Performance Modeling and Optimization of Resource Allocation in Cloud Computing Systems

Shahin Vakilinia

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

July 2015

CONCORDIA UNIVERSITY SCHOOL
OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Shahin Vakilinia

Entitled: Performance Modeling and Optimization of Resource Allocation in Cloud Computing Systems

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. A. Awasthi

_____ External Examiner

Dr. Z. Dziong

_____ External to Program

Dr. C.Assi

_____ Examiner

Dr. W-P.Zhu

_____ Examiner

Dr. Y. Liu

_____ Thesis Co-Supervisor

Dr. M. Mehmet Ali

_____ Thesis Co-Supervisor

Dr. D. Qiu

Approved by: _____

Dr. A.R. Sebak, Graduate Program Director

September 8, 2015 Dr. A. Asif, Dean

Faculty of Engineering & Computer Science

ABSTRACT

Performance Modeling and Optimization of Resource Allocation in Cloud Computing Systems

Shahin Vakilinia, Ph.D.
Concordia University, 2015

Cloud computing offers on-demand network access to the computing resources through virtualization. This paradigm shifts the computer resources to the cloud, which results in cost savings as the users leasing instead of owning these resources. Clouds will also provide power constrained mobile users accessibility to the computing resources. In this thesis, we develop performance models of these systems and optimization of their resource allocation.

In the performance modeling, we assume that jobs arrive to the system according to a Poisson process and they may have quite general service time distributions. Each job may consist of multiple number of tasks with each task requiring a virtual machine (VM) for its execution. The size of a job is determined by the number of its tasks, which may be a constant or a variable. In the case of constant job size, we allow different classes of jobs, with each class being determined through their arrival and service rates and number of tasks in a job. In the variable case a job generates randomly new tasks during its service time. The latter requires dynamic assignment of VMs to a job, which will be needed in providing service to mobile users. We model the systems with both constant and variable size jobs using birth-death processes. In the case of constant job size, we determined joint probability distribution of the number of jobs from each class in the system, job blocking probabilities and distribution of the utilization of resources for systems with both homogeneous and heterogeneous types of VMs. We have also analyzed tradeoffs for turning idle servers off for power saving. In the case of variable job sizes, we have determined distribution of the number of jobs in the system and average service time of a job for systems with both infinite and finite amount of resources. We have presented

numerical results and any approximations are verified by simulation. The performance results may be used in the dimensioning of cloud computing centers.

Next, we have developed an optimization model that determines the job schedule, which minimizes the total power consumption of a cloud computing center. It is assumed that power consumption in a computing center is due to communications and server activities. We have assumed a distributed model, where a job may be assigned VMs on different servers, referred to as fragmented service. In this model, communications among the VMs of a job on different servers is proportional to the product of the number of VMs assigned to the job on each pair of servers which results in a quadratic network power consumption in number of job fragments. Then, we have applied integer quadratic programming and the column generation method to solve the optimization problem for large scale systems in conjunction with two different algorithms to reduce the complexity and the amount of time needed to obtain the solution. In the second phase of this work, we have formulated this optimization problem as a function of discrete-time. At each discrete-time, the job load of the system consists of new arriving jobs during the present slot and unfinished jobs from the previous slots. We have developed a technique to solve this optimization problem with full, partial and no migration of the old jobs in the system. Numerical results show that this optimization results in significant operating costs savings in the cloud computing systems.

To

My parents, my Wife and my Grandparents

Acknowledgements

I would like to extend my utmost gratitude to Profs. Mustafa Mehmet Ali and Dongyu Qiu for their patience, for being generous with their time and for countless great ideas and advices that helped me overcome the difficulties of research. The completion of this thesis would not have been possible without their significant support. I am truly indebted to them for their insightful comments and uncompromising attention to details.

I would like to express my deep gratitude to my dear wife, Samaneh Mansouri, for her tolerance of my lifestyle which has also become hers. I am especially grateful to Ehsan Farahani, Dariush Ebrahimi and Behdad Heidarpour for their support and for being great friends. Last but not least, I owe more than thanks to my family members, which include my parents, my brother (Iman Vakilinia) and my grandparents. Without them, I would not be able to succeed throughout my life. I am always grateful to them for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BoT	Bag of tasks
CC	Cloud Computing
CG	Column Generation
CM	Centralized Model
CS	Core Switch
DC	Datacenter
DM	Distributed Model
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
IQP	Integer Quadratic Programming
HOL	Head of Line
LHS	Left hand side
LP	Linear programming
MCC	Mobile Cloud Computing
RMP	Restricted Master Problem
OPL	Optimization programming language
PaaS	Platform as a Service
PP	Pricing problem
PoD	Performance Optimized Modular Data Centers
QoS	Quality of Service
RHS	Right Hand Side
SaaS	Software as a service
SBP	Stochastic Bin Packing
SLA	Service Level Agreement
SI	Spot Instance
SRN	stochastic reward network

TOR	Top-Of-Rack
VM	Virtual Machine

GLOSSARY

Symbol	Definition
R	Number of classes of jobs
λ_r	Arrival rate of class *r* jobs
λ_T	Total job arrival rate
μ_r	Service rate of class *r* jobs
k_T	Total number of busy VMs
b_r	Number of VMs required by a class *r* job
n_r	Number of class *r* jobs in the system
λ_{rn}	Arrival rate of class *r* jobs to the *n*'th server pool.
λ_{Tn}	Total arrival rate of the jobs to the *n*'th server pool.
PB_{rn}	Probability that a class *r* job will be blocked by the *n*'th server pool.
PB_n	Overall job blocking probability at the *n*'th server pool.
$q_n(j)$	Probability of having j VMs will be busy in the n^{th} server pools.
g	Number of active server pools.
λ_{rn}	Arrival rate of class *r* jobs to the *n*'th server pool.
λ_{Tn}	Total arrival rate of the jobs to the *n*'th server pool.
PB_{rn}	Probability that a class *r* job will be blocked by the *n*'th server pool.
PB_n	Overall job blocking probability at the *n*'th server pool.
F	Number of resource types.
C_f	Number of units of resource *f*, $f = 1..F$.
$b_{\ell f}$	Number of units of resource *f* required by a type ℓ VM
$\lambda_{j\ell}$,	Arrival rate of class $j\ell$ jobs that require j number of type ℓ VMs,
$\mu_{j\ell}$	Service rate of class $j\ell$ jobs.
$n_{j\ell}$	Number of class $j\ell$ jobs in the system.
N	Number of jobs in the datacenter
T	Number of different types of servers.
M_t	Total number of type *t* servers

Q_t	Power usage of type *t* servers
$v_{n_h}^r$	Number of type *r* VMs required by job n_h
c_t^k	Type *k* resource capacity of a type *t* server
P_{n_h}	Power usage rate of communicating with a server serving VMs of job n_h.
i_r^k	Amount of type *k* resource required by a type *r* VM
$x_{r,n_h}^{m_t}$	Number of type *r* VMs in m^{th} type t server assigned to job n_h.
$\tilde{x}_{n_h}^{m_t}$	Number of VMs in m^{th} type t server assigned to serve job n_h.
y_{m_t}	Binary variable that denote *on* or *off* status of m^{th} type *t* server.
J_t	Total number of patterns of type *t* servers
m_{j_t}	Number of active type *t* servers with pattern j_t
$x_{r,n_h}^{j_t}$	Number of type *r* VMs assigned to job n_h over type *t* server with pattern j_t
$\tilde{J}_t$	Pattern $\tilde{j}_t$ of type *t* servers introduced by pricing problem *t*.
$\tilde{x}_{n_h}^{j_t}$	Number of VMs assigned to job n_h over a type *t* server with pattern j_t.

Chapter 1

Introduction

1.1 Introduction to Cloud Computing

Reduced costs of processing and storage technologies brought about the rapid growth of the computer industry. Recently, a new computing paradigm called cloud computing has emerged which provides on-demand network access to the computing resources through virtualization.

There have been many definitions of cloud computing, but one of the most referred-to definitions of it was published by the National Institute of Standards and Technology (NIST), which is given below,

"*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"

The cloud computing paradigm offers cost savings because users lease the computing resources from a service provider when needed instead of owning them. Cloud computing enables dynamic sharing of the computing resources among the users and allows them to submit and execute applications of many kinds. Different applications, however, require distinct types of resources. Applications such as interactive databases or web-based services lease the cloud resources and usually occupy various resources to maintain a high quality of service (QoS) level for their users. A service level agreement

(SLA) specifies the QoS to be provided to the user in terms of various performance parameters such as throughput, reliability, blocking probability and response time.

The enabling technology of cloud computing is virtualization. Virtualization is a technique for separating software applications from each other and the hardware resources. Virtual Machine (VM) also called instance refers to running of an individual copy of a particular user's application software or operating system in a virtual environment. Virtual systems feature multitenant capabilities through hypervisor. Hypervisor (also called a virtual machine monitor) is the virtualization software platform that allows simultaneous running of multiple instances on a server. Housing multiple virtual machines on a physical server utilize the physical server resources more efficiently. In a non-virtualized environment, most of the time servers are idle and it has been found that less than 10 percent of computing resources are used at any one time. These servers accrue maintenance and human resource costs along with the costs of the energy required to power and cool the hardware. After virtualization, server utilization is found to rise as high as 80 percent. Therefore, virtualization is one of the operative ways to mitigate datacenter (DC) expenses. Moreover, virtualized datacenters accommodate pools of resources rather than isolated servers. The goal is to pool resources and serve demands from these resource pools.

1.2 Cloud Computing Services

In this section, we describe the services provided by cloud computing systems. Cloud computing services may be classified into three types as Infrastructure-as-a-service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). IaaS refers to providing hardware equipment such as CPU, memory and storage as a service, PaaS refers to providing platforms such as software development frameworks, operating systems or multi-tenant application supports as a service and SaaS provides software and applications as a service. The research interest of this thesis is IaaS, which is described below.

Generally, the topology of a cloud computing center is hierarchical with racks containing a fixed number of blade servers. A blade server contains a number of processors each one consisting of several processing cores. The processing cores, memory and storage space are configured into VMs. IaaS is deployed in a cloud provider's datacenter (DC) in the form of VMs. The user of the IaaS acquires a resource and is charged for that resource based on the amount of resource used and the duration of that usage. IaaS allows access and management of systems from anywhere and thus helps organization in extending or shrinking their IT infrastructure. In IaaS, back-end hardware is administered and owned by the cloud service provider. Mobility, stability, agility, availability and elasticity in IaaS, is achieved by providing a user interface for the management of a number of resources. It enables the users to allocate a subset of the resources for their own use. Moreover, the interface provides the required functionality for operations, such as starting and stopping operating system instances.

Figure 1.1 simply depicts cloud computing platforms which provide IaaS, in the form of VMs. Cloud users request for VMs are specified in terms of resources such as CPU, memory and storage space.

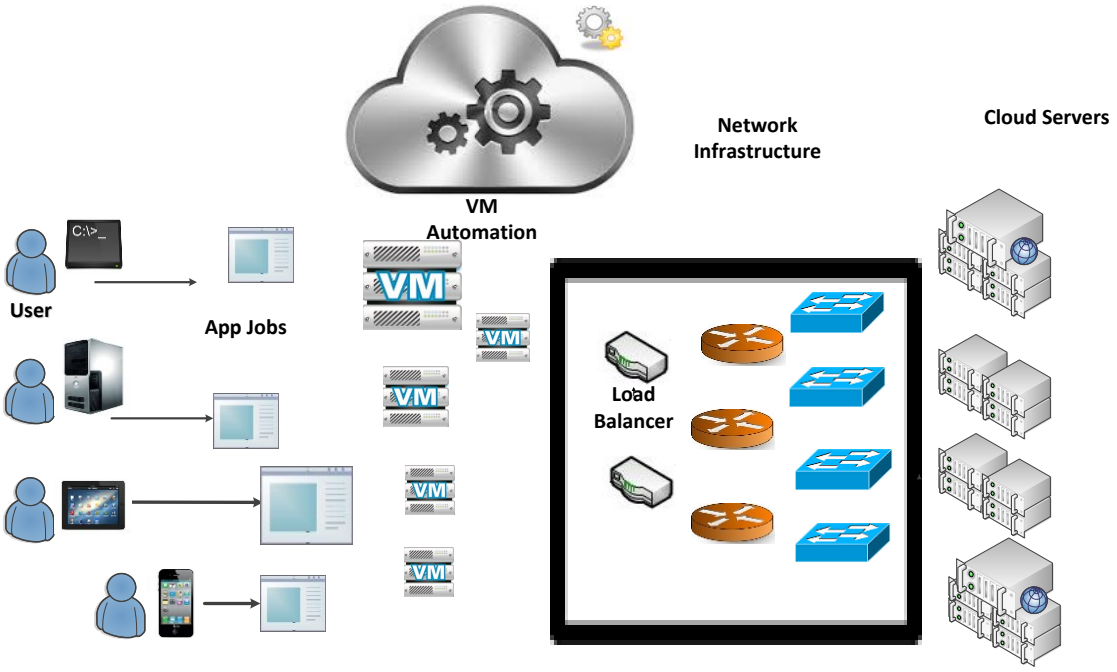

Fig 1.1 Dynamic resource allocation in a cloud computing center

1.3 Future of Cloud Computing

Cloud computing has been widely studied and applied in various fields as it can provide elastic computation platform and on-demand use and efficient sharing of resources. Further, clouds will provide mobile users access to computing resources, which is referred to as mobile cloud computing. This is very important as mobile devices are becoming the primary computing platform to many users and they have limited processing power and battery life. Thus many vendors and industry observers expect the expenditures on cloud services to increase dramatically. The following graph gives an insight into the investment growth in different cloud computing services over time. Gartner estimates that the global market size for cloud computing services will be around $210 billion in 2016. Gartner also expects annual growth of 41.3% for cloud computing services through 2016.

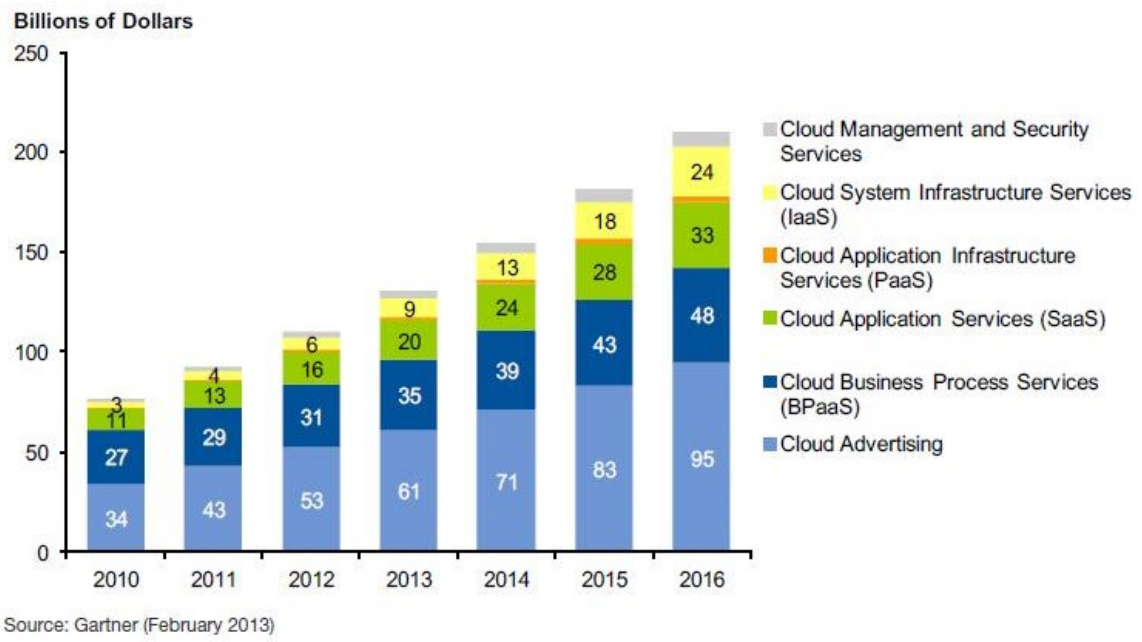

Figure 1.2: Public Cloud Service Market by segment, 2010-2016 provided by Gartner's "Worldwide IT Spending Forecast"

1.4 Motivations

Next, we will describe the motivation for the research in this thesis. We will discuss resource allocation models and associated algorithms of cloud Data-Centers (DCs) workloads with respect to power usage, and the QoS factor of service providers.

1.4.1 Modeling of the Resource Allocation in Cloud Computing Datacenters

One of the prime objectives of modern computing is to implement parallel computations on large distributed resources such as DCs. On demand allocation of resources in a cloud DC improves performance and reduces the deployment overheads significantly. However, efficient resource allocation requires accurate modeling of DCs. This is why more accurate modeling of jobs and resources that reflect real world scenarios, has attracted wide attention recently. Although the cloud computing technology is emerging

and growing rapidly, due to its complexity it suffers from a lack of standards [2], detailed models and optimal resource allocation algorithms.

The main issue that makes cloud resource allocation complex is heterogeneity of resources and workloads. Based on the types of applications served by the cloud computing center, there is a vast diversity in resource demand profiles which make the incoming workflow heterogonous. Besides heterogeneity of workloads, one of the essential characteristics of a cloud computing system is heterogeneity of server resources. As time goes by, DCs update their resources' configuration, processing capabilities, memory and storage spaces. They also introduce new platforms based on the new high performance servers along with older hardware, which makes the cloud platform heterogeneous.

Cloud workloads often have very large range of resource requirements, arrival rate and execution time. Jobs entering the cloud may demand different types of services. In the interactive applications, the users occupy cloud resources for a long period of time, whereas batch jobs require much shorter execution time. Hence, the execution time of the applications is notably divergent. In general, computing jobs such as web serving are more processing intensive, while database operations typically require high-memory support. Most of the jobs require parallel data analysis. This is the main reason for the recent development of MapReduce Programming model [3]. This model relies on parallel processing with a sequential functional approach. Job fragments are executed in parallel to speed up processing of the jobs. MapReduce has usually three phases: fan out, map and reduce. Applications such as Apache Hadoop [4] and platforms such as Pig [5] implement the MapReduce programming model. This model also applies to bag-of-tasks (BoTs) where a job consists of parallel and sequential tasks. The number of task executions in the mapping phase will be larger than the fan out and reduce phases. Hence, a job's load depends on the phase of execution.

As a result, high variety in resource and workload characteristics makes resource allocation in cloud computing centers very challenging.

1.4.2 Energy Efficient Resource Allocation in Cloud Computing Datacenters

Present day servers consume more power than those of a decade ago. Researchers believe that high performance servers impose more energy costs to the system. With the growing number of in-service servers, the worldwide expenditure on enterprise power usage and server cooling is estimated to be quite high. (Currently, server farms have been identified as one of the major electricity consumers in the world) [6]. Optimal resource allocation methods in datacenters can save up to 20% of the energy consumption. These savings may lead to an additional 30% saving in cooling energy requirements [6]. Given the rising cost of energy, cloud providers are presently looking for state-of-the-art solutions to ensure optimality of their power consumption. Service of a job at a datacenter results in power consumption because of the server processing and network communication demands of this processing.

Hence, one option can be VM placement in a way that minimizes the number of active servers and communication traffic among VMs at the same time in favor of cloud computing DCs. Dynamic power management aims to reduce power consumption in DCs by temporarily shutting down servers when they are not required. However, full reactivation of a server is delayed by setup time, which can adversely affect the system performance. Hence, in order to be able to manage the number of active servers dynamically, the amount of incoming workload needs to be determined and assign VMs to new arriving jobs. This VM assignment process includes all server resources; namely CPU, memory and storage; which leads to a “multidimensional bin packing problem”.

Beside power usage of servers, communication also impacts both performance and cost of the operations. Communication increases job execution latency and power consumption. One way to mitigate the Cloud Network (CN) power usage is to apply traffic aware VM placement methods [7], [8]. Thus, the optimal solution of VM placement problem would include both CN and servers power consumption.

1.5 Contributions

Next, the main contributions of this research are summarized below. The first part of the contributions is due to performance modeling of cloud computing systems, while the second part is due to the proposed optimal allocation of resources.

- We modeled a cloud computing system with multiple classes of jobs with constant sizes and with homogeneous VMs. Assuming Poisson arrival of jobs with arbitrary service distributions, we have determined the joint distribution of the number of jobs from each class in the system, job blocking probabilities of each class and the distribution of the utilization of resources under a single server, multiple-servers and multiple-server pool cases. In multiple-server case, we have determined fragmentation probability of a job's service among multiple servers. We show the applicability of our results to study a power management algorithm that reduces the power consumption while maintaining a plausible job blocking probability under time-varying traffic load. We have also derived job blocking probabilities and distribution of the utilization of resources with multiple classes of jobs with heterogeneous VMs. Probability distribution of the service time and average number of jobs for a system with constant job sizes and independent task completion times are determined. We have considered a system with jobs arriving to the system according to a Poisson process with variable job sizes in number of tasks. It is assumed that a job will generate new tasks randomly during its service time in the system. We have derived service time distribution of a job, distribution of the number of jobs and total number of tasks in the system.
- We developed an optimization problem that determines job scheduling such that the total power consumption of the cloud computing center is minimized. Then, the optimization problem using integer quadratic programming (IQP) and column generation (CG) are introduced. First, the optimization problem has been modeled as an IQP, then, we have applied the CG method to solve large scale optimization problem in conjunction with two different algorithms to decrease the complexity and the time to obtain a solution to the model, both

pertaining to the performance of the platform. Then, we have shown how to formulate and solve optimization problem of the system in discrete-time as the time evolves. At each discrete-time, the job load of the system consists of new arriving jobs during the present slot and unfinished jobs from the previous slots. We solve this optimization problem with full, partial and no migration of the old jobs in the system.

1.6 Thesis Organization

Next, we give the structure of the thesis. In chapter 2, a comprehensive literature survey of modeling and optimization of the dynamic resource allocation in cloud DCs is presented. Chapter 3 presents performance analyses of various cloud computing models. The systems with both constant and variable job sizes with both homogenous and heterogeneous types of VMs have been studied. Chapter 4 proposes an optimization platform for VM placement that minimizes the total power consumption of DC. The proposed optimization problem has been solved using the CG technique for both the ILP and IQP versions of the problem. The initialization patterns and heuristic termination approach for the CG technique, which reduces the complexity and time of obtaining a solution have been presented. Subsequently, an optimization problem for VM placement while there are still unfinished jobs from previous timeslots has been formulated and solved. Finally, concluding remarks are presented in chapter 5 with potential future work.

Chapter 2

Literature Review

In this chapter, we provide a survey of the modeling of cloud computing systems. Survey of the previous work has been divided into several groups depending on the objective of the works.

2.1 Performance Modeling of Cloud Computing Systems

The research work in this group studies performance of a DC under stochastic job arrival processes and service time distributions. The objective of these studies is to determine equilibrium distribution of the number of jobs in the system, job blocking probabilities and response times, and distribution of utilization of resources.

As explained in the previous chapter, when a job arrives at the system, VM are created and assigned to cloud computing resources to handle the job execution. This assignment has been referred to as VM placement[1] in the literature. The problem of efficient assignment of VMs to PMs (servers) in cloud computing centers is considered as a stochastic problem [9]. Knapsack [9] and stochastic bin packing (SBP) [10] problems are the typical stochastic approaches used to allocate resources in the context of cloud computing systems. In such problems, multiple input flows of several types of jobs are considered, with the mean service time depending on their type. There is a finite number of servers in which SBP server packing algorithm takes into account. Each new arriving

[1] Selecting the most appropriate server for the virtual machine is known as virtual machine placement

job is served immediately by being placed into the servers according to the resource sharing policy. Service times of jobs are independent of each other.

In [11], a cloud computing center has been modeled as a *M/G/m/m+r* queue, where *r* is the buffer size. New arriving jobs to a full buffer are lost. The steady-state distribution of the queue length is determined by writing down the transition probability matrix of the embedded Markov chain at the job arrival points and solving the equilibrium equations. This analysis makes the approximation that at most three jobs may be served during an inter-arrival time and the queue length distribution does not have a closed form. In [12] and [13] the work in [11] has been extended to a more complicated cloud computing center model. It has been assumed that the cloud computing center has a number of servers and each server has been configured as a number of VMs. A server may be in a hot (with running VMs), warm (turned on but without running VMs) or cold state (turned off). The amount of time it takes to launch a job on a server depends on the state of the server, a hot server requires the shortest amount of time. The system attempts to serve a job depending on availability in the following order on a hot, warm and a cold server. It is also assumed that each job contains a number of tasks chosen from a discrete probability distribution. The tasks may have general service time distributions and service of each task requires a VM. A server accepts a job if it has enough available idle instances of resources to start serving all the tasks of a job simultaneously. If a job cannot be accepted on any of the servers, then it will be blocked. The steady-state distribution of the queue length is determined by writing down the transition probability matrix of the embedded Markov chain at the arrival points and solving the equilibrium equations through fixed point iteration among the server states. In [14], performance of cloud computing systems has been studied using stochastic reward networks (SRNs) which are an extension of generalized stochastic Petri Nets (GSPNs). In [15], performance of cloud computing systems with fault recovery has been considered.

In [16], cloud computing capacity has been studied under time-varying traffic load using historical traces with the assumption that idle capacity is turned off through simulation. The time-axis has been divided into 5-minute slots. Various moving average and autoregressive models have been used to predict the job demand for the next slot

using the demands information for the present and past slots. Then, the required server capacity for the predicted load is determined using Erlang loss formula. As a result, extra capacity may be added or subtracted to/from the presently active capacity respectively. It is assumed that it takes one slot to turn on the extra capacity. The unneeded capacity is turned off after one slot to prevent unnecessary on-off turning of the servers. It is assumed that an arriving job will be blocked and lost if there is no available active capacity to serve it. Under this scheduling algorithm, the paper determined job blocking probabilities and unutilized server capacity for prediction models as well as for a model that maintains a fixed reserved capacity using simulation. It has been determined that fixed reserve capacity provides better performance than the prediction models.

In [17], throughput optimal load balancing models have been considered in systems that include clusters of servers. The work assumes heterogeneous types of VM configurations. The time-axis is slotted and in each slot, a number of job requests arrive to the system. Each job may request a single VM for a number of slots. When the system is busy the arriving jobs are stored in a central queue for each type of jobs. It is shown that server-by-server max-weight job scheduling with preemption and server reconfiguration in each slot is throughput optimal. A non-preemptive algorithm, which is nearly optimal, is also proposed. To reduce the communication overhead, a more distributed system is also considered where each server maintains its own queues. The paper also presents simulation results, which show that the mean delay performance of centralized and distributed queuing systems is not very different. The paper does not take into consideration QoS requirements of different types of jobs which may not be met in this process.

Recently, Amazon introduced a new cloud computing service that sells the idle instances of resources called Spot Instance (SI) through competitive bidding. The price of SI depends on the demand but, in general, it is lower because no reliability is provided for the services. In [18], a statistical modeling of the SI prices and inter-price durations have been provided through curve-fitting to the experimental data available from Amazon.

2.2 Heterogeneity of Cloud Computing Resources

The research work in this group studies impacts of the resource heterogeneity on the performance of cloud computing center. As mentioned earlier, due to inevitable platform upgrades or enhanced hardware resources, cloud platforms gradually become heterogeneous over time which makes the VM placement problem more complex.

In [19], the impact of hardware heterogeneity on the performance of public clouds has been investigated. Amazon EC2 and Rackspace cloud platforms providing IaaS and experienced several generations of hardware upgrades were selected to represent the hardware diversity. During a two-year period, the activities of DCs in US are measured to collect some useful benchmarks that might affect the dynamic resource allocation in cloud DCs. Then, these benchmarks such as CPU performance and network overhead of cloud communication are utilized to evaluate the impact of heterogeneity on the performance of heterogeneous cloud computing centers. For instance, CPUBench and UnixBench [20] are used to analyze CPU performance metrics such as processing time and CPU running and idle percentages for Amazon EC2 and Rackspace instances. TCPBench also is used to measure the networking of instances. It indicates that for some EC2 m1.small instances can acquire approximately 40% of CPU processing time, while m1.large instances can acquire 75 % of CPU time. For Rackspace, 4-GB instance type, one-process can acquire close to 100% CPU acquisition percentage; while for dual-core type, CPU acquisition percentage for each process varies between 95 and 99 percent, which is related to the administration overhead. Therefore, the task scheduling mechanism of hypervisors also has an effect on benchmarks. Moreover, the team uses a "trial-and-better" approach that has three steps for the incoming jobs into the system. First, arriving job should apply for certain number of instances from the cloud, then, the performance levels of the acquired instances will be checked and finally the better performing instances are kept and the other ones are discarded. Finally, based on the benchmarks and "trial-and-better" approach in which arriving jobs seek out better-performing instances, game theoretic analysis, and Nash equilibrium are discussed from the cloud user perspective. Then, the heuristic cost-saving optimization algorithm is proposed. Their results show that their proposed algorithm can achieve up to 30 percent costs saving while instances performance is satisfactory.

In [21] and [22], heterogeneous VMs are considered when jobs require different

amount of resources during their service times. After the completion of a service, the job releases all the resources that were allocated to it and leaves the system. However, it is assumed that each job has only a single task that requires one VM for its execution. The system is modeled in order to propose an optimum VM placement algorithm which minimizes the communication latencies of VMs. Due to elasticity characteristics, cloud computing centers need to provide VMs with various types of resources which may be specified in terms of its requirements for different resources. In [23], heterogeneity of workloads and PMs are also addressed. According to their characteristics, tasks are classified into classes with similar resource demands and performance characteristics. Different types of servers are also considered based on their platform ID and capacities on different resources. A time series-based estimator has been implemented to predict workload arrival rate. Then, a heterogeneity aware resource monitoring and management system dubbed "Harmony" was proposed to perform dynamic capacity provisioning to minimize the total energy consumption and scheduling delay considering heterogeneity as well as reconfiguration costs.

2.3 Optimal Resource Allocation in Cloud Computing Datacenters

Next, we describe the work on the optimal allocation of resources in cloud computing centers. The objective of optimization may be different performance metrics such as throughput, communication latency and power consumption. This type of work provides scheduling of the jobs that minimizes the chosen performance metric.

In cloud computing centers, communication latencies of applications may affect performance when the VMs for an application are split over multiple servers [7], [8] and [21]. Due to communication latencies, response time may exceed the QoS requirements of the users. So the usage of optimum resource allocation algorithms is critical to achieve optimum application performance in cloud systems [21]. A structural constraint-aware VM placement algorithm has been proposed in [24] where the objectives are the reduction of communication latencies and improving system availability. With awareness of availability and communication requirements, [25]

formulated VM placement as an optimization problem with DCs characteristics and applications as its inputs. [25] also minimizes the intra-datacenter traffic by proposing application-aware VM migration algorithm. [10], [26], [27] and [28] consider optimization of a cloud computing center with respect to communication bandwidth demands. The sum of the bandwidth requirements of VMs on a server may exceed the capacity of a server's network interface. Since bandwidth demands on the VMs are stochastic, statistical multiplexing may be used to place VMs on a minimum number of servers such that VMs bandwidth guarantees may be met probabilistically. This problem may be modeled as an SBP problem. Under the assumption that a VM's bandwidth consumption is normally distributed, [10] presents approximate online and offline algorithms for the optimal assignment of VMs to the servers.

On the other hand, many studies have been done on traffic aware VM placement. The VM placement problem taking into account network communication and server (PM) operation costs, is investigated in [22], then, an algorithm is proposed in order to minimize the network cost with fixed PM-cost. However, there is a trade-off between PM and network operation cost. Network cost is minimized when each job is assigned to a single server while PM cost is minimized when jobs are packed to the servers. This may result in job fragmentation among the servers which increase the network cost. In [21], [27], [28] and [29], the traffic between VMs are assumed to be known or fixed for the placement period, and the placement is proposed based on this assumption. In [30], the authors study the VM placement problem with the product traffic pattern in DCs. The product traffic pattern is defined as a product of the VMs communication activities in which the probability that a request belongs to a VM is defined as its activity. [30] proposed an optimal solution to minimize the network cost for a homogenous scenario by demonstrating that the more active VMs has to be placed on the PMs with higher capacity.

Due to high variety of the cloud network traffic, traffic awareness is impossible in practice. Therefore [22] and [30] relates network cost to the number of separated VMs of a tenant, by defining different cost functions in which the number of job fragmentations is the variable. [22] and [30] use a single dimensional resource allocation algorithm and define a slot to represent one resource unit (CPU/memory/disk), in a way that each slot

can host one VM. [30] also proposed a binary search based heuristic algorithm to achieve an optimum point in the tradeoff between PM-cost and network cost in order to minimize the cost according to the arbitrary assumption for the proposed cost functions.

[31] addresses the problem of traffic engineering in data center networks from a different aspect. In [31], each job is characterized by the set of VMs communication with each other. The problem of mapping traffic flows of each job into VLANs and selecting the most efficient spanning tree protocols with the objective of load balancing is investigated regarding the bandwidth requirements of VMs and bandwidth constraints. CG is proposed to solve the optimization problem reducing the complexity and search space and then a semi-heuristic decomposition approach is proposed to make it scalable.

Data access latencies became a challenge in delay sensitive cloud applications. One of the main components of the latency is the communication between the processors and data nodes. In order to overcome this problem, network distance between computation and storage has to be designed precisely. Considering the VM placement problem as a classic linear sum assignment problem (investigated in [32]), [33] took the MapReduce/Hadoop architecture into account, and investigated the delay intensive cloud applications. They used the Hungarian algorithm proposed in [32] to optimize the data access latencies under various cases in which each VM requires different data sources or several VMs demand for just a data source.

In general, tasks of a user may have different demands and number of the tasks may vary over the time. In [34], it is assumed that each user runs individual tasks, and each task is characterized by a demand vector, which specifies the amount of resources required by the task. In general, tasks of a user may have different demands and quantity of the tasks may vary over the time. Then, Dominant Resource Fairness (DRF) multi-resource allocation algorithm is proposed to equalize the dominant share. This factor is defined as the maximum of the ratios of any resource type allocated to each user in the entire cloud. [35], [36], [37] and [38] also extend several types of DRF algorithm by focusing on the scenarios with different forms of demand. [39] proposed Dominant Resource Fairness in Heterogeneous cloud (DRFH) by generalizing DRF to heterogeneity in both resources and demands. DRFH equalizes users' dominant share in a heterogeneous cloud that leads to higher resource utilization. Then a heuristic algorithm

is suggested to decrease complexity of DRFH, for implementation in real world scenarios.

