DATA CENTER RESOURCE MANAGEMENT WITH TEMPORAL DYNAMIC WORKLOAD

A DISSERTATION
in
Computer Networking
and
Telecommunications Networking

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
Haiyang Qian

M. S., Technical University of Denmark, Lyngby, Denmark, 2006
B. S., Nanjing University of Science & Technology, Nanjing, China, 2002

Kansas City, Missouri
2012

DATA CENTER RESOURCE MANAGEMENT WITH TEMPORAL DYNAMIC WORKLOAD

Haiyang Qian, Candidate for the Doctor of Philosophy Degree

University of Missouri–Kansas City, 2012

ABSTRACT

The proliferation of Internet services drives the data center expansion in both size and the number. More importantly, the energy consumption (as part of total cost of ownership (TCO)) has become a social concern. When the workload demand is given, the data center operators desire minimizing their TCO. On the other hand, when the workload demand is unknown while the requirements on quality (QoE) of experience of the Internet services are given, the data center operators need to determine appropriate amount of resources and design redirection strategies in presence of multiple data centers to guarantee the QoE.

For the first problem, we present formulations to minimize server energy consumption and server cost under three different data center scenarios (homogeneous, heterogeneous, hybrid hetero-homogeneous clusters) with dynamic temporal demand. Our studies show that the homogeneous model significantly differs the heterogenous model in computational time. To be able to compute optimal configurations in near real-time for large scale data centers, we propose aggregation by maximum and aggregation by mean modes for different Internet service requirements. In aggregation by maximum mode, the price of reducing computational time is over-provisioning (which causes extra energy consumption). In aggregation by mean mode, the price is the degradation of the timeliness of services. However, they still result in significant cost savings compared to the scenario when *all* servers are *on* during the entire duration. We first introduce an intuitive aggregation method: static aggregation. For each mode, dynamic aggregation is introduced to alleviate their individual drawbacks. The dynamic aggregation by maximum results in cost savings up to approximately 18% over the static aggregation by maximum. For three random distributed workload cases, the dynamic aggregation by mean can save up to approximately 50% workload reallocation compared to static aggregation. Dynamic Voltage/Frequency Scaling (DVFS) capacity is further considered in our model. Our numerical results show that adopting DVFS results in significantly reduction of energy consumption.

For the second problem, the data center provides resources via the cloud computing model. We propose a hierarchical modeling approach that can easily combine all components in the data center provisioning environment. Identifying interactions among the components is the key to construct such a model. In providing internet service by cloud computing hosted in data centers, we first construct four sub-models: an outbound bandwidth model, a cloud computing (hosted by data centers) availability model, a latency model and a cloud computing response time model. Then we use a data center redirection strategy graph to glue them together. We also introduce an all-in-one barometer to ease the QoE evaluation. The numeric results show that our model serves as a very useful analytical tool for data center operators to provide appropriate resources as well as design redirection strategies.

In addition, we study the redirection strategies (schemes) in a particular Internet service, agent-based virtual private networks architecture (ABVA). It refers to the environment where a third-party provider runs and administers remote access virtual private network (VPN) service for organizations that do not want to maintain their own in-house VPN servers. We consider the problem of optimally connecting users of an organization to VPN server locations in an ABVA environment so that request denial probability and latency are balanced. A user request needs a certain bandwidth between the user and the VPN server. The VPN server may deny requests when the bandwidth is insufficient (capacity limitation). At the same time, the latency perceived by a user from its current location to a VPN server is an important consideration. We present a number of strategies regarding how VPN servers are to be selected and the number of servers to be tried so that request denial probability is minimized without unduly affecting latency. These strategies are studied on a number of different topologies. For our study, we consider Poisson and non-Poisson arrival of requests under both finite and infinite population models to understand the impact on the entire system. We found that the arrival processes have a significant and consistent impact on the request denial probability and the impact on the latency is dependent on the traffic load in the infinite model. In the finite model, arrival processes have an inconsistent impact to the request denial probability. As to the latency in the finite model, arrivals that have a squared co-efficient of variation less than one is consistently largest, followed by Poisson case, then the case that the squared co-efficient of variation is more than one. Finally, a strength of this work is the comparison of infinite and finite models; we found that a mismatch between the infinite and the finite model is dependent both on the number of users in the system and the load.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled “Data Center Resource Management with Temporal Dynamic Workload,” presented by Haiyang Qian, candidate for the Doctor of Philosophy degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Deep Medhi, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Khosrow Sohraby, Ph.D.
Department of Computer Science & Electrical Engineering

Appie van de Liefvoort, Ph.D.
Department of Computer Science & Electrical Engineering

Kenneth Mitchell, Ph.D.
Department of Computer Science & Electrical Engineering

Cory Beard, Ph.D.
Department of Computer Science & Electrical Engineering

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Deep Medhi for all his guidance, mentoring and support throughout my doctoral research. Dr. Medhi's guidance covers all levels of research: from forming the ideas all the way down to the scratch.

I sincerely thank all of the other members of my committee, Dr. Khosrow Sohraby, Dr. Appie van de Liefvoort, Dr. Kenneth Mitchell and Dr. Cory Beard, for all their help when I have approached them with questions. Their comments have helped to clarify and improved this work a lot.

I am indebted to Dr. Kishor Trivedi from Duke University for his inspiring guidance on the hierarchical model and tutoring on SHARPE package. I would like to thank my master thesis advisor, Dr. Villy B. Iversen for his encouragement to pursue Ph.D. degree.

I would like to acknowledge the support I received from National Science Foundation under grant No. CT-08-31090 and CNS-09-16505. I would also like to thank all the lab mates, faculty, staff, colleagues, friends because even a small exchange of smiles may have enlightened my day.

Last and most importantly, I would like to thank my parents, QIAN Jinlong and GAO Yanyun for their patience, understanding, encouragement, and support for these years during my Ph.D. course. Their love has been and will always be my momentum to move forward.

CHAPTER 1

INTRODUCTION

The computing clusters are becomes more and more efficient. Consequently, the computational capacity provided by unit watt constantly rises [49], which conforms to Moore's Law [60]. However, driven by the the needs of society as well as the huge commercial profits of successful precedents, the proliferating and continuously scaling up Internet services, such as search(*e.g.*, Google [4], Bing [2]), social-networking(*e.g.*, Facebook [3], Twitter [7], Linkedin [5]), online-gaming (*e.g.*, Zynga [8]), real-time authentication products (*e.g.*, PhoneFactor [6, 74, 75]), *etc.*, demands more and more computing capacity at a even higher speed. Therefore, the power drawn by data centers constantly increases. Table 1 [49] presents estimated average power usage for three classes of servers.

The cost of data centers has become one of major social and economic concerns. It has been reported that the power consumption in data centers has increased 400% over the past decade [36]. Data centers consumed 61 billion kilowatt-hours of power in 2006, according to a report of U.S. Environmental Protection Agency (EPA) in 2007 [84]. That is 1.5 percent of all power consumed in the United States–at a cost of 4.5 billion U.S. dollars. Moreover, data center energy cost is approaching overall hardware cost [12]. The carbon footprint of data centers contribute to the global warming. People becomes more and more aware of the needs of "go green" for data centers. From both economic and environmental perspectives, reducing energy consumption in data centers is an ever high

Table 1: Estimated average power consumption per server class (W/Unit) from 2000 to 2006

Server Class	2000	2001	2002	2003	2004	2005	2006
Volume	186	193	200	207	213	219	225
Mid-range	424	457	491	524	574	625	675
High-end	5,534	5,832	6,130	6,428	6,973	7,651	8,163

appeal.

1.1 Data Center Cost Components

There are infrastructure expenditure (CAPEX) and operational expenditure (OPEX) in a data center. The energy consumption belongs to OPEX. The energy is consumed by multiple units in data centers, such as servers, cooling, power distribution loss. Table 2 presents a rough energy consumption cost guide given by [39] in a 50,000 server data center. In this dissertation, we focus on minimizing the costs of servers, i.e., energy consumption to run servers and CAPEX of servers.

Table 2: The energy cost components of a 50,000 server data center

Amortized Cost	Component	Sub-Components
45%	Servers	CPU, memory, storage systems
25%	Infrastructure	Power distribution and cooling
15%	Power draw	Electrical utility costs
15%	Network	Links, transit, equipment

1.2 Cloud Computing and Dynamics of Workload

The popular pay-as-you-go model adopted by many cloud computing providers (*e.g.*, Amazon EC2 [1], Rackspace Cloudservers$^{\text{TM}}$) is one of the most attractive features for customers whose demands on resources vary over time. Just like consuming any other utilities (*e.g.*, water, gas), customers pay for what they have consumed. Cloud computing providers use data centers as underlying infrastructure and provision resources to Internet services [11, 42, 90].

The real-world data center workload trace reveals that the workload is highly dynamic [10,13,26,28,65]. The workload fluctuates over timescales of minutes, hours, days. For example, the fluctuation of number of users logged on to Windows Live Messenger can be about 40% of the peak load within a day [28]. The fluctuation can be represented by "peak-to-mean" ratio [55] or the characteristics of distributions, such as, mean, variance and squared coefficient of variance(SCV) [74]. As another example from our previous work to characterize the pattern of the number arrivals (requests) in PhoneFactor [6] which is a double authentication service, a diary pattern was found. The weekends have much less number of requests and peak load appears on the mid morning for weekdays. Fig. 1 depicts this pattern from a 4-week trace from the PhoneFactor authentication service [74].

Nevertheless, the workload are very different from case to case due to the variety of Internet services synthesizing workload. The common point of Internet service workloads is they all vary over time. By characterizing history workload trace, predicting future workload demand pattern, the synthetic workload demand can be generated. There are many works on predicting workload in the data center environments, *e.g.*, [38, 86]. Thus we assume the workload is predictable in this dissertation. That is, we have the full knowledge of the workload. In this dissertation, the workload demand is given at a

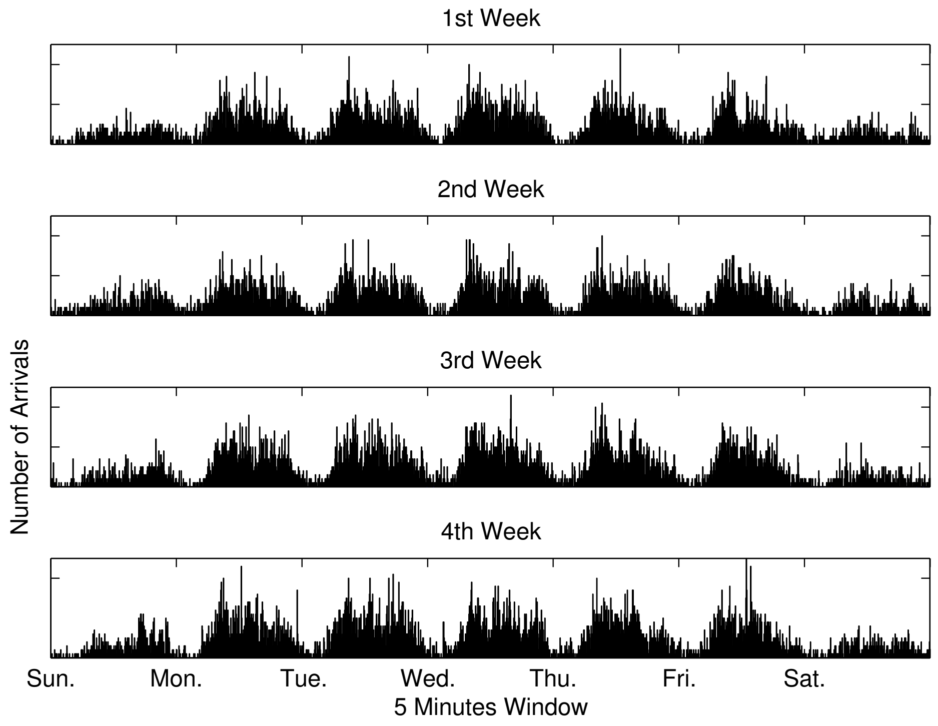


Figure 1: The number of arrivals (requests) of 5-minutes window of 4-week trace from PhoneFactor authentication service

5-minute interval. A 5-minute interval is called a time slot. The beginning of each slot is referred to as the *review point*. In this dissertation, we define the workload at certain time slot as the total amount of resources required by both new arriving jobs at the current review point and jobs that arrivals in previous review points and are still being served.

