subbiah, S., Gu, X. and Wilkes, J. (2011). "CloudScale: Elastic Resource Scaling for Multi-tenant Cloud Systems." *Proceedings of the 2Nd ACM Symposium on Cloud Computing*. SOCC '11, ACM, New York, NY, USA, 5:1–5:14.

Shivam, P., Babu, S. and Chase, J. (2006a). "Active and Accelerated Learning of Cost Models for Optimizing Scientific Applications." *Proceedings of the 32Nd International Conference on Very Large Data Bases*. VLDB '06, VLDB Endowment, 535–546.

Shivam, P., Babu, S. and Chase, J. S. (2006b). "Learning Application Models for Utility Resource Planning." *2006 IEEE International Conference on Autonomic Computing*. 255–264.

Shrivastava, V., Zerfos, P., w. Lee, K., Jamjoom, H., Liu, Y. H. and Banerjee, S. (2011). "Application-aware virtual machine migration in data centers." *2011 Proceedings IEEE INFOCOM*. 66–70.

Singh, R., Irwin, D., Shenoy, P. and Ramakrishnan, K. K. (2013). "Yank: Enabling Green Data Centers to Pull the Plug." *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*. nsdi'13, USENIX Association, Berkeley, CA, USA, 143–156.

Sladescu, M., Fekete, A., Lee, K. and Liu, A. (2012). "Event Aware Workload Prediction: A Study Using Auction Events." *Proceedings of the 13th International Conference on Web Information Systems Engineering*. WISE'12, Springer-Verlag, Berlin, Heidelberg, 368–381.

Sonnek, J., Greensky, J., Reutiman, R. and Chandra, A. (2010). "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration." *Proceedings of the 2010 39th International Conference on Parallel Processing*. ICPP '10, IEEE Computer Society, Washington, DC, USA, 228–237.

Stewart, C., Kelly, T., Zhang, A. and Shen, K. (2008). "A Dollar from 15 Cents: Cross-platform Management for Internet Services." *USENIX 2008 Annual Technical Conference*. ATC'08, USENIX Association, Berkeley, CA, USA, 199–212.

Sudevalayam, S. and Kulkarni, P. (2013). "Affinity-aware modeling of CPU usage with communicating virtual machines." *Journal of Systems and Software*, 86(10), 2627 – 2638.

Tang, H., Li, C., Bai, J., Tang, J. and Luo, Y. (2019). "Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloudâĂŞedge environment." *Computer Communications*, 134, 70 – 82.

Tesauro, G. (2005). "Online Resource Allocation Using Decompositional Reinforcement Learning." *Proc. of the Twentieth National Conference on Artificial IntelligenceProc. AAAI-05*. 9–13.

Tesauro, G., Jong, N. K., Das, R. and Bennani, M. N. (2007). "On the use of hybrid reinforcement learning for autonomic resource allocation." *Cluster Computing*, 10(3), 287–299.

TPC-W (2005). "TPC-W Data Set." `www.tpc.org/tpcw/` (Dec. 5, 2016).

Tseng, F., Wang, X., Chou, L., Chao, H. and Leung, V. C. M. (2018). "Dynamic Resource Prediction and Allocation for Cloud Data Center Using the Multiobjective Genetic Algorithm." *IEEE Systems Journal*, 12(2), 1688–1699.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M. and Tantawi, A. (2005a). "An Analytical Model for Multi-tier Internet Services and Its Applications." *SIGMETRICS Perform. Eval. Rev.*, 33(1), 291–302.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M. and Tantawi, A. (2005b). "An Analytical Model for Multi-tier Internet Services and Its Applications." *SIGMETRICS Perform. Eval. Rev.*, 33(1), 291–302.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P. and Wood, T. (2008). "Agile Dynamic Provisioning of Multi-tier Internet Applications." *ACM Trans. Auton. Adapt. Syst.*, 3(1), 1:1–1:39.

Urgaonkar, B., Shenoy, P. and Roscoe, T. (2002). "Resource Overbooking and Application Profiling in Shared Hosting Platforms." *SIGOPS Oper. Syst. Rev.*, 36(SI), 239–254.

Vaquero, L. M., Rodero-Merino, L., Caceres, J. and Lindner, M. (2008). "A Break in the Clouds: Towards a Cloud Definition." *SIGCOMM Comput. Commun. Rev.*, 39(1), 50–55.

Varasteh, A. and Goudarzi, M. (2017). "Server Consolidation Techniques in Virtualized Data Centers: A Survey." *IEEE Systems Journal*, 11(2), 772–783.

Wikimedia (2013). "Wikimedia." `http://dumps.wikimedia.org/other/pagecounts-raw` (June. 14, 2015).

Wilkes, J. (2011). "Google Cluster Trace." `http://code.google.com/p/googleclusterdata` (Nov. 15, 2015).

Williams, D., Jamjoom, H., Liu, Y.-H. and Weatherspoon, H. (2011). "Overdriver: Handling Memory Overload in an Oversubscribed Cloud." *SIGPLAN Not.*, 46(7), 205–216.

Williams, D., Jamjoom, H. and Weatherspoon, H. (2012). "The Xen-Blanket: Virtualize Once, Run Everywhere." *Proceedings of the 7th ACM European Conference on Computer Systems*. EuroSys '12, ACM, New York, NY, USA, 113–126.

Wood, T., Cherkasova, L., Ozonat, K. and Shenoy, P. (2008). "Profiling and Modeling Resource Usage of Virtualized Applications." *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Middleware '08, Springer-Verlag New York, Inc., New York, NY, USA, 366–387.