Many papers such as [28] address the traffic aware VM migration process. The service placement optimization problem belongs to the class of quadratic assignment problem (QAP), which is one of the hardest problems in the NP-hard class, and even its approximation is hard [40] and [41]. In [42], NetDEO, based on a swarm intelligence optimization model and search algorithm, is proposed to relocate VMs in order to adjust resource demands and resource availability. They defined the problem by considering a collection of jobs and servers. Traffic matrix is also considered to show the traffic rates among the jobs in the cloud computing system. In the traffic matrix, for jobs in the same server, the corresponding matrix element is zero. Each server has a capacity composite metric of process, memory and storage while jobs also are attributed by their resource requirements and traffic rates. Traffic stress of a node is defined as the root mean square of traffic rate between the node and all its communication peers. [42] relocates the VMs in order to minimize the total stress of the DC considering the initial condition of the CN. In the next step, they designed NetDEO that applies swarm intelligence optimization characteristics to improve the solution. However their stress definition is arbitrary with respect to to incoming and outgoing traffic.

2.4 Power Management in Cloud Computing Centers

Next, we describe the previous work on the power management in cloud computing centers. Generally, the cost of energy required for operation of a server is higher than its purchase price [43]. However, the consolidation of many servers in a cloud computing center empowers efficient usage of a server and provides better consumption of the power resources in the shared resource pools. Initiating from a small scale, [44] developed a model that considers the dynamic power usage of a server as a function of CPU utilization (it relies on the fact that CPU is the main power consumer in servers). [44] stated that the power consumption of a server grows linearly with increasing CPU utilization from the idle state upto fully utilized server. Then, according to the workload of DC, the number of servers in idle and busy states with different levels

of CPU utilization is estimated. [45] also found a strong relationship between CPU utilization and total power consumption of a server. Again, the proposed model assumes that the power consumption of a server increases linearly with the rise of CPU utilization. Finally, the total power consumption of a cloud computing center is estimated by summing up the power consumption of all the servers in the cloud.

In [46], several server pools have been considered in a DC. A reactive migration controller is proposed to detect and track the server load. Moreover, it dynamically enhances and reduces the number of active servers in the system to minimize the power usage. Their study shows that the controller approach, offers the best results in terms of quality of service and power usage.

In [43], the effectiveness of dynamic power management in data centers under an *M/M/k* queuing model via matrix analytic methods is investigated. Moreover, policies such as Delayed Off, *ON/OFF* and static power management have been considered and analyzed in [27] and [43]. Under *ON/OFF* policy, servers are in *off*, setup, or busy modes. If a new job arrives and all the active servers are already in the busy mode, then the job changes the status of a server in the *off* mode into setup mode. Also a server is shut down if there is no waiting job in the system. They propose Delayed Off policy which is the same as *ON/OFF* policy, except for the server going into the wait mode when the queue is empty. The waiting duration is the time that a server spends in the idle mode whenever there is no waiting job in the system. In [47], Balter et al. continue analysis in the heterogeneous workload case. They addressed heterogeneous types of jobs and workloads in the system. They considered several types of workloads and suggest a new method entitled “Auto Scale” which is independent of workload type. AutoScale scales the DC capacity and adjusts (by adding/removing) servers as needed. It maintains just the right amount of spare capacity to handle bursts in the request rate and it is robust to changes in the request rate, size and server efficiency. Under the AutoScale policy proposed in [47], each server decides autonomously when to turn itself off. When a server goes idle, rather than turning off immediately, it sets the duration to wait in the idle state. Timers prevent servers from going offline mistakenly just as a new arrival joins the system. However, timers can also waste power and capacity by leaving too many servers

in the idle state. Autoscale only keeps a small number of servers in the idle state by proposing a routing scheme that tends to concentrate jobs in a small number of servers. In order to implement AutoScale on a given cluster, such parameters have to be determined. The aforementioned parameter depends on the specifications of the system, such as the server type, the setup time, and the application, which do not change during the runtime. Under various conditions of loading, such as changes in the request size and in the server speed, as well as changes in the request rate, Autoscale has shown a better performance compared to the predictive algorithms.

2.5 Resource Allocation in Mobile Cloud Computing

Popularity of smartphones and related applications in various fields are significantly increasing in everyday life. These devices have a wide range of features (e.g., high-speed processors and supporting multiple wireless interfaces). Smartphones have become the primary computing platform for many users due to well-developed mobile applications in various realms such as commerce, learning, health care, computing, gaming, etc. While applications are becoming more and more complex, smartphones remain constrained due to limited processing power battery life. Most of the smartphone applications are QoS-sensitive and computation-intensive to perform on a mobile system. Mobile cloud computing (MCC) is a new concept in which mobile users access the cloud virtual resources via the Internet. [48],[49] and [50] give an overview of the MCC presenting definition, architecture, applications, and approaches, then, on the corresponding challenges at the operational, user, and application levels have been discussed. They introduced MCC as the dominant computing model for mobile applications in the future.

Mobile users usually need to maintain a low level of power consumption and thus computation must be performed in the cloud, which comes with a cost. Some researchers have studied power consumption in smartphones. It is beneficial to QoS improvement and battery power consumption to offload mobile data. The mobile computation offloading technique shares an application code between the cloud server and the mobile. A framework for smartphones is introduced in [51]. It shifts smartphone application processing into the cloud centers. It is based on the concept of smartphone virtualization

in the cloud and addresses lack of scalability by creating VMs of a complete smartphone system on the cloud. ThinkAir [51] provides on-demand resource allocation by dynamically managing VMs in the cloud via an execution controller. The execution controller handles decision-making and communication with the cloud server. It considers execution time, energy, and cost to make decision in order to achieve optimum performance. [52] suggests that cloud computing can potentially save energy through offloading of processing of applications with limited reliability and quality of service requirements. This reflects the fact that for some applications such as delay-sensitive ones, offloading to the clouds could not significantly offer energy savings to the smartphones while also satisfying QoS parameters.

Chapter 3

Performance Modeling of the Cloud Computing Centers

In this chapter, we will consider various cloud computing models that may be used in the dimensioning of these systems. Performance of these models will be determined under stochastic job arrival process and job service time distribution. The objective will be determining equilibrium distribution of the number of jobs in the system, job blocking probabilities, response time for different classes of jobs and distribution of the resource utilization.

3.1 Introduction of the models

In this chapter, we study several models which have been determined by a number of model parameters. These parameters are explained below.

- **Job Size parameter**

This parameter specifies the number of tasks in a job. This parameter may be a constant or variable. In the variable case, a job will initially require service to a single task but during its service time it will generate random number of tasks in the system. In the case of constant job type, a job will require service to a constant number of tasks at its arrival time.

- **Service Completion parameter**

This parameter determines service completion type of the tasks of a job. This parameter allows two types of service completion, simultaneous or individual. In the simultaneous category the service of all the tasks of a job will be completed simultaneously, while in the individual category service time of tasks will be independent of each other.

- **Resource Parameter**

This parameter determines the amount of resources available which maybe infinite or finite. In the infinite resource model, we assume that datacenter has infinite number of servers and in the finite case a datacenter has finite number of servers.

- **Virtual Machine Type parameter**

This parameter determines types of VMs which may be homogeneous or heterogeneous. In the first case, VMs have the same requirements (number of CPUs, memory and storage sizes), while in the second case, there may be different VM types which may differ from each other in their requirements.

- **Job arrival parameter**

We consider two types of job arrival processes, unsaturated and saturated cases. In unsaturated case, the job will arrive according to a Poisson process to the system as a function of time. In saturated case, there will be constant number of jobs in the system, where a new job will be inserted into the system as soon as service of a job is completed.
Each parameter value has been assigned an abbreviation. The combinations of the values of these parameters result in different models. These combinations result in a tree structure as shown in Fig. 3.1.

At each leave of the tree, the parameter of that leave as well as all of its ancestors are in effect. The numbers within parentheses at the leave of the tree give the subsection numbers where the analyses of these models are given in the chapter.

In the following sections of the chapter we present performance analysis of the cloud

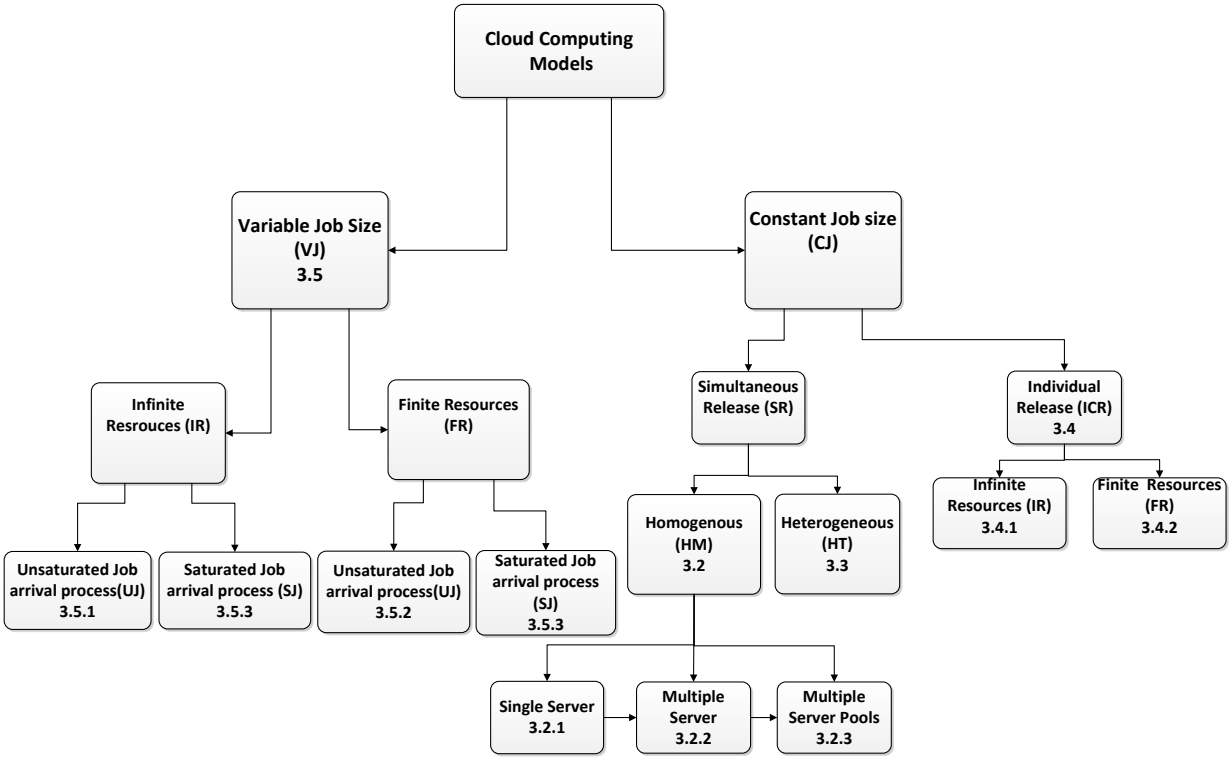

Fig. 3.1 Tree diagram of cloud computing models

computing models described in the above. In all cases, system is modeled using birth-death process. The models admit quite general service type distribution.

The remainder of this chapter is organized as follows. In section 3.2, we study systems with homogeneous VMs with constant job sizes and simultaneous task release times. Sections 3.3 and 3.4 extend the analysis of section 3.2 to systems with heterogeneous VMs and jobs with independent task release times respectively. In section 3.5, we present modeling of a system with variable job size. In section 3.6, we give a comparison of our results with the closest previous work that has been referred to in chapter 2 and section 3.7 contains the conclusions.

3.2 Modeling of a system with homogeneous VMs, constant job sizes and simultaneous release times (CJ, SR, HM)

In this section we will study the performance of a system with with homogenous VMs, constant job size and simultaneous release time. The abbreviation for the values of the model parameters have been listed in the above (CJ, SR, HM).

We assume multiple classes of jobs. Each class of jobs arrives at the system according to a Poisson process with a different parameter and each class has a different service rate and job size. The size of a job is determined by the number of tasks that it has and the job size remains constant during its service time. Each task requires a VM for its execution. Distribution of the service times of jobs may have rational Laplace transforms with a different mean service time for each class. Service time of a job begins with its arrival to the system and at the end of that service time all its tasks terminate simultaneously. In other words, resources related to an arriving job are provisioned and released together. The relevant notation has been introduced in Table 3.1.

Table 3.1 Parameter/Variable Definitions

Parameters	Indicator
R	number of classes of jobs
λ_r	arrival rate of class *r* jobs
λ_T	total job arrival rate
μ_r	Service rate of class *r* jobs
k_T	total number of busy VMs
b_r	number of VMs required by a class *r* job
n_r	number of class *r* jobs in the system

We will consider single and multiple servers and multiple server pools cases. We assume finite resources, thus a job will be blocked if there are no enough number of idle VMs to serve it. The objective of the following analysis will be to determine joint distribution of the number of jobs from each class, job blocking probabilities and distribution of the utilization of resources. We will also show applicability of our results

into power management in a cloud computing center.

Let us define state of the system as number of jobs from each class in the system, $\vec{n} = (n_1, n_2, \dots, n_r, \dots n_R)$, and $p(\vec{n})$ as the distribution of $\vec{n}$.

3.2.1 Single Server Model

First, we consider a system with finite resources of *S* VMs all located at a single server. In this case, an arriving job will be lost if there are not enough number of idle VMs to serve it. This model is same as blocking in shared resources environment studied in [53]. From there, the joint probability distribution of the number of jobs in the system is given by,

$$p(\vec{n}) = \frac{1}{G} \prod_{r=1}^{R} \frac{{\rho_r}^{n_r}}{n_r!} \tag{3.1}$$

where *G* is the normalization constant, which may be determined through a recursion [53] and $\rho_r = \frac{\lambda_r}{\mu_r}$. It may be seen that the joint probability distribution depends on the service time only through its mean value. Let *j* denote number of the busy VMs at the computing center, then, $j = \vec{n}.\vec{B}^T$ where $\vec{B} = [b_1, \dots b_r, \dots b_R]$. Defining probability distribution of the number of busy VMs in the computing center as,

$$q(j) = Pr\left(j = \vec{n}.\vec{B}^T\right)$$

from [53], $q(j)$ is given by the following recursion,

$$jq(j) = \sum_{r=1}^{R} b_r \rho_r q(j - b_r) \tag{3.2}$$

Then average number of busy VMs in the system is given by,

$$E[k_T] = \sum_{j=1}^{S} jq(j) \tag{3.3}$$

Let $\tilde{q}(\ell)$ denote probability distribution of the number of idle VMs , then, $\tilde{q}(\ell) = q(S - \ell)$. Defining PB_r as the probability that a class *r* job will be blocked, then from [53],

$$PB_r = \sum_{\ell=1}^{b_r-1} \tilde{q}(\ell) = 1 - \frac{G(S-b_r,R)}{G(S,R)} \quad (3.4)$$

where $G(S - b_r, R)$ may be calculated recursively [53].

$$G(j,i) = \sum_{\ell=0}^{\left\lfloor \frac{j}{b_i} \right\rfloor} \frac{{\rho_r}^{\ell}}{\ell!} G(j - \ell b_i, i-1) \;\; i = 2 \ldots . R, j = 0,1, \ldots, S$$

$$G(j,1) = \sum_{\ell=0}^{\left\lfloor \frac{j}{b_1} \right\rfloor} G(j - \ell b_i, i-1) \qquad j = 0,1, \ldots, S$$

The overall job blocking probability is given by,

$$P_b = \frac{1}{\lambda_T} \sum_{i=1}^{R} \lambda_i PB_i$$

where $\lambda_T = \sum_{i=1}^{R} \lambda_i$.

3.2.2 Multiple Servers Model

Next, we consider a system with *M* servers where each server has *S* VMs.

As before, an arriving job will be blocked if the total number of idle VMs in the computing center is less than the number of VMs needed to serve the arriving job. Thus as far as job blocking probabilities are concerned the system may be considered as a single server with a total of *MS* VMs. However, in this case, it is possible that no server may have enough number of idle VMs to serve an accepted job to the system and the job may need to be assigned VMs from multiple servers which will be referred to as fragmented service. As a result, these jobs will experience additional performance penalty due to the need for communication among the servers. Henceforth, we determine the probability that assigned VMs to an accepted job will be fragmented among servers. Let us introduce the following additional notation,

V = total number of VMs at the computing center.

j = total number of busy VMs in the computing center.

$\vec{\varepsilon} = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m, \ldots \varepsilon_M)$ where ε_m corresponds to the number of idle VMs at an arbitrary time in the m^{th} server.

The total number of VMs in the datacenter is given by:

$$V = MS \tag{3.5}$$

Let ℓ denote the total number of idle VMs in the computing center,

$$\ell = \sum_{m=1}^{M} \varepsilon_m = V - j \tag{3.6}$$

Since b_r denote the number of VMs required to provide service to a class *r* job, depending on the value of the ℓ, the following possibilities exist for a class *r* job:

$$\begin{cases} job\ will\ be\ blocked\ , & if \quad \ell < b_r \\ job\ may\ receive\ \ fragmented\ service, & b_r \leq \ell < Mb_r \\ job\ will\ receive\ service\ from\ a\ single\ server, & Mb_r \leq \ell \end{cases}$$

Assuming a load balancer is operating in the system, then probability distribution of the number of idle VMs in each server will be identical. Given that total number of idle VMs is equal to ℓ, let $P(\ell, M, b_r)$ denote the conditional probability that none of the *M* servers have b_r or more idle VMs:

$$P(\ell, M, b_r) = Pr(\varepsilon_1 < b_r, \dots, \varepsilon_m < b_r, \dots, \varepsilon_M < b_r) \tag{3.7}$$

Distribution of the number of idle VMs in servers is analogous to the traditional balls urn model, where each ball is placed into one of the urns with equal probability. Then, distribution of the number of idle VMs in each server will be the same as distribution of the balls in the urns model [57]. $P(\ell, M, b_r)$ does not have a closed form solution but it could be obtained recursively [57],

$$P(\ell + 1, m, b_r) = P(\ell, m, b_r) - \binom{\ell}{b_r} P(\ell - b_r, m - 1, b_r) \frac{(m-1)^{\ell - b_r}}{m^{\ell}} \tag{3.8}$$

with the following initial condition,

$$P(\ell, m, b_r) = 1 \quad for \quad \{1 \leq \ell \leq b_r, 1 \leq m \leq M\}$$

The following result may be used to simplify the above recursion,

$$\binom{\ell}{b_r}\frac{(m-1)^{n-b_r}}{m^n} = \binom{m-1}{m}\binom{\ell}{\ell-b_r}\left\{\binom{\ell-1}{b_r}\frac{(m-1)^{\ell-1-b_r}}{m^{\ell-1}}\right\} \tag{3.9}$$

Note that $P(\ell, M, b_r)$ gives the probability of a class *r* job receiving fragmented service when $b_r \leq \ell < Mb_r$. If $Mb_r \leq S$ then $\ell \leq S$, then all assignment combinations of idle instances of *V* resources into servers are feasible. But if $S < Mb_r$ it is possible that $\ell > S$, then some assignment of idle VMs to the servers will not be admissible because it will result in allocation of more idle VMs to a server than the capacity of that server. The non-admissible assignments of idle VMs have to be excluded through normalization. Let $\tilde{P}(\ell, M, b_r)$ denote the probability that a class *r* job receives fragmented service, thus:

$$\tilde{P}(\ell, M, b_r) = \begin{cases} P(\ell, M, b_r), & \text{if } \ell \leq S \\ \frac{P(\ell, M, b_r)}{1-\sigma}, & \ell > S \end{cases} \tag{3.10}$$

where $\sigma = \sum_{k=S}^{\ell} P(\ell, M, k)$ and $P(\ell, M, k)$ is obtained from (3.8). Next, unconditioning the above result wrt the distribution of the number of idle VMs leads to the probability that a class *r* job will receive fragmented service. Defining,

PF_r= *Pr*(an accepted class *r* job receives fragmented service)

Then, it is given by,

$$PF_r = \frac{\sum_{\ell=b_r}^{Mb_r-1} \tilde{P}(\ell, M, b_r)\tilde{q}(\ell)}{1-PB_r} \tag{3.11}$$

In the above, denominator normalizes the fragmentation probability with the probability of accepting a job.

Next, we present numerical and simulation results for a computing center with multiple servers. Discrete event-based simulation has been developed to determine accuracy of the assumption in the analysis that the number of idle VMs is uniformly distributed over multiple servers. Simulation implements a practical load balancer to be described below to achieve fair distribution of the load among the servers. In simulation also it has been assumed that jobs arrive into the system according to a Poisson process and job service time are exponentially distributed.

We consider a system with *M=5* servers with $S = 50$ VMs per server. We assume 4 classes of jobs with the following VM requirements and job arrival rates,

$$\vec{B} = [b_1\ b_2\ b_3\ b_4] = [1\ \ 2\ \ 3\ \ 4] \tag{3.12}$$

$$\lambda = [\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4] = [0.4\ 0.3\ .2\ 0.1]\lambda_T \tag{3.13}$$

It may be seen that jobs with smaller VMs requirements have been assigned higher arrival rates. Fig. 3.2 presents blocking probabilities of different classes of jobs as a function of the total job arrival rate. As may be seen, blocking probabilities increases with the number of VMs required by a job class. As expected, there is total agreement between numerical and simulation results as the analysis for calculation of job blocking probabilities is exact.

Next we present the results concerning service fragmentation. In simulation, using a load balancer, it is assumed that a server selection algorithm attempts to achieve fair distribution of the load among the servers. An accepted job if possible will be given service without fragmentation otherwise with fragmentation. If a job receives service without fragmentation, then it is assigned to the server with highest number of idle VMs. On the other hand, if a job receives service with fragmentation the scheduling algorithm aims to minimize the number of fragments depending on the distribution of the number of idle VMs in the servers. In Fig.s 3.3 and 3.4, we present average number of idle VMs in a server and job fragmentation probabilities for each class as a function of the total job arrival rate. The jobs with higher VM requirements experience higher fragmentation

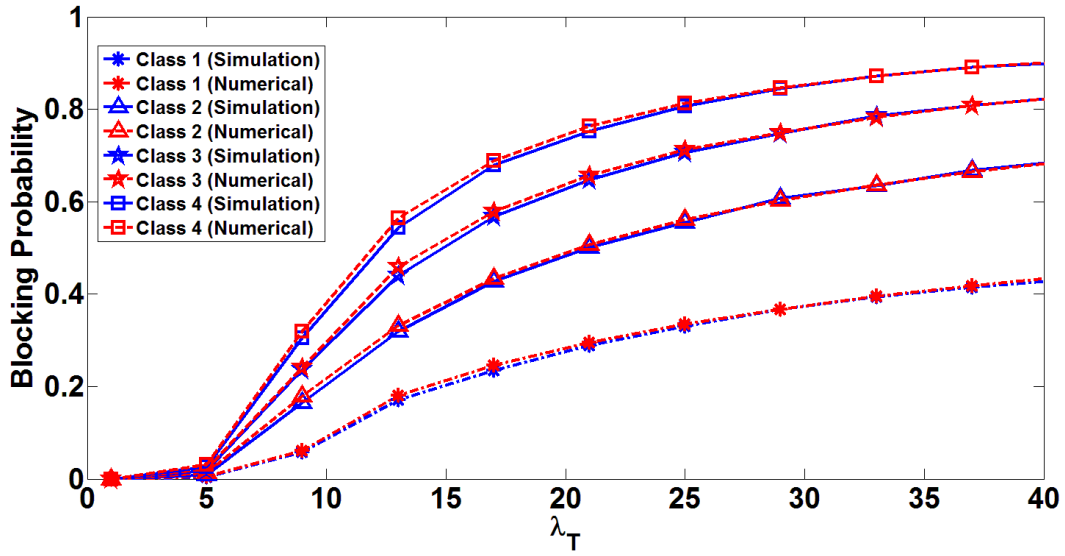

Fig. 3.2 Numerical and simulation results for blocking probabilities of different classes of jobs as a function of total job arrival rate

probabilities at any total arrival rate. From Fig. 3.4, the fragmentation probability of class 4 jobs reaches to %30 at the total job arrival rate of 30. Job fragmentation will increase the communication latency between the VMs, which will increase job service times. As may be seen, there is a close agreement between numerical and simulation results in both figures, which validates the assumption in the analysis that the number of idle VMs is uniformly distributed across the multiple servers.

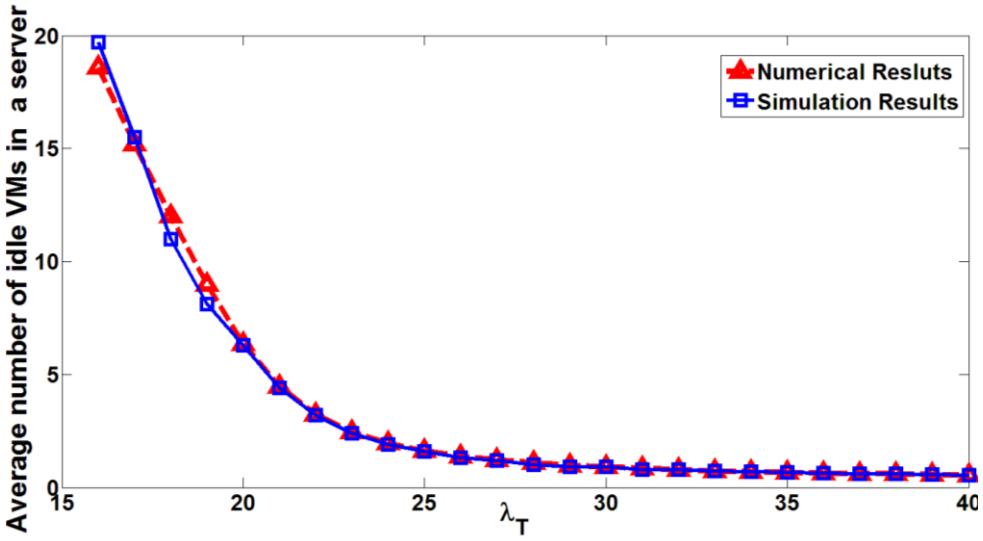

Fig. 3.3 Numerical and simulation results for the average number of idle instances of resources per server as a function of total job arrival rate.

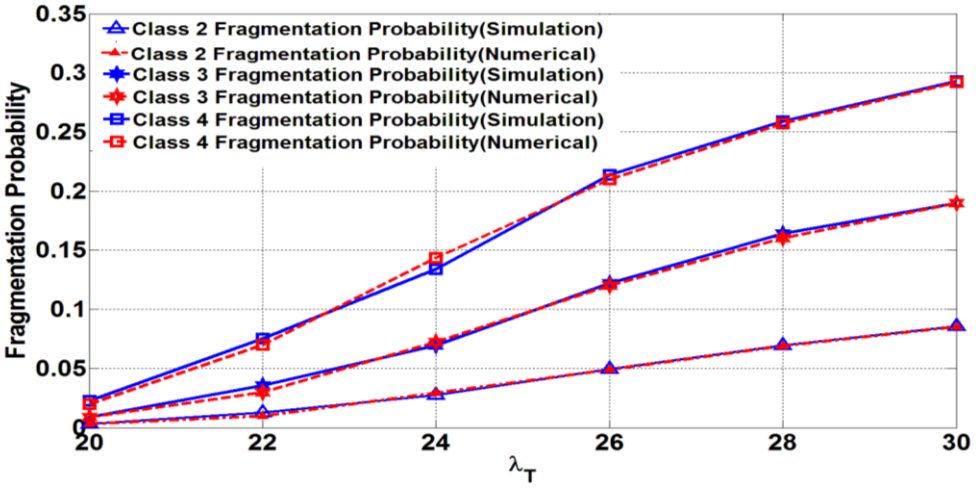

Fig. 3.4 Numerical and simulation results for fragmented service probabilities of different classes of jobs as a function of total job arrival rate.

3.2.3 Multiple Server Pools Model

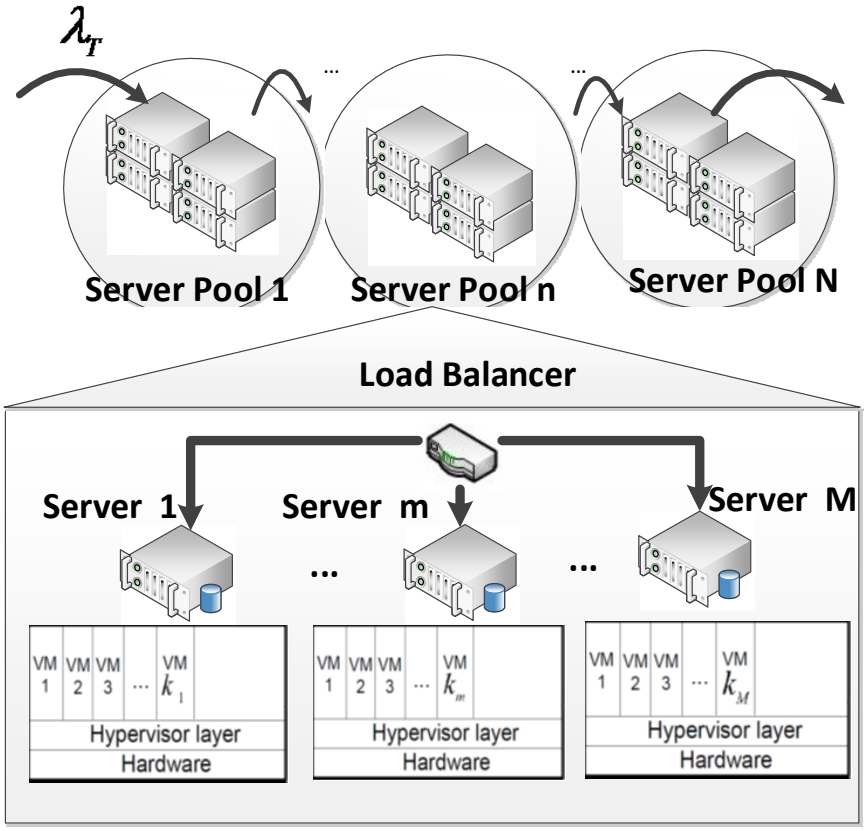

Fig. 3.5 Topology of the cloud computing center

In this subsection, we extend our model to cloud computing centers with pools of servers. Pool management techniques attempt to reduce power consumption of the system, which represents a significant component of the operating cost of a cloud computing center. Topology of the cloud computing center under consideration is shown in Fig. 3.5. These techniques turn off a server pool to save power if its servers are not currently serving any job. Let us assume that there are N server pools in the system, which are numbered as, n=1..N. We assume that scheduling algorithm always assigns a job to the server pool with the smallest index number that has enough idle resources. It is assumed that a job will not be assigned resources from multiple server pools to keep communication overhead low. Thus a job will be served by the pool n+1 with enough idle resources if pool n does not have enough idle resources. As before the total job arrival process at the system will be according to a Poisson process. The first pool of servers will see the total job arrival process while any other pool of servers will see the overflow traffic from the preceding pool. We assume that the overflow processes are Poisson which is an approximation to be verified by simulation. Within a pool, if possible, a job will be placed in a single server otherwise it will be fragmented. Thus VMs of each pool may be considered as a completely shared resource without the need to make a distribution among its servers. Let us define,

λ_{rn}= arrival rate of class r jobs to the n'th server pool.
λ_{Tn}= total arrival rate of the jobs to the n'th server pool.
PB_{rn}= probability that a class r job will be blocked by the n'th server pool.
PB_n= overall job blocking probability at the n'th server pool.
$q_n(j) = \Pr(j \text{ VMs will be busy in the } n^{th} \text{ server pool})$
g = number of active server pools.
$g_n = \Pr(g = n)$
Then, we have the following,

$$\lambda_{rn} = \lambda_{r(n-1)} PB_{r(n-1)} = \lambda_{r1} \prod_{i=1}^{n-1} PB_{ri} \ , \quad n \geq 2.$$

$$\lambda_{Tn} = \sum_{r=1}^{R} \lambda_{rn}$$

where $\lambda_{T1} = \lambda_T,\ \lambda_{r1} = \lambda_r$

$$PB_n = \frac{1}{\lambda_{Tn}} \sum_{r=1}^{R} \lambda_{rn} PB_{rn} \tag{3.14}$$

Assuming that each pool has *M* servers with *S* VMs per server, then, $q_n(j)$ will be determined by (3.2) with finite resources of *MS* and overflow traffic from the pool *(n-1)* as job arrival process. Then,

$$g_n = \prod_{i=n+1}^{N} q_i(0) \tag{3.15}$$

$$\vec{g} = [g_0, \dots, g_n, \dots, g_N]$$

We have tested the accuracy of the Poisson approximation of the overflow processes in the analysis through discrete event based simulation. In simulation also arrival of the jobs is according to a Poisson process and job service times are exponentially distributed. We assumed four job classes defined in (3.12, 3.13) with *N*=5 server pools, *M*=5 servers/pool and S =50 VMs/server. In Fig.s 3.6 and 3.7, we have plotted numerical and simulation results for the average number of idle VMs and the probability distribution of the number of active server pools in the system as a function of the total job arrival rate respectively. As may be seen, there is a close agreement between numerical and simulation results, which justifies Poisson assumption of overflow processes. From Fig. 3.7, it is seen that at any arrival rate with probability one there will be only single number of active server pools except in the narrow transition regions. This plot shows that system operation does not result in frequent *on-off* switching of the server pools if the job arrival rate is not time-varying. Fig. 3.8 presents overall job blocking probabilities of the server pools as a function of the total job arrival rate. As may be seen, job blocking probabilities of server pools drop with the increasing index value with PB_5 giving the overall job blocking probability of the system. The results in this figure may be used to determine number of needed active server pools to support a given traffic load at an acceptable level of job blocking probability.