1.3 Dynamic Provisioning

The data center operators have to provision enough resources to satisfy the peak workload demand. Statically provisioning results in over-provisioning for off-peak workload demand. The the power needed at peak workload is called peak power. It has been found that there is significant power consumption when the processor is idle (*i.e.,* it is at "base power" [65]). It is measured that idle servers consume more than 66% of the peak power [28, 35]. The base power cannot be reduced unless unused hosts are powered off. Processor (CPU) utilization can be used as an indicator of power consumption because the I/O and memory activities are correlated to processor utilization and power consumption is a monotonically increasing function with regard to processor utilization [20]. Many processors have the capability of DVFS, which allows processors to scale the frequency up or down as needed. The cubic relationship between the power consumption and frequency is commonly used [29, 32]. This relationship is given by:

$$\text{Power Consumption} = P_{fixed} + P_f \times (\text{frequency})^3, \tag{1.1}$$

where P_{fixed} is the fixed component and P_f is the coefficient of frequency related component.

Virtualization, such as, Linux VServer, Parallels, VMware, Xen, is the technology that enables different services to run in a virtually isolated environment and allows resources that are allocated to these services scale up and down transparently and seamlessly [24]. Cloud computing providers use one virtual machine (VM) (It is called instance in Amazon EC2) as the finest granularity of resources to offer the service. Consolidation is the technology utilizes the capability of migration to consolidate workload into minimum number of running servers and maximize the number of idle servers [1] [81]. VMware "Distributed Power Management" (DPM) in VMware's VSERVER adopts the

[1]"Minimum number of running servers" does not refer to fully utilize the server since we may not fully utilize servers on purpose to guarantee Quality of Service (QoS).

consolidation concept to improve the power efficiency [85]. The way of assigning servers is called workload dispatching. Assigning resource (servers) in the Last-Released-First-Used (LRFU) manner (equivalent to Last-Come-First-Served(LCFS) or Last-In-First-Out (LIFO) can minimize the needs of migration for consolidation [55]. That is, we use the server that is released most recently when the workload increases. We always assume LRFU job dispatching in this dissertation.

Above facts suggest that data center operators consolidate jobs into minimum number of servers and switch idle servers off. However, switching servers on/off results in wear-and-tear cost and consolidation cost. The hard disk is the most vulnerable part in data center infrastructure – majority (78%) of hardware failure/replacement is due to hard disks [87]. To represent the "wear-and-tear" costs of switching on and off, we amortize the CAPEX of servers by dividing the average price of server by the average number of switching on/off cycles of the hard disk. In addition, the source server need to run additional time to keep the states of running application when applying consolidation. This additional time consumes extra energy. The presence of relationships between the states of adjacent time slots requires a global optimization framework in the entire planning horizon. We use integer programming to determine the optimal number of running servers at review points such that the costs of energy consumption and CAPEX of servers are minimized in the entire planning horizon. The optimal solution should be computed in an approximately real-time manner, even for large scale data centers, such as, data centers with thousands of servers.

1.4 Quality of Experience of Internet Services

The users are served by the Internet service providers (ISeP) who in turn is the customer of the cloud computing service provider (CSP) [11]. And CSP use data centers as their infrastructure. *Quality of Experience (QoE)* is critical for service providers since their revenue depends on it. [48]. Being able to receive service and the time needed to wait comprises QoE for online service users. ISeP desire to be able to predict the QoE of their users. The service performance described in a service level agreement (SLA) includes response time, utilization, throughput, and availability, just to name a few. [89]. However, these aspects consist of a promised service level that is delivered from data center to ISePs. None of these aspects is suitable to be used as a direct barometer of QoE of the online service. Note the response time in a data center, however, does not necessarily correlate to QoE since (a) the response time experienced by users is the summation of Internet delay and the response time in the data center, (b) the redirection functionality further complicates the QoE. In addition, it is desirable for ISePs to be able to evaluate the QoE by a single metric which comprehensively addresses the service delivered by CSP and underlying data centers, the Internet's conditions, and redirection strategies.

1.5 ABVA FrameWork

The ABVA framework has been originally described in [57, 68]. A high-level conceptual view of the ABVA framework with users, routers, RPs, and organizations is presented in Fig. 2. The RPs are marked as RP1, RP2. Here, users U1(a) and U1(b) are associated with organization ORG1, and user U2 with organization ORG2. The Internet serves as the native network that has IP routers identified as R1, R2, and so on. When user U1 wants to connect to organization ORG1 for remote-access VPN services, it is actually connected to RP1, which in turn connects to organization ORG1; similarly, user U2 connects to organization ORG2. The underlying path from the user to the RP is over the Internet; thus, a user can access it from anywhere in the Internet. Certainly, the Rendezvous Points must have presence in the Internet. We will interchangeably use the terms, Rendezvous Points and VPN servers.

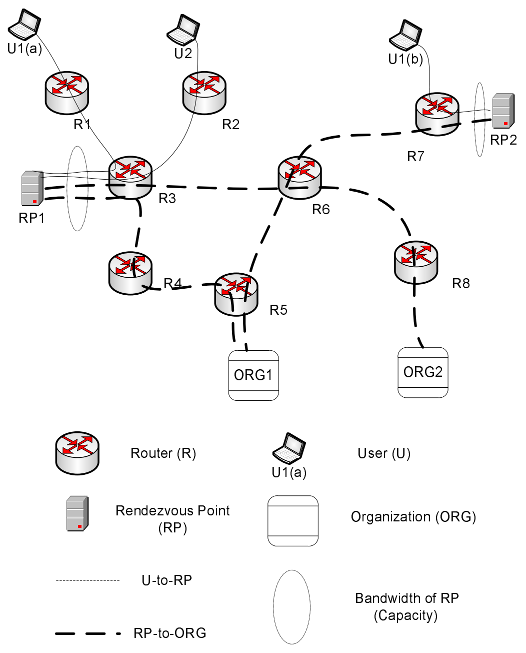


Figure 2: ABVA: basic conceptual framework

The path from an RP to an organization can be over the public Internet or over a private network; for ease of illustration, the figure shows only the former case. Usually, this path is provided based on service level agreements between the ABVA provider and the native Internet service provider, by specifying acceptable quality of service and bandwidth guarantees. For instance, the path from an RP to an organization can be set up as an MPLS tunnel that can provide such guarantees; or, private dedicated circuits may be used between an RP and an organization.

With the above background, we are now ready to provide further clarity to the latency and bandwidth denial aspects of the problem. Based on an arriving user's request, latency refers to latency for the segment between the user and the RP it is connected to; the part from an RP to the organization is bounded by the SLA and is, therefore, not required to be included in this consideration. Certainly, this part must be acceptable to begin with. The latency from a user to an RP is important to consider for two primary reasons: 1) the latency is entirely over the public Internet, 2) once connected, during the entire duration of the session, this latency (with some acceptable fluctuation) is perceived by the user; for example, when downloading a file, watching a video clip, or accessing email on the agent-based VPN. It should be noted that in general, a user location is not fixed; a user may move from a particular location to another overnight and access from a different location (e.g., hotel instead of home), and thus, a particular user should not always be tied to a specific RP. Since the bandwidth guarantee is to be ensured on the path from an RP to the organization, it means that if a specific RP does not have the ability to handle a request, the request will be denied. That is, bandwidth guarantee is not factored in directly on the public Internet part (i.e., from the user location to an RP); however, a user (and the organization it belongs to), if accessing from a particular access technology, might have certain expectations. For instance, if the access technology is

DSL, the user would expect/perceive the bandwidth to be at a certain rate, as opposed to a dial-up connection. Based on the above discussion as well as the agent-based VPN service offering of [67], it is reasonable to build the system model by considering latency from a user to an RP (VPN server) given bandwidth guarantee at the entrance of the RP to the provider service. Thus, starting in the next section, we focus on the framework of modeling this system.

1.6 Scope and Contribution of this Dissertation

In this dissertation, we focus on two aspects of data center resource management. When the resource requirements are already given, how to manage the resource. It is desired to minimize the total cost comprised by energy consumption of running servers and CAPEX of servers in data centers. For sake of brevity, we call the total cost *server operational cost*. The data centers operators can use our proposed methods to manage resources (*e.g.*, CPU resources) such that data center server operational cost is minimized.

The other one is when the data center operators only have knowledge of the requirements on quality of experience (QoE) between the user and the Internet service provider (ISeP). They want to determine how many resources are needed to to provide to the ISeP to guarantee QoE. The cost in this problem is the abstraction of the resource assigned to the service to achieve certain reliability, outbound bandwidth, *et. al.* Our proposed modeling approach can help data center operators allocate resources to the services in their geographically dispersed data centers such that QoE is guarantted by minimum resources. [2] Both problems fall into data center resource management scope.

Understanding the characteristics of data centers as well as Internet services is indispensable to solve both problems. Actually, the characteristics of data centers and Internet services are challenging part of the work. In the following subsections we list the characteristics of data centers and Internet services we need to be aware of to solve the problems. The contributions of this dissertation are as follows.

- We formulate the data center server operational cost minimization problem in three data center server environments: homogenous, heterogeneous, hybrid hetero-homogeneous data centers when the resource workload requirements are temporally dynamic. Workload demand aggregation is proposed to reduce computational time to solve the problem. Part of this work has been published in [70].
- We further formulate the same problem with taking Dynamic voltage/frequency scaling (DVFS) into considering. This work has been published in [72].
- We propose a hierarchical modeling approach to evaluate the QoE of users. This model can be used to determine the amount of data center resources for a Internet services. This work has been published in [73].
- We conduct a study on a particular Internet service, called agent-based VPN, and propose redirection strategies (schemes) to balance the service denial probability and latency. Based on this work, a conference paper was initially published in [68], which was later extended with additional results for a journal paper [69].

[2] On the hand, Internet service provider can use our models to select data centers. These are two sides of the same story.