Wood, T., Ramakrishnan, K. K., Shenoy, P., Van Der Merwe, J., Hwang, J., Liu, G. and Chaufournier, L. (2015). "CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines." *IEEE/ACM Trans. Netw.*, 23(5), 1568–1583.

Wood, T., Shenoy, P., Venkataramani, A. and Yousif, M. (2007). "Black-box and Gray-box Strategies for Virtual Machine Migration." *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*. NSDI'07, USENIX Association, Berkeley, CA, USA, 17–17.

Wu, Y. and Zhao, M. (2011). "Performance Modeling of Virtual Machine Live Migration." *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*. CLOUD '11, IEEE Computer Society, Washington, DC, USA, 492–499.

Xu, F., Liu, F., Liu, L., Jin, H., Li, B. and Li, B. (2014). "iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud." *IEEE Trans. Comput.*, 63(12), 3012–3025.

Xu, J., Zhao, M., Fortes, J., Carpenter, R. and Yousif, M. (2008a). "Autonomic Resource Management in Virtualized Data Centers Using Fuzzy Logic-based Approaches." *Cluster Computing*, 11(3), 213–227.

Xu, J., Zhao, M., Fortes, J., Carpenter, R. and Yousif, M. (2008b). "Autonomic Resource Management in Virtualized Data Centers Using Fuzzy Logic-based Approaches." *Cluster Computing*, 11(3), 213–227.

Xu, L., Wang, J., Nallanathan, A. and Li, Y. (2016). "Resource Allocation Based on Double Auction for Cloud Computing System." *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. 1538–1543.

Xu, W., Zhu, X., Singhal, S. and Wang, Z. (2006). "Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers." *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*. 115–126.

Yang, B., Li, Z., Chen, S., Wang, T. and Li, K. (2016). "Stackelberg Game Approach for Energy-Aware Resource Allocation in Data Centers." *IEEE Transactions on Parallel and Distributed Systems*, 27(12), 3646–3658.

Yang, J., Liu, C., Shang, Y., Mao, Z. and Chen, J. (2013). "Workload Predicting-Based Automatic Scaling in Service Clouds." *2013 IEEE Sixth International Conference on Cloud Computing*. 810–815.

Zhang, J., Ren, F. and Lin, C. (2014). "Delay guaranteed live migration of Virtual Machines." *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 574–582.

Zheng, J., Ng, T. S. E., Sripanidkulchai, K. and Liu, Z. (2013). "Pacer: A Progress Management System for Live Virtual Machine Migration in Cloud Computing." *IEEE Transactions on Network and Service Management*, 10(4), 369–382.

Zheng, W., Bianchini, R., Janakiraman, G. J., Santos, J. R. and Turner, Y. (2009). "JustRunIt: Experiment-based Management of Virtualized Data Centers." *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*. USENIX'09, USENIX Association, Berkeley, CA, USA, 18–18.

Zhu, X., Young, D., Watson, B. J., Wang, Z., Rolia, J., Singhal, S., McKee, B., Hyser, C., Gmach, D., Gardner, R., Christian, T. and Cherkasova, L. (2008). "1000 Islands: Integrated Capacity and Workload Management for the Next Generation Data Center." *2008 International Conference on Autonomic Computing*. 172–181.

List of Publications

Journal Publications

- **Bane Raman, R.** and Annappa, B. (2015). "Improving Resource Utilization in Datacenters by Accurately Triggering Live VM Migration", *International Journal of Information Processing*, Vol. 09 Issue 04, 1-14.

- **Bane Raman, R.** and Annappa, B. (2017). "Autonomic Resource Management Framework for Virtualized Environments", *International Journal of Internet Technology and Secured Transactions*, DOI: 10.1504/IJITST.2018.10011187. Appeared as forthcoming article.
  http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=IJITST

Conference Publications

- **Bane Raman, R.** and Annappa, B. (2014). "Survey of Dyamic Resource Management Approaches in Virtualized Data Centers." *Proc., IEEE International Conference on Computational Intelligence and Computing Research (ICCIC 2014)*, IEEE , Madurai, India, pp.26-31.

- **Bane Raman, R.** and Annappa, B. (2015). "Virtual Machine Migration triggering using Application Workload Prediction in Data Centers". *Proc., 11^{th} International Conference on Communication and Networks (ICCN-2015)*, Elsevier Procedia - Computer Science, Vol .13, 167-176.

- **Bane Raman, R.** and Annappa, B. (2017). "Prediction based Dynamic Resource Provisioning in Virtualized Environments". *Proc., 35^{th} International Conference on Consumer Electronics (ICCE-2017) held at LAS VEGAS, USA, during 09-11 January, 2017*, IEEE, LAS VEGAS, USA, pp. 100-105.

- **Bane Raman, R.** and Annappa, B. (2017). "Dynamic Resource Allocation Using Fuzzy Prediction System". *IEEE International Conference for Convergence of Technology (I2CT) , Pune. 6-8 April, 2018*, IEEEXplore, USA.

Brief Bio-Data

Bane Raman Raghunath

Research Scholar

Department of Computer Science and Engineering

National Institute of Technology Karnataka, Surathkal

P.O. Srinivasnagar

Mangalore - 575025

Phone: +91 9422632740

Email: ramanbane@gmail.com

Permanent Address

Bane Raman Raghunath

S/o Bane Raghunath Arjun

Bagwe-wadi

Kasal - 416603

Kudal (Tq.), Sindhudurg (Dist.)

Maharashtra, INDIA

Qualification

M. Tech. in Computer Science and Engineering, Walchand College of Engineering, Sangli, Maharashtra, 2005.

B. E. Computer Science and Engineering, Shivaji University, Kolhapur, Maharashtra, 1998.