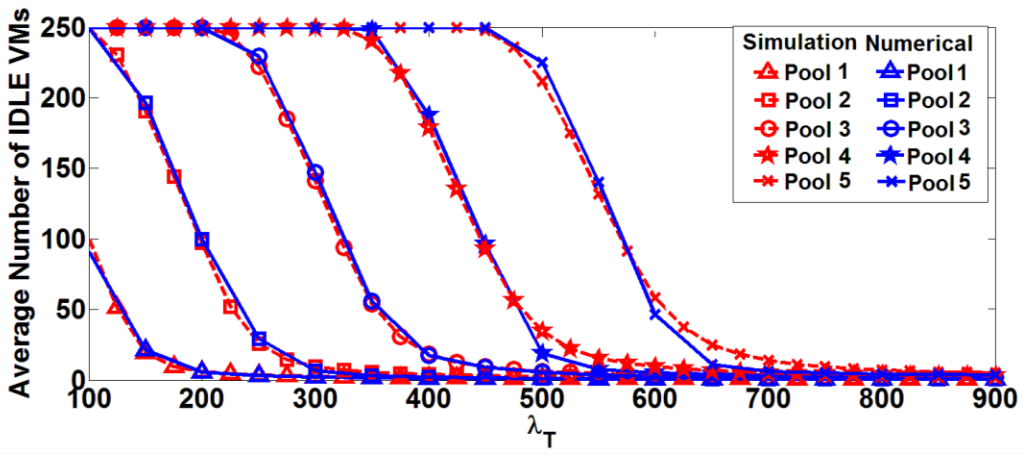

Fig. 3.6 Numerical and simulation results for the average number of idle VMs of different server pools as a function of total job arrival rate

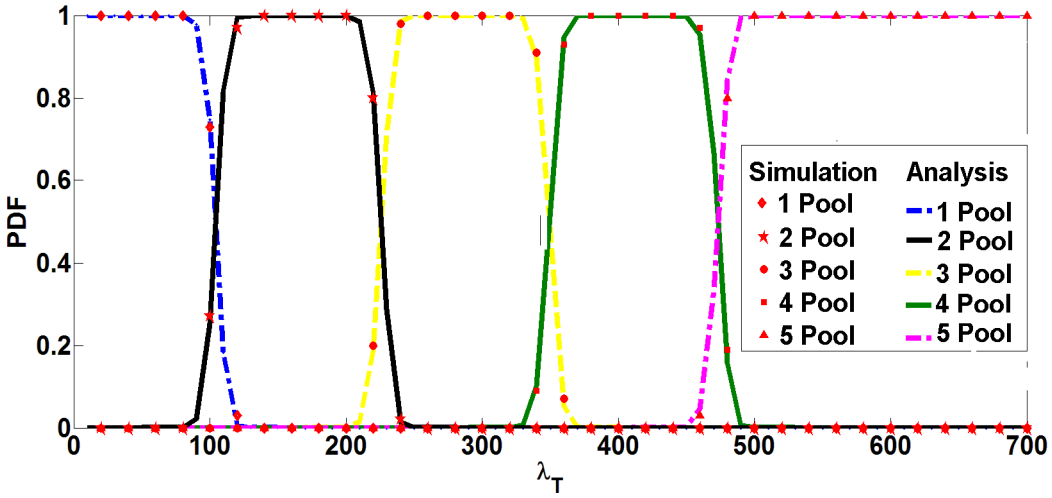

Fig. 3.7 Numerical and simulation results for probability distributions of number of active server pools as a function of total job arrival rate

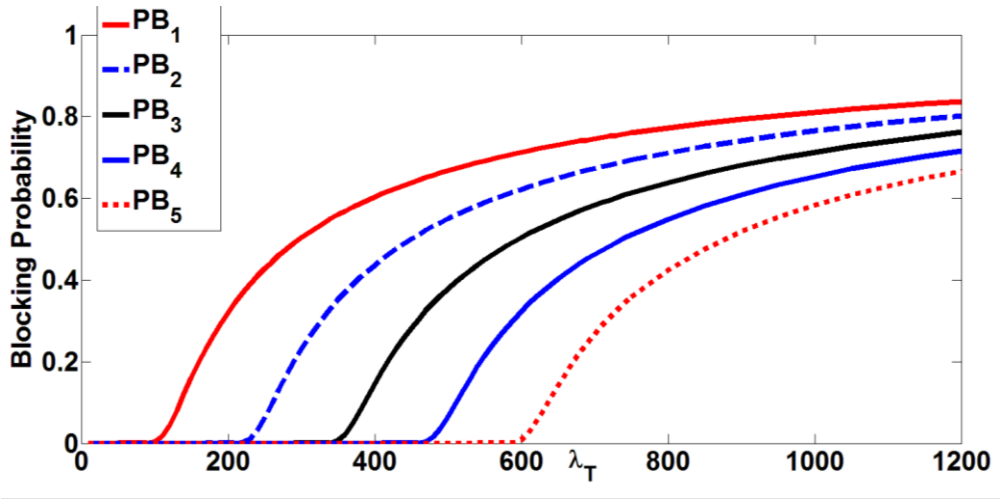

Fig. 3.8 Job blocking probabilities of server pools as a function of total job arrival rate.

Next, we will assume that the total job arrival rate to the system is time-varying. It will be assumed that job arrival rate will be changing according to a discrete-time Markov chain. The time-axis will be slotted with slot durations equaling to server set-up time. We will let number of active servers to denote state of the system with the state of the system changing at the discrete-times. There will be set-up times for turning an off machine to on, while turning an on machine off will be instantaneous. As may be seen from the previous results, the domain of the total arrival rate may be divided into intervals during which number of active server pools has a non-zero probability only for a single value during an interval. Let λ'_{Tn} denote the total arrival rate at the midpoint of the interval for $g_n = 1$. In calculation of job blocking probabilities during the transition from state i to state j, where $j > i$, we will assume that the total job arrival rate is given by λ'_{Tj}.

Letting p_{ij} denote the transition probability from state i to state j and P the corresponding transition probability matrix, then the steady-state probability distribution of the number of active server pools is determined by,

$$\vec{g} = \vec{g}P \tag{3.16}$$

Defining $\bar{g}$ as average utilization of the server pools in the system,

$$\bar{g} = \frac{1}{N}\sum_{i=0}^{N} i\, g_i \tag{3.17}$$

Given the rising cost of energy, with the growing scale of cloud computing datacenters, the expenditure on enterprise power usage and server cooling prevents facility owners to keep all server pools active. On the other hand, switching a server pool on requires setup time, which can adversely affect system performance in terms of job blocking rate. Hence, we consider a dynamic power management approach similar to that in [47] aiming to reduce power wastage while keeping job blocking probabilities and consequently loss of revenue at an acceptable level. In the following we consider four schemes, which will be referred to as always-on, reactive, proactive and optimal prediction and compare their performances. In the always-on case, there is no power management and all the idle server pools remain on. In the reactive case, idle server pools are turned off and they are turned on according to the demand. This scheme includes set-up times during which job losses occur. Reactive scheme responds to load increases with the time lag of one slot. In proactive case, an additional pool is kept in idle state to meet any load increases. The optimal prediction scheme predicts the job arrival rate for the next slot and turns on enough number of off servers to meet the demand.

Let k_p denote the cost of per unit power consumption (standard fee per watt) and k_r denote per hour rental rate of a VM. Also, p_{on} and p_{idle} denote the average power usage of a VM in active and idle states respectively. Next, we determine the net cost of transition (NC) to a higher state per slot for each of the four schemes, which is the difference between revenue and cost of power consumption. In the following equations, earned and lost revenue has negative and positive signs respectively.

$$NC_{always-on} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{k_p(N-i)MSp_{idle} - \sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left(k_r r\lambda'_{rj} PB_{ri} \frac{1}{\mu_r}\right)\right\} \tag{3.18}$$

$$NC_{reactive} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{\sum_{j=i+1}^{N} p_{ij} \left[k_p(j-i)MSp_{on} + \sum_{r=1}^{R} \left(k_r r\lambda'_{rj} PB_{ri} \frac{1}{\mu_r}\right)\right]\right\} \tag{3.19}$$

$$NC_{proactive} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{k_p MSp_{on} + \sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left(k_r r\lambda'_{rj} PB_{r(i+1)} \frac{1}{\mu_r}\right) -\right.$$

$$\sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left[k_r r\lambda'_{rj} (PB_{ri} - PB_{r(i+1)}) \frac{1}{\mu_r} \right] \Big\} \quad (3.20)$$

$$NC_{Optimal\ Prediction} =$$

$$\bar{\zeta} \sum_{i=0}^{N-1} g_i \left\{ \sum_{j=i+1}^{N} p_{ij} \left[k_p (j-i) MSp_{on} - \sum_{r=1}^{R} \left(k_r r\lambda'_{rj} PB_{ri} \frac{1}{\mu_r} \right) \right] \right\} \quad (3.21)$$

In the above, the terms with k_p and k_r correspond to cost and revenue items respectively. Clearly, the scheme with the most negative net cost value will be performing better than the others. We need to know transition probabilities of the imbedded Markov chain for calculation of the net cost of the transitions. In practice, these values will be determined from the measurements, however, next we illustrate the utilization of our results through an example. We assume the same job classes that have been defined in (3.12) with the additional parameter values given below,

$k_p = 0.055 \frac{\$}{kWh}$ $(HydroQuebec\ rate)$

$k_r = 0.085 \frac{\$}{hr} (Microsof\ Azure\ Small\ VM\ )$

$N = 5, M = 5,\ S = 50,\ R = 4$

$p_{on} = 405w$, $p_{idle} = 225w$, (Intel Atom Centerton 1.6 GHz CPU)

$\bar{\zeta} = 300\ sec$

where p_{on} is the required power to turn a CPU on. Next we assume that the transition probabilities for the discrete-time Markov chain are given by,

$$p_{ij} = \begin{cases} {\gamma_i}^{i-j}, & 0 \le j < i \le N \\ \alpha_i, & j = i \\ {\beta_i}^{j-i}, & 0 \le i < j \le N \end{cases} \quad (3.22)$$

where α_i, β_i and γ_i are state dependent parameters. As may be seen the transition probability between states *i* and *j* is given by a power of β_i or γ_i where the power is determined by the distance between the two states. Thus probability of transition between two states decreases with the increasing distance between them. Next, we will relate state

dependent parameters α_i, β_i to each other. It has been found that average utilization of a cloud computing center is presently about 30%, $\bar{g} = 0.3$ [58]. As a result, the system will spend more time in state 1 than the other states. We will designate state 1 as the base state and express all the α_i, β_i as functions of α_1, β_1 respectively. Next we assumed that $\beta_i = \tau^{|1-i|}\beta_1, \alpha_i = \sigma^{|1-i|}\alpha_1$ where σ, τ are proportionality constants, $0 \leq \sigma, \tau \leq 1$. We note that γ_i is determined from the normalization condition of the transition probabilities of each state. High value of α_1 (low values of β_1, γ_1) indicates a system with slowly varying job arrival rate, on the other hand low value of α_1 (higher values of β_1, γ_1) indicates a system with fast varying job arrival rate, the latter being a more dynamic system.

In Fig.s 3.9 and 3.10, we present plots of NC for the four schemes as a function of β_1 and α_1 respectively. As expected, in both cases, optimal prediction gives the best performance as its net cost has the most negative value. In Fig 3.9, reactive scheme always performs better than always-on and most of the time better than proactive scheme because the system spends a lot of time in state 1 due to high value of α_1. In Fig. 3.10, the system is more dynamic for low values of α_1 compared to its high values. Since reactive scheme's response has a lag time, it gives the worst performance for $\alpha_1 < 0.65$. It may be seen that the performance of various schemes depend on degree of time-variation of the traffic load. Fig. 3.11 shows the utilization of the system as a function of parameter α_1 with the other parameters fixed. As may be seen, utilization increases with increasing value of α_1.

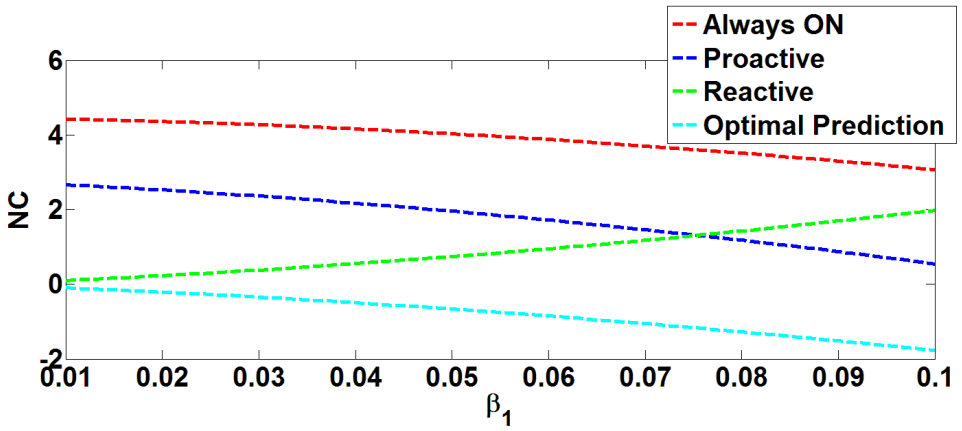

Fig 3.9 Net cost of a transition for *always-on, reactive, proactive* and *optimal prediction* schemes as a function of β_1 for $\alpha_1 = 0.88$, $\sigma = \tau = 0.1$.

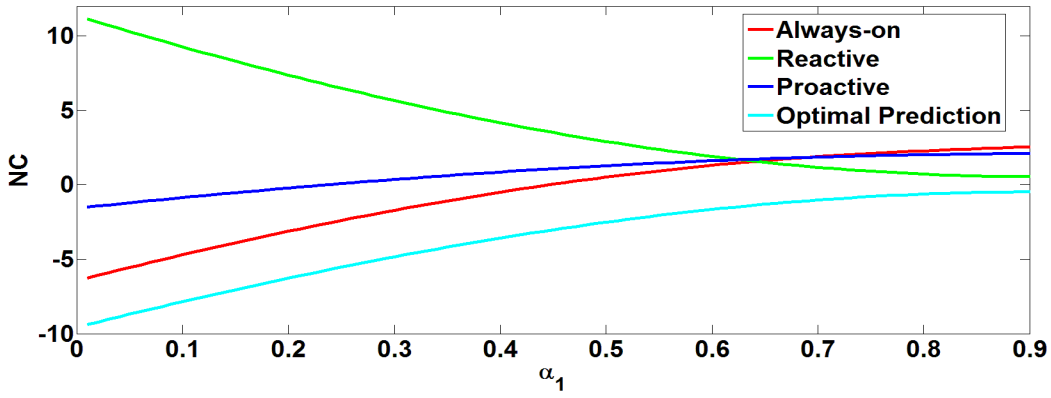

Fig. 3.10 Net cost of a transition for *always-on*, *reactive*, *proactive* and *optimal prediction* schemes as a function of α_1 for $\beta_1 = 0.05$, $\sigma = \tau = 0.1$.

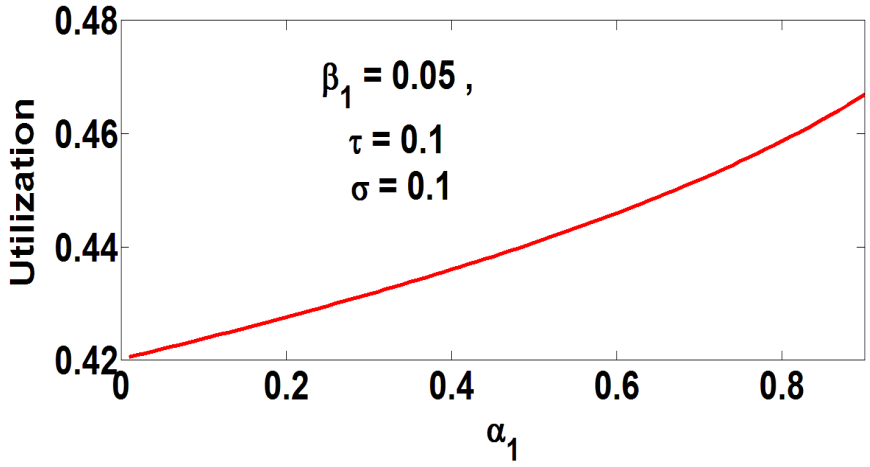

Fig.3.11 Utilization as a function of α_1

Findings in this section may give insight to the selection of appropriate system operation policy, i.e. proactive to reactive or vice versa. For example, in a static scenario (large values of α_1) reactive approach is good enough while for more dynamic systems the proactive approach gives better performance.

3.3 Modeling of a system with heterogeneous VMs, constant job size and simultaneous release times (CJ, SR, HT)

In this section, we extend the results of the previous section to a single server with heterogeneous types of VMs. The VM types may differ from each other in the amount of resources allocated to a VM, such as in number of CPUs, memory and storage sizes. We assume that there are L types of VMs and a job may request up to J VMs of a single type. The type and number of VMs requested will define class of a job. Thus a class $j\ell$ job will request j VMs of type ℓ, $j = 1..J, \ell = 1..L$. Let us introduce the following notation,

$F =$ number of resource types.

$C_f =$ number of units of resource f, $f = 1..F$.

$b_{\ell f}$ = number of units of resource f required by a type ℓ VM, $\ell = 1..L, f = 1..F$.

$\lambda_{j\ell} =$ arrival rate of class $j\ell$ jobs that require j number of type ℓ VMs, $j = 1..J, \ell = 1..L$.

$\mu_{j\ell} =$ service rate of class $j\ell$ jobs.

$n_{j\ell}$ = number of class $j\ell$ jobs in the system.

$\vec{b}_\ell = (b_{\ell 1}, \dots b_{\ell f}, \dots, b_{\ell F})$

$\vec{C} = (C_1, \dots C_f, \dots, C_F)$

$$\vec{n} = (n_{11}, \dots, n_{j1} \dots, n_{J1}, \dots, n_{1\ell}, \dots, n_{j\ell}, \dots n_{J\ell}, \dots, n_{1L}, \dots, n_{jL}, \dots, n_{JL})$$

$\overrightarrow{n_{j\ell}^-} = (n_{11}, \dots, n_{j1} \dots, n_{J1}, \dots, n_{1\ell} - 1, \dots n_{J\ell}, \dots n_{1L}, \dots, n_{jL}, \dots, n_{JL})$

Total arrival rate of the jobs is given by,

$$\lambda_T = \sum_{j=1}^{J} \sum_{\ell=1}^{L} \lambda_{j\ell}$$

Defining B as the resource matrix of VM types,

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1f} & \cdots & b_{1F} \\ \vdots & & b_{\ell f} & \ddots & \vdots \\ b_{L1} & \cdots & b_{Lf} & \cdots & b_{LF} \end{bmatrix}$$

Next defining N and Λ as matrices of the number of each class of jobs and their arrival rates respectively,

$$N = \begin{bmatrix} n_{11} & \cdots & n_{1\ell} & \cdots & n_{1L} \\ \vdots & & n_{j\ell} & \ddots & \vdots \\ n_{J1} & \cdots & n_{J\ell} & \cdots & n_{JL} \end{bmatrix} \tag{3.23}$$

$$\Lambda = \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1\ell} & \cdots & \lambda_{1L} \\ \vdots & & \lambda_{j\ell} & \ddots & \vdots \\ \lambda_{J1} & \cdots & \lambda_{J\ell} & \cdots & \lambda_{JL} \end{bmatrix}$$

As before, we assume that the distribution of the service time of each class of jobs has a rational Laplace transform.

We note that this model is an extension of blocking in shared resources environment studied in [53] to a system with multiple types of resources. Following the analysis in [53], we will determine joint probability distribution of the number of jobs in the system and derive a multi-dimensional recursion for the distribution of the utilization of resources. First, we will write the local balance equation (LBE) of this system. An LBE equates the flow due to a departure of a job from a network state to the flow due to an arrival of a job to a network that will return the system to the same state, thus,

$$n_{j\ell}\mu_{j\ell}p(\vec{n}) = \lambda_{j\ell}p\left(\overrightarrow{n_{j\ell}^{-}}\right) \tag{3.24}$$

Let us assume the following joint probability distribution of the number of different classes of the jobs in the system,

$$p(\vec{n}) = \frac{1}{G}\prod_{j=1}^{J}\prod_{\ell=1}^{L}\frac{{\rho_{j\ell}}^{n_{j\ell}}}{n_{j\ell}!} \tag{3.25}$$

where G is the normalization constant and $\rho_{j\ell} = \frac{\lambda_{j\ell}}{\mu_{j\ell}}$.

It may be shown by substitution that (3.25) satisfies (3.24). Since $p(\vec{n})$ satisfies the LBE, it also satisfies the global balance equations (GBEs), and therefore (3.25) is the correct distribution. It may be seen again that the joint probability distribution depends on service only through it mean.

Let us define,

u_f = number of units of resource *f* that is busy.

$$\vec{u} = (u_1, .., u_f, .., u_F) \tag{3.26}$$

Let $q(\vec{u})$ denote joint probability distribution of the utilization (number of busy units) of different type of resources. Next, we derive the following multi-dimensional recursion for determining this distribution,

$$u_f q(\vec{u}) = \sum_{l=1}^{L}\sum_{j=1}^{J} jb_{\ell f}\rho_{j\ell}q(\vec{u} - j\vec{b}_{\ell}) \tag{3.27}$$

Proposition 3.1: $q(\vec{u})$, probability distribution of the utilization of resources may be determined by following multi-dimensional recursion,

$$u_f q(\vec{u}) = \sum_{l=1}^{L}\sum_{j=1}^{J} jb_{\ell f}\rho_{j\ell}q(\vec{u} - j\vec{b}_{\ell})$$

Proof: Let us define,

a_{ℓ} = number of type ℓ VMs that is busy.

$\vec{a} = [a_1, a_2, \ldots, a_{\ell}, \ldots a_L]$

$\vec{j} = (1,2 \ldots, j, \ldots J)$

From the above definitions, we have,

$$a_\ell = \sum_{j=1}^{J} j n_{j\ell} \ , \ u_f = \sum_{\ell=1}^{L} a_\ell b_{\ell f} = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell r} n_{j\ell} \tag{3.28}$$

Then,

$$\vec{a} = \vec{j}N \qquad \vec{u} = \vec{a}B \tag{3.29}$$

$q(\vec{u})$ is given by,

$$q(\vec{u}) = Pr(\vec{a}B = \vec{u}) = \sum_{\vec{n}|\vec{a}B=\vec{u}} p(\vec{n}) \tag{3.30}$$

Let us rewrite LBE in equation (3.24) as follows,

$$n_{j\ell} p(\vec{n}) = \rho_{j\ell} p\left(\overrightarrow{n_{j\ell}^{-}}\right) \tag{3.31}$$

Multiplying both sides of (3.31) by $j b_{\ell f}$ and summing over j and ℓ,

$$p(\vec{n}) \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} n_{j\ell} = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p\left(\overrightarrow{n_{j\ell}^{-}}\right)$$

Substituting from (3.28) on the LHS,

$$u_f p(\vec{n}) = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p\left(\overrightarrow{n_{j\ell}^{-}}\right) \tag{3.32}$$

Next let us sum both sides of equation (3.32) over the states ($\vec{n}|\vec{a}B = \vec{u}$),

$$\sum_{\vec{n}|\vec{a}B=\vec{u}} u_f p(\vec{n}) = \sum_{\vec{n}|\vec{a}B=\vec{u}} \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p\left(\overrightarrow{n_{j\ell}^{-}}\right)$$

Substituting from (3.28) on the LHS and interchanging the order of summations on the RHS,

$$u_f q(\vec{u}) = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p \sum_{\vec{n}|\vec{a}B=\vec{u}} p\left(\overrightarrow{n_{j\ell}}\right) \tag{3.33}$$

We note from (3.28), $(\vec{n}|\vec{a}B = \vec{u}) = (\vec{n}|\vec{j}NB = \vec{u})$

Then $(\vec{n}|\vec{a}B = \vec{u})$ means that,

$$\left(\overrightarrow{n_{j\ell}}|\vec{j}\,\overrightarrow{n_{j\ell}}B = \vec{u} - j\overrightarrow{b_\ell}\right) \tag{3.34}$$

Substituting (3.34) in (3.33) completes the proof.

Then, the average utilization vector is given by,

$$E(\vec{u}) = \sum_{\vec{u}|\,(\forall f \in F,\ u_f \leq C_f)} \vec{u} q(\vec{u}) \tag{3.35}$$

The probability that demand for a type ℓ VM will be blocked is given by,

$$PB_\ell = \sum_{\vec{u}|\,(\forall f \in F,\, u_f + b_{lf} > C_f)} q(\vec{u}) \tag{3.36}$$

Next we will give an example based on a system with three VM types given in Table 3.2 with the following resource vector,

Table 3.2
Representative VMs Specifications

VM type	Memory	CPU cores	Storage
Standard	2(GB)	2	100 (GB)
High Memory Extra Large	16(GB)	6	400 (GB)
High CPU Extra Large	8(GB)	10	200 (GB)

$$\vec{C} = (160GB, 200\ Core, 10000\ GB) \tag{3.37}$$

From Table 3.2, resource matrix of VM types is given by,

$$B = \begin{bmatrix} 2 & 2 & 100 \\ 16 & 6 & 400 \\ 8 & 8 & 200 \end{bmatrix} \tag{3.38}$$

Assuming the following arrival rate matrix for classes of jobs with (*J =4*),

$$\Lambda = \begin{bmatrix} 0.2 & 0.1 & 0.1 \\ 0.15 & 0.075 & 0.075 \\ 0.1 & 0.05 & 0.05 \\ 0.05 & 0.025 & 0.025 \end{bmatrix} \lambda_T \tag{3.39}$$

It should be noted that in the above job classes with higher resource requirements have lower arrival rates. Figures 3.12, 3.13 and 3.14 show the cumulative probability distributions of memory, CPU and storage utilizations respectively with the total job arrival rate as a parameter. These results may be used to determine bottleneck resources and redundancy in the system. It may be seen that at the total job arrival rate of 10, the values of memory, CPU and storage corresponding to cumulative probabilities of unity are 160Gb, 130cores and 4500Gb respectively. Since at this arrival rate all the available memory may be busy, the system cannot support a higher traffic load. As a result, the number of cores beyond 130 and storage beyond 4500Gb will not be utilized and they will be redundant.

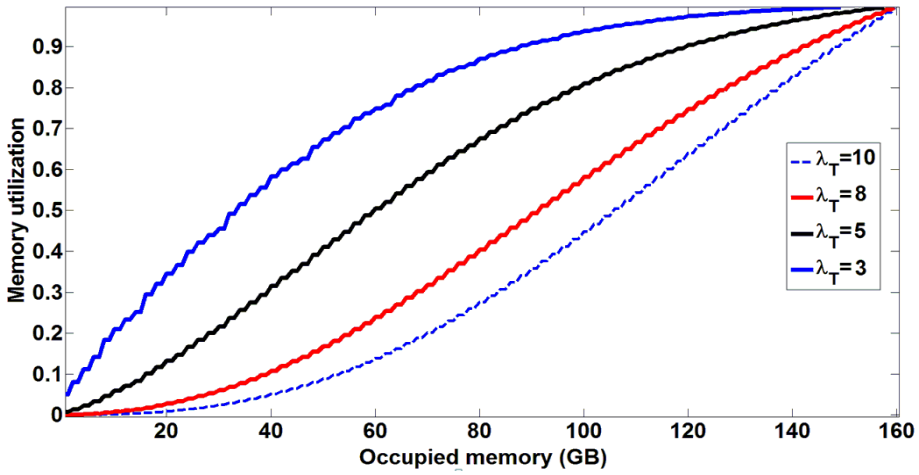

Fig. 3.12 Cumulative Distribution of memory utilization with λ_T as a parameter

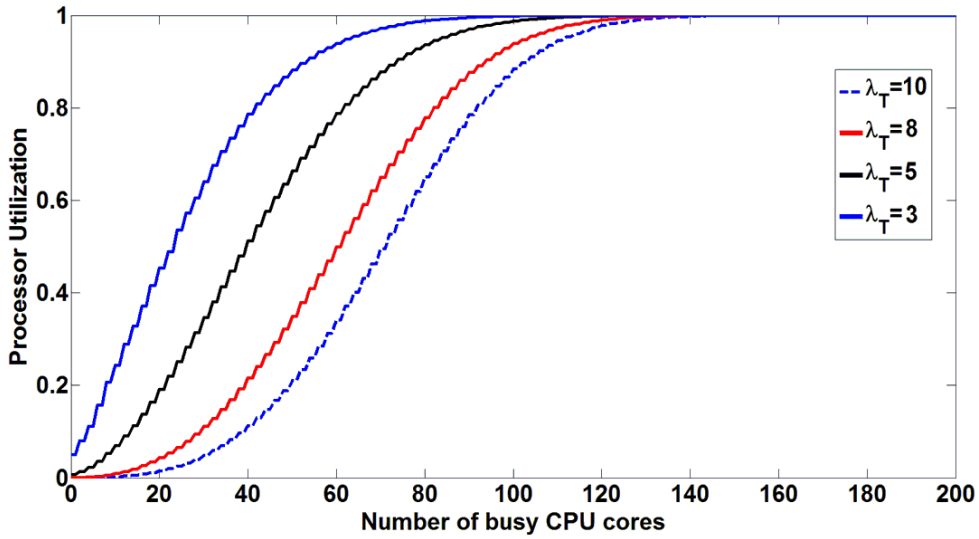

Fig. 3.13 Cumulative Distribution of CPU utilizatiion with λ_T as a parameter

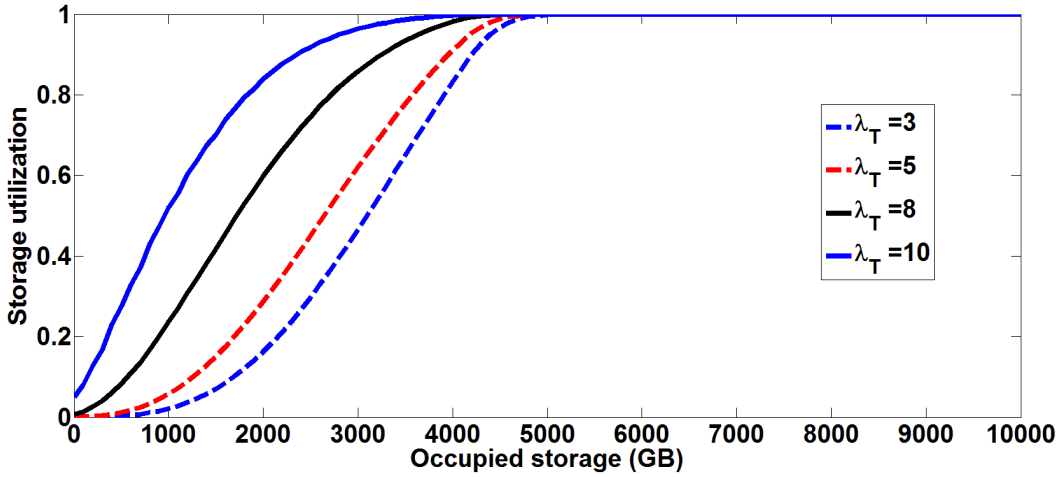

Fig. 3.14 Cumulative Distribution of storage utilization with λ_T as a parameter

Figure 3.15 shows the blocking probabilities of the requests for different types of VMs as a function of the total job arrival rate. As may be seen, VMs differ in their blocking probabilities pertaining to their resource requirements.

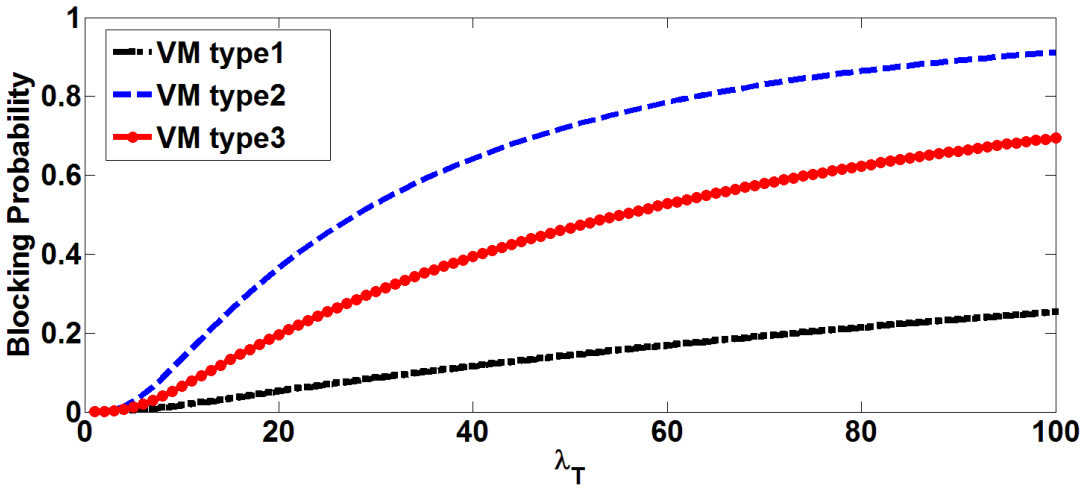

Fig. 3.15 Blocking probabilities of different types of VMs as a function of job arrival rate λ_T

3.4 Modeling of the system with Constant Job size, Homogeneous VMs and Independent Release times (CJ, HM, IR)

In this section, as in section 3.2, we assume constant job sizes with multiple classes as defined in Table 3.1. This model differs from the model of that section in the service given to the tasks. Defining system state as the total number of the tasks in the system, the state-transition rate diagram of the system is given by Fig. 3.16. It is assumed that service times of the tasks of a job are i.i.d with exponential distribution with parameter μ, which results in the independent as opposed to simultaneous task completion times.

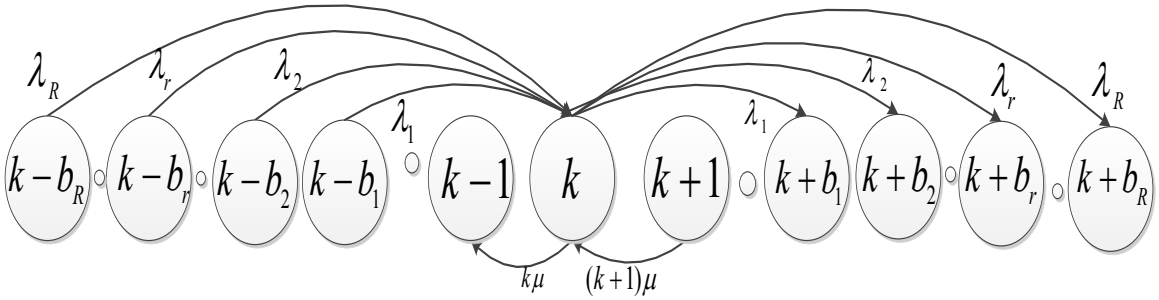

Fig. 3.16 State-transition rate diagram of the Could computing system (independent release Times)

3.4.1 Infinite Resource Model

We first analyze the system for a datacenter with infinite number of servers serving different classes of jobs ($S = \infty, r = 1, \dots R$). Let p_j denote probability that there will be *j* tasks in the system, then equilibrium equations can be written as follows,

$$\begin{cases} (\sum_{r=1}^{R} \lambda_r)\, p_0 = \mu p_1 & j = 0 \\ (\sum_{r=1}^{R} \lambda_r + j\mu) p_j = (j+1)\mu p_{j+1} + \sum_{r=1}^{R} p_{j-r}\lambda_r & j > 0 \end{cases} \tag{3.40}$$

After multiplying by z^j and summing over *j*, we have:

$$\sum_{j=1}^{\infty}(\sum_{r=1}^{R} \lambda_r + j\mu) p_j\, z^j =$$

$$\sum_{j=1}^{\infty}(j+1)\mu p_{j+1}\, z^j + \sum_{j=1}^{\infty}\sum_{r=1}^{R} p_{j-r}\lambda_r\, z^j \tag{3.41}$$

Taking out z^r from the internal summation leads to:

$$\sum_{j=1}^{\infty}(\sum_{r=1}^{R} \lambda_r + j\mu) p_j\, z^j = \sum_{j=1}^{S}(j+1)\mu p_{j+1}\, z^j + (\sum_{r=1}^{R} \lambda_r z^r \sum_{j=r}^{S} p_{j-r}\, z^{j-r}) \tag{3.42}$$

Let us define $\Lambda(z) = \sum_{r=1}^{R} \lambda_r\, z^r$ and $p(z) = \sum_{j=0}^{S} p_j\, z^j$ then with substitution of the variable $j' = j - r$ we obtain:

$$\Lambda(1)(p(z) - p_0) + z\mu p(z) = \mu(p(z) - p_1) + \Lambda(z)p(z) \tag{3.43}$$

This could be simplified to:

$$(\Lambda(1) - \Lambda(z))p(z) + (z-1)\mu p\acute{(}z) = \Lambda(1)p_0 - \mu p_1 \tag{3.44}$$

From (3.40) we find that $\Lambda(1)p_0 - \mu p_1 = 0$. Hence, after solving the first order differential equation $p(z)$ is given by,

$$p(z) = e^{\int \frac{\Lambda(z)-\Lambda(1)}{\mu(z-1)}\partial z} = e^{\frac{-\sum \lambda_r \sum_{i=1}^{r} \frac{1}{i} + \sum \lambda_r \sum_{i=1}^{r} \frac{z^i}{i}}{\mu}} \tag{3.45}$$

where $\frac{\Lambda(z)-\Lambda(1)}{z-1} = \frac{\sum_{r=1}^{R}\lambda_r(z^r-1)}{z-1} = \sum_{r=1}^{R}\lambda_r\,(\sum_{i=0}^{r-1} z^i)$ and the constant part of the PGF is obtained by applying normalization condition $p(z)|_{z=1} = 1$.