1.7 A Literature Survey

A significant amount of work focuses on saving energy on servers in data centers [16–19,22,28,29,32,51,55,64,65], initiated by the pioneering work [27,65] a decade ago. Aiming at reducing data center server operational cost, these works using different techniques, such as mathematical programming, queuing theory, control theory, machine learning to model the problem, develop algorithms and heuristics. A recent comprehensive survey can be found at [15].

Reducing server energy consumption can be performed at different levels. Each level has cost components associated with it. At the individual physical server level, the capacity of server/its components is scaled according to load. The components can be processor, memory or storage system. The load of processor is usually called as utilization. As we have mentioned before, processor utilization can be used to indicate the power consumption. Processors with DVFS capacity (*e.g.*, Transmeta's Crusoe processor and Intel's Pentium with SpeedStep) can scale the voltage/frequency in light with load and therefore reduce the power consumption. Barroso and Hölzle proposed the term *energy proportinality* to refer to the energy consumption proportional to load [13]. The cost associated with physical server level is the power consumption.

At the cluster level, the decisions are made regarding how many servers are needed. Switching the server on and off as needed was originally proposed by Pinheiro *et. al.* [65] (named as "VOVO" in their work) to save energy. The cost associated with cluster level is the wear-and-tear cost.

At virtualization level, migration and consolidation are used to consolidate workload into less number of virtual machines, *i.e.,* VMware's Dynamic Power Management [85]. The cost associated with virtualization level is consolidation cost.

Many works in the literature consider more than one level of factors. [65] uses the Proportional-Intergral-Differential (PID) method based on the control theory to predict the demand for next decision point. Basically, this method takes the current demand status, previous accumulated demand status, and demand change speed into consideration and gives different weights to decide the demand for the next decision point. Bichler *et. al.* use mixed-integer programming to formulate the capacity planning problems for virtualized servers [19]. They also identify the "static server allocation problem" as an instance of bin packing problem. However, DVFS is not considered in their work. Their evaluation is based on historical traces. Petrucci *et. al.* consider both turning on/off and DVFS to formulate the so called "virtual server cluster configuration problem" by mixed integer programming and present an algorithm to dynamically manage server clusters [62–64]. They propose to devise a control loop to periodically run the optimization problem to adapt the time-varying incoming workload of multiple applications. They formulate the problem in single time period and thus can only take the turning on/off cost based on adjacent time periods into consideration, which results in a local optimum. Chen *et. al.* first formulate the objective function, then predict the first and second moment of the next interval arrivals and finally calculate the SLA constraint in terms of delay based on $G/G/m_i$ queue [29]. They also propose an approach based on control theory and hybrid approach of queueing and control theory as alternatives. Most of these works evaluated their approaches based on a relatively small server cluster size (such as around 10), while some have considered a cluster size around 100.

How the dynamic workload structure affects the optimal solution and how to use the workload structure to determine the optimal solution has not been paid a lot of attention. [55, 71] decompose the workload into certain substructures and utilize the substructures to determine the optimal solution. Lu and Chen also presents method to solve the problem when workload is unpredictable [55].

Garfinkel conducted a number of experiments to evaluate the Amazon's Grid Computing services. Li *et. al.* benchmarked different cloud services [52]. A number of cloud computing hardware reliability issues are identified for future research in [87]. Kalyanakrishnan *et. al.* used collected data to study the host reliability from the perspective of the users [44]. Bodik *et. al.* discussed the affect of turning machines on and off on reliability in [21].

In general, the Agent-based virtual private networks architecture (ABVA) forms a service overlay network (SON). There has been significant research on SON in recent years [30], [54]. Duan *et al.* [30] took QoS, traffic demand distributions and bandwidth costs into account to study the bandwidth provisioning problem of a general service overlay problem. Liu *et al.* [54] studied content delivery service of SON and proposed algorithms to place replicas to maximize the total throughput of the service delivery tree. There have been a number of works on VPNs [37], [61], [78]. Duffield *et al.* [31] studied the resource management problem of proposed "hose" VPN service model which simplifies SLA for users while complicates resource management for providers compared with traditional customer-pipe model. To the best of our knowledge, our work is the first one to study system management problems in the ABVA environment.

Accessing nearby servers/caches for a variety of network environments to either minimize delay or maximize the utilization of network resources has drawn attention for many years [41], [80], [66], [66]. Guyton and Schwartz [41] analyzed the performance of several techniques for locating nearby server replicas. Shaikh *et al.* [80] have proposed using DNS servers as a redirection service to select the closest server. Locating nearby server though an important problem, is separate from our main concern. Plaxton *et. al.* [66] proposed an efficient algorithm with small additional memory for each node to access the nearby copies of objectives that are continuously being created, replicated, and destroyed. Krishnan *et al.* [50] presented algorithms on where to place caches to minimize the overall flow or delay. The RPs in ABVA are static in the sense that their locations and capacity are fixed and these information is *priori* known. We consider the dynamics of user arrival and server service time in this work. It is not surprising that accessing nearby servers has been widely applied to other SON optimization problems, such as Content Distributions Network (CDN) [23], P2P network [82].

The service denial problem, due to capacity limitation, has been extensively studied in various loss models since the seminal work by A. K. Erlang [33], [34]. An SON environment (*i.e.*, ABVA) often forms a hybrid of loss and overflow models where a user can not be served by one resource is overflowed to another resource and it might be lost finally. However, traditional overflow model has the same pre-designated ordering of primary resource, secondary resource and so on for all users [43]. In ABVA environment, this ordering is different from user to user due to distributed manner of users and servers. Furthermore, the delay caused by overflows is not considered in traditional overflow models. Motivated by the hunting schemes used by circuits to carry traffic in conventional telephone networks [43], we propose the hunting schemes used by users to seek resource. Our work presented here focuses on balancing user request denial probability and latency in a capacitated ABVA environment considering a number of arrival models for three different topologies.

1.8 Organization

The rest of this dissertation is organized as follows. Chapter 2 formulates the data center resource allocation problem and present time slot aggregation methods to reduce the computational complexity when solving the problem. DVFS is considered along with switching servers on and off to formulate the problem in chapter 4. Chapter 5 introduces the hierarchical model to evaluate Quality of Experience for Internet Services hosted in data centers. Chapter 6 proposes the redirection schemes (strategies) in the agent-based VPN. Chapter 7 concludes this dissertation.

CHAPTER 2

ENERGY-AWARE DATA CENTER RESOURCE MANAGEMENT WHEN RESOURCE REQUIREMENTS ARE GIVEN

The proliferation of Internet services drives the data center expand in both size and the number. The data centers desire minimizing their total cost of ownership (TCO). More importantly, the power consumption (as part of TCO) becomes a social concern now. We present formulations to minimize server energy consumption and server cost under three different data center server models (homogeneous, heterogeneous, hybrid hetero-homogeneous clusters) with dynamic temporal demand.

The contribution of our work is: (a) We formulate three models to determine optimal number of running servers that for three different data center types in terms of their heterogeneity. (b) Workload aggregation methods are proposed to reduce the computational time of computing optimum. Two workload aggregation modes are introduced for different needs of applications. To combat the pitfalls of static aggregations for both modes, we further introduce dynamic aggregations.

The remainder of this chapter is organized as follows. The notions and notations are introduced in Section 2.1. We present optimization formulations in Section 2.2. Two aggregation modes: (a) aggregation by maximum and (b) aggregation by mean and two aggregation methods: (a) static aggregation and (b) dynamic aggregation are presented in Section 2.3. Our numeric studies are discussed in Section 2.4. Finally, Section 2.5 concludes this chapter.

2.1 Notions and Notations

We introduce the notions and notations used in this chapter in this section. The rest of notions and notations will be introduced when they are used. There are two fonts used to denote sets. We use blackboard bold font to denote conventional sets, for example, $\mathbb{R}$ denotes the set of real numbers; $\mathbb{Z}$ denotes the set of integers. We use calligraphy to denote sets of indexes, for instance, $\mathcal{I}$ denotes the set of server numbering index. We use $0, 1$ to denote binary integer sets. We use $\mathbb{R}_+$ and $\mathbb{R}_{++}$ to denote the set of non-negative real numbers and positive real numbers, respectively. Vectors are denoted by lower-case boldface letters. An array of elements which are delimited by commas and embraced by a pair of parentheses is a column vector. And an array of elements delimited by spaces and embraced by a pair of square brackets is a row vector. For example, $x_1, x_2, x_3 \in \mathbb{R}$ are three elements, we have

$$(x_1, x_2, x_3) = [x_1 \; x_2 \; x_3]^{\mathbf{T}}.$$

The set of real n-vector is denoted by $\mathbb{R}^n$. The matrix is denoted by a boldface capital letter and its elements are denoted by italicized lower-case letters having subscripts to denote their positions in the matrix. A matrix $\mathbf{A}$ having m rows and n columns is referred to as an $m \times n$ matrix. For example, $\mathbf{A} = [a_{ij}]$ is a notation of matrix by specifying the general notation of its elements. If all elements in matrix $\mathbf{A}$ are real numbers, we say $\mathbf{A} \in \mathbb{R}^{m \times n}$.

2.2 Problem Formulation

There is a data center or a cluster in a data center with I servers. Let $\mathcal{I}$ denote the set of servers, while the cardinality of this set is denoted by I (i.e., $\#(\mathcal{I}) = I$). This optimization problem is considered in a duration of Υ hours. In practice, we expect Υ to be six to eight hours during which the workload can be accurately predicted according to history observations. There are many techniques can be used to achieve accurate prediction, *e.g.*, [38,86]. However, the prediction techniques are out of the scope this paper. The duration Υ hours are divided into T equal *time slots* or *periods*[1] and the duration of a time slot, called slot size, is $\tau = \Upsilon/T$ hours. We assume that workload on CPU is predicted at the beginning of the planning duration. The servers are reconfigurable at the beginning of time slots. These points are called *review points*. The capacity that server i at time slot t can offer is denoted by v_{it}. We assume the capacity offered by server i is the same for all time slots. Thus, we have:

$$v_{it} = v_i, \ t = 1, \cdots, T. \tag{2.1}$$

We want to determine how to configure the servers at review points such that total cost of energy consumption and server CAPEX is minimized over the entire planning period.

2.2.1 Heterogenous Server Model

In this model, we assume all servers to be heterogenous and this model is denoted by *Model-Het*. The power consumption per time unit of running server $i \in \mathcal{I}$ is denoted

[1]The terms time slot and period are used interchangeably in this paper.

by c^p_{it}. We assume the energy cost are the same at all time slots. That is,

$$c^p_{it} = c^p_i,\ t = 1, \cdots, T. \tag{2.2}$$

Let binary decision variables x_{it} denote if server i is turned on at time t. The switching costs of turning server i on and off are denoted by c^{s+}_{it} and c^{s-}_{it}, respectively. We assume the switching cost are the same at all time slots. Thus,

$$c^{s+}_{it} = c^{s+}_i,\ t = 1, \cdots, T; \tag{2.3}$$

$$c^{s-}_{it} = c^{s-}_i,\ t = 1, \cdots, T. \tag{2.4}$$

$x_{it} = 1, 0$ represent the states of server i is "on" and "off" at time slot t, respectively. Let "1" represents state change while "0" stands for no change between adjacent time slots. Then the state change from "on" to "off" can be represented by $x_{it} \cdot (x_{it} - x_{i(t-1)})$. Similarly, the state change from off to on can be represented by $x_{i(t-1)} \cdot (x_{i(t-1)} - x_{it})$. Our objective is to minimize the energy cost as well as the cost of switching servers on or off over the entire time horizon. Therefore, the objective function is given by:

$$F = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} c^p_i \cdot x_{it} + \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \left(c^{s+}_i \cdot x_{it} \cdot (x_{it} - x_{i(t-1)}) + c^{s-}_i \cdot x_{i(t-1)} \cdot (x_{i(t-1)} - x_{it})\right) \tag{2.5}$$

Next we consider the constraints. There is only one set of constraints in this problem: the demand must be satisfied in every time slot t. Thus we have:

$$\sum_{i \in \mathcal{I}} v_i \cdot x_{it} \geq d_t,\ t = 1, 2, \cdots, T \tag{2.6}$$

We notice that the objective function (2.5) is a binary quadratic function. We can transform this special form of quadratic function into a linear function by introducing additional variables and constraints without resorting to any approximation. We introduce two binary variables x_{it}^{+} and x_{it}^{-} to represent switching on and off at review point of time slot t. $x_{it}^{+} = 1$ represents that server i is turned on at time t. Similarly, $x_{it}^{-} = 1$ means server i is turned off at time slot t. $x_{it}^{+} = x_{it}^{-} = 0$ indicates that the state of server i does not change from time slot $t-1$ to t. Thus, we have:

$$x_{it} - x_{i(t-1)} - x_{it}^{+} + x_{it}^{-} = 0,\ i = 1, 2, \cdots, I;\ t = 1, 2, \cdots, T. \tag{2.7}$$

Because x_{it}^{+} and x_{it}^{-} *cannot* be both 1. we use following inequalities to enforce this requirement:

$$x_{it}^{+} + x_{it}^{-} \leq 1,\ i = 1, 2, \cdots, I;\ t = 1, 2, \cdots, T. \tag{2.8}$$

With the aid of x_{it}^{+} and x_{it}^{-}, the original quadratic objective function (2.5) is transformed to the following linear function:

$$F = \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} c_i^p \cdot x_{it} + \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} (c_i^{s+} \cdot x_{it}^{+} + c_i^{s-} \cdot x_{it}^{-}). \tag{2.9}$$

Since we assume all servers are set to off status at the review point of time slot 0, we have the initial set of conditions as follows:

$$x_{i0} = 0,\ i = 1, \cdots, I. \tag{2.10}$$

In Model-Het, the objective is to minimize (2.9) which is subject to (2.6), (2.7) and (2.8). And the initial conditions are given by (2.10). It is a binary LP problem. This problem

constants

c_i^p energy consumption of server i in a time slot

c_i^{s+} cost of switching server i on

c_i^{s-} cost of switching server i off

v_i capacity of server i

d_t demand at t

$x_{i0} = 0$ initial (time slot 0) state of server i

variables

x_{it} $= 1$ if server i is running at time slot t; 0, otherwise

x_{it}^+ $= 1$ if server i is turned on at the review point of time slot t; 0, otherwise

x_{it}^- $= 1$ if server i is turned off at the review point of time slot t; 0, otherwise

objective

minimize $F = \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}} c_i^p \cdot x_{it} + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}(c_i^{s+}\cdot x_{it}^+ + c_i^{s-}\cdot x_{it}^-)$

constraints

$\sum_{i\in\mathcal{I}} v_i \cdot x_{it} \geq d_t, \quad t = 1, 2, \cdots, T$

$x_{it} - x_{i(t-1)} - x_{it}^+ + x_{it}^- = 0, \quad i = 1, 2, \cdots, I;\ t = 1, 2, \cdots, T$

$x_{it}^+ + x_{it}^- \leq 1, \quad i = 1, 2, \cdots, I;\ t = 1, 2, \cdots, T$

Figure 3: Model-Het: Binary IP: heterogenous server scenario

is a special case (the case without dynamic voltage frequency scaling) of our previous work [72]. For sake of clarity, we summarize this model in Fig. 2.2.1.

It is easier to understand the physical meaning of the problem in normal representation. However, matrix representation is more convenient to observe and explore the structure of the problem. Hence, we present the matrix representations of the Model-Het. The matrix representation for other two models can be derived in a similar way and are special cases of the Model-Het. Therefore, we only derive and present the matrix representation for Model-Het in this dissertation.

Let I-vector $\mathbf{x}_t = (x_{1t}, x_{2t}, \cdots, x_{It})$ represent the status of servers at time slot t. Then by column-wise concatenating T such I-vectors we obtain $I \times T$-matrix $\mathbf{x} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_T]$, which denotes the status of all servers at all time slots. The I-vector

$\mathbf{c}_t^p = [c_{1t}^p \; c_{2t}^p \; \cdots \; c_{It}^p]$ is the coefficient vector of power consumption for 1 to I servers at time slot t. And $I \times T$-matrix $\mathbf{c}^p = (\mathbf{c}_1^p, \mathbf{c}_2^p, \cdots, \mathbf{c}_T^p)$ is the coefficient vector of power consumption for 1 to I servers from time slot 1 to T. As we can see, the server status have two dimensions: one is server set $\mathcal{I}$, the other one is time slot set $\mathcal{T}$. If these coefficients do not change over time, the subscript t can be dropped. We define I-column vector $\mathbf{x}_t^+ = (x_{1t}^+, x_{2t}^+, \cdots, x_{It}^+)$ to represent the state of turning on actions at (the review point of) time slot t. Let $I \times T$-matrix $\mathbf{x}^+ = [\mathbf{x}_1^+ \; \mathbf{x}_2^+ \; \cdots \; \mathbf{x}_T^+]$ denote the states of turning on actions for all servers at all time slots. Similarly, I-column vector $\mathbf{x}_t^- = (x_{1t}^-, x_{2t}^-, \cdots, x_{It}^-)$ is the status of the state of turning off actions at (the review point of) time slot t. And $I \times T$-matrix $\mathbf{x}^- = [\mathbf{x}_1^- \; \mathbf{x}_2^- \; \cdots \; \mathbf{x}_T^-]$ is the states of turning off actions for all servers at all time slots. The variable matrix for the whole problem can be represented by column-wise concatenation of $\mathbf{x}$, $\mathbf{x}^+$ and $\mathbf{x}^-$. That is, $\mathbf{x} = [\mathbf{x} \; \mathbf{x}^+ \; \mathbf{x}^-]$. We denote $\mathbf{a}' = [1 \; \mathbf{0}^{IT-1} \; -1 \; \mathbf{0}^{IT-1} \; 1]$, $\mathbf{a}'' = [-1 \; \mathbf{0}^{I-1} \; \mathbf{a}']$. We define matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}' & \mathbf{0}^{IT-1} & & & \\ \mathbf{0}^1 & \mathbf{a}' & \mathbf{0}^{IT-2} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{0}^{I-1} & \mathbf{a}' & \mathbf{0}^{I(T-1)} \\ \hdashline \mathbf{a}'' & \mathbf{0}^{IT-I-1} & & & \\ \mathbf{0}^1 & \mathbf{a}'' & \mathbf{0}^{IT-I-2} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{0}^{IT-I-1} & \mathbf{a}'' & \end{bmatrix} \tag{2.11}$$

We have integrated the state initialization (2.10) in $\mathbf{A}$ by partitioning it into two blocks.

The upper block is the coefficients for the states at first time slot. The lower block represents the coefficients of the states at the remaining $(T-1)$ time slots. Therefore, (2.10) has already been implied. Hence we have following matrix presentation for (2.7):

$$\mathbf{Ax} = 0, \quad \text{where } \mathbf{A} \in \{-1, 0, 1\}^{IT \times 3IT} \quad \mathbf{x} \in \{0, 1\}^{3IT}. \tag{2.12}$$

To represent the (2.8) we define matrix $\mathbf{G}$ as follows

$$\mathbf{G} = \begin{bmatrix} \mathbf{0}_{IT} & 1 & \mathbf{0}_{IT-1} & 1 & \mathbf{0}_{IT-1} \\ & \ddots & \ddots & \ddots & \\ & \mathbf{0}_{2IT-1} & 1 & \mathbf{0}_{IT-1} & 1 \end{bmatrix} \tag{2.13}$$

And (2.8) can be written as:

$$\mathbf{Gx} \preceq \mathbf{1}, \quad \text{where } \mathbf{G} \in \{-1, 0, 1\}^{IT \times 3IT} \quad \mathbf{x} \in \{0, 1\}^{3IT}. \tag{2.14}$$

Let $\mathbf{c} = (\mathbf{c}^p, \mathbf{c}^{s+}, \mathbf{c}^{s-})$. The matrix representation of (2.5) is

$$F = \mathbf{cx}. \tag{2.15}$$

For the set of demand constraints, we define I-row vector $\mathbf{v}_t = (v_{1t}, v_{2t}, \cdots, v_{It})$. Then we have $T \times IT$ capacity coefficient matrix $\mathbf{V}$

$$\mathbf{V}' = \begin{pmatrix} \mathbf{v}_1 & & & \mathbf{0} \\ & \mathbf{v}_2 & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{v}_T \end{pmatrix} \tag{2.16}$$

The matrix representation of (2.6) is given by:

$$\mathbf{V}'\mathbf{x}' \geq \mathbf{d}, \quad \text{where} \mathbf{V} \in {\mathbb{R}_+}^{T \times IT}, \mathbf{x}' \in \{0, 1\}^{IT}, \mathbf{d} \in {\mathbb{R}_+}^{T}. \tag{2.17}$$

constants

$\mathbf{c} \in \mathbb{R}_{+}^{3IT}, \mathbf{A} \in \{-1,0,1\}^{IT\times 3IT}, \mathbf{G} \in \{-1,0,1\}^{IT\times 3IT}, \mathbf{V} \in \mathbb{R}_{+}^{T\times 3IT}$

variables

$\mathbf{x} \in \{0,1\}^{3IT}$

objective

minimize $F = \mathbf{cx}$

constraints

$\mathbf{Vx} \succeq \mathbf{d}$

$\mathbf{Ax} = \mathbf{0}^{IT}$

$\mathbf{Gx} \preceq \mathbf{1}^{IT}$

Figure 4: The matrix representation of Model-Het

By appending a $T \times 2IT$ 0 matrix to matrix $\mathbf{V}'$, we obtain a new matrix $\mathbf{V}$

$$\mathbf{V} = [\mathbf{V}'\ \mathbf{0}^{T\times 2IT}] \tag{2.18}$$

Then we have

$$\mathbf{Vx} \leq \mathbf{d}, \quad \text{where}\, \mathbf{V} \in \mathbb{R}_{+}{}^{T\times 3IT}, \mathbf{x} \in \{0,1\}^{3IT}, \mathbf{d} \in \mathbb{R}_{+}^{T}. \tag{2.19}$$

The matrix representation of the Model-Het is summarized in Fig. 2.2.1.