The average number and variance of occupied VMs in the system is equal to:

$$E[k_T] = \frac{dp(z)}{dz}|_{z=1} = \frac{\sum_{r=1}^{R} r\lambda_r}{\mu} \tag{3.46}$$

$$VAR_{k_T} = (\frac{d^2p(z)}{dz} + \frac{dp(z)}{dz} - \frac{dp(z)}{dz}^2)|_{z=1} = \frac{\sum_{r=1}^{R} r(\frac{r+3}{2})\lambda_r}{\mu} \tag{3.47}$$

3.4.2 Finite Resource Model

In this subsection, we assume finite resources with *S* VMs and model the system with birth-death processes. GBE of the system may be written as,

$$\begin{cases} (\sum_{r=1}^{R}\lambda_r + j\mu)p_j = (j+1)\mu p_{j+1} + \sum_{r=1}^{R} p_{j-b_r}\lambda_r\,, 0 < j < S \\ \sum_{r=1}^{R}\lambda_r\, p_0 = \mu p_1 \quad , \quad j = 0 \\ S\mu p_S = \sum_{r=1}^{R} p_{S-b_r}\lambda_r \quad , \quad j = S \end{cases} \tag{3.48}$$

The above equations cannot be solved through the transform analysis, but the distribution of the number of busy VMs may be determined from the above recursive equations together with the normalization condition. Then average of the total number of the busy VMs is given by,

$$E[k_T] = \sum_{j=0}^{S} jp_j$$

Let P_{B_r} denote the blocking probability of class *r* jobs, then it is given by,

$$P_{B_r} = \sum_{j=S-r+1}^{S} p_j$$

Next we will determine pdf of the service time of a class *r* job. Let T_r and $f_{T_r}(t)$ this service time and its pdf respectively. Then,

$$T_r = \max(t_1, t_2, \dots, t_j, \dots, t_r)$$

where t_j is the service time of the j^{th} task. Since service times of the tasks are i.i.d. with exponential distribution,

$$Pr(T_r < t) = \prod_{j=1}^{r} P(t_j < t)$$

From the above, the pdf of T_r is given by,

$$f_{T_r}(t) = r\mu e^{-\mu t}(1 - e^{-\mu t})^{r-1}$$

The average service time of a class *r* job is given by,

$$\overline{T_r} = \frac{1}{\mu}\sum_{i=1}^{r} \frac{\binom{r}{i}}{i}(-1)^{i+1} \tag{3.49}$$

Let n_r denote number of class *r* jobs in the system, then from the Little's result its average is given by,

$$E[n_r] = \lambda_r(1 - P_{B_r})\overline{T_r} \tag{3.50}$$

Fig. 3.17 presents probability distribution of the number of busy VMs for a system with four classes of jobs with equal arrival rates with total arrival rate as a parameter for a fixed number of VMs in the system. As may be seen, probability distribution shifts to the right with increasing total arrival rate. Further, the distribution has the largest spread at the medium job arrival rate. Figure 3.18 presents the average number of jobs from each class in the system as a function of the total arrival rate. It may be observed that average of the number of class 4 jobs in the system decreases faster than the other classes with increasing total arrival rate.

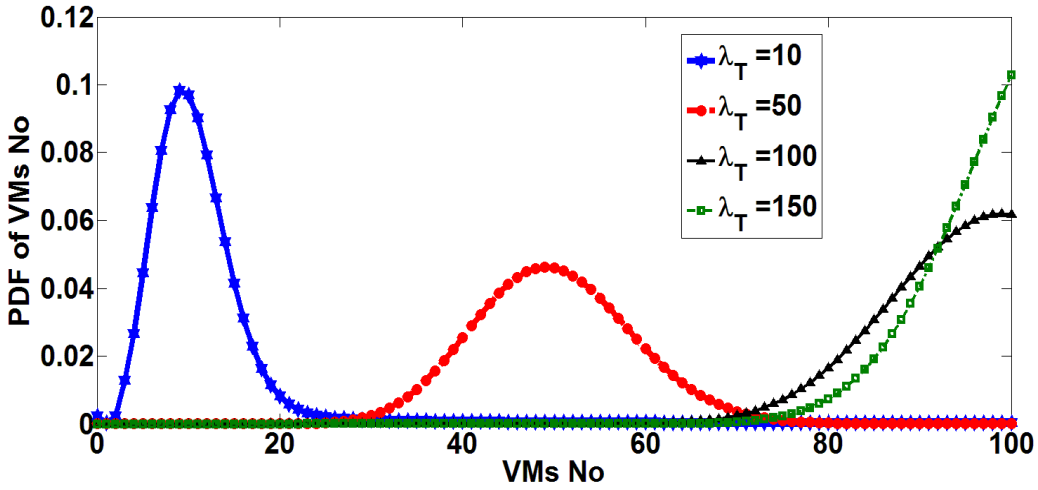

Fig. 3.17 Distribution of busy VMs under low, medium, heavy and very heavy load (*R=4, S =100, μ=1)*

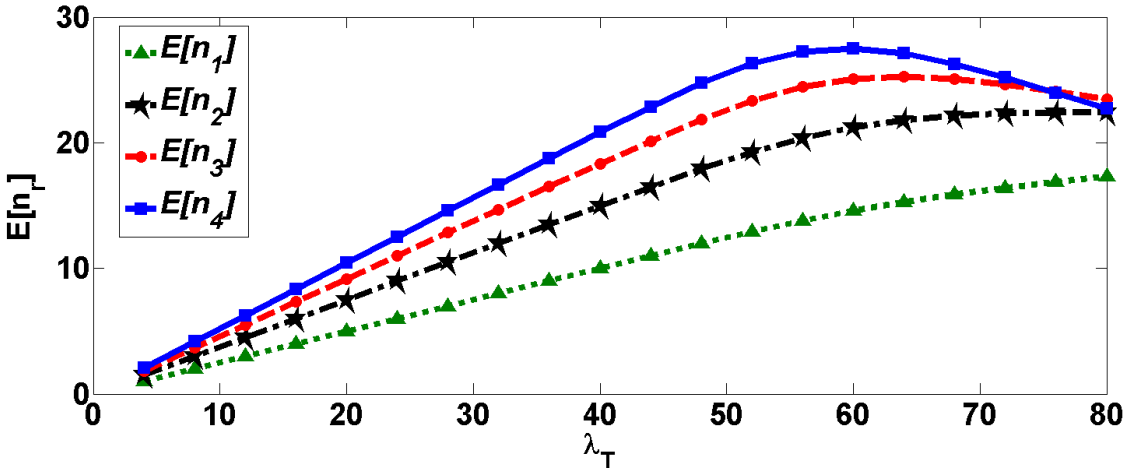

Fig. 3.18 Average number of jobs from each class as a function of the total job arrival rate (*R=4, S=100, μ=1*)

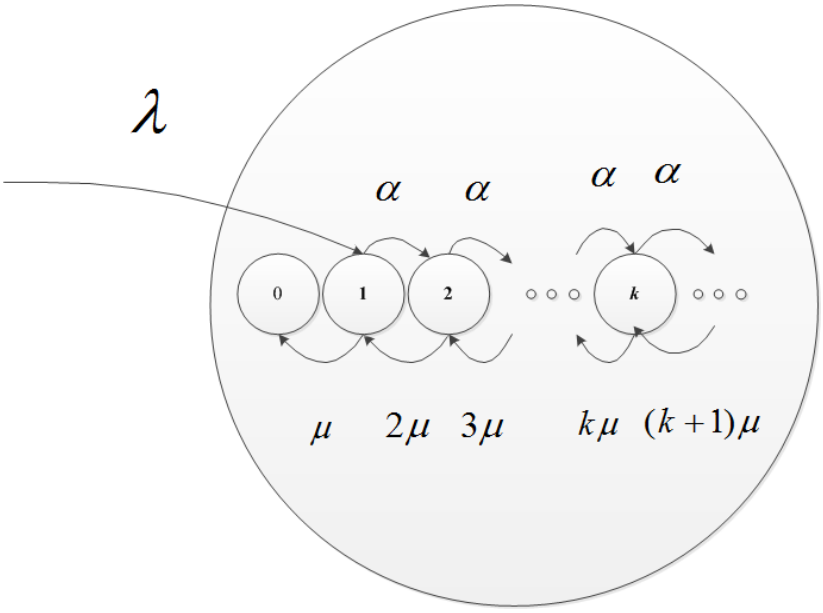

Fig. 3.19 State-transition-rate diagram for the tasks of a job in the system

3.5 Modeling of the system with Variable Job Size (VJ)

In this section, we propose a performance model for systems with dynamic service demand where job size in number of tasks varies during service. As explained in chapter 1, this model will be more appropriate to mobile cloud computing systems. We assume that the size of a job in number of tasks varies randomly during the time that job is in the system. The arrival of the jobs to the system will be according to a Poisson process with parameter λ jobs/sec. We assume that a new arriving job to the system initially demands service for a single task. A job generates random number of tasks according to a Poisson process with parameter α task/job/sec during its service time in the system. We assume that each task requires a VM for its execution and task execution times are exponentially distributed with parameter μ. Service time of a job begins with its arrival to the system and it is completed when there are no more tasks belonging to that job left in the system. Clearly, a job will have a general service type distribution. In this section, a birth-death process is proposed to model this type of cloud computing systems. Figure 3.19 shows the state transition diagram for the tasks of a job in the system. The objective of this analysis is to determine distribution of the number of jobs in the system, service time distribution of a job and average of the total number of tasks. We will consider systems

with both infinite and finite number of VMs.

3.5.1 Infinite Resource model (VD, IR, UJ)

First, we consider infinite resource model where there is always an idle VM available for the execution of each newly generated task to begin immediately. In this case the number of jobs in the system can be modeled as an $M/G/\infty$ queuing system. Next, we will determine main performance measures of this system.

i) Distribution of the number of jobs in the system

Let p_n denote the steady state probability of having *n* jobs in the system and $N(z)$ its probability generating function (PGF). From the results for the $M/G/\infty$ queuing system [59],

$$p_n = \frac{(\lambda\bar{x})^n}{n!} e^{\lambda\bar{x}} \tag{3.51}$$

$$N(z) = e^{-\lambda\bar{x}(1-z)} \tag{3.52}$$

where $\bar{x}$ denotes the average service time of a job which is determined below.

As stated above, each job initially requires service for a single task; however, it generates new tasks according to a Poisson process during its service time in the system. Since we have assumed infinite resource model, each newly generated task immediately begins to receive service. Since task execution times are also exponentially distributed, service time of a job corresponds to the busy period of an $M/M/\infty$ queue, where the number of customers served during the busy period corresponds to the total number of

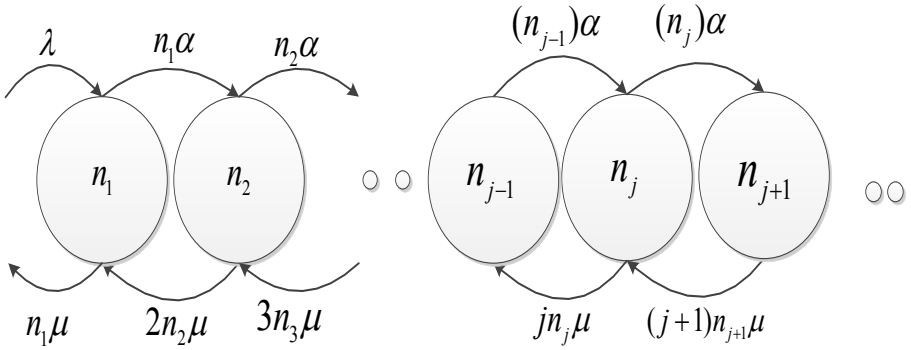

Fig. 3.20 State-transition diagram for the stages of the system

tasks generated by the job. Figure 3.20 shows the state-transition-rate diagram for the tasks of a job in the system. From [60], Laplace transform of the probability distribution of the busy period of an $M/M/\infty$ queue with arrival and service rates of α and μ is given by,

$$B(s) = 1 + \alpha^{-1}(s - (\int e^{-st-\alpha\int_0^t(1-G(v))dv})) \tag{3.53}$$

where $G(v)$ denotes the service time distribution of a task in the system, which has exponential distribution.

Then average service time of a job is given by the mean busy period of $M/M/\infty$ queue,

$$\bar{x} = \frac{e^{\alpha/\mu-1}}{\alpha} \tag{3.54}$$

From the Little's result the average number of jobs in the system is given by:

$$E[n] = \lambda\bar{x} \tag{3.55}$$

ii) Average number of tasks generated by a job during its lifetime in the system

Next, we determine average of the total number of tasks generated by a job during its life-time in the system, which is given by the ratio of average service time of a job to the service rate seen by its tasks in the system. Thus, first, we will determine the service rate seen by the tasks of a job.

Let q_k denote probability that there will be k customers in an $M/M/\infty$ queuing system at the steady-state. From [59], q_k has Poisson distribution given by,

$$q_k = \frac{(\alpha/\mu)^k}{k!} e^{-\alpha/\mu}, k \geq 0 \tag{3.56}$$

Letting q'_k denote probability that there will be k customers at an arbitrary time during a busy period in an $M/M/\infty$ queuing system, then:

$$q'_k = \frac{q_k}{1-q_0}, k \geq 1 \tag{3.57}$$

Let μ_k denote service rate of the tasks of a job, which has k tasks in the system at an

arbitrary time. Since $\mu_k = k\mu$, the average service rate of the tasks generated by a job is given by,

$$\bar{\mu} = \mu \sum_{k=1}^{\infty} k q'_k = \frac{\alpha}{1 - e^{\frac{-\alpha}{\mu}}} \tag{3.58}$$

Defining $\bar{r}$ as the average number of tasks generated by a job during its service time in the system, then it is given by,

$$\bar{r} = \frac{\bar{x}}{\bar{\mu}} = e^{\frac{\alpha}{\mu}} \frac{(1 - e^{\frac{-\alpha}{\mu}})^2}{\alpha^2} \tag{3.59}$$

iii) Joint distribution of the number of jobs in each stage of the system

We define a job to be in stage j if it has j tasks in execution at that time within the system. Let n_j denote number of jobs in stage j at an arbitrary time. Next, we will determine joint distribution of the number of jobs in each stage of the system.

Proposition 3.2. n_j *has a Poisson distribution.*

Proof. Let us define Bernoulli random variable k_{ij} as,

$$k_{ij} = \begin{cases} 1 & i^{th} \ job \ has \ \ j \ tasks \ in \ the \ system \\ 0 & otherwise \end{cases} \tag{3.60}$$

Then, PGF of the distribution of k_{ij} is given by,

$$K_{ij}(z) = q'_j z + 1 - q'_j \tag{3.61}$$

From the above, n_j may be expressed as,

$$n_j = \sum_{i=1}^{n} k_{ij} \tag{3.62}$$

Let $N_j(z)$ denote PGF of the probability distribution of n_j, then,

$$N_j(z) = N(z)|_{z=K_{ij}(z)} = e^{-\lambda \bar{x} q'_j (1-z)} \tag{3.63}$$

where we substituted from (3.56) and (3.61) in the above. The inversion of the above PGF gives,

$$p_{n_j} = \frac{(\lambda \bar{x} q_j)^{n_j}}{n_j!} e^{-\lambda \bar{x} q_j} \tag{3.64}$$

which completes the proof.

Now, we will determine the joint distribution of the number of jobs at each stage of the system. Let state of the system denoted by the vector $\vec{n} = (n_1, \dots, n_i, \dots n_\infty)$. We will show that the joint probability distribution of $\vec{n}$ has a Poisson distribution given by,

$$p(\vec{n}) = \prod_{j=1}^{\infty} \left[\frac{(\lambda \bar{x} q'_j)^{n_j}}{n_j!} e^{-\lambda \bar{x} q'_j} \right] \tag{3.65}$$

Let us define the following vectors that differ from $\vec{n}$ at most in two components by unit value:

$$\vec{n}_j^{+} = (n_1, \dots, n_j^{+}, \dots n_\infty)$$

$$\vec{n}_j^{-} = (n_1, \dots, n_j^{-} \dots n_\infty)$$

$$\vec{n}_{ij}^{+-} = (n_1, \dots, n_i^{+}, \dots, n_j^{-}, \dots n_\infty) \tag{3.66}$$

$$\vec{n}_{ij}^{-+} = (n_1, \dots, n_i^{-}, \dots, n_j^{+}, \dots n_\infty)$$

where $n_j^{+} = n_j + 1, n_j^{-} = n_j - 1$.

Next, we will write the LBEs for the state $\vec{n}$,

$$\begin{cases} jn_j \mu p(\vec{n}) + n_j \alpha p(\vec{n}) = (j+1)\left(n_{j+1}+1\right)\mu \; p\left(\vec{n}_{j,j+1}^{-+}\right) \\ \quad +\left(n_{j-1}+1\right)\alpha p\left(\vec{n}_{j-1,j}^{+-}\right), \quad j > 1 \\ n_1 \mu p(\vec{n}) + n_1 \alpha p(\vec{n}) = 2(n_2+1)\mu \; p(\vec{n}_{12}^{-+}) + \lambda \, \mathrm{p}(\vec{n}_1^{-}), j = 1 \end{cases} \tag{3.67}$$

By means of substitution it can be shown that (3.65) satisfies the LBEs in (3.67) and therefore it is the correct distribution.

iv) Distribution of the total number of tasks in the system

Next, we will determine distribution of the total number of tasks in the system. Let us introduce the following notation,

$r_j = jn_j$

$\vec{r} = (r_1, \dots, r_j, \dots, r_\infty)$

$\vec{z} = (z_1, \dots, z_j, \dots, z_\infty)$

$\vec{n} = (n_1, \dots, n_j, \dots, n_\infty)$

where, r_j corresponds to the total number of tasks that belong to the jobs in stage j. Let us define PGF of the distribution of $\vec{r}$ as,

$$R(\vec{z}) = E[\vec{z}^{\vec{r}}] = E\left[\prod_{j=1}^{\infty} z_j^{r_j}\right] = E[\prod_{j=1}^{\infty} z_j^{jn_j}] = E[\prod_{j=1}^{\infty}(z_j^j)^{n_j}]$$

$$R(\vec{z}) = E[\prod_{j=1}^{\infty}(z_j^j)^{n_j}] \tag{3.68}$$

$$R(\vec{z}) = \sum_{n_1=0}^{\infty} \cdots \sum_{n_j=0}^{\infty} \cdots \sum_{n_\infty=0}^{\infty} \left[\prod_{j=1}^{\infty}(z_j^j)^{n_j}\right] p(\vec{n})$$

Substituting for $p(\vec{n})$ from (3.57),

$$R(\vec{z}) = \sum_{n_1=0}^{\infty} \cdot \cdot \sum_{n_j=0}^{\infty} \cdot \cdot \sum_{n_\infty=0}^{\infty} \left[\prod_{j=1}^{\infty} e^{-\lambda\bar{x}\, q_j'} \frac{(\lambda\bar{x} q_j' z_j{}^j)^{n_j}}{n_j!}\right]$$

Interchanging the order of summations and multiplications,

$$R(\vec{z}) = \prod_{j=1}^{\infty} e^{-\lambda\bar{x}\, q_j'} e^{\lambda\bar{x}\, q_j' z_j^j} = \prod_{j=1}^{\infty} e^{-\lambda\bar{x}\, q_j'(1-z_j^j)}$$

$$R(\vec{z}) = e^{-\lambda\bar{x}\, \sum_{j=1}^{\infty} q_j'(1-z_j^j)} = e^{-\lambda\bar{x}\,(1-\sum_{j=1}^{\infty} q_j' z_j^j)} \tag{3.69}$$

Next let us define k_T as the total number of tasks in the system and $K_T(z)$ as the PGF of its distribution, then,

$$k_T = \sum_{j=1}^{\infty} r_j$$

$$K_T(z) = E[z^{k_T}] = R(\vec{z})|_{z_i=z, i=1,\ldots,\infty} = e^{-\lambda\bar{x}\,(1-\sum_{j=1}^{\infty} q_j' z^j)} \tag{3.70}$$

Substituting in the above from (3.51), (3.70) gives,

$$K_T(z) = e^{-\lambda\bar{x}\left[1-\frac{e^{\frac{-\alpha(1-z)}{\mu}} - e^{\frac{-\alpha}{\mu}}}{1-e^{\frac{-\alpha}{\mu}}}\right]} \tag{3.71}$$

Finally, from the above average of the total number of tasks in the system are given by,

$$E[k_T] = \frac{\lambda\alpha\bar{x}}{\mu(1-e^{\frac{-\alpha}{\mu}})} = \frac{\lambda}{\mu} e^{\frac{\alpha}{\mu}} \tag{3.72}$$

Figure 3.21 presents average of the total number of the tasks in the system as a function of the task arrival rate with job arrival rate as a parameter. Fig. 3.22 presents the average service time of a job with dynamic service time and the independent release time of the previous section from equations (3.54) and (3.49) respectively. We plotted the results for class 3 and 4 jobs for the independent release times. For fair comparison, average of the number of tasks generated by a job with dynamic service time, (3.54), has been set equal to the number of tasks in each class of jobs for the independent release time. Thus for each value of $\boldsymbol{\mu}$, task generation parameter $\boldsymbol{\alpha}$ has been chosen such that $\bar{\boldsymbol{r}} = \boldsymbol{r}$. As may be seen, under these assumptions the average service times of a job in the two models are close to each other.

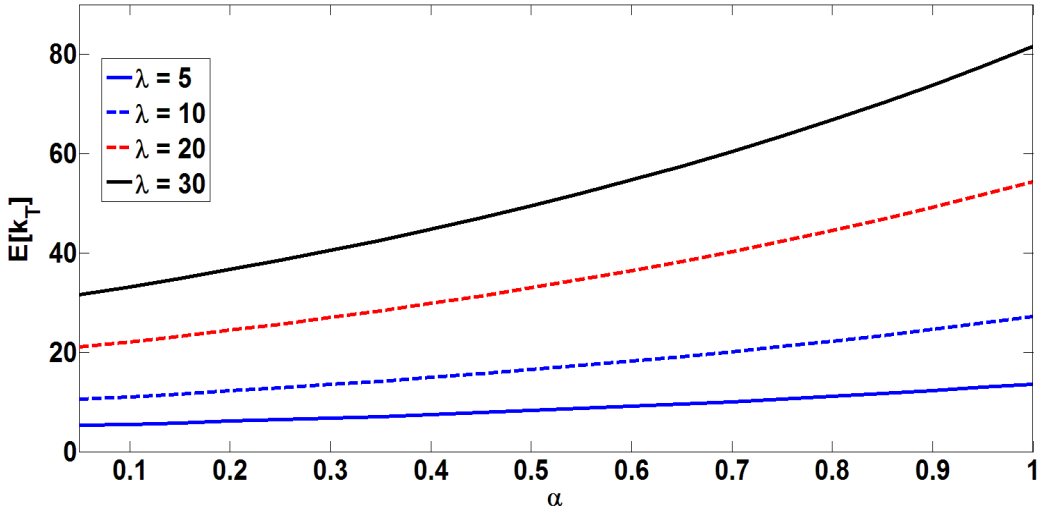

Fig. 3.21 Average of the total number of the tasks as a function of α and λ as a parameter

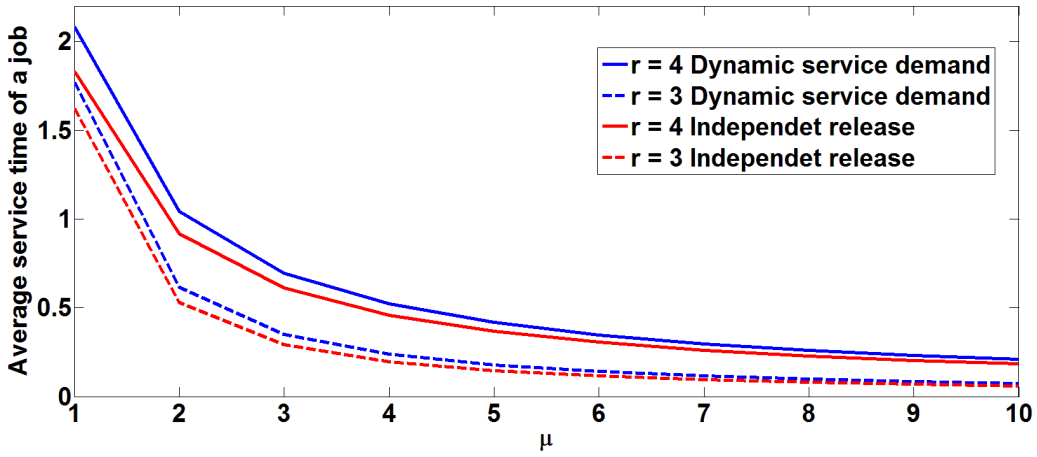

Fig. 3.22. Average service time of a job as a function of task service rate for dynamic service and independent release time models

3.5.2 Finite Resource Model (VJ, IR, UJ)

Next, we consider the finite resource model where the computing center has finite number of VMs given by *S*. A new arriving job will be blocked if all the VMs are occupied. In this model, we assume that each job is assigned a fixed number of VMs, *c*, for its service. When the number of tasks belonging to a job is more than *c*, then the excess tasks are queued. Let us assume that *S* is an integral multiple of *c*, then the number of jobs in the system can be modeled as an *M/G/N/N* queuing system where $N = S/c$.

The service time of a job may be modeled by the busy period of an $M/M/c$ queue, where customers are the tasks generated by the job. The average service time of a job is given by the mean busy period of the $M/M/c$ queue, which is from [61],

$$\bar{x} = \begin{cases} \dfrac{1}{\mu\left(1-\frac{\alpha}{c\mu}\right)} & \text{for } c \leq 2 \\ \dfrac{1}{\alpha}\left[\dfrac{\left(\frac{\alpha}{\mu}\right)^c}{\left(1-\frac{\alpha}{c\mu}\right)c!} + \dfrac{1}{\alpha}\sum_{k=1}^{c-1}\dfrac{\left(\frac{\alpha}{\mu}\right)^k}{k!}\right] & \text{for } c > 2 \end{cases} \tag{3.73}$$

Let *k* denote the number of tasks in the system that belongs to a job, then it may be determined from the distribution of the number of customers in an *M/M/c* queuing system, [61],

$$\Pr(k=i)=\begin{cases}\frac{\Pr(k=0)}{1-\Pr(k=0)}\frac{\left(\frac{\alpha}{\mu}\right)^{i}}{i!} & 0<i\le c\\ \frac{\Pr(k=0)}{1-\Pr(k=0)}\frac{\left(\frac{\alpha}{\mu}\right)^{k}}{c!c^{k-c}} & i>c\end{cases} \tag{3.74}$$

where ,

$$\Pr(k=0)=\left[\sum_{k=0}^{c-1}\frac{\left(\frac{\alpha}{\mu}\right)^{k}}{k!}+\frac{\left(\frac{\alpha}{\mu}\right)^{c}}{c!(1-\frac{\alpha}{c\mu})}\right]^{-1}$$

Let y denote the number of busy VMs from those that assigned to a job, then,

$$\Pr(y=i)=\begin{cases}\Pr(k=i) & 0<i<c\\ \sum_{\ell=c}^{\infty}\Pr(k=\ell) & i=c\end{cases} \tag{3.75}$$

From the *M/G/N/N* queuing results, probability distribution of the number of jobs in the system is given by, [62],

$$p_n=\begin{cases}p_0\frac{(N\rho)^n}{n!} & n<N\\ p_0\frac{(N\rho)^N}{N!} & n=N\end{cases} \tag{3.76}$$

where $p_0=\left[\sum_{j=0}^{N-1}\frac{(N\rho)^j}{j!}+\frac{(N\rho)^N}{N!}\right]^{-1}$ and $\rho=N\lambda\bar{x}$.

We note that blocking probability of a job is given by p_N. Let k_T denote total number of tasks in the system, and then its average is given by,

$$E[k_T]=E[n]E[k] \tag{3.77}$$

The above average needs to be determined numerically from (3.74) and (3.76).

Figure 3.23 shows the average number of VMs in the system as a function of task arrival rate and job arrival rate as a parameter. We assumed that *N*=40 and *c*=10. As illustrated, due to hard limitation on maximum number of tasks of a job, task arrival rate is dominant in creation of VMs compared to job arrival rate. With increasing the job arrival rate, job saturation probability shifts to the left.

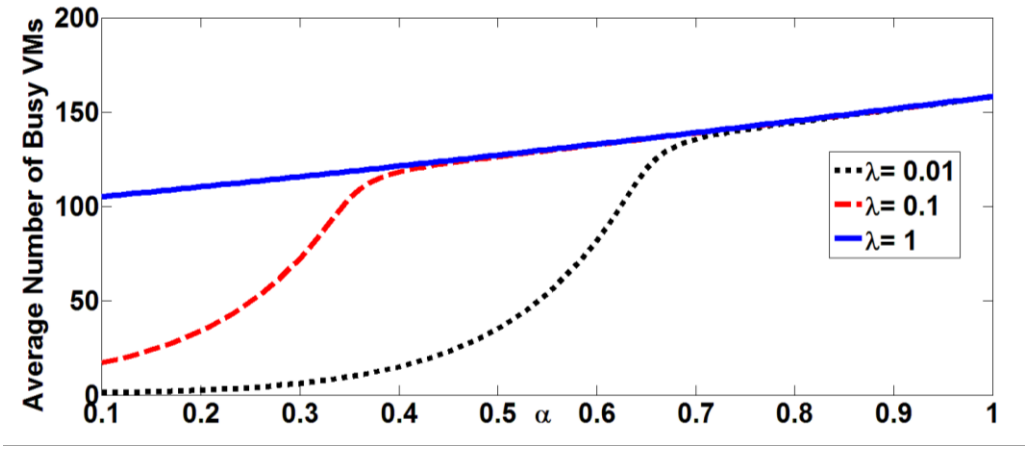

Fig. 3.23 Average number of the VMs as a function of task arrival rate and job arrival rate as a parameter. (*c=10, N=40)*

3.5.3 Saturated job arrival Process (VJ, SJ)

In this part, we consider a system in which there are always *N* jobs in service. When service of a job is completed, a new job immediately enters the system. Also, the new job initially requires service for a single task and generates new tasks according to a Poisson process. The service time of a job remains as in the previous case. The objective of analysis is to determine total number of tasks in the system. Since each job has at least a single task in the system, the minimum number of tasks in the system will be *N*. Defining k_T to be the total number of tasks as the state of the system, and then the system may be modeled as a birth-death process with birth and death coefficients:

$$\alpha_k = N\alpha \quad , \quad \mu_{k_T} = k_T\mu \, , \quad k_T \geq N \, , \tag{3.78}$$

Thus the distribution of the total number of tasks in the system is given by the product form solution,

$$p_{k_T}(k) = p_N \prod_{i=N}^{k-1} \frac{\alpha_i}{\mu_{i+1}}, \quad k \geq N \tag{3.79}$$

Substituting (3.78) in the above,

$$p_{k_T}(k) = p_N \frac{N!}{k!} \left(\frac{N\alpha}{\mu}\right)^{k-N} \, , \quad k \geq N \tag{3.80}$$

where p_N is determined from the normalization condition, $\sum_{k=N}^{\infty} p_{k_T}(k) = 1$,

$$p_N = \frac{1}{\sum_{k=N}^{\infty} \frac{N!}{k!}\left(\frac{N\alpha}{\mu}\right)^{k-N}} = \frac{\left(\frac{N\alpha}{\mu}\right)^N}{N!\left(e^{\frac{N\alpha}{\mu}} - \sum_{i=0}^{N-1} \frac{\left(\frac{N\alpha}{\mu}\right)^i}{i!}\right)} \tag{3.81}$$

Then substituting (3.81) in (3.78) will lead to:

$$p_{k_T}(k) = \left(e^{\frac{N\alpha}{\mu}} - \sum_{i=0}^{N-1} \frac{\left(\frac{N\alpha}{\mu}\right)^i}{i!}\right)^{-1} \frac{\left(\frac{N\alpha}{\mu}\right)^k}{k!}, \quad k \geq N \tag{3.82}$$

From the above, average number of tasks in the system is obtained as:

$$E[k_T] = \sum_{k=N}^{\infty} k\, p_{k_T}(k) = \left(\frac{N\alpha}{\mu}\right)\left(1 + \left(e^{\frac{N\alpha}{\mu}} - \sum_{i=0}^{N-1} \frac{\left(\frac{N\alpha}{\mu}\right)^i}{i!}\right)^{-1} \frac{\left(\frac{N\alpha}{\mu}\right)^{N-1}}{(N-1)!}\right) \tag{3.83}$$

In the finite resource model, we assume that the system has finite number of VMs to execute the tasks, denoted by *S*. In this model, we only consider saturated job arrival process and there will always be *N* jobs in service. Service of a job is completed, whenever a job does not have any more tasks left in the system. Following the service completion of a job, a new job is immediately inserted into the system. The new job also initially requires service for a single task and generates new tasks according to a Poisson process. We note that $S \geq N$ and maximum number of the tasks that can be executed simultaneously equals to *S*. When number of tasks in the system exceeds *S* the remainder will be queuing. The objective of the analysis is again to obtain the total number of tasks, k_T, in the system. Here, we define the system state as total number of tasks currently in the system. We model the system as a birth-death process with the following coefficients,

$$\alpha_k = N\alpha\,, \qquad \mu_{k_T} = \begin{cases} k_T\mu & N \leq k_T \leq S \\ S\mu & k_T > S \end{cases} \tag{3.84}$$

The distribution of the total number of tasks in the system is given by the product form solution in (3.79). Substituting from (3.84) in (3.79),

$$p_{k_T}(k) = \begin{cases} p_N \prod_{j=1}^{k-1} \frac{N\alpha}{(i+1)\mu} = p_N \frac{N!}{k!}\left(\frac{N\alpha}{\mu}\right)^{k-N}, & N \le k < S \\ p_N \prod_{i=N}^{S} \frac{N\alpha}{(i+1)\mu} \prod_{i=S+1}^{k} \frac{N\alpha}{S\mu} = p_N \frac{N!}{S!} \frac{\left(\frac{N\alpha}{\mu}\right)^{k-N}}{S^{k-S}}, & k \ge S \end{cases} \tag{3.85}$$

Normalization condition gives p_N as,

$$p_N = \frac{1}{\sum_{k=N}^{S} p_N \frac{N!}{S!}\frac{\left(\frac{N\alpha}{\mu}\right)^{k-N}}{S^{k-S}} + \sum_{k=S}^{\infty} \frac{N!}{k!}\left(\frac{N\alpha}{\mu}\right)^{k-N}} = \frac{\left(\frac{N\alpha}{\mu}\right)^{N}}{N!\left(\sum_{i=N}^{S} \frac{\left(\frac{N\alpha}{\mu}\right)^{i}}{i!} + \frac{S^S}{S!}\sum_{i=S}^{\infty}\left(\frac{N\alpha}{S\mu}\right)^{i}\right)} \tag{3.86}$$

Finally, average number of tasks in the system is given by,

$$E[k_T] = \sum_{k=N}^{\infty} k\, p_{k_T}(k) = \frac{N\alpha}{\mu}\left(1 + p_N \frac{\left(\frac{N\alpha}{\mu}\right)^{S}}{(S-1)!\left(1-\frac{N\alpha}{S\mu}\right)^{2}}\right) \tag{3.87}$$

Next, we present numerical results for variable service demand models.