2.2.2 Homogeneous Server Model

In the homogeneous server model, all servers are identical. This model is denoted by *Model-Hom*. Although the heterogeneous model introduced in the previous section is general enough to be applied to this model, we present a newly formulated model where the number of variables and constraints are reduced significantly. Let c^p be the power consumption of running a server in a time slot. Let y_t denote the number of *running* homogeneous servers at time slot t. Therefore, we can reduce the I binary variables in Model-Het to a single integer variable for each time slot in Model-Hom. The power consumption for each server per time slot is denoted by c^p. We have:

$$c_i^p = c^p,\ i = 1, \cdots, I. \tag{2.20}$$

The cost of switching one server on and off is denoted by c^{s+} and c^{s-}, respectively. We have:

$$c_i^{s+} = c^{s+};\ c_i^{s-} = c^{s-},\ i = 1, \cdots, I. \tag{2.21}$$

The capacity of each server can provide is denoted by v. We have:

$$v_i = v,\ i = 1, \cdots, I. \tag{2.22}$$

Performing workload dispatching is the overhead in this model. Because to achieve this model, we need to use the resource in a “stack” (LIFO) manner. That is, the latest released resource should be used first when receiving new requests. Implementing workload dispatching algorithms is out of the scope of this paper. It has been studied in [28,55]. However we need to keep in mind that this overhead happens for the homogeneous server model as well as heterogeneous homogeneous server cluster model which will be introduced in Sec 2.2.3.

We now formulate this problem. Similar to Model-Het, the first constraint is the workload requirements:

$$v \cdot y_t \geq d_t,\ t = 1, \cdots, T. \tag{2.23}$$

Let y_t^+ be the number of servers that are switched on at the review point of time slot t.

Then y_t^+ should take the maximum between 0 and $y_t - y_{t-1}$. That is:

$$y_t^+ = \max\{0, y_t - y_{t-1}\},\ t = 1, \cdots, T. \tag{2.24}$$

Let y_t^- be the number of servers that are switched off at the review point of time slot t. Similar to y_t^+, we have:

$$y_t^- = \max\{0, y_{t-1} - y_t\},\ t = 1, \cdots, T. \tag{2.25}$$

The objective is to minimize the total energy and switching on/off cost, which is given by:

$$F = \sum_{t \in \mathcal{T}} (c^p \cdot y_t + c_t^{s+} \cdot y_t^+ + c_t^{s-} \cdot y_t^-). \tag{2.26}$$

Because this is a minimization problem with the cost coefficient being non-negative, we can substitute (2.24) by following two linear inequalities:

$$y_t^+ \geq y_t - y_{t-1},\ t = 1, \cdots, T. \tag{2.27}$$

$$y_t^+ \geq 0,\ t = 1, \cdots, T. \tag{2.28}$$

Likewise, we can substitute (2.25) by following two linear inequalities:

$$y_t^- \geq y_{t-1} - y_t,\ t = 1, \cdots, T. \tag{2.29}$$

$$y_t^- \geq 0,\ t = 1, \cdots, T. \tag{2.30}$$

The last constraints is the number of running servers should not be larger than total number of servers:

$$y_t \leq I. \tag{2.31}$$

constants

c^p energy consumption per server per time slot

c^{s+} cost of switching a server on

c^{s-} cost of switching a server off

v capacity of a server

d_t demand at t

$y_0 = 0$ initial (time slot 0) state of servers

variables

$y_t \in \mathcal{N}$ # of running servers at time slot t

$y_t^+ \in \mathcal{N}$ # of servers switched on at time slot t

$y_t^- \in \mathcal{N}$ # of servers switched off at time slot t

objective

minimize $F = \sum_t (c^p \cdot y_t + c^{s+} \cdot y_t^+ + c^{s-} \cdot y_t^-)$

constraints

$v \cdot y_t \geq d_t,\ t = 1, \cdots, T$

$y_t \leq I,\ t = 1, \cdots, T$

$y_t^+ \geq 0,\ t = 1, \cdots, T$

$y_t^+ \geq y_t - y_{t-1},\ t = 1, \cdots, T$

$y_t^- \geq 0,\ t = 1, \cdots, T$

$y_t^- \geq y_{t-1} - y_t,\ t = 1, \cdots, T$

Figure 5: Model-Hom: IP: homogeneous server scenario

Because it is a minimization problem with the cost coefficient being non-negative, this constraint can actually be omitted. Since all servers are off at time slot 0, we have the initial condition:

$$y_0 = 0. \tag{2.32}$$

In the Model-Hom, we want to minimize (2.26) which is subject to (2.23), (2.27), (2.28), (2.29) and (2.30). And the initialization condition is given by (2.32).

For sake of clarity, we summarize this model in Fig. 2.2.2

2.2.3 Heterogeneous Homogeneous-Server-Cluster Model

In a more realistic data center, there are different clusters of servers and each cluster has a certain number of homogeneous servers while servers may be different from one cluster to another. Thus, this is a hybrid of the afore-formulated two models, the hybrid hetero-homogenous cluster model. This model can also be used when homogenous servers are required to partition into multiple clusters for ease of management. We denote this model by *Model-HH*. Denote the set of clusters by $\mathcal{J}$ and its cardinality, $J = \#(\mathcal{J})$, where $1 \leq J \leq I$, represents the number of clusters. Actually, two previous models correspond to two extremes in this model: when $J = 1$, it is Model-Hom; when $J = I$, it becomes Model-Het. Let the energy consumption of running a server in cluster j at a time slot be z_j^p. Denote the number of servers in cluster j by I_j, where $\sum_j I_j = I$. We denote the set of the number of running servers for cluster j by $\mathcal{N}_j$, which can be $0, 1, \cdots, I_j$. Let z_{jt} be the number of running servers in cluster j at time slot t. As we have mentioned in Sec. 2.2.2, the workload need to be dispatched in the LIFO manner. The cost of running a server in cluster j is denoted by c_j^p. The costs of switching a server in cluster j on and off are represented by c_j^{s+} and c_j^{s-}, respectively. The capacity of a server in cluster j is given by v_j. Another way to look at this problem is to consider it as the superposition of multiple homogeneous cases.

We now introduce constraints in this problem. First, the workload requirements need to be satisfied all the time:

$$\sum_{j \in \mathcal{J}} v_j \cdot z_{jt} \geq d_t, \; t = 1, \cdots, T. \tag{2.33}$$

Secondly, the number of running servers cannot be larger than the total number of servers in that cluster:

$$z_{jt} \leq I_j,\ j = 1, \cdots, J;\ t = 1, \cdots, T. \tag{2.34}$$

The third set of constraints is similar to (2.24) and (2.25) in Model-Hom. But they need to be applied to each of J clusters. Let z^+_{jt} be the number of servers turned on in cluster j at the review point of time slot t while z^-_{jt} be the number of servers turned off in cluster j at the review point of time slot t. We use the same technique in Model-Hom to transform the constraints to linear constraints:

$$z^+_{jt} \geq 0,\ j = 1, \cdots, J;\ t = 1, \cdots, T. \tag{2.35}$$

$$z^+_{jt} \geq z_{jt} - z_{j(t-1)},\ j = 1, \cdots, J;\ t = 1, \cdots, T. \tag{2.36}$$

$$z^-_{jt} \geq 0,\ j = 1, \cdots, J;\ t = 1, \cdots, T. \tag{2.37}$$

$$z^-_{jt} \geq z_{j(t-1)} - z_{jt},\ j = 1, \cdots, J;\ t = 1, \cdots, T. \tag{2.38}$$

And the objective function is given by:

$$F = \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}} (c^p_j \cdot z_{jt} + c^{s+}_j \cdot z^+_{jt} + c^{s-}_j \cdot z^-_{jt}) \tag{2.39}$$

Because we assume at the beginning of planning period, the servers are all off, we have:

$$z_{j0} = 0,\ j = 1, \cdots, J. \tag{2.40}$$

Therefore, the objective is to minimize (2.39) which is subject to (2.33), (2.34), (2.35), (2.36), (2.37) and (2.38). And the initial conditions are given by (2.40). We present our formulation in Fig. 2.2.3.

constants

c_j^p energy consumption of a class j server in a time slot
c_j^{s+} cost of switching a class j server on
c_j^{s-} cost of switching a class j server off
v_j capacity of a class j server
d_t demand at time slot t
$z_{j0} = 0$ initial (time slot 0) state of class j server

variables

$z_{jt} \in \mathcal{I}_j$ # of class j running servers at t
$z_{jt}^+ \in \mathcal{I}_j$ # of class j servers turned on at t
$z_{jt}^- \in \mathcal{I}_j$ # of class j servers turned off at t

objective

minimize $F = \sum_t \sum_j (c_j^p \cdot z_{jt} + c_j^{s+} \cdot z_{jt}^+ + c_j^{s-} \cdot z_{jt}^-)$

constraints

$\sum_j v_j \cdot z_{jt} \geq d_t,\ t = 1, \cdots, T$
$z_{jt} \leq I_j,\ j = 1, \cdots, J;\ t = 1, \cdots, T$
$z_{jt}^+ \geq 0,\ j = 1, \cdots, J;\ t = 1, \cdots, T$
$z_{jt}^+ \geq z_{jt} - z_{j(t-1)},\ j = 1, \cdots, J;\ t = 1, \cdots, T$
$z_{jt}^- \geq 0,\ j = 1, \cdots, J;\ t = 1, \cdots, T$
$z_{jt}^- \geq z_{j(t-1)} - z_{jt},\ j = 1, \cdots, J;\ t = 1, \cdots, T$

Figure 6: Model-HH: IP: heterogeneous homogeneous-server-cluster scenario

2.3 Aggregating Demand to Reduce Computational Time

The number of variables for the heterogenous case, the homogenous case, the heterogeneous homogeneous-server-cluster case, presented above are $3 \times I \times T$, $3 \times T$, and $3 \times J \times T$, respectively. Since the computational time grows with the number of integer variables, the time complexity for the heterogenous and heterogeneous homogeneous-server-cluster cases differs significantly for larger scale data centers.

To reduce the computational time, we need to either reduce the number of variables in the original problem or use a heuristic to approximate the original problem. In this paper, we propose aggregating a certain number of continuous time slots into a single aggregated slot to reduce the total number of time slots for workload. The value of workload demand on an aggregated time slot is decided by the workload of the original time slots and the service-level agreement (SLA). Let $\hat{T}$ denote the reduced number of time slots.

2.3.1 Aggregation by Maximum

If an SLA stringently requires that the workload should be satisfied all the time, then the workload of an aggregated time slot takes the maximum of the demand of original time slots. For example, we want to aggregate $[d_k, d_{k+1}, \cdots, d_{k+\ell}]$ into one time slot; then the new demand $\hat{d}_k$ with ℓ times of the original slot size is given by:

$$\hat{d}_k = \max\{d_k, d_{k+1}, \cdots, d_{k+\ell}\}. \tag{2.41}$$

This method introduces an artificial increase in the demand, which is not needed by users. Increasing demand causes extra consumption of power energy. On the other hand, aggregation smoothes out the "regularity" of workload which has effects on switching cost. That is, we trade off the time complexity of running the model at the expense of extra cost on energy consumption.

2.3.1.1 Static Aggregation

In this method, we aggregate every M contiguous time slots into one. We have $\hat{T} = \lceil T/M \rceil$. The slot size of the aggregated workload (except the last slot) is M times of the slot size of the original demand. The slot size of the last slot is $T - M \times (\hat{T} - 1)$.