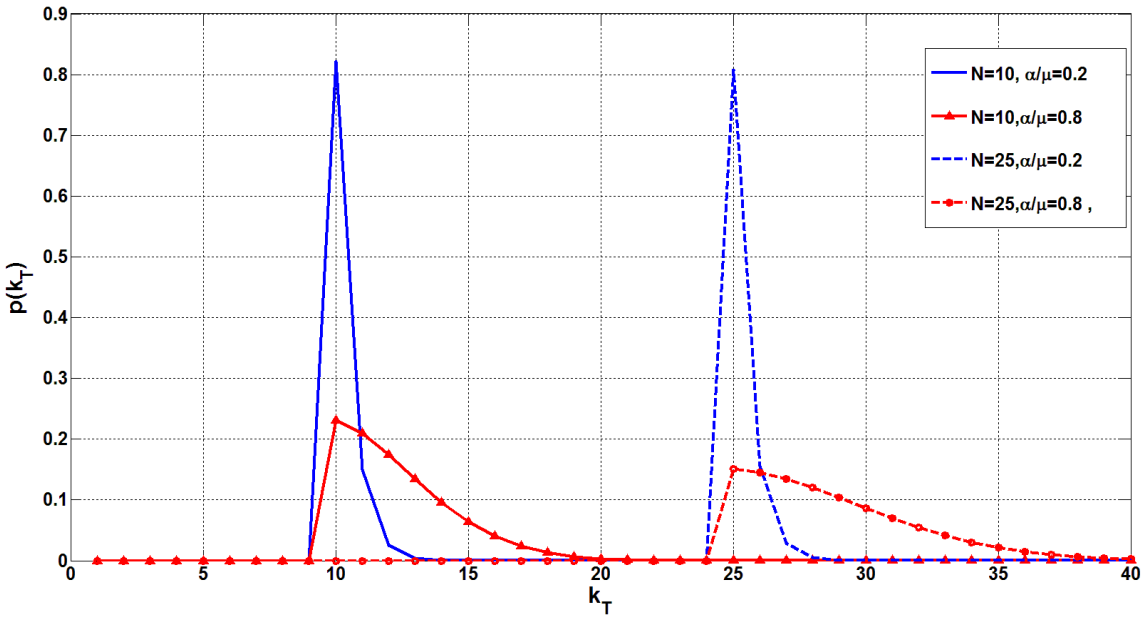

Fig 3.24 Probability distribution of number of tasks in the system with N and α as the parameters

Figure 3.24 illustrates the distribution of total number of tasks in the system with N , α as parameters. Note that, the spread of the probability distribution increases with the

growth of task arrival rate while probability distribution shifts to the right with increasing number of jobs.

Figures 3.25, 3.26 show the average number of tasks in the infinite and finite resource models respectively as functions of task arrival rate with number of jobs as a parameter. As it is shown in Fig. 3.25, when task arrival rate increases, the average number of tasks in the system will increase. However, for larger values of α this growth is more tangible. Moreover, for larger number of jobs in the system, the total number of tasks in the system is higher. Figure 3.26 also indicates that when task arrival rates increase, until the system is saturated, the total number of the tasks in the system will also increase. In addition, for low task arrival rates, the average number of tasks in the system will remain almost the same for different job arrival rates.

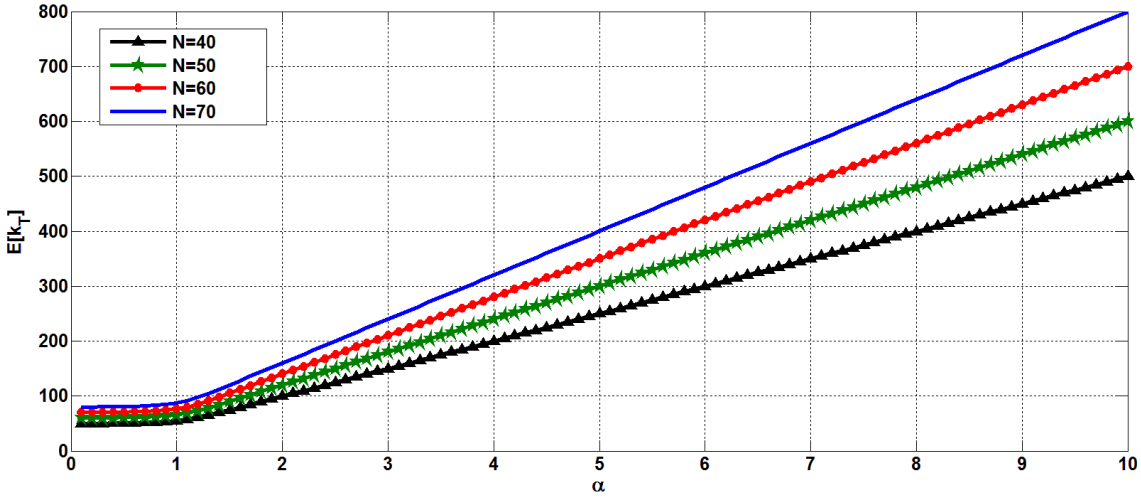

Fig. 3.25 Average of the total number of tasks for infinite resource model with saturated job arrival process as a function of task arrival rate and number of jobs, *N*, as a parameter and $\mu = 1$.

Figure 3.27 also compares the average number of tasks from several modeling perspectives namely saturated, unsaturated infinite resource models and finite resource models. As illustrated, the average number of tasks in unsaturated infinite resource model is much higher than in the other two. $E[k_T]$ associated with infinite server model under unsaturated job arrival process is also included in the figure with $E[n] = \lambda\bar{x} = N$ given in

(3.55). N is assumed to be a constant, thus for all values of α, λ is determined such that average number of jobs in the system remains fixed.

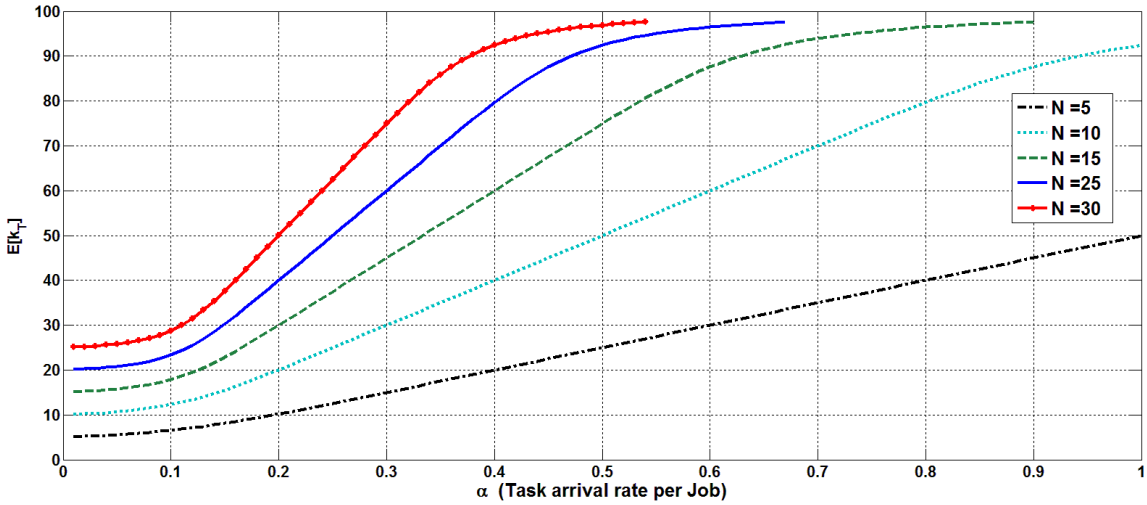

Fig. 3.26 Average number of tasks in finite resource model as a function of task arrival rate and number of jobs, N, as a parameter and $\mu = 1, S = 100$.

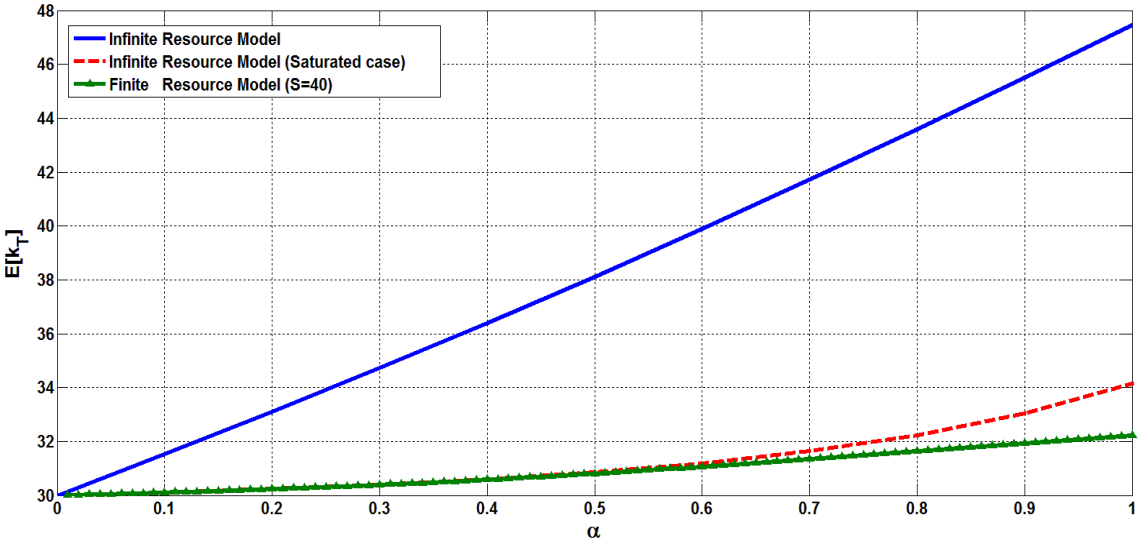

Fig. 3.27 Comparison of the average number of tasks for saturated and unsaturated infinite resource model and finite resource model (S=40) as a function of task arrival rate ($\mu = 1, N = 30$).

3.6 Comparison of the Performance Modeling Results with the Previous Work

In this section, we give a comparison of the performance analysis of cloud computing systems developed in this chapter with the previous work that has been introduced in the previous chapter. There is an overlap between the work in this chapter and that in [11-13], though we have studied several more models not considered in those works.

In [11], a cloud computing center has been modeled as a $M/G/m/m+r$ queue, where m is the number of VMs in the system and r is the size of the buffer that stores the waiting jobs. A new arriving job to a full buffer is lost and the jobs in the buffer are served on FCFS basis. It is assumed that each job requires a single VM for its execution. The steady-state distribution of the queue length is determined by writing down the transition probability matrix of the embedded Markov chain at the arrival points. The analysis makes the approximation that at most three jobs may be served during an inter-arrival time. The equilibrium equations had to be solved numerically, thus the queue length distribution could not be obtained in a closed form. This model corresponds to our single server model with one class of jobs, when no buffering is allowed, $r=0$. In Fig. 3.28, we plot average number of busy VMs for both our and their model under the assumption of no buffering, $r=0$, as a function of the job arrival rate. The results have been plotted both for exponential and deterministic service times. As may be seen, the approximate results of [11] are very close to our exact results.

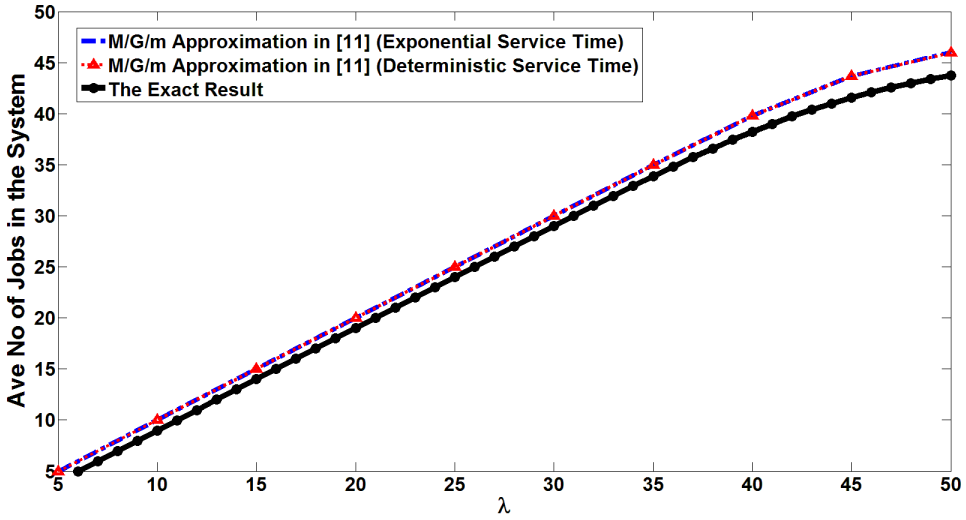

Fig. 3.28 Average number of the jobs in the system as a function of job arrival rate for M/G/m approximation and the exact results for $\mu = 1$.

In [12], [13], the analysis in [11] has been extended to the jobs where each job contains random number of tasks and execution of each task demands a VM. In this model, the tasks of a waiting job are stored in the buffer with each task occupying one position. All the tasks of a job need to start execution simultaneously. If the tasks of a new arriving job cannot be served immediately and there is no enough storage in the buffer to store all the tasks, then that job is rejected. Since jobs are still served on a FCFS basis, this results in head-of-line (HOL) blocking until enough servers become available to serve the HOL job. Service times of the tasks are i.i.d with a general distribution, thus the tasks of a job have independent release times. Letting number of tasks to denote the system state, then the system has been analyzed by embedding a Markov chain at the job arrival points. Similar to the original model, it is assumed that a VM cannot serve more than three tasks during a job interarrival time. The transition probabilities are determined assuming constant number of tasks in a job, which needs to be unconditioned numerically afterwards. Further, an important weakness of the analysis is that the probabilities involving transitions from a state with number of idle VMs require knowledge of the distribution of the idle VMs, which is part of the solution that is being determined. The distribution of the number of idle VMs had to be determined through simulation. After determination of the transition probability matrix, which is quite tedious, the equilibrium

equations have been solved numerically. This model becomes identical to our model for a system with multiple classes of jobs and independent task release times under the assumptions of exponential task service times and no buffering, $r = 0$ (section 3.4). For this case, we also determine distribution of the service time of a job and average number of jobs from each class in the system (latter is plotted in Fig. 3.14), which are not available in [24]. We note that our single server models apply to systems with multiple job classes and simultaneous task completion times for both homogeneous and heterogeneous VMs under no queuing assumption. The joint distribution of the number of jobs is presented in equations (3.1) and (3.25) for homogeneous and heterogeneous VMs cases respectively.

In [14], the performance of cloud computing systems has been studied using stochastic reward networks (SRNs). It is assumed that cloud center has N servers that may support upto M VMs where $N \geq M$. The arrival of the jobs is either according to a homogeneous Poisson process or a Markov Modulated Poisson Process (MMPP) which allows time variations in the arrival rate. It is assumed that each job requires a single VM for its execution and service times are exponentially distributed. However, mean service time is a function of the number of busy VMs on a server. The system has a finite queue, which is managed according to the FCFS discipline and a job arriving to a full queue is lost. The models of [14] and [11] become identical for Poisson arrivals and exponentially distributed service times with a constant mean value, when number of servers and number of VMs are equal to each other, $N=M$. For this case, the two models have been compared in [14] and the presented numerical results show very close agreement. This also means that our results agree with that of [14] for the case of single task per job scenario with no buffering, since all the three models become same for this special case. The main weakness of the model in [14] is that it is numerical and lacks closed form results.

In [15], performance of cloud computing systems has been studied considering fault recovery. It is assumed that arrival of jobs is according to a general stochastic process and each job has random number of tasks. The system has a server with S VMs and each task requires a VM for its execution. Task service times are i.i.d with exponentially distribution, which results in independent task completion times. The system has a finite

queue and each task of a job occupies a position in the queue. A job is lost if not all of its tasks can be accepted to the system. The system has been modeled as $GI^X/M/S/N$ queue where N corresponds to the maximum number of allowed tasks in the system. The steady-state probability distribution of the number of tasks in the system is determined by writing down the transition probability matrix for the embedded Markov chain and solving numerically the equilibrium equations. We note that, the analysis doesnot result in the distribution of the number of jobs in the system. For fault modeling, it is assumed that VMs fail according to a Poisson process and VM recovery times are exponentially distributed. Following recovery, the execution of a task resumes from the point of failure. Under the approximation that all the tasks of a job begin receiving service simultaneously, job service times have been determined. However, probability distribution of the number of tasks in the system with fault tolerance could not be obtained because the queuing model only allows exponential service times. Again, this model under the assumption of Poisson arrival of jobs with no queuing and simultaneous service completion of the tasks of a job corresponds to our single server model with single class of jobs (section 3.2.1). Simultaneous service completion means that whenever a VM assigned to a task fails, all the tasks belonging to the job as the failed task are also delayed until recovery. Then for fault tolerance scenario, our model gives the distribution of the number of jobs in the system from equation (1), since the analysis applies for any service time distribution. Let μ,γ denote parameters of the exponential distributions for service and recovery times respectively and α parameter of the Poisson distribution for failure. Then, mean service time of a job is given by,

$$\bar{\bar{b}} = \frac{\alpha + \gamma}{\mu\gamma}$$

In Fig. 3.29, we plotted average number of jobs in the system as a function of the job arrival rate λ with α as a parameter for constant values of μ,γ and S. It is assumed that number of tasks per job is four. As may be seen, average number of jobs in the system increases with increasing value of VM failure rate at any job arrival rate.

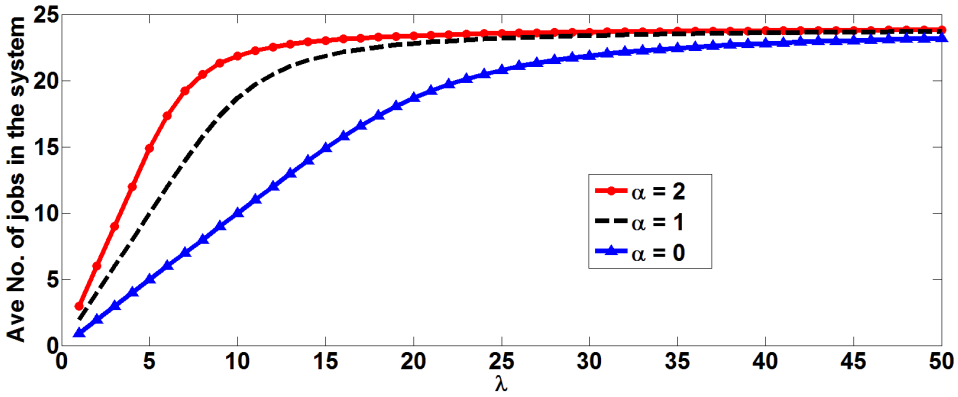

Fig. 3.29 Average number of the jobs in the system as a function of job arrival rate with α as a parameter for $= \gamma = 1$, $S = 100$ and four tasks per job.

The optimal resource allocation approach and its objective in this thesis also is similar to works in [22] and [23], except for the solutions methods. [22] and [23] proposed heuristic method to find the solution while column generation technique is proposed in this thesis to find the optimal resource allocation. Further, similar to [39] our analysis allows heterogeneous VMs with different resource requirements, while [22] considered the problem into a one dimensional resource type. We also develop a technique for the optimal allocation of the resources as a function of the time under a stochastic job arrival process with and without migration of the VMs belonging to incomplete jobs.

3.7 Conclusion

In this chapter, we have studied performance modeling of cloud computing systems. We have derived joint distribution of the number of jobs from each class in the system, job blocking probabilities and distribution of the utilization of resources as a function of the traffic load under various scenarios for systems with both homogenous and heterogeneous VMs. We have shown that joint distribution of the number of jobs depend on the service time only through its mean. We have determined service fragmentation probabilities and have shown application of the derived results in power management techniques under time-varying loads. We have obtained results for systems that resource requirements of jobs may vary dynamically during their service times, which may be appropriate to mobile cloud computing environment. The derived results advance the

state-of-the-art on performance modeling of cloud computing systems and they will be useful in dimensioning of cloud computing systems.

Chapter 4

Job Scheduling with Optimization of Power Consumption in Cloud Computing Centers

In this chapter, we propose an optimization model for VM placement in the cloud computing centers. The VM placement scheduler should minimize the power consumption of servers and inter-VMs communications.

In previous chapter, performance of the cloud computing models is determined under stochastic job arrival process and job service time distribution. The objective of that analysis was determining equilibrium distribution of the number of jobs in the system, job blocking probabilities, response time for different classes of jobs and distribution of the resource utilization. In this chapter, given the number of jobs and occupied VMs in the system, we develop an optimization model that determines the job schedule, which minimizes the total power consumption of a cloud computing center

The problem of VM placement for power minimization is NP-hard [22]. Due to similarity between our optimization problem and cutting stock problem, we utilized column generation (CG) technique to solve this large scale optimization problem. Moreover, initialization and heuristic termination algorithms are also proposed to mitigate the complexity of the optimization problem. The model also has been extended to the case where communication rate and computation level are random variables to make the model more realistic.

The remainder of this chapter is organized as follows: Job scheduling with optimization of power consumption is defined in Section 4.1. In Section 4.2, we extend the optimization problem of Section 4.1 to include communication network infrastructure and bandwidth constraints in order to have a more realistic model. Section 4.3 discusses

the probabilistic model. Dynamic job scheduling with optimization of power consumption as a function of time is studied in Section 4.4. Complexity order analysis followed by initialization, and heuristic rounding algorithm for finding the ILP solution from the relaxed LP solution is discussed in section 4.5. Section 4.6 discusses the numerical results and section 4.7 presents conclusion.

4.1 Job Scheduling with Optimization of Power Consumption in a Cloud Computing Center

In this section, we will develop an optimization problem that determines the job schedule, which minimizes the total power consumption of a cloud computing datacenter. It is assumed that power consumption in a datacenter is due to communications and server operations. We assume a distributed model, where a job may be assigned VMs on different servers. There will be a need for communications among the VMs assigned to a job on different servers. This demand will be proportional to the product of the number of VMs assigned to each job on each pair of servers. We assume a server will be on if it has at least one VM assigned to at least one of the jobs and otherwise it will be *off*. It will be assumed that an on server consumes constant power and an *off* server zero power. In this optimization problem, we will ignore power consumption of communication network infrastructure since that is topology dependent. However, in the follow up section, optimization problem will be expanded to include this power consumption for a hierarchical network topology.

We assume that a datacenter that has T types of servers, where each server type is determined by the amount of different types of resources that it contains. A server type may have K different types of resources such as bandwidth, storage, CPU and memory. A unique resource vector determines the amount of resources that each server type has. We let M_t denote number of type t servers in the datacenter, $t \in \{1, \dots, T\}$. Power consumption of an on type t server will be denoted by Q_t and otherwise it will be zero. We assume that a server may have R different VM configurations. Each VM configuration is determined by the amount of different types of resources that it contains. We let i_r^k denote the type k resource requirement of type r VM. We assume that there are

H types of jobs, where each job type requires a random number of VMs from a group of VM types and it has geometrically distributed service time with a different parameter. We let N_h denote number of type h jobs in the datacenter, $h \in \{1, \dots, H\}$. Let also $v_{n_h}^r$ denote the number of type r VMs that job n_h requires, $n_h \in \{1_h, \dots, N_h\}$. N is defined as the number of jobs in the datacenter, then, $N = \sum_{h=1}^{H} N_h$. The notation for this optimization problem has been summarized in Table 4.1.

Table 4.1 Parameter/Variable Definitions

Parameters	Definitions
R	number of VM types
N	number of jobs in the datacenter
K	number of resource types
T	number of different types of servers.
M_t	total number of type t servers
Q_t	power usage of type t servers
$v_{n_h}^r$	number of type r VMs required by job n_h
c_t^k	type k resource capacity of a type t server
P_{n_h}	power usage rate of communicating with a server serving VMs of job n_h.
i_r^k	Amount of type k resource required by a type r VM
Variables	**Definitions**
$x_{r,n_h}^{m_t}$	number of type r VMs in m^{th} type t server assigned to job n_h.
$\tilde{x}_{n_h}^{m_t}$	number of VMs in m^{th} type t server assigned to serve job n_h.
y_{m_t}	binary variable denote *on* or *off* status of m^{th} type t server.

The total power will be minimized if the job load is served by minimum number of servers and each job is assigned VMs from as few servers as possible. Next, optimization problem using IQP and CG will be introduced. First, IQP is used to model the optimization problem. Then, CG will be introduced to solve the optimization problem with less complexity.

4.1.1 Integer Quadratic Programming Model

In this subsection, we will develop an IQP model for the optimization problem

described in the above. We assume that communication power consumption between two VMs assigned to a job depends on the type of job but not on the types of VMs. We let P_{n_h} denote communication power consumption between two VMs assigned to the job n_h. The scheduling variable $x_{r,n_h}^{m_t}$ represents number of type *r* VMs in m^{th} type *t* server assigned to serve job n_h, where $m_t = 1_t..M_t$. We are interested in finding optimal values of $x_{r,n_h}^{m_t}$s that minimize the DC power consumption. Similarly defining connectivity variable $\tilde{x}_{n_h}^{m_t}$ as number of VMs assigned to job n_h on the m^{th} type *t* server. Then, the optimization problem is given by,

$$Min\ \sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h} P_{n_h}\ \sum_{t=1}^{T}\sum_{m_t=1_t}^{M_t}\left[\sum_{t'=1}^{T}\sum_{m_{t'}'=1_{t'}}^{M_{t'}}\left(\tilde{x}_{n_h}^{m_t}\tilde{x}_{n_h}^{m_{t'}'}\right)-\left(\tilde{x}_{n_h}^{m_t}\right)^2\right]+$$

$$\sum_{t=1}^{T} Q_t \sum_{m_t=1_t}^{M_t} y_{m_t} \tag{4.1}$$

Subject to.

$$\tilde{x}_{n_h}^{m_t} = \sum_{r=1}^{R} x_{r,n_h}^{m_t} \qquad \forall\, n_h \in \{1,\dots,N_h\}, m_t \in \{1_t,\dots,M_t\} \tag{4.2}$$

$$\sum_{t=1}^{T}\sum_{m_t=1_t}^{M_t} x_{r,n_h}^{m_t} \geq\ v_{n_h}^{r} \quad \forall\, r \in \{1,\dots,R\}, n_h \in \{1_h,\dots,N_h\}, h \in \{1,\dots,H\} \tag{4.3}$$

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\sum_{r=1}^{R} x_{r,n_h}^{m_t} i_r^k \leq\ c_t^k \ \ \forall\, k \in \{1,\dots,K\}, m_t \in \{1_t,\dots,M_t\} \tag{4.4}$$

$$y_{m_t} = \begin{cases} 1 & \sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} > 0 \\ 0 & \sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} = 0 \end{cases} \tag{4.5}$$

where y_{m_t} denotes *on* and *off* status of m^{th} type *t* server.

In the objective function the first and second terms correspond to communications and server power consumptions of the datacenter respectively. Constraint group (4.3) ensures that VM requirements of each type of job are satisfied and group (4.4) guarantees that resource demands of jobs scheduled on a server do not exceed that server's resource capacities. In the above optimization problem, objective function is quadratic and constraints group (4.5) is nonlinear. We would like to simplify the optimization problem to IQP with linear constraints by converting nonlinear constraints in (4.5) to linear. This can be achieved by replacing the constraints in (4.5) by the following two pairs of linear constraints for all servers,

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} - y_{m_t} \geq 0 \tag{4.6}$$

$$\theta y_{m_t} - \sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} \geq 0 \quad (4.7)$$

That implies $\sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} = 0 \Leftrightarrow y_{m_t} = 0$ and $\sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \tilde{x}_{n_h}^{m_t} > 0 \Leftrightarrow \mathrm{y}_{m_t} = 1$. θ denotes an integer much larger than the maximum value of the above positive integer. For the remainder of the Chapter, constraints in (4.5) will be replaced by the constraints in (4.6) and (4.7), which will be referred as "positive integer to binary linear conversion constraints" (IBLC). As a result of the replacement, the above optimization problem may be expressed as,

Min (4.1)

$ST.$ (4.2) - (4.4), (4.6) and (4.7)

We note that $m_t \in \{1_t, \dots, M_t\}$ stands $\forall\ t\ \in \{1, \dots, T\}$. In the above ST is the abbreviation for "Subject To" .

It should be noted that solution of the IQP model gives exact results.

4.1.2 Column Generation Model

The optimization problem, which has been developed in the previous subsection is NP hard. For large scale datacenters, finding the global optimum point of the IQP becomes overly complex and time consuming. In this subsection, we will use column generation technique to provide an alternative solution to our problem. This technique originally had been applied to cutting-stock problem. In column generation approach, the optimization problem is divided into restricted master and pricing problems [63], [79]. The Restricted Master Problem (RMP) determines if the explored patterns satisfy the job demand constraints. The pricing problem finds a new pattern to feed the RMP. The objective function of the pricing problem is in fact the reduced cost coefficient of the RMP. The RMP and pricing problems collaborate until reduced cost coefficients (objectives) of the pricing problems are negative indicating optimal solution has been reached. In our problem, there are *T* pricing problems, one for each server type.

Let us define a pattern as a distinct combination of number of VMs from each type of VMs that a server can accommodate. Let j_t denote such a pattern and J_t total number of patterns available for a type *t* server, then $j_t \in \{1_t, \dots, J_t\}$. The new introduced notation is explained in Table. 4.2. We also define, m_{j_t} as the number of times pattern j_t is used in

scheduling of type *t* servers. Let also $x_{r,n_h}^{j_t}$ denote number of VMs of type *r* that has been assigned to job n_h by pattern j_t.

Table 4.2 Parameter/Variable Definitions for CG

Parameters	Definitions
J_t	Total number of configuration patterns collection of type *t* servers
Variables	**Indicator**
m_{j_t}	number of active server type t with pattern j_t
$x_{r,n_h}^{j_t}$	number of VM type *r* of job n_h over the server type *t* in pattern j_t
$\tilde{j}_t$	Pattern $\tilde{j}_t$ of type t servers introduced by pricing problem *t*.
$\tilde{x}_{n_h}^{j_t}$	number of VMs of job n_h over the server type *t* in pattern j_t

Similarly, $\tilde{x}_{n_h}^{j_t}$ denotes total number of VMs assigned to job n_h by pattern j_t. Then, we have the following equality between the two variables,

$$\tilde{x}_{n_h}^{j_t} = \sum_{r=1}^{R} x_{r,n_h}^{j_t} \tag{4.8}$$

Number of communication links of type *t* servers with pattern j_t dedicated to job n_h, is given by,

$$\left[\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_t}\left(m_{j_t}\tilde{x}_{n_h}^{j_t}.m_{j'_{t'}}\tilde{x}_{n_h}^{j'_{t'}}\right)-m_{j_t}\left(\tilde{x}_{n_h}^{j_t}\right)^2\right]$$

Then, the optimization for the RMP is given by,

$$Min\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}P_{n_h}\ \sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t}\left[\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_t}\left(m_{j_t}\tilde{x}_{n_h}^{j_t}.m_{j'_{t'}}\tilde{x}_{n_h}^{j'_{t'}}\right)-m_{j_t}\left(\tilde{x}_{n_h}^{j_t}\right)^2\right]+$$
$$\sum_{t=1}^{T}Q_t\sum_{j_t=1_t}^{J_t}m_{j_t} \tag{4.9}$$

$ST.$ (4.8)

$$\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t}x_{r,n_h}^{j_t}m_{j_t}\geq v_{n_h}^{r}\ \ \forall\, r\in\{1,\dots,R\}, n_h\in\{1_h,\dots,N_h\}, h\in\{1,\dots,H\} \tag{4.10}$$

$$\sum_{j_t=1_t}^{J_t}m_{j_t}\ \leq\ M_t\quad \forall\, t\in\{1,\dots,T\} \tag{4.11}$$

In the above objective function, first and second terms correspond to VMs

communication and server power consumption respectively. Constraint group (4.10) ensures that VM requirements of jobs are satisfied. Constraint (4.11) verifies that number of needed patterns for a server type does not exceed number of servers of that type.

Next, we present the *T* pricing problems one for each server type. The pricing problem for server type *t* attempts to introduce the new pattern $\tilde{j}_t$ to the RMP.

$$Max \sum_{n=1}^{N} \sum_{r=1}^{R} u_{r,n_h}^{t} (x_{r,n_h}^{\tilde{j}_t}) \tag{4.12}$$

$$ST. \ \sum_{n=1}^{N} \sum_{r=1}^{R} x_{r,n_h}^{\tilde{j}_t} i_r^k \ \leq c_t^k \ \forall k \in \{1, \dots, K\} \tag{4.13}$$

Where $x_{r,n_h}^{\tilde{j}_t}$ represents number of type *r* VMs assigned to job n_h by pattern $\tilde{j}_t$. The pricing problem's objective function is the reduced cost function of the RMP with respect to server type *t*. u_{r,n_h}^{t} denotes the dual variables of the RMP for type *t* server. Constraint groups (4.13) ensure resource constraints of the servers are satisfied.

In the column generation technique, RMP and pricing problems are solved iteratively. In each iteration, a new pattern for each server type will be introduced to the RMP. The new pattern maximizes the objective function of the pricing problem for that server type. The iterations continue, as long as there are reduced cost functions with positive values. The algorithm terminates when all the reduced cost functions are negative and no new pattern is introduced to the RMP.

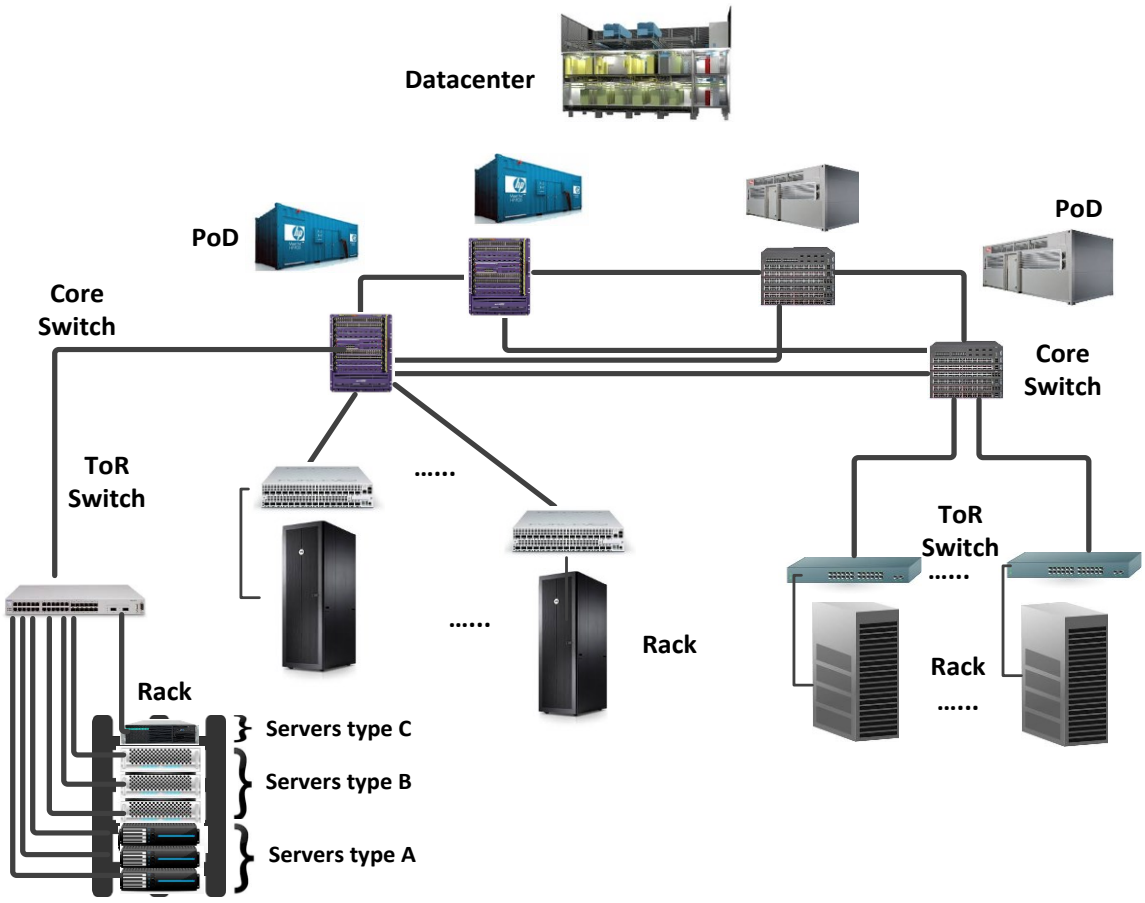

Fig. 4.1 Hierarchical Architecture of a Datacenter

4.2 Job Scheduling with Power Consumption Optimization including Network infrastructure

In this section, we extend the optimization problem of the previous section to include communication network infrastructure and bandwidth constraints in order to have a more realistic model. The extension will include power consumption of the switches and traffic congestion in the network. Clearly, this extension depends on the network topology. We chose hierarchical network topology as it is commonly used in the datacenters shown in Fig. 4.1. We assumed that a datacenter consists of a collection of Performance Optimized modular Datacenters (PoD). Each PoD consists of a number of racks and each rack contains a collection of servers. We considered a typical two-tier datacenter network [64], [65], which has servers housed in a rack connected to a Top-of-Rack (ToR) switch. The ToR switch provides connectivity among the servers of a rack and also

connects the rack to the Core Switch (CS) of its host PoD. Core switches depending on the datacenter topology such as Hyper-X, clique or fat-tree [64] may have different types of connectivity that provides varying amounts of bandwidths for communication among the PoDs.

In the assumed model, there is no communication congestion between the servers in the same rack because their connected to their ToR switch with high capacity links. The communication congestion may occur either in the (ToRS-CS) links or in PoD links (CS-CS). Hence, resource allocation has to consider the communication constraints of datacenter topology. We assume that a ToR switch will be turned off if none of the servers in that rack are being utilized. Similarly CS in a PoD will be turned off if all the servers connected to its racks are *off*. We note that an on switch consumes a constant power plus load dependent variable power; the former will be referred to as static and the latter as dynamic power respectively. We will let $PS_{\ell,e}^{ToRS}, PS_{\ell}^{CS}$ denote static power consumption of the ToR switch on rack $a_{\ell,e}$, and CS switch in PoD ℓ respectively. Similarly, we will let $PD_{\ell,e}^{ToRS}, PD_{\ell}^{CS}$ denote dynamic power consumptions of these switches for per bit transmission rate. We also let PW_{NIC} denote the power consumption at the network interface card (NIC) of a server per bit transmission rate. We define $\eta_{\ell,e}$ as a variable that determines whether ToR switch serving to rack e on pod ℓ is active or not. Similarly, ξ_{ℓ} determines status of the CS serving PoD ℓ. In addition to the newly introduced notation that is given in Table. 4.3, the notation of Table 4.1 remains valid for this model. From Table 4.3,

Table 4.3 Parameter/Variable Definitions

Parameters	Indicator
L	number of PoDs in the data center.
d_{ℓ}	number of racks in pod ℓ.
b_{ℓ}	set denoting racks in pod ℓ.
$a_{\ell,e}$	set denoting servers on rack e in pod ℓ.
ϑ_{n_h}	data rate of VMs serving job n_h
$S_{\ell,e}$	capacity of the link connecting rack e to its pod ℓ CS switch.
$CP_{\ell,\ell'}$	the capacity of the link connecting CS switches of pods ℓ and ℓ'.
$M_{\ell,e}^{t}$	number of type *t* servers in rack e of pod ℓ

$M_{\ell,e}$	total number of servers in rack e of pod ℓ
PS_{ℓ}^{CS}	static power usage rate of the CS switch in PoD ℓ
$PS_{\ell,e}^{ToRS}$	static power usage of the ToR switch on rack $a_{\ell,e}$ and
$PD_{\ell,e}^{ToRS}$	dynamic communication power usage of e^{th} rack ToR switch of pod ℓ.
PD_{ℓ}^{CS}	dynamic communication power usage of pod ℓ CS switch.
PW_{NIC}	dynamic communication power usage of server NIC card switch (for bit per second).
$P_{m_t,m'_{t'}}^{n_h}$	dynamic communication power usage between two VMs serving job n_h allocated in servers m_t and $m'_{t'}$.
$P'^{n_h}_{\ell,e;\ell',e'}$	dynamic communication power usage between two VMs serving job n_h allocated in a server in rack e of PoD ℓ and in a server in rack e' in PoD $\ell'$of.
$PR_{\ell,e}$	power supply of rack e on PoD ℓ.

Variables	Indicator
$\eta_{\ell,e}$	binary variable that assumes the value of one if at least one server on rack e in pod ℓ is active and otherwise zero.
ξ_{ℓ}	binary variable that assumes the value of one if at least one server in pod ℓ is active and otherwise zero.
$m_{\ell,e}^{j_t}$	number of active type t servers with pattern j_t in the e^{th} rack of pod ℓ.
$\varphi_{\ell,e}^{f,j_t}$	binary variable indicate whether server f type t in the e^{th} rack of pod ℓ is active pattern j_t or not.

$$M_t = \sum_{\ell=1}^{L} \sum_{e=1}^{d_\ell} M_{\ell,e}^{t} \tag{4.14}$$

$$M_{\ell,e} = \sum_{t=1}^{T} M_{\ell,e}^{t} \tag{4.15}$$

$$a_{\ell,e} = \{1_{\ell,e}, \dots, m_{\ell,e}, .. M_{\ell,e}\} \tag{4.16}$$

$$\xi_{\ell} = \begin{cases} 1 & \sum_{e=1}^{d_\ell} \eta_{\ell,e} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

Also from the definition of $\eta_{\ell,e}$,

$$\eta_{\ell,e} = \begin{cases} 1 & \sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \sum_{t=1}^{T} \sum_{m_t \in a_{\ell,e}} \tilde{x}_{n_h}^{m_t} > 0 \\ 0 & \sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \sum_{t=1}^{T} \sum_{m_t \in a_{\ell,e}} \tilde{x}_{n_h}^{m_t} = 0 \end{cases} \quad \forall e \in \{1, \dots, d_\ell\}, \forall \ell \in \{1, \dots, L\}.$$

4.2.1 Integer Linear Programming Model

In this subsection, we will develop an integer linear programming (ILP) model of the optimization problem introduced in the above. Dynamic communication power consumption between two VMs located in servers m_t and $m'_{t'}$ serving job n_h, $P_{m_t,m'_{t'}}^{n_h}$, is given by,

$$P^{n_h}_{m_t,m'_{t'}} =$$

$$\left\{\begin{array}{ll} 0, & if\ \ m_t = m'_{t'} \qquad (a) \\ \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToRS}_{\ell,e}\right), & if\ \ m_t, m'_{t'} \in a_{\ell,e}\ , m_t \neq m'_{t'} \qquad (b) \\ \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToRS}_{\ell,e} + PD^{CS}_{\ell} + PD^{ToRS}_{\ell,e'}\right), if\ m_t \in a_{\ell,e}, m'_{t'} \in a_{\ell,e'}\ , e \neq e' \qquad (c) \\ \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToRS}_{\ell,e} + PD^{CS}_{\ell} + PD^{CS}_{\ell'} + PD^{ToRS}_{\ell',e'}\right), if\ m_t \in a_{\ell,e}, m'_{t'} \in a_{\ell',e'}, \ell \neq \ell' (d) \end{array}\right\} \tag{4.18}$$

In the above, (4.18a) corresponds to dynamic power consumption of communication between two VMs of a type h job located at the same server. (4.18b) corresponds to the power consumption of communication between two VMs of a job n_h located at the same rack but different servers. As may be seen, power consumption depends on data rate, and NICs and ToRS dynamic power consumption per bit. (4.18c) corresponds to dynamic power consumption of communication between two VMs of a type h job located in two servers at two different racks of a PoD. As seen in (4.18c) dynamic power consumption in this case depends on the data rate, and NICs, ToRSs and CS dynamic power consumption per bit. (4.18d) corresponds to the dynamic power consumption of communication between two VMs of a job placed in two servers at two different racks in separated PoDs. In this case, as seen in (4.18d), in addition to data rate, and NICs, ToRSs power consumption per bit, dynamic power consumption of communication between two VMs of a job also depends on the CSs dynamic power consumption per bit. Then, the optimization problem is given by,

$$Min\left\{\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\sum_{t=1}^{T}\sum_{t'=1}^{T}\sum_{m_t=1_t}^{M_t}\sum_{m_{t'}=1_{t'}}^{M_t} P^{n_h}_{m_t,m_{t'}}\left(\tilde{x}^{m_t}_{n_h}\tilde{x}^{m'_{t'}}_{n_h}\right) + \sum_{\ell=1}^{L}\left(\xi_\ell PS^{\ell}_{CS} + \sum_{e=1}^{d_\ell}\eta_{\ell,e}PS^{\ell,e}_{ToR}\right) + \sum_{t=1}^{T} Q_t \sum_{m_t=1}^{M_t} y_{m_t}\right\} \tag{4.19}$$

$$ST.\ (4.2), (4.3), (4.4), (4.6), (4.7)$$

$$\sum_{m_t \in a_{\ell,e}} y_{m_t} - \eta_{\ell,e} \geq 0 \quad \forall\ e\ \in \{1, \dots, d_\ell\},\ \forall\ \ell \in \{1 \dots L\} \tag{4.20}$$

$$\theta\eta_{\ell,e} - \sum_{m_t \in a_{\ell,e}} y_{m_t} \geq 0\ \ \forall\ e\ \in \{1, \dots, d_\ell\},\ \forall\ \ell \in \{1 \dots L\} \tag{4.21}$$

$$\sum_{e=1}^{d_\ell}\eta_{\ell,e} - \xi_\ell \geq 0 \qquad \forall\ \ell \in \{1 \dots L\} \tag{4.22}$$

$$\theta\xi_\ell - \sum_{e=1}^{d_\ell}\eta_{\ell,e} \geq 0 \qquad \forall\ \ell \in \{1 \dots L\} \tag{4.23}$$

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\vartheta_{n_h}\sum_{m_t\in a_{\ell,e}}\left[\sum_{\ell'=1}^{L}\sum_{e'=1}^{d_{\ell'}}\sum_{m'_{t'}\in a_{\ell',e'}}\left(\tilde{x}_{n_h}^{m_t}\tilde{x}_{n_h}^{m'_{t'}}\right)-\right.$$

$$\left.\sum_{m'_{t'}\in a_{\ell,e}}\left(\tilde{x}_{n_h}^{m_t}\tilde{x}_{n_h}^{m'_{t'}}\right)\right]\leq S_{\ell,e}\ \ \forall\, e\in\{1,\dots,d_\ell\},\forall\,\ell\in\{1\dots L\} \tag{4.24}$$

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\vartheta_{n_h}\sum_{e=1}^{d_\ell}\sum_{m_t\in a_{\ell,e}}\sum_{e'=1}^{d_{\ell'}}\sum_{m'_{t'}\in a_{\ell',e'}}\left(\tilde{x}_{n_h}^{m_t}\tilde{x}_{n_h}^{m'_{t'}}\right)\leq CP_{\ell,\ell'}\ \ \forall\,\ell,\ell'\in\{1\dots L\}$$

$$,\ell\neq\ell' \tag{4.25}$$

In the objective function, the first term corresponds to the dynamic part of the communication power consumption. Second term represents the static part of communication power consumption and finally the last term expresses the power consumption of the servers. It should be noted that traffic congestion may occur only in ToRS to CS and CS to CS links, with their capacities $S_{\ell,e}$, $CP_{\ell,\ell'}$ defined in Table 4.3. As explained before, there is no traffic congestion in server communications within a rack. The constraints (4.24) and (4.25) ensure that bandwidth demands do not violate the capacities of ToRs to CS and CS to CS links respectively. The rest of constraints, (4.20)-(4.23) are used for IBLC. Constraints (4.21), (4.22) make the connection between $\eta_{\ell,e}$, y_{m_t} such that $\eta_{\ell,e}=0\leftrightarrow\sum_{m_t\in a_{\ell,e}}y_{m_t}=0$ and $\eta_{\ell,e}=1\leftrightarrow\sum_{m_t\in a_{\ell,e}}y_{m_t}>0$. Constraints (4.22), (4.23) make the connection between ξ_ℓ , $\eta_{\ell,e}$ such that $\xi_\ell=0\leftrightarrow$ $\sum_{e=1}^{d_\ell}\eta_{\ell,e}=0$ and $\xi_\ell=1\leftrightarrow\sum_{e=1}^{d_\ell}\eta_{\ell,e}>0$.

4.2.2 Column Generation Model

Due to large-scale of the optimization problem, once again we are interested in applying column generation technique which provides an alternative solution to our problem. Here, $m_{\ell,e}^{j_t}$ represents number of active type t servers with pattern j_t in the e^{th} rack of PoD ℓ. Hence, the state of rack e on pod ℓ as active or not may be expressed as,

$$\eta_{\ell,e}=\begin{cases}1 & \sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t}m_{\ell,e}^{j_t}>0\\ 0 & otherwise\end{cases}$$

We note that dynamic communication power consumption between two VMs depend on the rack locations of the servers housing the VMs and not on the server locations

within the racks. Let us define $P'^{n_h}_{\ell,e;\ell',e'}$ as dynamic communication power consumption between a VM allocated in a server in rack e of PoD ℓ and a VM allocated in a server in rack e'in PoD ℓ' serving job n_h, which is given by,

$$P'^{n_h}_{\ell,e;\ell',e'} = \begin{Bmatrix} \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToR}_{\ell,e}\right) & if \quad \ell = \ell', e = e' \\ \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToRS}_{\ell,e} + PD^{CS}_{\ell} + PD^{ToRS}_{\ell,e'}\right), & if \quad \ell = \ell', e \neq e' \\ \vartheta_{n_h}\left(2PW_{NIC} + PD^{ToRS}_{\ell,e} + PD^{CS}_{\ell} + PD^{CS}_{\ell'} + PD^{ToRS}_{\ell',e'}\right), & if \quad \ell \neq \ell' \end{Bmatrix} \quad (4.26)$$

It should be noted that communication power consumption between VMs of a job located in the same server is considered to be zero similar to the assumption made in the previous subsections. Then the optimization problem may be expressed as,

$$Min \ \left\{\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\sum_{\ell=1}^{L}\sum_{e=1}^{d_\ell}\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t}\left\{\left[\sum_{\ell'=1}^{L}\sum_{e'=1}^{d_{\ell'}}\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_{t'}} P'^{n_h}_{\ell,e;\ell',e'}\left(m^{j_t}_{\ell,e}\tilde{x}^{j_t}_{n_h}.\ m^{j'_{t'}}_{\ell',e'}\tilde{x}^{j'_{t'}}_{n_h}\right)\right] - P'^{n_h}_{\ell e;\ell e} m^{j_t}_{\ell,e}\left(\tilde{x}^{j_t}_{n_h}\right)^2\right\} + \sum_{\ell=1}^{L}\left[\xi_\ell PS^{CS}_{\ell} + \sum_{e=1}^{d_\ell}\eta_{\ell,e}(PS^{\ell,e}_{ToR})\right] + \sum_{\ell=1}^{L}\sum_{e=1}^{d_\ell}\sum_{t=1}^{T} Q_t \sum_{j_t=1_t}^{J_t} m^{j_t}_{\ell,e}\right\} \quad (4.27)$$

$ST.$ (4.22), (4.23) and

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\vartheta_{n_h}\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t} m^{j_t}_{\ell,e}\tilde{x}^{j_t}_{n_h}\left[\left(\sum_{\ell'\in\{1..L\},\ell\neq\ell'}\sum_{e'=1,}^{d_{\ell'}}\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_{t'}} m^{j'_{t'}}_{\ell',e'}\tilde{x}^{j'_{t'}}_{n_h}\right) + \left(\sum_{e'\in b_{\ell,e'}\neq e}\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_{t'}} m^{j'_{t'}}_{\ell,e'}\tilde{x}^{j'_{t'}}_{n_h}\right)\right] \leq S_{\ell,e} \quad (4.28)$$

$$\sum_{h=1}^{H}\sum_{n_h=1_h}^{N_h}\vartheta_{n_h}\sum_{e=1}^{d_\ell}\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t}\sum_{e'=1}^{d_{\ell'}}\sum_{t'=1}^{T}\sum_{j'_{t'}=1_{t'}}^{J_{t'}}\left(m^{j_t}_{\ell,e}\tilde{x}^{j_t}_{n_h}.m^{j'_{t'}}_{\ell',e'}\tilde{x}^{j'_{t'}}_{n_h}\right) \leq CP_{\ell,\ell'}$$

$$\forall\ \ell,\ell' \in \{1\dots \mathrm{L}\},\ \ell' \neq \ell \quad (4.29)$$

$$\sum_{\ell=1}^{L}\sum_{e=1}^{d_\ell}\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t} x^{j_t}_{r,n_h} m^{j_t}_{\ell,e} \geq v^r_{n_h} \quad (4.30)$$

$$\sum_{j_t=1_t}^{J_t} m^{j_t}_{\ell,e} \ \leq \ M^t_{\ell,e} \quad (4.31)$$

$$\sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t} m^{j_t}_{\ell,e} - \ \eta_{\ell,e} \geq 0 \quad (4.32)$$

$$\theta\eta_{\ell,e} - \sum_{t=1}^{T}\sum_{j_t=1_t}^{J_t} m^{j_t}_{\ell,e} \geq 0 \quad (4.33)$$

where constraints (4.30) are $\forall\, r \in \{1,\dots,R\},\ n_h \in \{1_h,\dots,N_h\}, h \in \{1,\dots,H\}$ constraints (4.31) are $\forall\, t \in \{1,\dots,T\}, \forall\, e\ \in \{1,\dots,d_\ell\}, \forall \ell \in \{1 \dots L\}$ and constraints (4.28), (4.32) and (4.33) are $\forall e \in \{1,\dots,d_\ell\}, \forall \ell \in \{1 \dots L\}$.

In the objective function (4.27), the first term corresponds to power consumption of the interface cards and dynamic power consumption of active switches due to communication load, second term to static power consumption of active switches and the third term to power consumption of active servers. The constraint (4.28) and (4.29) ensures that bandwidth demands of the jobs do not violate the capacities of the ToRS to CS links and CS to CS links respectively. Constraint (4.30) ensures that job and VM requirements are satisfied. Constraint (4.31) ensures that the number of type *t* servers in each rack, does not exceed the maximum number of type *t* servers in the rack. Constraints (4.32) and (4.33) are used for IBLC and connect $m_{\ell,e}^{j_t}$ and $\eta_{\ell,e}$ variables.

The pricing sub-problems are similar to those in (4.12), (4.13). In this case, the existence of the non-linear constraints (4.28), (4.29) and objective function creates problems for column generation, which requires their linearization. We use LP conversion method in order to convert the IQP to ILP.

Let us define binary variables $\varphi_{\ell,e}^{f,j_t}$ as follows,

$$\varphi_{\ell,e}^{f,j_t} = \begin{cases} 1 & if\ f^{th} \text{ type } t \text{ server on rack } e \text{ in pod } \ell \\ & \text{is active and has pattern } j_t \\ 0 & otherwise \end{cases}$$

Hence we have,

$$m_{\ell,e}^{j_t} = \sum_{f=1}^{M_{\ell,e}^t} \varphi_{\ell,e}^{f,j_t} \tag{4.34}$$

Then, product $m_{\ell,e}^{j_t}\, m_{\ell',e'}^{j_{t'}}$ that appears in non-linear constraints may be expressed as,

$$m_{\ell,e}^{j_t}\, m_{\ell',e'}^{j'_{t'}} = \sum_{f=1}^{M_{\ell,e}^t} \sum_{f'=1}^{M_{\ell',e'}^{t'}} \varphi_{\ell,e}^{f,j_t} \varphi_{\ell',e'}^{f',j'_{t'}} \tag{4.35}$$

Next let us further define a new binary variable $\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}}$ as,

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} = \varphi_{\ell,e}^{f,j_t} \varphi_{\ell',e'}^{f',j'_{t'}}$$

then,

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} = \begin{cases} 1 & if\ \ \varphi_{\ell,e}^{f,j_t} = \varphi_{\ell',e'}^{f',j'_{t'}} = 1 \\ 0 & otherwise \end{cases}$$

Thus the product term in (4.35) may be expressed in linear form as follows,

$$m_{\ell,e}^{j_t}\, m_{\ell',e'}^{j'_{t'}} = \sum_{f=1}^{M_{\ell,e}^{t}} \sum_{f'=1}^{M_{\ell',e'}^{t'}} \psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} \tag{4.36}$$

Binary multiplication can be linearized through adding the following constraints,

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} \geq \varphi_{\ell,e}^{f,j_t} + \varphi_{\ell',e'}^{f',j'_{t'}} - 1 \tag{4.37}$$

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} \leq \varphi_{\ell',e'}^{f',j'_{t'}} \tag{4.38}$$

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} \leq \varphi_{\ell,e}^{f,j_t} \tag{4.39}$$

$$\psi_{\ell\ell',ee'}^{ff',\, j_t j'_{t'}} \geq 0 \tag{4.40}$$

(4.37), (4.38), (4.39) and (4.40) are $\forall\, \ell, \ell' \in \{1 \ldots \mathrm{L}\}, \forall\, t, t' \in \{1, \ldots, T\}, \forall\, e' \in \{1_{\ell'}, \ldots, d_{\ell'}\}, \forall e \in \{1_{\ell}, \ldots, d_{\ell}\}, \forall j_t \in \{1_t, \ldots, J_t\},\ \forall j'_{t'} \in \{1_{t'}, \ldots, J_{t'}\},$

$\forall\, f \in \{1, \ldots, M_{\ell,e}^{t}\}, f' \in \left\{1, \ldots, M_{\ell',e'}^{t'}\right\}$

Using and substituting mentioned conversion in the objective function and constraints, CG problem becomes linear which reduces the complexity order at the expense of increasing number of variables and constraints drastically.

4.3 Probabilistic Model

In the previous sections, we assumed that the traffic (communication) rates and processing (computation) levels of different VM types were deterministic; however, in reality they are random and vary as a function of time. In this section, we extend the optimization problem of the previous section to a more realistic model, in which PM computation levels and VM communication rates are considered as random variables. In this model, the data rate between two VMs serving to a type h job, ϑ_{n_h}, becomes a random variable. As a result, bandwidth constraints given in (4.28, 4.29) become probabilistic. In particular, (4.28) may be expressed as,

$$\Pr\left(\sum_{h=1}^{H} \vartheta_{n_h} \sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} > S_{\ell,e}\right) \leq p \tag{4.41}$$

Where Ψ_{ℓ,e,n_h} denotes the total number of external communication flows of job n_h in rack e of PoD ℓ .For instance, Ψ_{ℓ,e,n_h} for IQP model is given by,

$$\Psi_{\ell,e,n_h} = \sum_{m_t \in a_{\ell,e}} \left[\sum_{\ell'=1}^{L} \sum_{e'=1}^{d_{\ell'}} \sum_{m'_{t'} \in a_{\ell',e'}} \left(\tilde{x}_{n_h}^{m_t} \tilde{x}_{n_h}^{m'_{t'}} \right) - \sum_{m'_{t'} \in a_{\ell,e}} \left(\tilde{x}_{n_h}^{m_t} \tilde{x}_{n_h}^{m'_{t'}} \right) \right] \quad (4.42a)$$

and parameter p is used to control the probability of link congestion or system failure.

As in [66], we assume that traffic rate follows a Gaussian distribution, from the Central Limit Theorem which remains a good model for the total link traffic even if the individual streams are non-Gaussian [67], [68].

Next, we assume that ϑ_{n_h} has a Gaussian distribution with mean λ_h and standard deviation σ_h. Then, the constraint (4.42) may be expressed as,

$$\sum_{h=1}^{H} \left(\lambda_h \sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} \right) + \zeta \sqrt{\sum_{h=1}^{H} {\sigma_h}^2 \left(\sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} \right)^2} \leq S_{\ell,e} \quad (4.43)$$

where $\zeta = \Phi^{-1}(1-p)$ and Φ^{-1} is the inverse function of the normal CDF. From [71], we note that LHS of the above constraint may be bounded as follows,

$$\sum_{h=1}^{H} \lambda_h \sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} + \zeta \sqrt{\sum_{h=1}^{H} {\sigma_h}^2 \left(\sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} \right)^2} \leq$$
$$\sum_{h=1}^{H} \left[(\lambda_h + \zeta \sigma_h) \sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} \right] \quad (4.44)$$

We decided to use the above upper bound in inequality (4.44) in order to eliminate the nonlinearity introduced by the square-root function, which results in,

$$\sum_{h=1}^{H} \left[(\lambda_h + \zeta \sigma_h) \sum_{n_h=1_h}^{N_h} \Psi_{\ell,e,n_h} \right] \leq S_{\ell,e} \quad (4.45)$$

In the previous sections, power consumption of a type t server is assumed to be constant denoted by Q_t. In fact, power consumption of a server is random and depends on processing utility, I/O, load, memory usage etc. Let q_t denote power consumption of a type t server. From [66], q_t has a general probability distribution and varies in the range $[0.5Q_t, Q_t]$ with mean and standard deviation denoted by ω_t, δ_t. When total power consumption, reaches to 96% of rated capacity at rack level or 72% at data center level [74]), then system failure, overheating, circuit break tripped may occur. It is better to avoid high power consumption at the rack level in order to prevent such a malfunction. As a result, we introduce the following constraint,

$$Pr\left(\sum_{t=1}^{T} q_t \sum_{j_t=1_t}^{J_t} m_{\ell,e}^{j_t} > PR_{\ell,e}\right) \leq p \tag{4.46}$$

Where $PR_{\ell,e}$ denotes the power supply of rack e on PoD ℓ. From the Central limit theorem we can assume that the total power consumption at the rack level has a Gaussian distribution. Similar to the analysis that has been done to find Eq. (4.45), Eq. (4.46) can be linearized as follows,

$$\sum_{t=1}^{T} \omega_t \sum_{j_t=1_t}^{J_t} m_{\ell,e}^{j_t} + \zeta \sum_{t=1}^{T} \delta_t \sum_{j_t=1_t}^{J_t} m_{\ell,e}^{j_t} \leq PR_{\ell,e} \tag{4.47}$$

Hence, the optimization problem has to consider uncertainty of computation and communication. The new optimization problem has two more constraints namely (4.45) and (4.46). These constraints provide margin against power failure and link congestion and therefore, leads to a more reliable system.

4.4 Dynamic Job Scheduling

In this section, we would like to study job scheduling with optimization of power consumption as a function of time. As a result, it will be assumed that time-axis is slotted and VMs are assigned to jobs in units of slot times. We will assume that arrival of jobs to the system is according to a Poisson process, though the analysis is applicable to other arrival processes. The new arriving jobs during the present slot and leftover jobs from the present slot will be scheduled for service in the next slot. We will consider two types of service disciplines, a job either releasing its assigned VMs simultaneously or individually according to Bernoulli trials at the end of each slot. In the former case, a leftover job will require full complement of its VMs and in the latter case a subset of the VMs it's currently holding. At the beginning of the next slot, the system will schedule the new arriving jobs and the leftover old jobs from the previous slot such that power consumption is minimized. For the scheduling of leftover jobs, there are two options depending whether or not VM migration is allowed. If VM migration is allowed, then leftover jobs are scheduled like the new jobs, on the other hand, if no migration is allowed then the new jobs can only be scheduled to VMs not utilized by the leftover jobs. As a result of migration, the system may end up in a state that consumes less power,

however, migration has communication and processing overhead that optimization needs to take into account. Let G_r denote normalized power consumption cost of migration of type *r* VMs. Optimization will allow VM migration if power saving due to migration offsets the cost of migration. As a result, the optimization may result in partial VM migration.

We consider dynamic resource allocation model with and without VM migration. Since jobs release their VMs according to Bernoulli trials, number of leftover jobs to the next slot will be a random variable with Binomial distribution. However, for simplicity we will assume that number of leftover jobs is a constant given by the mean of the Binomial distribution. Let N_h' denote number of the type *h* leftover jobs from the current slot and N_h total number of jobs to be scheduled in the next slot, which include both leftover as well as new arriving jobs. We note that $N_h \geq N_h'$ and $n_h \in (1_h, .., N_h', .., N_h)$ and the first N_h' jobs in the set correspond to the leftover jobs from the current slot. Next, we will develop both dynamic ILP and CG models.

4.4.1 Dynamic ILP Model

First, we will consider the job scheduling that allows VM migration. Let us consider ${n_h}^{th}$ job, which is in the system in the current slot and will continue to receive service in the next slot. Let ${x'}_{r,n_h}^{m_t}$, $x_{r,n_h}^{m_t}$ denote the number of type *r* VMs assigned to this job over the m^{th} type *t* server during the current and next slots respectively. Based on the new notation introduced in Table. 4, we define the following binary variable,

$$\beta_{r,n_h}^{m_t} = \begin{cases} 1 & \text{if } x_{r,n_h}^{m_t} - {x'}_{r,n_h}^{m_t} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.48}$$

The value of $\beta_{r,n_h}^{m_t}$ shows whether type *r* VMs required by job n_h have migrated or not. In the case of VM migration from this type of server, then $x_{r,n_h}^{m_t} < {x'}_{r,n_h}^{m_t}$ and $\beta_{r,n_h}^{m_t}$ will have a nonzero value and in all other cases a zero value. The objective function of this optimization problem will be given by,

$$Min \left[(4.19) + \sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h'} \sum_{r=1}^{R} \sum_{\ell=1}^{L} \sum_{e=1}^{d_\ell} \sum_{t=1}^{T} \sum_{m_t \in a_{\ell,e}} G_r \beta_{r,n_h}^{m_t} \left| x_{r,n_h}^{m_t} - {x'}_{r,n_h}^{m_t} \right| \right]$$

where absolute value of $(x_{r,n_h}^{m_t} - x'^{m_t}_{r,n_h})$ corresponds to number of VM migrations. In the above, migration of a VM will be allowed if it results in power saving larger than power cost of migration.

Table 4.4 Parameter/Variable Definitions for Dynamic Job Scheduling

Parameters	Definitions
$x'^{m_t}_{r,n_h}$	number of type r VMs of server m_t assigned to serve job n_h at the current time slot.
N'_h	total number of current type h jobs
$v'^{r}_{n_h}$	number of type r VMs required by job n_h at current time slot left in the system.
$x'^{j_t}_{r,n_h}$	number of type r VMs serving job n_h on a type t server with pattern j_t at current time slot
$m'^{j_t}_{\ell,e}$	number of active type t servers with pattern j_t in the rack of pod ℓ at the current time slot.
$\varphi'^{f,j_t}_{\ell,e}$	Binary parameter represents whether type t server on rack e in pod ℓ which has pattern j_t at current time slot is active or not.
G_r	power consumption related to the migration of type r VMs
I	Number of iterations among RMP and pricing problems to find the best cutting patterns
D_{sub}	expectation of the time required to solve a sub-problem
D_{MT}	expectation of time required to solve the relaxed RMP
D_{Int}	time required to convert RMP relaxed LP to ILP optimal solution.
$x_{r,h}^{\tilde{I}_t}$	number of VM type r for type h jobs over the server type t in pattern $\tilde{I}_t$ introduced by initialization
R_h	Different VM types demanded by type h jobs.

Job scheduling without VM migration can be achieved by setting G_r to a very large value. This will prevent migration as its cost cannot be offset by any power saving. As a result, old jobs will preserve their VM assignments.