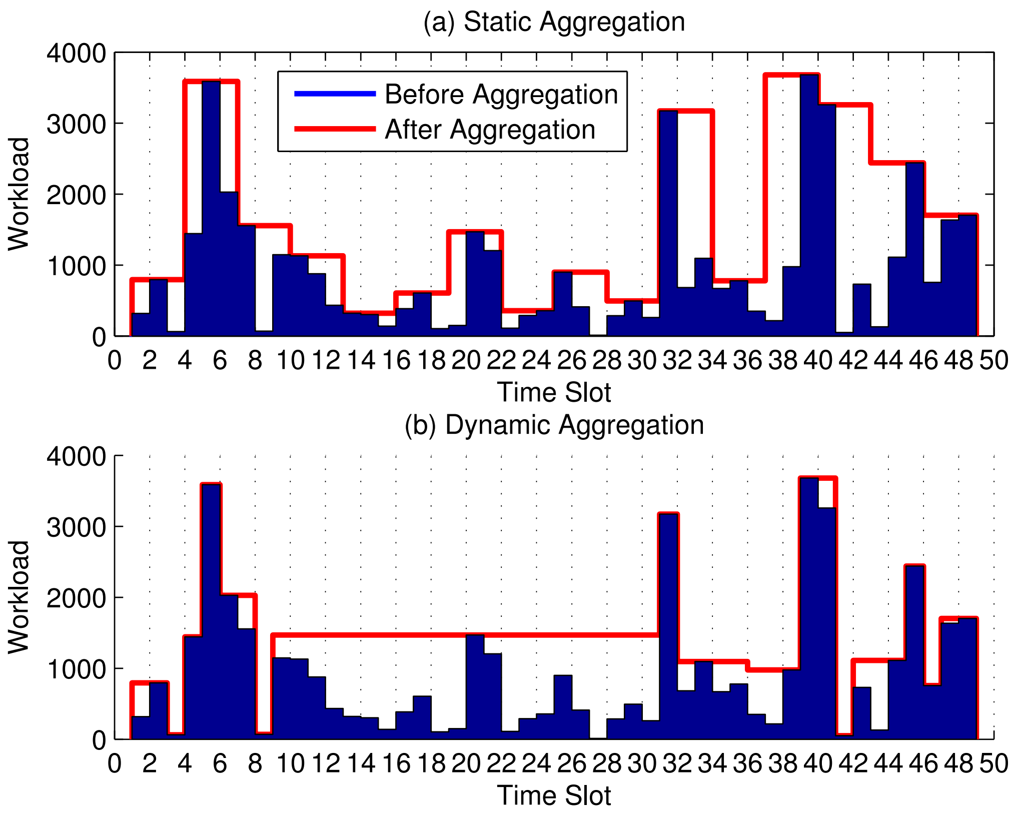


Figure 7: An illustration of aggregating workload by maximum

The aggregated workload $\lceil t/M \rceil$ is given by (except the workload for the last slot which is trivial):

$$\hat{d}_{\lceil t/M \rceil} = \max\{d_{(\lceil t/M \rceil - 1)\cdot M+1}, \cdots, d_{(\lceil t/M \rceil -1)\cdot M+M}\} \tag{2.42}$$

The static aggregation is illustrated in Fig. 7(a).

2.3.1.2 Dynamic Aggregation

Although some waste of energy consumption is inevitable, we can alleviate waste by using dynamic aggregation. Instead of aggregating workload in equal numbers of original time slots statically, the adaptive aggregation method aggregates an arbitrary number of time slots as long as the number of aggregated time slots is $\hat{T}$ such that the sum of the difference between aggregated workload and the original workload is minimized. That is, we need to minimize

$$\sum_{k=1}^{\hat{T}} \sum_{\ell=1}^{T_k} (d_k^{max} - d_{kl}), \text{ where } d_k^{\text{max}} = \max\{d_{k1}, \cdots, d_{kT_k}\}, \tag{2.43}$$

which is subject to

$$\sum_{k=1}^{\hat{T}} T_k = T. \tag{2.44}$$

Fig. 7(b) illustrates an example of dynamic aggregation.

To this end, we need to choose $\hat{T} - 1$ review points out of $T - 1$, requiring $\binom{T-1}{\hat{T}-1}$ operations. This extra computational time complexity is contradictory to the purpose of doing aggregation. Thus, we propose *local smooth* heuristics to implement this idea. We aggregate the time slots that are locally "smoother" together. We first define the smooth index of workloads. The smooth index is the absolute value of difference between the workload demand of adjacent time slots (adjacent workloads for short). Therefore, we obtain $T - 1$ smooth indices. Then we pick the smallest non-zero smooth index and compare two adjacent workload associated with this smooth index. The smaller workload are aggregated into maximum workload over these slots; the slot size of the aggregated workload is the sum of two slot sizes of adjacent slots and the new workload for the aggregated slot takes the maximum of these demands. The smooth index is updated for the new aggregated demand series. We repeat this procedure until the target number of slots is reached. We call this procedure the *local smooth algorithm* [2] (see Algorithm 1). We can make further improvements. For this, denote the smooth index vector by si. Scalar ap records the aggregation point. Vector ss, which is initialized to T ones, records the size of each time slot. Let "InMin" be the procedure to find the index of minimum value in a vector. "Max" is the procedure to find the maximum value while "Mean" is the procedure taking the average which is used in aggregation by mean. The value of the workload in time slot i is denoted by $d[i]$ here. The *improved local smooth algorithm* is presented in Algorithm 2.

Algorithm 1 Local Smooth Algorithm

```
 1: $j \leftarrow T$                                          Initialization
 2: while $j > \hat{T}$ do
 3:     for $i := 2 \rightarrow j$ do
 4:         $si[i-1] \leftarrow |d[i] - d[i-1]|$                Compute smooth index
 5:     end for
 6:     $ap \leftarrow$ InMin(si)                            Find aggregation point
 7:     $d[ap] \leftarrow$ Max/Mean$(d[ap], d[ap+1])$
 8:     $d[ap+1] \leftarrow$ Max/Mean$(d[ap], d[ap+1])$                  Aggregate
 9:     $ss[ap] \leftarrow ss[ap] + ss[ap+1]$      Compute the size of aggregated slots
10:     if $ap \neq j-1$ then
11:         for $k := ap+1 \rightarrow j-1$ do
12:             $d[k] = d[k+1]$    Adjust the index of slots behind the aggregation point
13:             $ss[k] = ss[k+1]$              Adjust the corresponding slot size
14:         end for
15:     end if
16:     $j \leftarrow j - 1$                       Decrease the number of slots by 1
17: end while
18: return d, ss
```

[2]This algorithm was conceived through discussions with Fu Li.

Algorithm 2 Improved Local Smooth Algorithm

```
1: $j \leftarrow T$                                              Initialization
2: for $i := 2 \rightarrow j$ do
3:     $si[i-1] \leftarrow (d[i] - d[i-1])$                     Compute smooth index
4: end for
5: while $j > \hat{T}$ do
6:     $ap \leftarrow$ InMin(|si|)                               Find aggregation point
7:     $d[ap] \leftarrow$ Max/Mean$(d[ap], d[ap+1])$
8:     $d[ap+1] \leftarrow$ Max/Mean$(d[ap], d[ap+1])$           Aggregate
9:     $si[ap-1] \leftarrow d[ap] - d[ap-1]$                     Update smooth index
10:    $si[ap] \leftarrow d[ap+2] - d[ap+1]$                     Update smooth index
11:    $ss[ap] \leftarrow ss[ap] + ss[ap+1]$                     Compute the size of aggregated slots
12:    if $ap \neq j-1$ then
13:        for $k := ap+1 \rightarrow j-1$ do
14:            $d[k] \leftarrow d[k+1]$                          Adjust the index of slots behind the aggregation point
15:            $ss[k] \leftarrow ss[k+1]$                        Adjust the corresponding slot size
16:            if $k < j-1$ then
17:                $si[k] \leftarrow si[k+1]$                    Adjust the smooth index
18:            end if
19:        end for
20:    end if
21:    $j \leftarrow j-1$                                        Decrease the number of slots by 1
22: end while
23: return d, ss
```

These algorithms are efficient to aggregate the workloads. In Algorithm 1, we can achieve $\mathcal{O}(T^2)$. In Algorithm 2, we further improve the efficiency by recomputing two smooth indexes adjacent to the selected aggregation point only and an $\mathcal{O}(T)$ complexity is achieved. However, since it is based on the local information, there is no guarantee that this method will reach the global optimal solution. In some rare cases, it is possible that this solution is worse than static allocation in terms of the optimum. Note that the computational time for the dynamic aggregation has no significant difference with its static aggregation counterpart since they have the same number of variables and constraints. To differentiate the proposed dynamic aggregation and implemented dynamic aggregation, we call proposed dynamic aggregation as *strict* dynamic aggregation.

2.3.2 Aggregation by Mean

The workload demand of the aggregated time slot can also take a certain percentile of workload demand of the original slots. "Aggregation by maximum" actually takes 100-th percentile of the original slots. For many long-live jobs which does not need to executed on real-time, such as data warehousing, scientific computation, the workload can be arranged over time as long as the average workload over time is satisfied. In this paper, we introduce aggregation by mean: the workload demand of the aggregated time slot takes the mean of the workload demand of the original slots. For example, We aggregate $[d_k, d_{k+1}, \cdots, d_{k+\ell}]$ into one time slot, then the new demand $\hat{d}_k$ with ℓ times of the original slot size is given by:

$$\hat{d}_k = \text{mean}\{d_k, d_{k+1}, \cdots, d_{k+\ell}\}. \tag{2.45}$$

Compared with aggregation by maximum, aggregation by mean does not introduce the artificial increase of the workload demand and smoothes out the regularity of the workload.

2.3.2.1 Static Aggregation

We aggregate every M continuous time slots into 1. The only difference is that the aggregated workload takes the average of original workload:

$$\bar{d}_{\lceil t/M \rceil} = \text{mean}\{d_{(\lceil t/M \rceil - 1)\cdot M+1}, \cdots, d_{(\lceil t/M \rceil - 1)\cdot M+M}\} \tag{2.46}$$

The application of method is based on the ability of re-arranging user request within certain time limitation for some applications that do not requires real-time execution. In the other words, the requested load can be either executed in advance or delayed in the data center. An example of static aggregation by average is shown in Fig. 8(a).

2.3.2.2 Dynamic Aggregation

To improve the user experience, we need to avoid delay or advance workload as much as possible. Therefore, we have the counterpart of dynamic aggregation of what we have in aggregation by max. The objective function is to minimize:

$$\sum_{k=1}^{\hat{T}} \sum_{\ell=1}^{T_k} |d_{kl} - \bar{d}_k|, \text{where} \bar{d}_k = \text{mean}\{d_{k1}, \cdots, d_{kT_k}\} \tag{2.47}$$

which is subject to (2.44).