Finally, we have to add the following constraints into the problem in order to linearize equation (4.48),

$$x_{r,n_h}^{m_t} - x'^{m_t}_{r,n_h} + \theta\beta_{r,n_h}^{m_t} < 1 \tag{4.49}$$

$$x_{r,n_h}^{m_t} - x'^{m_t}_{r,n_h} + \theta\beta_{r,n_h}^{m_t} \geq 0 \tag{4.50}$$

Where (4.49), (4.50) are $\forall\, r \in \{1, \dots, R\}, m_t \in \{1, \dots, M_t\},\ t \in \{1, \dots, T\},\ n_h \in \{1_h, \dots, N_h\}, h \in \{1, \dots, H\}$

4.4.2 Dynamic Column Generation Model

As in the above, first let us consider job scheduling with VM migration. Assume that ${n_h}^{th}$ job is in the system in the current slot and will continue to receive service in the next slot. Let ${x'}_{r,n_h}^{j_t}$, $x_{r,n_h}^{j_t}$ denote the number of type *r* VMs assigned to this job over the ${j_t}^{th}$ pattern during the current and next slots respectively. Similarly, ${\varphi'}_{\ell,e}^{f,j_t}$, $\varphi_{\ell,e}^{f,j_t}$ are binary variables indicating whether f^{th} type *t* server on rack e in pod ℓ is active and has pattern j_t during the current and next slots respectively. In this model, we define the binary variables $\beta_{\ell,e,r,n_h}^{f,t}$ that show whether or not *r* type VMs required by job n_h have migrated or not from a server as follows,

$$\beta_{\ell,e,r,n_h}^{f,t} = \begin{cases} 1 & if \ \sum_{j_t=1_t}^{J_t} \left(x_{r,n_h}^{j_t} \varphi_{\ell,e}^{f,j_t} - {x'}_{r,n_h}^{j_t} {\varphi'}_{\ell,e}^{f,j_t} \right) < 0 \\ 0 & otherwise \end{cases} \tag{4.51}$$

We note that the summation in the above allows the use of a different pattern at the server as long as it preserves the number of VMs assigned by the original pattern to this job. The objective function of this optimization problem is given by,

$$Min \left[(4.27) + \sum_{h=1}^{H} \sum_{n_h=1_h}^{N'_h} \sum_{r=1}^{R} G_r \sum_{\ell=1}^{L} \sum_{e=1}^{d_\ell} \sum_{t=1}^{T} \sum_{f=1}^{M_{\ell,e}^t} \beta_{\ell,e,r,n_h}^{f,t} \sum_{j_t=1_t}^{J_t} \left| x_{r,n_h}^{j_t} \varphi_{\ell,e}^{f,j_t} - {x'}_{r,n_h}^{j_t} {\varphi'}_{\ell,e}^{f,j_t} \right| \right] \tag{4.52}$$

As in the previous subsection, job scheduling without VM migration can be achieved by setting G_r to a very large value. Finally, similar to the previous subsection, we have to add the following constraints to the problem in order to linearize Eq. (4.51),

$$\sum_{j_t=1_t}^{J_t} \left(x_{r,n_h}^{j_t} \varphi_{\ell,e}^{f,j_t} - {x'}_{r,n_h}^{j_t} {\varphi'}_{\ell,e}^{f,j_t} \right) + \theta \beta_{\ell,e,r,n_h}^{f,t} < 1 \tag{4.53}$$

$$\sum_{j_t=1_t}^{J_t} \left(x_{r,n_h}^{j_t} \varphi_{\ell,e}^{f,j_t} - {x'}_{r,n_h}^{j_t} {\varphi'}_{\ell,e}^{f,j_t} \right) + \theta \beta_{\ell,e,r,n_h}^{f,t} \geq 0 \tag{4.54}$$

where (4.53), (4.54) are $\forall\, r \in \{1, \dots, R\}, f \in \{1, \dots, M_{\ell,e}^t\}, t \in \{1, , , T\}, n_h \in \{1_h, \dots, N_h\}, h \in \{1, \dots, H\}$.

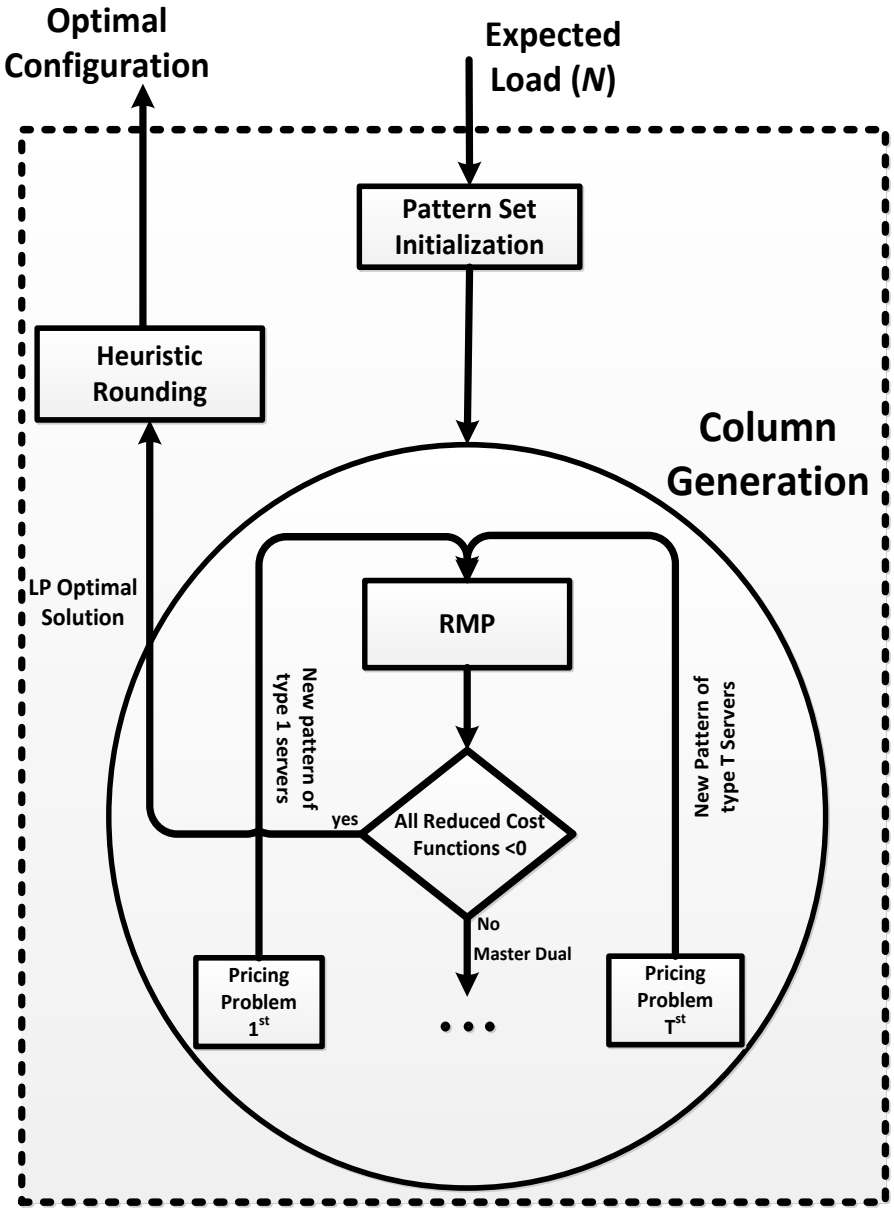

Fig. 4.2 Optimization Module Structure

4.5. Optimization Structure and Complexity Reduction

In addition to the complexity of the search region and large scale of the number of variables, computing time constraint is also an important factor which has to be taken

into account. As discussed earlier, we have used the CG technique to solve the optimization problem. The aforementioned optimization problem cannot be solved in a plausible short time frame. Hence, we are interested in steps that will reduce the computing time.

The main optimization problem consists of several sub-problems: RMP, *T* pricing sub-problems and the problem of finding the exact ILP solution from the relaxed LP-solution of RMP. Let T_{sch} denote the amount of time it takes to solve the optimization problem using CG technique. T_{sch} may be expressed in terms of new variables $I, D_{MT}, D_{sub}, D_{Int}$ introduced in Table. 4 as follows,

$$T_{sch} = I(D_{MT} + TD_{sub}) + D_{Int}$$

T_{sch} may be reduced through the following steps:

1- Reduction of the number of iterations, *I*, by offline initialization.
2- Simultaneous, instead of sequential, execution of sub-pricing problems that results in replacing T by 1 in the expression.
3- Reduction of computing time to find the ILP solution from the LP-relaxed solution, D_{Int}, through the use of a proposed heuristic.

Figure 4.2 depicts the proposed optimization platform. Given a number of different types of jobs, first, we solve the offline optimization problem explained in subsection 4.5.1 for each server type to obtain initial server configuration patterns. Then, RMP is initialized with these patterns. RMP is solved using the barrier optimizer, which applies a primal-dual logarithmic algorithm to determine the optimal solution. The solution yields the dual vector of variables to the pricing problems.

The pricing problem in (4.12), (4.13) is solved for different server types, which introduces a new set of patterns to the RMP. Pricing problems use branch and cut algorithms to solve the integer programming problems. As long as values of the reduced cost functions are positive, the algorithm (collaboration among RMP and pricing problems) continues, but once the reduced cost functions all together become negative, the pricing problem terminates and does not introduce any new candidate pattern set to the RMP. Instead of using the branch and bound technique, we use a heuristic rounding algorithm explained in Subsection 4.5.2 to find the ILP solution from the relaxed LP

solution of the RMP.

4.5.1 CG Initialization

We use offline initialization to reduce computation time for the solution of the optimization problem. Without initialization, in the first iterations, the RMP does not contain adequate columns to provide beneficial dual information to pricing sub-problems [76]. An appropriate initialization helps to reduce number of iterations RMP and pricing problems to reach to the solution through introduction of optimum patterns. Optimum patterns maximize resource utilization of active servers. We use the notation introduced in Table. 4.4 and define the initialization (optimization) problem as follows,

$$Max \ \sum_{r=1}^{R} x_{r,h}^{\tilde{I}_t} i_r^k \tag{4.55}$$

$$ST. \sum_{r=1}^{R_h} x_{r,h}^{\tilde{I}_t} i_r^{k'} \leq c_t^{k'} \quad \forall k' \in \{1, \dots, K\} \tag{4.56}$$

We solve this problem for each $\{k, t, h\}$ and find the best Υ_t patterns for different types of jobs. Then, for a type t server we will have $\Upsilon_t HK$ initial patterns. To obtain $x_{r,n_h}^{\tilde{I}_t}$s, which are introduced in the previous sections and are related to the initial pattern $\tilde{I}_t$, $x_{r,h}^{\tilde{I}_t}$ is assigned to a type h job while other jobs are set to zero. Hence, for each $x_{r,h}^{\tilde{I}_t}$ vector there would be N_h different patterns. Thus, initial number of patterns for server type t will be equal to $\sum_{h=1}^{H} N_h \Upsilon_t K$. So in the proposed initialization, we may have separate candidate patterns for each job. After collaboration of the pricing problems and RMP, new patterns that consider different jobs in a server will be introduced by pricing problems.

4.5.2 Heuristic Rounding Termination Algorithm

As mentioned earlier, LP problem (solvable in polynomial time) has less complexity compared to ILP problem (NP-hard optimization problem). In the CG solution of our optimization problem, RMP is LP and pricing problems are ILP type. As a result, we need to determine the optimal ILP solution of the RMP from its relaxed LP solution following the termination of the iterative process. Typically, this is done through the branch and

bound algorithm [63], which is time consuming. In the following, we propose a heuristic method to find the ILP solution from the relaxed LP solution that satisfies the scheduling time constraint [76], [77]. The proposed method will round up and down the values of the scheduling variables, $m_{\ell,e}^{j_t}$, in the relaxed LP solution [75], [78]. This operation will be carried out after $m_{\ell,e}^{j_t}$ have been sorted according to their priorities. $m_{\ell,e}^{j_t}$s more likely to be rounded down will be given higher priority. Following this operation, it is possible that all the servers of a rack will become inactive in that case ToR switch serving to that rack will be turned off to save power.

First, let us define $s_{\ell,e}$ as the set of scheduling variables for the rack $a_{\ell,e}$,

$$s_{\ell,e} = \left\{ m_{\ell,e}^{j_t}, j_t \in \{1_t, \dots, J_t\}, t \in \{1, \dots, T\} \right\}$$

and define set S as the set with its elements given by the subsets $s_{\ell,e}$ as given below,

$$S = \left\{ s_{\ell,e} | 1_\ell \le e \le d_\ell, 1 \le \ell \le L \right\}$$

Next, we split S into two mutually exclusive subsets,

$$S = \{S_1, S_2\}$$

where S_1 consists of all $s_{\ell,e}$ whose elements have values strictly less than one and S_2 otherwise. The elements of S_1 denote potentially inactive racks, while elements of S_2 active racks. From the above definition, elements of $S_1$is given higher priority than S_2 in rounding operation.

First, we sort set S_1 according to the number of active servers in a rack, $\sum_{t=1}^{T} \sum_{j_t=1_t}^{J_t} m_{\ell,e}^{j_t}$, in ascending order. Thus, the number of active servers in elements of S_1 will increase from left to right. We note that the order of the elements of S_2 isnot significant for rounding operation.

Next, we sort scheduling variables, $m_{\ell,e}^{j_t}$, within each $s_{\ell,e}$ wrt two performance measures in ascending order. These performance measures are efficiencies of server types (t) and patterns (j_t). First, $m_{\ell,e}^{j_t}$ will be sorted according to server type efficiency. Then, the ties among $m_{\ell,e}^{j_t}$ with the same service type will be broken through sorting according to pattern efficiency. Next, we explain each of these sorting algorithms.

i) Server Type efficiency-based sorting:

Depending on the job load some resources become critical and may become performance bottleneck [80], [81]. As a result, we first sort resources according to their criticalities. For a given job load, let L^k denote the total demand for resource type k,

$$L^k = \sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \sum_{r=1}^{R} v_{n_h}^r \, i_r^k \quad \forall \, k \in \{1, \dots, K\}$$

Then, the resource types may be ordered according to their criticality using the following formula

$$max_k \frac{L^k}{\sum_{t=1}^{T} M_t c_t^k} \tag{4.57}$$

Thus higher is the ratio of total demand to total amount of that resource in the datacenter, then higher will be the criticality of that resource. Next, we define efficiency of a server type with respect to resource type k as the ratio of (c_t^k / Q_t) with higher value indicating higher efficiency. Next, we order server types according to their efficiency for the critical resource. In the case of a tie, server efficiencies wrt second critical resource will be used to break down the ties and so on and so forth. System will prefer to use the server types with higher efficiencies. Set for each rack will be sorted in ascending order according to the efficiency of the server type of each element. The ties between the elements having the same server type will be broken through pattern-based sorting.

ii) Pattern efficiency-based sorting:

The patterns of each server type will be sorted in ascending order according to their resource utilization $\sum_{h=1}^{H} \sum_{n_h=1_h}^{N_h} \sum_{r=1}^{R} x_{r,n_h}^{j_t} \, i_r^k$. The sorting function is illustrated in Algorithm 4.1.

Algorithm 4.1. Terminating the optimization (non-Integer to Integer Conversion)

Data : optimal LP values of $m_{\ell,e}^{j_t}$ s
Result : Integer Values of $m_{\ell,e}^{j_t}$s

1 *For* ℓ=1 :L *and for* $e = 1_\ell : d_\ell$
2 *Find* $(\ell, e) | \forall\, t \in \{1, \dots, T\}, j_t \in J_t \;\; m_{\ell,e}^{j_t} \leq 1$
3 *Prioritize servers in* S_1
Case S_1
Prioritize $s_{\ell,e}s$ *according to* $min(\sum_{t=1}^{T} \sum_{j_t \in J_t} m_{\ell,e}^{j_t})$
4 *Sort* $m_{\ell,e}^{j_t}$ *according to the type ();*
5 *For t=1:T*
Sort servers according to the patterns(); end
Case S_2
6 *Sort* $m_{\ell,e}^{j_t}$ *according to the types();*
7 *For t=1:T*
8 *Sort servers according to the patterns(); end*
9 *Round up all the* $m_{\ell,e}^{j_t}s$;
10 *For all* $m_{\ell,e}^{j_t}s$
$m_{\ell,e}^{j_t} = m_{\ell,e}^{j_t} - 1$;
If $\sum_{\ell=1}^{L} \sum_{e=1}^{d_\ell} \sum_{t=1}^{T} \sum_{j_t \in J_t} (x_{n,r}^{j_t}) m_{\ell,e}^{j_t} < v_n^r$
$m_{\ell,e}^{j_t} = m_{\ell,e}^{j_t} + 1$ *end*
Go for the next highest priority $m_{\ell,e}^{j_t}$

Following the completion of sorting, all the $m_{\ell,e}^{j_t}$s within the set *S* have been assigned priority with the first element of the set having the highest priority in rounding down operation. First, we round up all the $m_{\ell,e}^{j_t}$ variables is the set *S* with non-integer values. Then, rounding down operation is applied from the highest to lowest priority $m_{\ell,e}^{j_t}$s one by one. In this operation, each $m_{\ell,e}^{j_t}$ is decremented by one if the demand constraints are not violated. Steps 4 and 6 can be done offline such that all the server types are sorted according to c_t^k / Q_t. The complexity order of the mentioned algorithm is approximated by,

$$O(\sum_{t=1}^{T} M_t J_t \, (min(log(\sum_{t=1}^{T} M_t J_t), NR))$$

in which M_t, N, R and J_t are number of type *t* servers, number of jobs, number of VM types and number of pattern of type t servers respectively.

$\left(\sum_{t=1}^{T} M_t J_t \, (log(\sum_{t=1}^{T} M_t J_t)\right)$ is due to the sorting part and $NR \sum_{t=1}^{T} M_t J_t$ is because of

the checking the demands constraints part.

4.6 Numerical Results

In this section, we present some numerical results regarding the analysis in this chapter. Numerical results plot a performance metric either at a random time or as a function of discrete time. In the first case, number of jobs is assumed to be either a constant or a variable. In the latter case, new jobs arrive to the datacenter according to a Poisson process.

We compare performance of our optimum resource allocation algorithms with two heuristic scheduling methods namely deterministic and random. The deterministic method is similar to the scheduling algorithm proposed in section 3.3 of chapter 3 that assigns a job to the PoD and rack with the smallest index number that also has enough idle resources to serve the job. In the random method, each VM of a job is placed to a randomly chosen rack of a PoD with enough idle resources given that communication demand does not violate the link capacities; otherwise a new rack is randomly chosen for the placement of VM.

IBM ILOG CPLEX is used as a platform to model and solve the optimization problems. We assume a datacenter with the topology shown in Fig. 4.1. We presume that the datacenter has 4 PoDs and each PoD having 25 racks. In consonance with [82], we assumed that each rack contains 40 to 80 servers and racks of each PoD has the same server composition. It should be noted that solution of the IQP model always gives the exact results. However, for large scale datacenters, finding the global optimum point of the IQP becomes overly complex and time consuming.

Next, we present the parameters of the system used in generation of numerical results.

i) Servers and Server Types

Table 4.5a presents number of servers per server type per rack at each PoD. Table 4.5b shows number of servers per server type per PoD, which is obtained by multiplication of each entry of Table 4.5a by 25. Considering Amazon instances and Google clusters, we consider $T=12$ server types with two resource types, CPU cores and memory. Table 4.6

presents the amount of resources and power consumption of each server type.

Table 4.5a No. of servers per type per PoD

PoD No(ℓ) Server type(t)	ℓ =1	ℓ =2	ℓ =3	ℓ =4
1	300	0	550	150
2	100	0	200	200
3	150	0	200	150
4	200	0	150	150
5	300	0	200	0
6	200	0	100	100
7	0	300	0	700
8	0	250	0	250
9	0	150	0	100
10	0	50	0	200
11	0	250	0	0
12	0	100	0	0
Total No. of servers	1250	1100	1400	2000

Table 4.5b. No. of servers per type per rack

$M_{\ell,e}^{t}$ ℓ-> t	ℓ =1	ℓ =2	ℓ =3	ℓ =4
1	12	0	22	6
2	4	0	8	8
3	6	0	8	6
4	8	0	6	6
5	12	0	8	0
6	8	0	4	4
7	0	12	0	28
8	0	10	0	10
9	0	6	0	4
10	0	2	0	8
11	0	10	0	0
12	0	4	0	0
No. of Servers	50	44	56	80

Table 4.6 Characteristics of Server types

Index (t)	Model	No. of Cores $c_{t,1}$	Memory $c_{t,2}$	No. of PMs M_t	Power Supply Q_t	ω_t , δ_t
1	Dell PE T110	4	16GB	1000	350W	200W, 20W
2	Dell PE T410	8	128GB	300	580W	400W, 20W
3	Dell PE M910	32	512GB	200	2750W	1500W, 100W
4	Dell PE R810	16	512 GB	250	2200W	1200W, 100W
5	Dell PE M915	64	1TB	100	2750W	1500W, 100W
6	Dell PE R910	40	2TB	150	3000W	1500W, 100W
7	HP DL320e Gen8	4	32GB	1000	350W	200W, 20W
8	HP DL360e Gen8	8	384GB	500	750W	400W, 50W
9	HP DL380p Gen8	8	768GB	250	1200W	700W, 50W
10	HP DL360 G7	4	768GB	250	1200W	700W, 50W
11	HP DL385p G7	16	768 GB	150	2000W	1200W, 100W
12	HP DL370 G6	16	2 TB	100	2300W	1150W, 100W

ii) Communication Network Parameters

Network power consumption parameters, $PD_{\ell,e}$ and $PS_{\ell,e}$, are the same as discussed in [83], [84] and [85]. We also assume that dynamic power consumption of a NIC is given by $\mathrm{PW_{NIC}} = 0.6\,\mathrm{microW}$. ToR switches offer a combination of internal (int) and external (ext) interfaces. The internal interfaces connect to NIC of the blade-servers while the external interfaces connect to Core switches. It is assumed that internal and external interfaces support up to 10 Gbps and 40 Gbps respectively. Table 4.7 presents the performance characteristics of the chosen switches for the network structure.

Table 4.7. Specification of Typical Switches

Name	Switch Type	Data Rate	No.	Power (static) $PS_{\ell}^{ToR/CS}$	Power (Dynamic) $PD_{\ell,e}^{ToR/CS}$
NEC IP8800	ToR	10 GbpS int 40GbpS ext	100	25W	65 nano w/bps
HP A12500	Core Switch	200Gbps	4	200W	10 nano w/bps

iii) Parameters of VM Types

We presume that number of VM types is *R=18* with their resource requirements given in Table 4.8. Resources of VMs consist of number of CPU cores and amount of memory. It is assumed that each physical core of a CPU is utilized as a virtual CPU (vCPU). In order to balance CPU, memory and network resources, Amazon t2 and m3 series are appropriate for many applications and servers, Microsoft SharePoint, and enterprise applications. c3 series with higher ratio of vCPU to memory represent compute-optimized Amazon instances which are appropriate for high-traffic web sites, on-demand batch processing, distributed analytics, web servers, and high performance science and engineering applications. r3 series represent memory optimized amazon instances and are recommended for memory bound applications such as high performance databases and distributed cache, in-memory analytics, genome assembly, and larger deployments of SAP. cg1 and g2 are also considered for game streaming, video encoding, 3D application streaming and other server-side graphic workloads.

Table 4.8 VM Types

Type (r)	Model	vCPU(i_r^1)	Mem (GiB) (i_r^2)
1	t2.micro	1	1
2	t2.small	1	2
3	t2.medium	2	4
4	m3.medium	1	3.75
5	m3.large	2	7.5
6	m3.xlarge	4	15
7	c3.large	2	3.75
8	c3.xlarge	4	7.5
9	c3.2xlarge	8	15
10	c3.4xlarge	16	30
11	c3.8xlarge	32	60
12	r3.large	2	15.25
13	r3.xlarge	4	30.5
14	r3.2xlarge	8	61
15	r3.4xlarge	16	122
16	r3.8xlarge	32	244
17	g2.2xlarge	8	15
18	cg1.xlarge	16	22.5

iv) Parameters of job types

We assume that the number of job types equals to *H = 9*. Table 4.9 presents

requirements and appropriate applications for each job type. It may be seen that with increasing job type h value, requirements for one or more of the following resources also increases, the number of VMs, VM sizes and VM traffic rates. Thus jobs with higher h type have higher resource demands. The type of each job is determined probabilistically according to α_h values given in the table. We assumed that arrival rates of job types are an inverse function of their demand requirements. A job belonging to each type may request VMs with different types. From Amazon recommendations in [69] and [70], the table presents types of VMs for each job type. As given in the table, the number of VMs required by a job is either a constant C_h or a uniformly distributed random number between Min_h and Max_h for type h jobs.

After determining the type of a job and the number of VMs it requires, the next step is determination of the types of its VMs. The type of each VM of each job type is determined probabilistically according to the percentages given in the table.

We assume that the traffic rate between two VMs of a job type is either a random variable or a constant. In the former case, we assumed that traffic rate for each job type has a Gaussian distribution with the mean and standard deviation given in the table. In the latter case, the traffic rate for each job type is a constant that equals to the mean of the Gaussian variable.

We considered both individual and simultaneous VM release service disciplines for a type h job at the end of a slot according to a Bernoulli trial with probability ρ_h. Then, we assume that ρ_h=*0.3* $\forall\, h \in \{1, \ldots, H\}$.

Finally for the power constraint in the probabilistic model, we assume that power supply of a rack is given by $PR_{\ell,\mathrm{e}} = 25\mathrm{kW}$ [74]. We assumed that power overloading probability of the racks should be less than *p=0.02*.

Table 4.9 Jobs Types and their Requirements

Index (h)	Job Types	VM Type	VM types Percentage	C_h	$Min_h - Max_h$	α_h	Traffic rates, ω_h, σ_h (Mbps)
1	General Processing Jobs	t2.micro, t2.small, t2.medium, m3.medium, m3.large, m3.xlarge	30% 20% 20% 10% 10% 10%	10	1-20	0.46	3, 0.2
2	Graphical Processing Jobs	g2.2xlarge, cg1.xlarge	%70 %30	50	10-100	0.02	3, 0.2
3	Scientific Jobs 1	c3.large, c3,xlarge, c3.2xlarge, c3.4xlarge, c3.8xlarge	30% 30% 20% 10% 10%	100	10-200	0.02	0.7, 0.05
4	Scientific Jobs 2	r3.large, r3,xlarge, r3.2xlarge, r3.4xlarge, r3.8xlarge	30% 30% 20% 10% 10%	100	20-200	0.02	0.7, 0.05
5	Scientific Jobs 3	m3.medium, m3.large, m3.xlarge	50% 30% 20%	100	10-200	0.02	12, 2
6	Web Services	m3.medium,m3.large, 3.xlarge +c3.large, c3,xlarge, c3.2xlarge,	50% 30% 20% + 50% 30% 20%	10 + 10	1-20 + 0-20	0.4	5, 0.5
7	Data Search	m3.xlarge, r3.large, r3,xlarge, r3.2xlarge, r3.4xlarge, r3.8xlarge	30% 20% 20% 10% 10% 10%	100	10--200	0.02	1, 0.1
8	Enterprise Infrastructure Services	t2.micro,t2.small, t2.medium, m3.medium,m3.large, 3.xlarge	30% 20% 20% 10% 10% 10%	100	5-200	0.02	3, 0.5
9	Peer 2 Peer Services	c3.large,c3,xlarge,c3.2xlarge, c3.4xlarge, c3.8xlarge + r3.large,r3, xlarge,r3.2 xlarger3.4 xlarge, r3.8xlarge	30%, 30%, 20%, 10% 10% + 30%, 30% 20%, 10% 10%	100 + 50	5 -100 + 5- 100	0.02	10, 1

Fig. 4.3 presents optimal power consumption of the datacenter with Poisson arrival of new jobs as a function of the number of time slots. For these results, we assumed constant server power consumption and deterministic traffic rates between VMs. We considered optimization both with/without VM migration of the leftover jobs with individual VM release service discipline. For the VM migration scheme, we assumed zero power cost for migration. In this figure, we also plot consumption of the deterministic heuristic. Optimal power consumption with migration is lower compared to the without VM migration, once we have zero cost VM migration. We note that power consumption varies as a function of time because of the random job arrival process. It may be seen that there is a significant power usage gap (*100KW*) between optimal and heuristic algorithms power consumption, which shows value of the optimization.

Fig. 4.4 shows number of active racks as a function of the time for the two schemes, as expected VM with migration results in lower values compared to without migration scheme. For the same system, Fig. 4.5 plots optimal power consumption of the datacenter as a function of time for both with/without VM migration schemes with simultaneous VM release service discipline. As may be seen power consumption of the two schemes are closer to each other compared to individual VM release service discipline.

Figure 4.6 shows the cumulative distribution function (CDF) of bandwidth demand for the ToRS-CS links between the racks and core switches for fixed number of jobs in the datacenter, *N=350* jobs. CDF resulting from the optimization is given both for deterministic and random traffic rates with Gaussian distribution between two VMs. It may be seen that probabilities for a given demand is 18% or more less for the random than deterministic traffic rates due to statistical averaging. The figure also plots CDF of the bandwidth demand for the random placement of the VMs of a job in the datacenter without optimization of power consumption. The random heuristic results in higher communication demand than the optimized placement of the VMs. It may be seen that probabilities for a given demand is 45% or more higher for random than optimized placement of the VMs.

Figure 4.7 shows the number of active servers as a function of the number of jobs in the datacenter with number of VMs in each job type as a constant parameter. Results have been plotted for the C_h, Min_h, Max_h values of the parameter. For Min_h, Max_h results

have been plotted only for constant VM traffic rates, while for C_h for both constant and random traffic rates. As depicted, the number of active servers increases exponentially. It is rationally related to the lack of high performance servers as jobs increase. Hence more number of the servers is required to serve the jobs. Moreover upper and lower number of VMs for each type of jobs also has been considered in order to investigate the impact of number of demanding VMs on number of required PMs. As demonstrated in Figure 4.7, for the maximum number of VMs, system cannot support more than 250 jobs. Hence, for the same number of jobs, different number of VMs may change the required PMs dramatically. Moreover, due to the high communication rate among VMs of jobs, it is observed that for a fixed number of loads in terms of VMs, lower number of jobs with higher number of VMs per job requires more number of servers compared to the case of having higher number of jobs with lower number of VMs. In other words, smaller numbers of jobs with bigger number of VMs require more infrastructures with higher bandwidth which may lead into usage of bigger number of the servers. Furthermore, considering the probabilistic case, due to the reservation of the bandwidth for random traffic, the bandwidth usage becomes more critical and number of the servers required to serve the jobs is more than the case with a fixed traffic. Also, the constraints on power usage and external bandwidth of racks may cause activation of less efficient servers leading to a larger number of active servers. However, in a probabilistic case reliability and resistance against congestion will be higher, which prevent latency and rack power failure.

Figure 4.8 plots the total power consumption as a function of the number of jobs in the datacenter for optimal and random placement of the VMs of a job. For optimal placement of VMs, results have been plotted both for constant and random server power consumption cases, while for random placement only for constant server power consumption. Random allocation algorithm allocates VMs in DC randomly. As it shown, there is huge power usage gap (*5MW*) between maximum constant optimum resource allocation and random algorithm and (*1MW*) between maximum constant optimum resource allocation and deterministic heuristic for half loaded DC which shows we can achieve the optimal solution for power saving by using the proposed optimal resource allocation method.

Figure 4.9 also presents the number of active racks for different PoDs as a function of number of jobs in the datacenter. It may be seen that the number of active PoDs increases with the job load. Similarly, the number of active racks in an activated PoD increases almost linearly with the job load. Thus optimization keeps only needed number of PoDs and racks active to serve the job load and others are turned off.

Figure 4.10 compares optimal results gap of CG using heuristic termination. As it depicted, the optimality gap (objective difference over objective optimal value) among the CG using heuristic termination results is less than %0.01 percent for $N=50$. Hence, the difference between our proposed heuristic method and branch and bound method is negligible and we can say it is less than half a percent of the optimal value. However, it is possible to face a bigger optimality gap for larger values of N.

We also examined the quality of the obtained solutions. In Table 4.10, the difference among values of the objective functions of CG/Proposed rounding, IQP are represented. Moreover, results of random rounding algorithm of the relaxed CG RMP solution is considered to represent the upper bound for the performance of our optimization model. It can be seen that the optimality gap between the exact results and upper bound is up to 6 % and the gap between the solution of CG/proposed rounding and that of the IQP is less than 1% for $N<50$. The optimality gap of CG/proposed rounding and the upper bound is attributed to the heuristic nature of the methodology followed for mapping the pure relaxed solution to the integers. This shows that the better and more effective employment of the relaxed to integer conversion results in smaller optimality gap.

Next, we look at the run time of the optimization models in Table 4.11. It can be noticed that as the workload (number of jobs) in the datacenter increases, the runtime of both IQP and CG increase. However, the runtime of the IQP grows exponentially while that of CG almost linearly due to the fact that CG scan much smaller number of configurations.

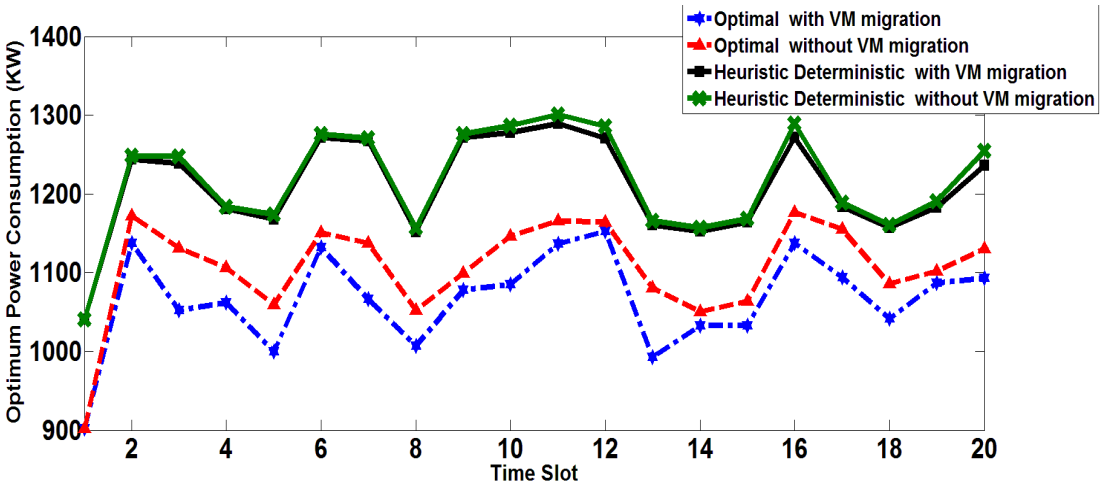

Fig. 4. 3. Optimal power consumption with/without VM migration and power consumption of heuristics with VM migration (with independent VM release time) as a function of time.

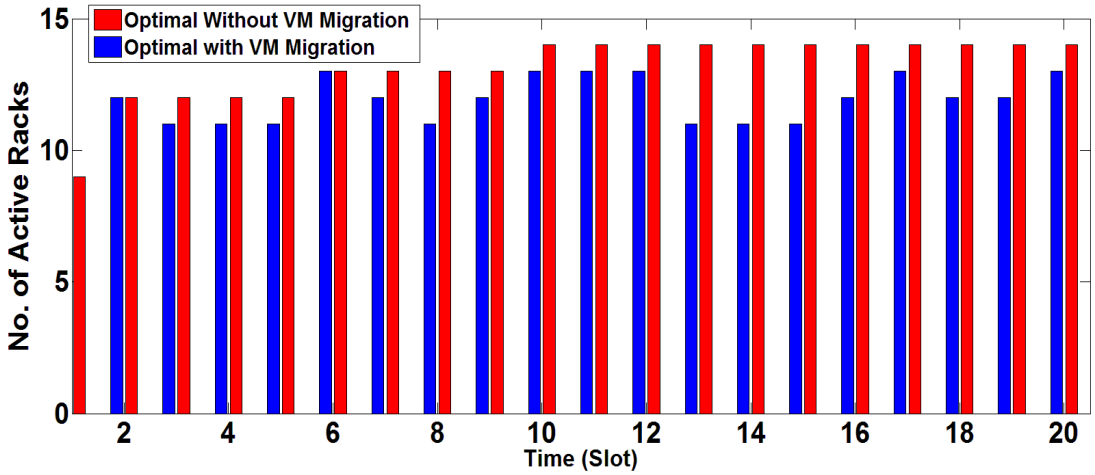

Fig. 4.4. Number of active racks as a function of time with/without VM migration with independent VM release time.

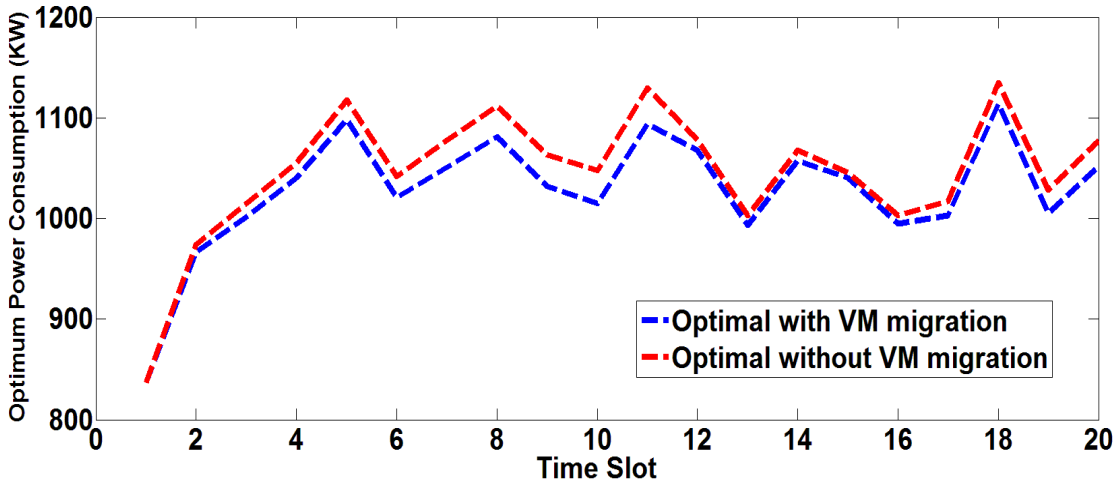

Fig. 4. 5. Optimal power consumption as a function of time with/without VM migration with simultaneous VM release time.

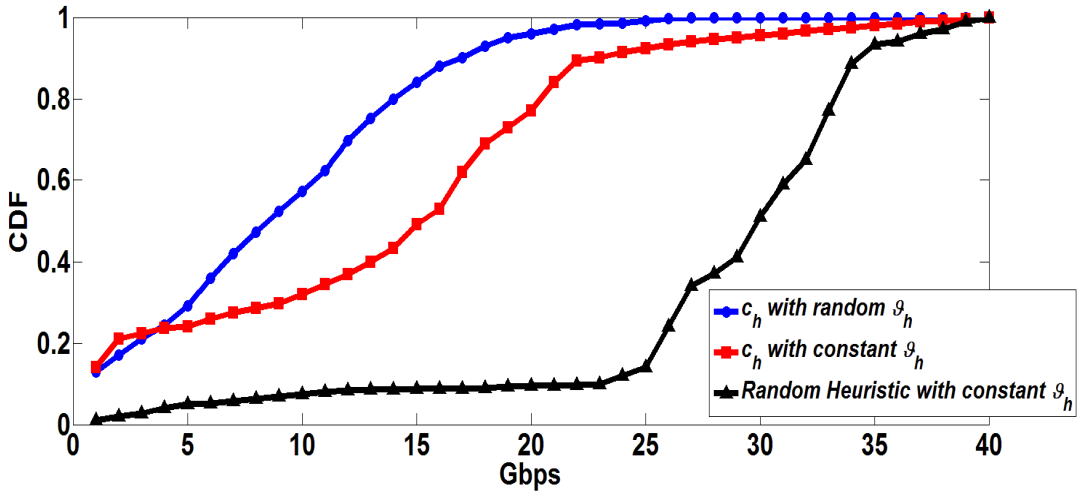

Fig. 4.6. Numerical results of CDF of ToRS to CS links of different models for *N= 350*,

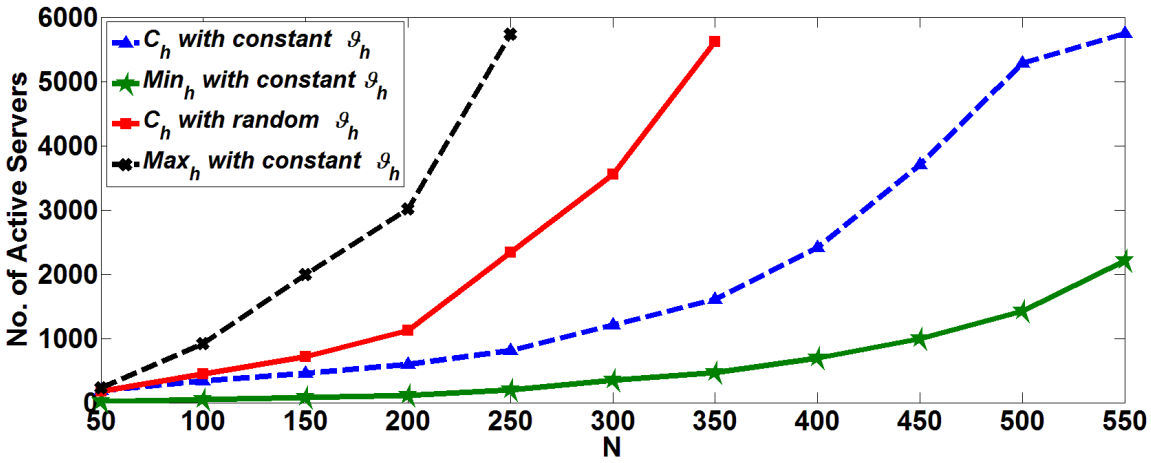

Fig. 4. 7. Number of active servers as a function of total number of jobs in the DC.

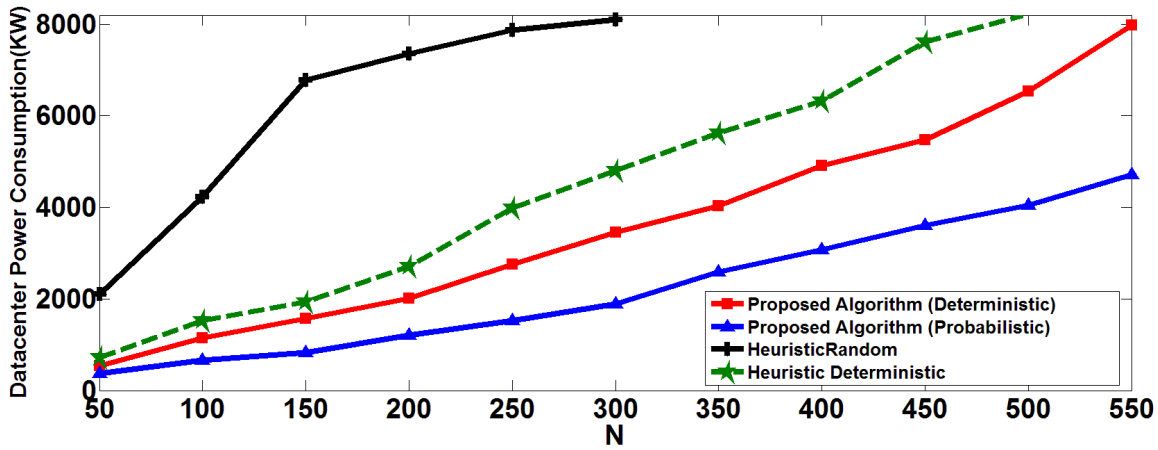

Fig. 4. 8. Optimum power consumed in DC as a function of total number of jobs in the DC

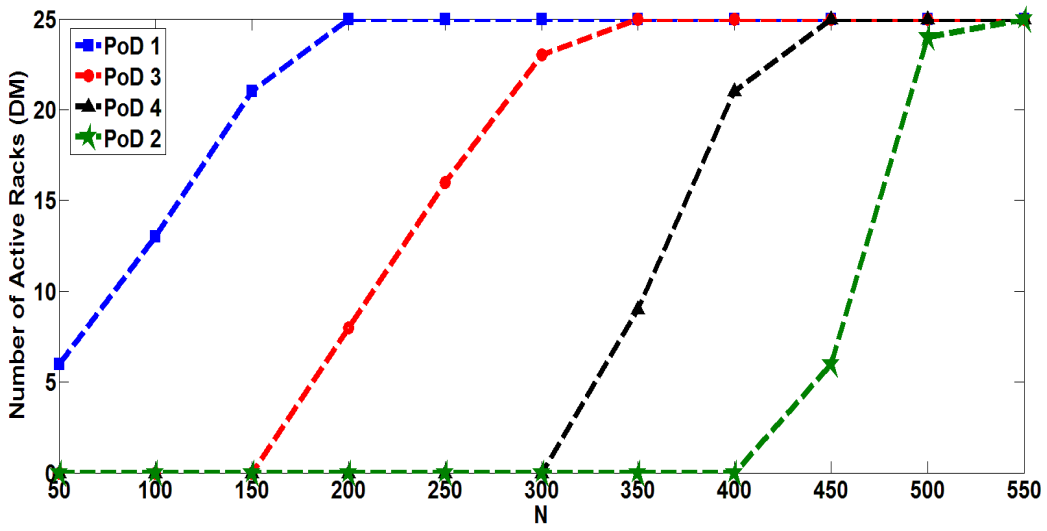

Fig. 4. 9. Number of Active Racks in each PoD as a function of number of jobs

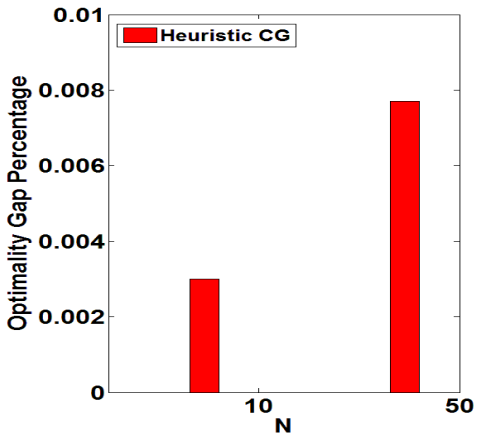

Fig. 4. 10 Numerical results of optimality gap for CG using proposed heuristic rounding method

Table 4.10 Comparison of values of the objective functions among IQP, CG/ Proposed Rounding and CG/Random Rounding

Optimization Method	Value of the objective for different *N*				
	10	20	30	40	50
IQP	16.92	29.2	49.2	92	422
CG/Proposed Rounding	16.95	29.25	49.95	92.35	426.5
CG/Random Rounding	20.7	36.85	59.6	121.5	453.3

Table 4.11 Comparison of the run time between IQP and CG/proposed rounding

Optimization Method	Run Time (hour) for different *N*				
	10	20	30	40	50
IQP	8	31	78.2	111.3	169.2
CG/Proposed Rounding	0.3	1.7	4.5	8	13.2

4.7 Conclusion

In this chapter, we considered the optimization of resource allocation in a cloud computing center. The objective of the optimization problem optimization was scheduling of incoming workloads among servers such that total power consumption of cloud computing center is minimized; both network and server power consumption have been taken into account. First, we formulated energy efficient VM placement problems. Then, a CG based algorithm is presented to determine the number, type and location of the servers that should be used to serve the workloads in order to minimize power consumption of the datacenter. Subsequently, we optimized VM placement problem while there are still unfinished jobs from the previous timeslots. We developed a technique to solve the optimization problem that allows full, partial and no migration of VMs belonging to unfinished jobs. Finally, pattern initialization and heuristic termination algorithms are proposed to reduce complexity of the optimization problem. Numerical results show that the heuristic algorithm yields near to optimal solution under random job arrival process. Optimal results show significant savings power consumption.

Chapter 5

Conclusions and Future Work

In this chapter, we present the conclusions of the research done in this thesis and discuss the future work.

5.1. Conclusions

In this thesis, we have studied performance modeling of cloud computing systems and optimization of resource allocations in these systems. On the topic of performance modeling, we have assumed Poisson arrival of jobs to the system, where a job may consist of multiple numbers of tasks with each task requiring a virtual machine (VM) for its execution. The studied models admit quite general job service time distributions. We considered both constant and variable job sizes in the number of tasks during their service times. In the case of constant job size, we allow different classes of jobs, which are determined through their arrival and service rates and number of tasks in a job. In the variable case a job generates randomly new tasks during its service time. The latter case requires dynamic assignment of VMs to a job, which will be needed in the mobile cloud. In both cases, the system is modeled using birth-death processes. In the case of constant job size, we have derived joint distribution of the number of jobs from each class in the system, job blocking probabilities and distribution of the utilization of resources as a function of the job load under various scenarios for systems with both homogenous and heterogeneous VMs. We have shown that joint distribution of the number of jobs in the system depends on the job service times only through its mean value. We have determined service fragmentation probabilities and have shown application of the derived results in power management techniques under time-varying traffic loads. In the case of

variable job sizes, we have determined distribution of the number of jobs in the system and the average service time of a job for both infinite and finite resources systems.

Next we have studied optimization of resource allocation for a given cloud computing center architecture. We have developed an optimization model that determines the job schedule, which minimizes the total power consumption of the datacenter. It is assumed that power consumption in a datacenter is due to communications and server activities. We assumed a distributed model, where a job may be assigned VMs on different servers, resulting in fragmented job service. In this model, communications among the VMs of a job on different servers is proportional to the product of the number of VMs assigned to the job on each pair of servers which results in a quadratic network power consumption in the number of job fragments. We have applied the CG method to solve this optimization problem for large scale colud computing systems in conjunction with two different algorithms that reduces the complexity and the amount of time it takes to obtain the solution. We have also investigated the impact of stochastic communication rate and computation level on the optimization of resource allocation. Afterwards, we have extended the model to the periodical application of the optimization problem. The extended model solves the optimization problem at discrete-time instants where the load includes new arriving jobs in the present slot as well as unfinished jobs from the previous timeslots. An important contribution of this thesis is development of a technique that solves this optimization problem such that it allows full, partial and no migration of VMs belonging to the unfinished jobs. Numerical results show that the proposed platform yields an approximate optimal solution (optimality gap less than 2 percent) within a limited computing time. The numerical results also show that optimal VM placement results in significant power consumption savings. Thus the proposed optimization will provide significant cost savings to the operators of cloud computing systems.

The main contributions of this thesis are published in [86] and [87].

5.2. Future Work

Next, we present proposals for future research.

5.2.1 Performance modeling of cloud computing systems under nonstationary conditions

The performance modeling in this thesis assumes equilibrium conditions in the cloud computing system. However, the application workloads are time-varying, which results in nonstationary resource demands over time. Thus, we propose to study performance of cloud computing systems under nonstationary conditions. This may be achieved through the prediction of the future workload through the use of historical data. The referenced literature on forecasting includes a time series prediction of the status of distributed system resources such as CPU and available memory based on historical information captured throughout monitoring of the systems. Nevertheless, time series approaches such as Linear Regression (LR) and Moving Window Average (MWA) are not powerful estimators for load prediction. Non-stationary space of the job arrival process makes the LR and MWA approaches too error prone. However, powerful estimators such as Kalman, and particle filters may be used in the prediction of the workloads and available resources. Then, the predicted load may be used to study nonstationary behaviour of the system as a function of the system.

5.2.2 Performance Modeling of Cognitive Cloud Computing Systems

Cloud computing allows different services to be offered by the service providers in the cloud. Among the provided services are access to interactive databases and some web based applications. For these services, the service providers lease cloud resources for a long period of time in order to meet QoS requirements of their users. However, as the result of fluctuation in the total user load, some of the resources leased by the service provider may be idle significant periods of time. The service provider may rent out these resources to the secondary users on the condition that QoS given to the primary users of the service provider is not affected. This could be achieved by giving pre-emptive priority

to primary users over the secondary users. This problem is similar to the sharing of a communication channel by the primary and secondary users in a cognitive radio system, hence the name cognitive cloud computing system. We would like to study performance seen by the secondary users in cognitive cloud computing system.

5.2.3 In Depth Study of VM Migration Policy

We propose an in depth study of the VM migration policy. There are three main objectives of VMs migration in a cloud computing center: reduction of communication network traffic over the DC, reduction of power consumption, and avoidance of server failure. After the prediction of the mentioned events, VM migration policy should be used to determine movement of the computation load away from irresolute servers to other appropriate servers. We note that VM migration has a cost due to interruption of the processing in the migrating VM (VM downtime), the additional power required for the migration and increased communication network utilization. The downtime of VM migration may be modeled as a random variable. The frequency of VM migration should be limited to avoid high cost of VM migration.

In this thesis, we modeled the network considering communication amongst VMs inside the DC. However, inter-traffic communication (out-of-band cloud signals) also exists, which could be as important as the server power consumption and internal traffic in the resource allocation and relocation processes of VMs.

As mentioned earlier, any VM migration causes a slight performance degradation of the application hosted by the VM. The time needed to transfer the VM memory from the source to the target server may vary from a few seconds up to two minutes in the worst cases. As discussed earlier, each VM has a lifetime in the DC. So it is possible that there is a VM existing in a network where its lifetime is going to end. In this situation it might be preferable not to migrate the VM. So in addition to the traffic amongst VMs in the cloud, their lifetime also has to be considered in order to make an optimum decision on VM migration. For example, in the dynamic service demand case the amount of traffic the VM will generate and communicate over the network should also consider job service time, which has been calculated in this thesis. For calculating service time of a job, two different scenarios where VMs service times are totally independent have been investigated. In the scenario where VMs release the system simultaneously, this has been

presented in this thesis. The other scenario is for a correlation between VM service times of a job in the DC. Under these circumstances, it is possible to approximate the service times of a job using the cross correlation of processes presented above.

References

[1] P. Mell and T. Grance. "The NIST definition of cloud computing", National Institute of Standards and Technology, vol. 53, no. 6, pp. 50, 2009.

[2] S. S. Islam, M. B. Mollah, M. I. Huq, M. A. Ullah, “Cloud Computing for Future Generation of Computing Technology”, Proceedings of the 2nd IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, pp. 1-6. May 2012.

[3] D. Jeffrey, and S. Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.

[4] T. White, “Hadoop: the definitive guide”, O'Reilly, 2012.

[5] C. Lam, “Hadoop in action”, Manning Publications Co., 2010.

[6] A. Berl, , E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Quan Dang, and K. Pentikousis. "Energy-efficient cloud computing." The computer journal, vol. 53, no. 7, pp. 1045-1051, 2010.

[7] M. Alicherry, T.V. Lakshman "Network Aware Resource Allocation in Distributed Clouds", Proceedings of IEEE INFOCOM, pp. 963-971, 2012.

[8] X. Meng “Improving the scalability of datacenter networks with traffic-aware Virtual Machinne placement”, Proceedings of INFOCOM, pp. 1-9, 2010.

[9] S .T. Maguluri, R. Srikant, L.Ying, “Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters”, Proceedings of IEEE INFOCOM, pp. 702-710, 2012.

[10] A. Stolyar, “An infinite server system with general packing constraints”, Operations Research Journal, vol. 61, no. 5, pp. 1200-1217, 2013.

[11] H. Khazaei, J. Misic, and V. B. Misic. "Performance analysis of cloud computing centers using M/G/m/m+ r queuing systems." IEEE Transactions on Parallel and Distributed Systems, vol. 23, no 5, pp. 936-943, 2012.

[12] H. Khazaei, J. Miˇsi´c, V. B. Miˇsi´c, and S. Rashwand, “Analysis of a pool management scheme for cloud computing centers,” IEEE Transactions on Parallel and Distributed Systems, vol. 99, no. 5, pp. 849-861, 2012.

[13] H. Khazaei, J. Miˇsi´c, V. B. Miˇsi´c, “Performance of cloud centers with high degree of virtualization under batch task arrivals”, IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 12, pp. 2429-2438, 2013.

[14] D. Bruneo, “A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems”, IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 560-569, March 2014.

[15] B. Yang, F. Tan, Y. Dai, “Performance Evaluation of Cloud Service Considering Fault Recovery”, Journal of Supercomputing, Springer, Vol. 65, pp. 426-444, 2013.

[16] B. Bouterse and H. Perros, “Scheduling Cloud Capacity for Time-Varying Customer Demand”, Proceedings of IEEE Cloud Networking (CLOUDNET), pp. 137-142, 2012.

[17] S .T. Maguluri, R. Srikant, L.Ying, “Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters”, Proceedings of INFOCOM, 2012.

[18] B. Javadi, R. K. Thulasiram, and R. Buyya, “Statistical modeling of spot instance prices in public cloud environments”, Proceedings of the UCC 2011.

[19] Z. Ou, , H. Zhuang, A. Lukyanenko, J. Nurminen, P. Hui, V. Mazalov, and A. Yla-Jaaski. "Is the Same Instance Type Created Equal? Exploiting Heterogeneity of Public Clouds", IEEE Transaction on Cloud Computing, vol. 1 no. 1, pp. 201-214, 2013.

[20] “UnixBench,” http://freecode.com/projects/ unixbench, 2013.

[21] Y. Guo, A L., and A. Walid, "Shadow-Routing Based Dynamic Algorithms for Virtual Machine Placement in a Network Cloud", Proceedings of INFOCOM, pp. 620-628, 2013.

[22] X. Li, n, J. Wu, S. Tang, and S. Lu, "Let’s Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers." Proceedings of the INFOCOM, pp. 1842-1850, 2014.

[23] Q. Zhang, R. Boutaba, L. Hellerstein et al., “Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud”, IEEE Transactions on Cloud Computing, vol. 2, no. 1, pp. 14-28, 2014.

[24] A. Deepal, et al, "Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware Virtual Machine placement", Proceedings of IEEE Services Computing (SCC), pp. 72-79, 2011.

[25] Sh. Vivek, P. Zerfos, K. Lee, H. Jamjoom, Y. Liu, and S. Banerjee, "Application-aware Virtual Machine migration in data centers", Proceedings of INFOCOM, pp. 66-70, 2011.

[26] D. Breitgand and A. Epstein, "Improving Consolidation of Virtual Machines with Risk-Aware Bandwidth Oversubscription in Compute Clouds", Proceeding of INFOCOM, pp. 2861-2865, 2012.

[27] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. "Joint Virtual Machine placement and routing for data center traffic engineering", Proceedings of INFOCOM 2012, pp. 3158–3162, 2012.

[28] X. Meng, V. Pappas, and L. Zhang. "Improving the scalability of data center networks with traffic-aware Virtual Machine placement", Proceedings of INFOCOM, pp. 1-9, 2010.

[29] J. Zhu, D. Li, J.Wu, H. Liu, Y. Zhang, and J. Zhang, "Towards bandwidth guarantee in multi-tenancy cloud computing networks", 20th IEEE International Conference on Network Protocols (ICNP), pp. 1-10, 2012.

[30] K. You, B. Tang, and F. Ding. "Near-optimal Virtual Machine placement with product traffic pattern in data centers". Proceedings of IEEE ICC, pp. 3705-3709, 2013.

[31] C. Assi, Chadi, S. Ayoubi, S. Sebbah, and K. Shaban, "Towards Scalable Traffic Management in Cloud Data Centers.", IEEE Transaction on communications, vol. 62, no. 3, pp: 1-13, 2014.

[32] D. Dolev, D. Feitelson, J. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources", Proceedings of the 3rd Innovations in Theoretical Computer Science ACM Conference, pp. 68-75, 2012.

[33] A. Gutman and N. Nisan, "Fair allocation without trade.", in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, vol. 2, pp. 719-728., 2012.

[34] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types", NSDI, vol. 11, pp. 24-38, 2011.

[35] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: fairness-efficiency tradeoffs in a unifying framework.", IEEE/ACM Transactions on Networking (TON), vol. 21, no. 6, pp. 1785-1798, 2012.

[36] D. Dolev, D. Feitelson, J. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources", Proceedings of the 3rd Innovations in Theoretical Computer Science ACM Conference, pp. 68-75, 2012.

[37]A. Gutman and N. Nisan, "Fair allocation without trade.", Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 719-728, 2012.

[38] A. Gutman, N. Nisan, D. Parkes, A. Procaccia, and N. Shah, “Beyond dominant resource fairness: Extensions, limitations, and indivisibilities”, Proceedings of the 13th ACM Conference on Electronic Commerce, pp. 808-825, 2012.

[39] W. Wei, B. Li, and B. Liang. "Dominant Resource Fairness in Cloud Computing Systems with Heterogeneous Servers", Proceedings ofINFOCOM, pp. 583-591, Toronto, 2014.

[40] S. Sahni and T. Gonzalez, “P-complete approximation problems”, J. ACM on Applied Math., vol. 23, pp. 555–565, July 1976.

[41] R. Hassin, A. Levin, and M. Sviridenko, “Approximating the minimum quadratic assignment problems,” ACM Trans. Algorithms, vol. 6, pp. 181–189, 2009.

[42] Z.Wu, Y. Zhang, V. Singh, G. Jiang, and H. Wang. "Automating Cloud Network Optimization and Evolution", IEEE Journal on Cloud computing, vol. 31, no. 12, pp. 2620-2631, 2013.

[43] A.Gandhi, M.Harchol-Balter. "How data center size impacts the effectiveness of dynamic power management", Proceeding of 49th IEEE Annual Allerton Conference on Network Management, pp. 1164-1169, 2011.

[44] V. Venkatachalam, M. Franz, “Power reduction techniques for microprocessor systems”, ACM Computer Survey journal (CSUR), vol. 37, no. 3, pp. 195–237, 2005.

[45] X. Fan, W.D. Weber, L.A. Barroso, “Power provisioning for a warehouse-sized computer”, in Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), ACM New York, pp. 13–23, 2007.

[46] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. "Resource pool management: Reactive versus proactive or let’s be friends", Elsevier Computer Networks, vol. 53, no. 17, pp. 2905-2922, 2009.

[47] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M.A. Kozuch, "Autoscale: Dynamic, robust capacity management for multi-tier data centers.", ACM Transactions on Computer Systems (TOCS), vol. 30, no. 4, pp. 14-26, 2012.

[48] MR Rahimi, J Ren, CH Liu, AV Vasilakos, N Venkatasubramanian, “Mobile cloud computing: a survey, state of art and future directions”, Mobile Netw. Appl., vol. 19, no. 2, pp. 133–143, 2014.

[49] N Fernando, SW Loke, W Rahayu, “Mobile cloud computing: a survey”, Future Generation Comput. Syst., vol. 29, no. 1, pp. 84–106, 2013.

[50] H. T. Dinh, C. Lee, D. Niyato and P. Wang, “A Survey of mobile cloud computing: architecture, applications and approaches”, Wireless Communications and Mobile Computing, vol. 13, no. 18, pp. 1587-1611. 2013.

[51] S Kosta, A Aucinas, P Hui, R Mortier, X Zhang, “ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading”, Proceedings of INFOCOM, pp. 945–953, 2012.

[52] K. Kumar, Y.H. Lu, “Cloud Computing for mobile users: Can Offloading Computation Save Energy”, IEEE Computer Magazine, vol. 4, no. 1, pp. 51-56, April 2010.

[53] J. F. Kaufman, “Blocking in a Shared Resource Environment”, IEEE Trans. Communications, vol 29, pp. 1474–1481, 1981.

[54] M. Mehmet-Ali. "Call–burst blocking and call admission control in a broadband network with bursty sources", Performance Evaluation, vol. 38, no. 1, pp. 1-19, 1999.

[55] J.W. Roberts, “A service system with heterogeneous user requirements”, Performance of Data Communications Systems and their Applications, North-Holland, Amsterdam, pp. 423–431, 1981.

[56] Z. Dziong, J.W. Roberts, Congestion probabilities in a circuit-switched integrated services network, Performance Evaluation, vol. 7, no. 1, pp. 267–284, 1987.

[57] M.V. Ramakrishna, “An exact probability model for finite hash table”, Proceedings of IEEE Fourth International Conference on Data Engineering, pp. 362-368, February 1988.

[58] A. Gandhi, M. Harchol-Balter, R. Das, and Ch. Lefurgy. "Optimal power allocation in server farms." ACM SIGMETRICS Performance Evaluation Review, vol. 37, no. 1, pp. 157-168, 2009.

[59] L. Kleinrock, “Queuing Systems”, vol. I. John Wiley& Sons, 1976.

[60] M. A. M. Ferreira and M. Andrade, “The M/G/∞ queue busy period distribution exponentially”, Journal of Appl. Math. vol. 4, no. 3, pp. 249-260, 2011.

[61] J. R. Artalejo, and M. J. L Herrero, "Analysis of the busy period for the M/M/c queue: An algorithmic approach", Journal of Applied Probability, pp. 209-222, 2001.

[62] T. Kimura, "A transform-free approximation for the finite capacity M/G/s queue", Operations Research, vol. 44, no. 6, pp. 984-988, 1996.

[63] V. Chvatal, “Linear Programming. Macmillan”, W. H. Freeman and Company, New York - San Francisco, 1983.

[64] F. Cedric, H. Liu, B. Koley, X. Zhao, V. Kamalov, and V. Gill. "Fiber optic communication technologies: What's needed for datacenter network operations", IEEE Communications Magazine, vol. 48, no. 7, pp. 32-39, 2010.

[65] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks", ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 242-253, 2011.

[66] D. Xu, X. Liu, and Z. Niu, “Joint Resource Provisioning for Internet Datacenters with Diverse and Dynamic Traffic”, IEEE Transactions on Cloud Computing, vol. PP, no 99, pp. 1-14, 2014.

[67] H. Xu and B. Li, “Cost Efficient Datacenter Selection for Cloud Services”, Proceeding of IEEE 1st International Conference on Communications in China (ICCC), pp. 51-56. 2012.

[68] D. Niu, C. Feng, B. Li, “Pricing Cloud Bandwidth Reservations under Demand Uncertainty”, ACM SIGMETRICS Performance Evaluation Review, vol. 40, no. 1, pp. 151-162, 2012.

[69] K. Mills, J. Filliben, and Ch. Dabrowski. "Comparing vm-placement algorithms for on-demand clouds." In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 91-98, 2011.

[70] G. Juve , E. Deelman, V. Karan Vahi, M. Gaurang, B. Berriman, P. Berman, and P. Maechling, "Scientific workflow applications on Amazon EC2", 5th IEEE International Conference on In E-Science Workshops, pp. 59-66. 2009.

[71] M Wang, X Meng, L. Zhang, “Consolidating virtual machines with dynamic bandwidth demand in data centers”, Proceedings of INFOCOM, pp. 71–75, 2011.

[72] R. Stanojevic and R. Shorten, “Distributed dynamic speed scaling”, Proceedings of INFOCOM, pp. 1-5, 2010.

[73] G. Chen, and et.al, “Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services,”, USENIX NSDI, vol. 8, no. 1, pp. 337-350., 2008.

[74] X. Zhang, H. Wang, Z. Xu, and X. Wang. "Power Attack: An Increasing Threat to Data Centers.", Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS, pp. 132-147, 2014.

[75] KC. Poldi, and M. Nereu Arenales, "Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths", Computers & Operations Research, vol. 36, no. 6, pp. 2074-2081, 2009.

[76] A. I. Hinxman, "The trim-loss and assortment problems: A survey", European Journal of Operational Research, vol. 5, no. 1, pp. 8-18, 1980.

[77] G.Wäscher, and Th. Gau. "Heuristics for the integer one-dimensional cutting stock problem: A computational study." Operations-Research-Spectrum, vol. 18, no. 3, pp. 131-144, 1996.

[78] D. Carvalho, JM Valério, "LP models for bin packing and cutting stock problems", European Journal of Operational Research, vol. 141, no. 2, pp. 253-273, 2002.

[79] A. Wolsey, “Integer programming”, New York: Wiley, vol. 42, 1998.

[80] Sh. Srikantaiah, A. Kansal, and F. Zhao. "Energy aware consolidation for cloud computing", Proceedings of the IEEE conference on Power aware computing and systems. vol. 10, 2008.

[81] A. Beloglazov, J. Abawajy, and R. Buyya. "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", Future generation computer systems, vol. 28, no. 5, pp. 755-768, 2012.

[82] B. Barroso, L. André, J. Dean, and U. Holzle. "Web search for a planet: The Google cluster architecture." IEEE Micro Journal, vol. 23, no. 2, pp. 22-28, 2003.

[83] S. Aleksic, “Analysis of power consumption in future high-capacity network nodes”, IEEE/OSA Journal of Optical Communications and Networking, vol. 1, no. 3, pp. 245-258, 2009.

[84] P. Pries, M. Rastin, M. Jarschel, D. Schlosser, M. Klopf, and P. Tran-Gia, "Power Consumption Analysis of Data Center Architectures", Green Communications and Networking, Springer Berlin Heidelberg, pp. 114-124, 2012.

[85] K. Kant, "Power control a high speed network interconnects in data centers", Proceeding of INFOCOM Workshops, pp. 1-6, 2009.

[86] S.Vakilinia, MM. Ali, and D. Qiu. "Modeling of the Resource Allocation in Cloud Computing Centers", Computer Networks, 2015.

[87] S. Vakilinia, D. Qiu, and MM. Ali. "Optimal multi-dimensional dynamic resource allocation in mobile cloud computing." EURASIP Journal on Wireless Communications and Networking 2014, no. 1. pp. 1-14, 2014.