Fig. 8(b) illustrates the dynamic aggregation by average. It is worthy to note that although (2.43) and (2.47) look similar, the objectives of dynamic aggregation in

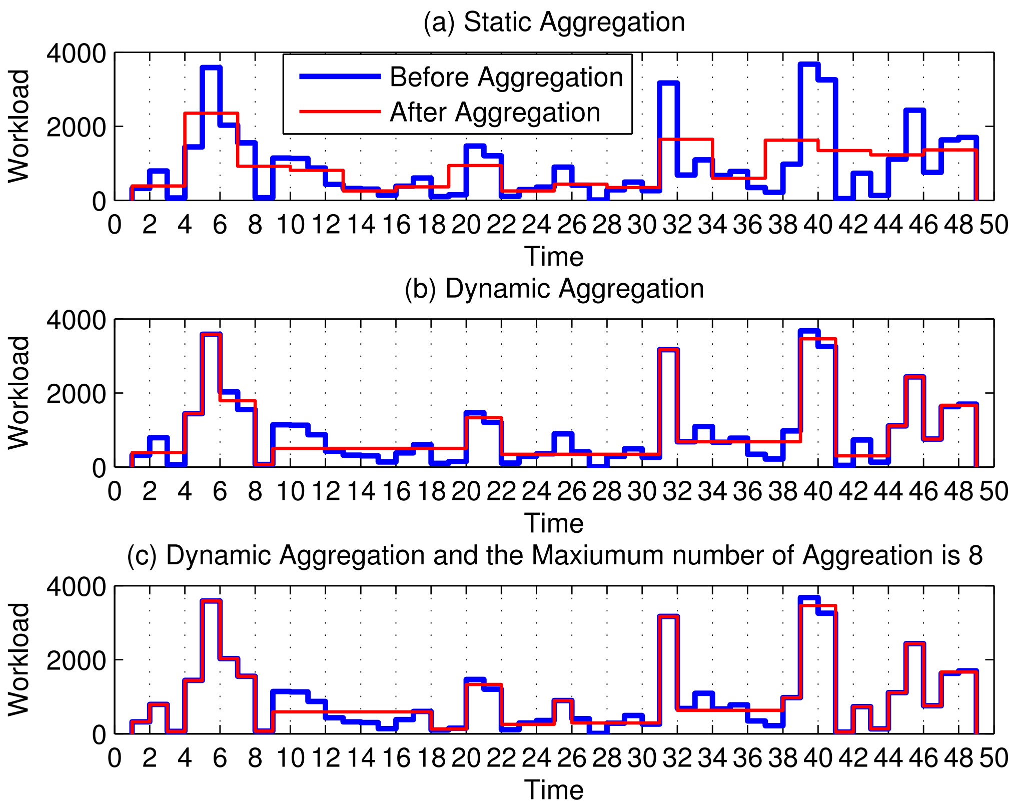


Figure 8: An illustration of aggregating workload by mean

aggregation by maximum and aggregation by mean are different: 1) in aggregation by maximum, it aims to reduce "wasted energy"; 2) in aggregation by mean, it targets to reduce the movement of workload. Thus compared with static aggregation, in aggregation by maximum, it results in less energy cost while in aggregation by average, the energy cost is always the same.

However, the proposed dynamic aggregation method has no constraints on the number of original slots for aggregated slot. In practical applications, the workload can only be executed in advance or delayed for certain time. Let S be the maximum number of continuous time slots that can be aggregated. Consequently, this problem is also subject to:

$$\max\{s_1, \cdots, s_{\hat{T}}\} \leq S. \tag{2.48}$$

The exact solution based on improve local smooth algorithms requires $n!$ time. We suggest a low complex approximation that relaxes the target number of aggregated workload slots to guarantee the movement of workload is less than certain threshold. The modified algorithm is given by 3. Fig. 8(c) presents an example of dynamic aggregation constrained by $S = 8$. The problem with this implementation is that it may not have a solution. We can swap line 22 and 23 in 3 to relax the target number of time slots to guarantee there is a solution.

2.4 Results and Discussions

The aggregation methods are proposed to reduce the computational complexity for large scale data centers. In practice, if the workload cannot be rearranged over time, aggregation by maximum should be used; otherwise, aggregation by mean can be adopted. The price of aggregation by maximum is over-provisioning which causes extra energy consumption. The price of aggregation by mean is the workload rearrangement. Dynamic aggregation is proposed to alleviate over-provisioning and workload rearrangement in aggregation by maximum and by mean, respectively. In this section we quantitatively study the pros and cons of proposed aggregation methods.

Algorithm 3 Improved Local Smooth Algorithm with Constraints on Advance and Delay

```
1: $j \leftarrow T$                                                   Initialization
2: for $i := 2 \rightarrow j$ do
3:     $si[i-1] \leftarrow (d[i] - d[i-1])$                           Compute smooth index
4: end for
5: while $j > \hat{T}$ do
6:     if $ss[ap] + ss[ap+1] \leq S$ then
7:         $ap \leftarrow$ InMin(|si|)                                 Find aggregation point
8:         $d[ap] \leftarrow \text{Mean}(d[ap], d[ap+1])$
9:         $d[ap+1] \leftarrow \text{Mean}(d[ap], d[ap+1])$           Aggregate
10:        $si[ap-1] \leftarrow d[ap] - d[ap-1]$                       Update smooth index
11:        $si[ap] \leftarrow d[ap+2] - d[ap+1]$                       Update smooth index
12:        $ss[ap] \leftarrow ss[ap] + ss[ap+1]$       Compute the size of aggregated slots
13:        if $ap \neq j-1$ then
14:            for $k := ap+1 \rightarrow j-1$ do
15:                $d[k] \leftarrow d[k+1]$   Adjust the index of slots behind the aggregation point
16:                $ss[k] \leftarrow ss[k+1]$            Adjust the corresponding slot size
17:                if $k < j-1$ then
18:                    $si[k] \leftarrow si[k+1]$                    Adjust the smooth index
19:                end if
20:            end for
21:        end if
22:        $j \leftarrow j-1$                            Decrease the number of slots by 1
23:    end if
24: end while
25: return d, ss
```

2.4.1 Experiment Setup

In our study, the server CPU frequency set and power consumptions are adopted from [29] except that we use the maximum frequency only. For ease of computation, the capacity of each server is normalized to 1. Greenberg *et. al.* [39] use \$.07 per killowatt-hours (kWh) as the utility price. We use the same utility price. The server energy consumption in a time slot is the product of the power consumption, the utility price and the slot size. Table 3 presents the specifications of server CPU.

Google reported [14] that the personnel cost for each repair is \$100 and the replacement cost is 10% of the server cost (\$2000). We assume the lifetime for a disk to be 60,000 switching-on-and-off cycles. Using this, we arrive at 0.5 cents for the switching cost per switching-on-off cycle. Out of 0.5 cents, we assume turning on to be a higher cost than turning off; thus, we split this value to 0.3 cents and 0.2 cents for turning on and turning off, respectively. We also assume that the power consumption for turning on and turning off is 0.02 cents and 0.005 cents, respectively, since turning on draws much more power than turning off in most cases. This analysis of the turning on/off cost is similar to one in [29] except that we differentiate the cost of switching on and off. The cost of switching on is 0.32 cents and the cost of switching off is 0.22 cents. When consolidation is performed, the source server needs to run for additional time to sustain the state of running applications. We assume the average extra time to be 77 seconds. We arrive at 0.015 cents per server switching off as the cost of consolidation. Therefore, the total cost of switching off is 0.22 cents. We assume that the utilization of the data center is 20%. Therefore, the average workload to the cloud is $I \times 0.2$. We also assume that the workload

Table 3: Specification of the server CPU

Frequency	Power Consumption	Power Cost	Normalized Capacity
2.6GHz	100watts	$.7\tau$	1

can be perfectly forecasted and profiled every 5 minutes. Due to the diurnal behavior associated with human beings' working cycles, we chose the 8 hour work time as the planning horizon where the dynamically changing workload from one time slot to another is generated for our study. Sinusoidal function and three different random distributions with the same average are used to generate temporally dynamic demand. The three random distributions are Erlang-2 (smooth), exponential, and two-state hyper-exponential (bursty). Each random distribution is generated 101 times using 101 independent random streams. Note that for the demand over the maximum capacity, i.e., I, we truncate the demand to I to make the problem feasible. We also want to study the workload that can be represented by certain deterministic cyclic functions. Assume that a day of workload can be represented by a full cycle of Sinusoid function and the 8 hour workload is in the range of 0 degree and 120 degree. We first generate the value given by a "plain" Sinusoid function in the range of 0 degree and 120 degree:

$$\tilde{d_t} = \sin(t \times 2 \times \pi/3/96), \forall t = 1, \cdots, 96. \tag{2.49}$$

Then the 8 hour workload load demand with average of $0.2 \times I$ is given by:

$$d_t = (\tilde{d_t} - \sum_{t \in \mathcal{T}} \tilde{d_t}/96 + 1) \times 0.2 \times I, \forall t = 1, \cdots, 96. \tag{2.50}$$

Fig. 9 presents an example of the 8 hour sinusoid workload derived from (2.50) given $I = 100$. Compared with three workload generated by random distributions, since this

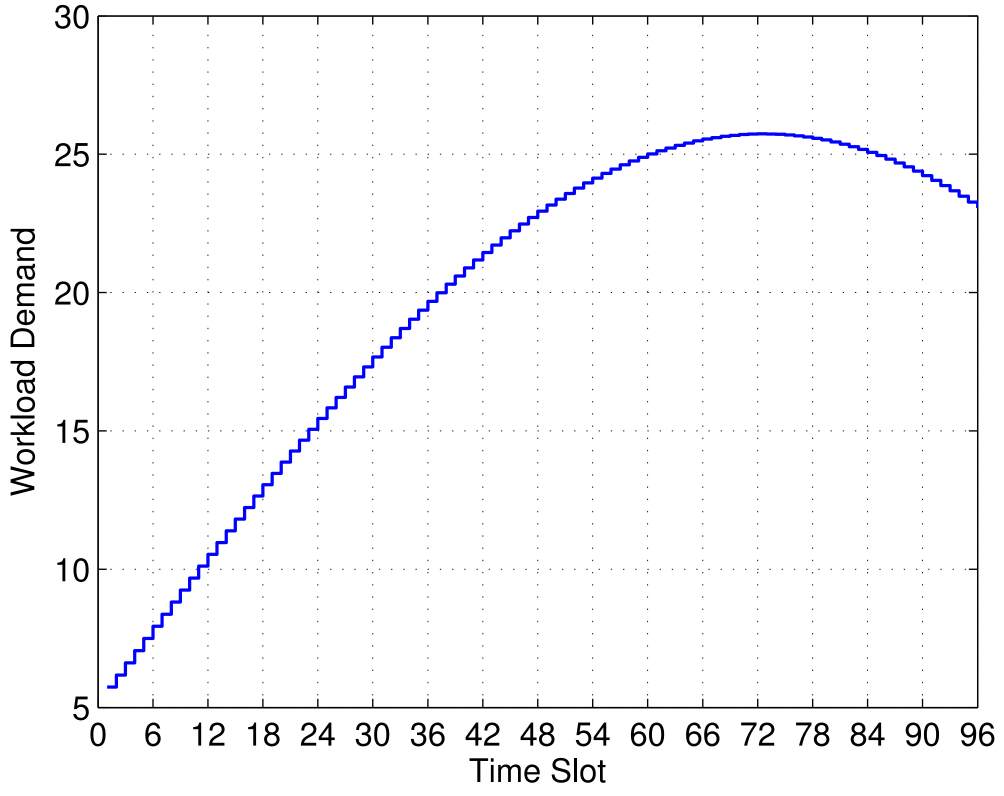


Figure 9: A sinusoid workload example when $I = 100$

workload is deterministic, we also call it deterministic sinusoidal workload. The sinusoid workload is the most smooth one among all the workload cases.

However, the cost components may change due to the fluctuation of power cost and technology advancements. We define a cost model to consider this factor. We weight the utility price by β and the wear-and-tear cost by $1-\beta$. We define cost of running a 100-watt server for 5 minutes as $\beta \cdot 0.014 \cdot 100/12$. The switching on cost due to wear-and-tear is defined by $(1-\beta)\cdot 0.6$. The switching off cost due to wear-and-tear is defined by $(1-\beta)\cdot 0.4$. The power consumption of switching a server on and off is defined by $\beta \cdot 0.04$ and $\beta \cdot 0.01$, respectively. The power consumption to do consolidation when switching off a server is defined by $\beta \cdot 0.03$. Thus we have cost of $\beta \cdot 7/60$ for running a 100 watt server for 5 minutes, cost of $0.6-0.56\beta$ for switching a server on and $0.4-0.36\beta$ for switching a server off. Previously derived costs can be obtained by letting $\beta = 0.5$. Except for studying the impacts of varying β, we fix $\beta = 0.5$. Note that $\beta = 0.5$ results in the above derived parameters.

We ran the optimization model using CPLEX through Matlab on an Intel(R) Core(TM)2 Duo CPU U9400 1.40GHz with 4GB memory. Due to the memory has limitation to hold the matrixes, we were not able to run the scenarios that have a large number of variables. Nevertheless, our goal here is to understand performance and cost savings.

2.4.2 Computational Complexity

We first study how the number of servers and the number of time slots of workload affects computational time in Model-Het. We fix the number of time slots in workload as

96 and vary the number of servers from 10 to 100 with incremental step 10. The optimal cost and computational time is shown in Fig. 10(a) and (b), respectively. The optimal cost and computational time is both linear with respect to the number of servers. Then we fix the number of servers to 100 and vary the number of time slots in workload from 10 to 100 with incremental step 10. The optimal cost and computation time is shown in Fig. 11(a) and (b), respectively. Note that Fig. 11(b) is on log scale on y-axis. In Fig. 11(a), the optimal cost is linear with respect to the number of number of slots. In Fig. 11(b), the computational time is approximately exponential increasing with respect to the number of slots[3]. The computational time increase linearly with respect to the number of servers while exponentially with respect to the number of time slots because increasing the number of servers does not increase the number of constrains while increasing the number of time slots does.

To quantify the difference of computation time among three proposed models, we run the same problem in three models. The cloud system consists of 100 identical servers (i.e., $I = 100$). The workload with 96 time slots is generated according to four aforementioned distributions. This problem fits into Model Hom. To run Model Het, we "cheat" that the servers are different. Thus we assign a binary variable for each server at each time slot and model this problem by Model Het. By dividing the servers into 10, 20 and 50 clusters, we model these three problems by Model HH; these three cases are denoted by HH-10, HH-20 and HH-50, respectively. Fig. 12 shows that the homogenous case has the least computational time while the heterogenous case has the most computational time

[3]The pattern is really between linear and exponential. The reason is that the computational time is too small to observe the pattern.

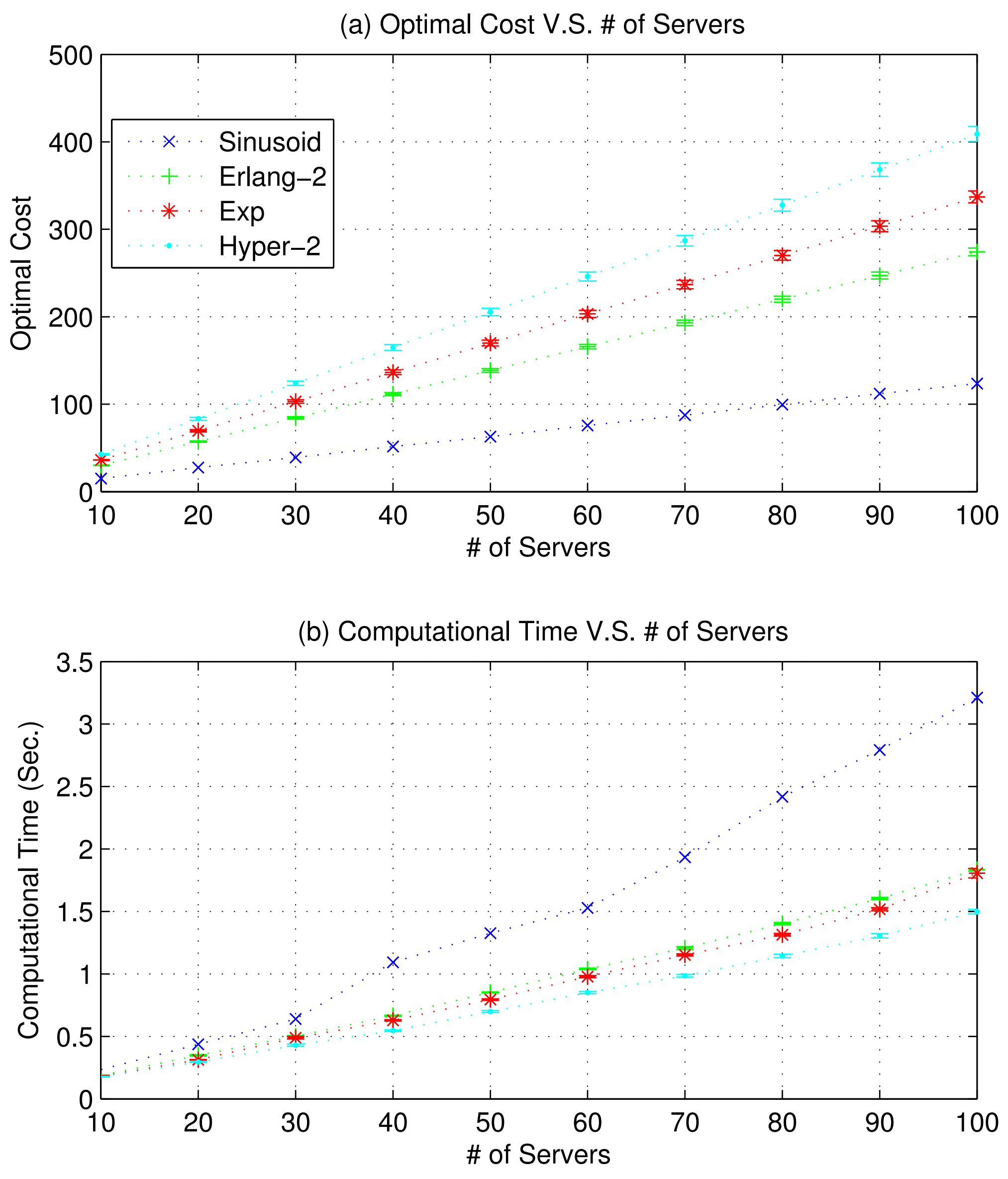


Figure 10: The cost and computational time with respect to different number of servers in a heterogeneous model with 96 time slots

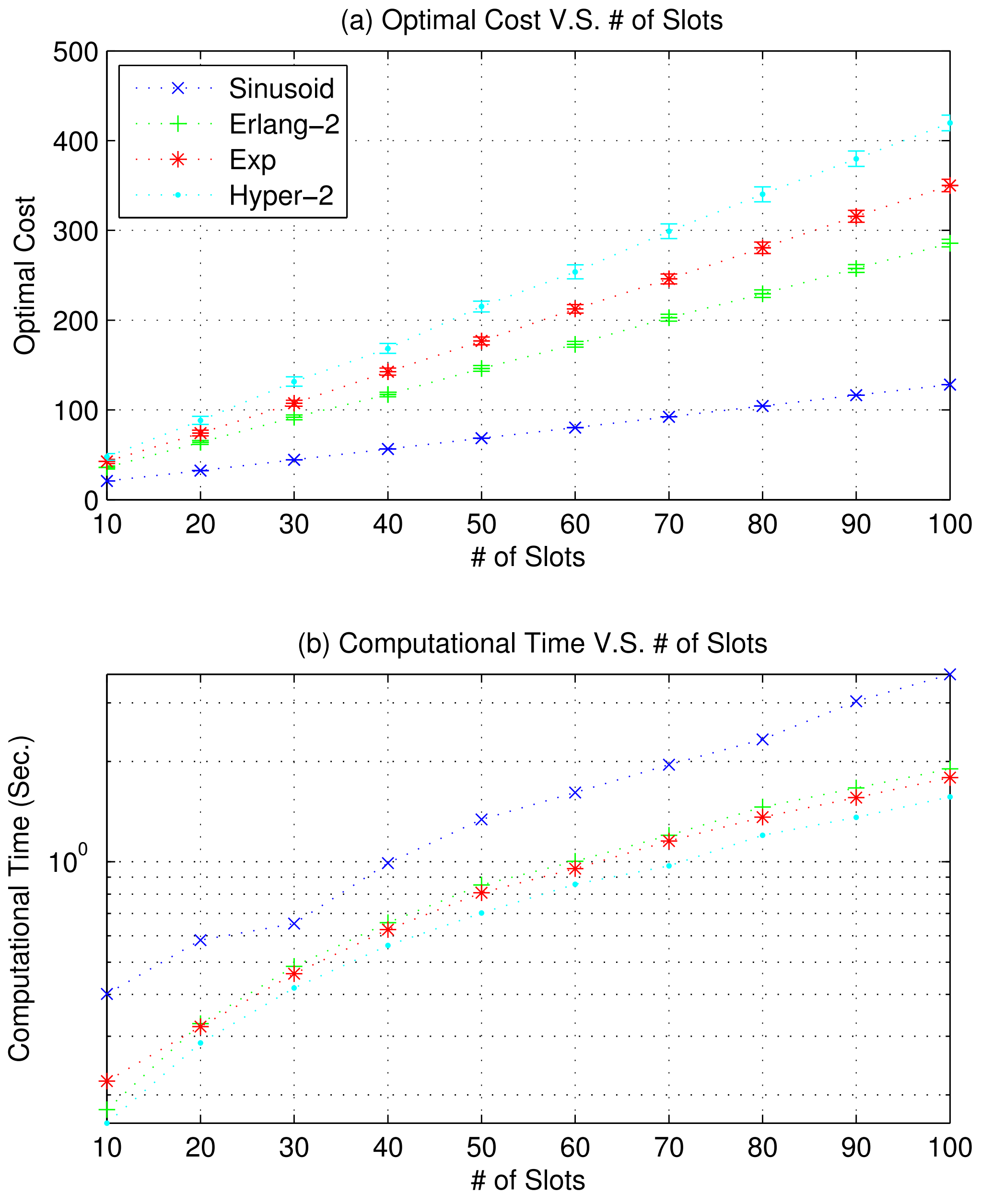


Figure 11: The cost and computation time with respect to different number of slots in a heterogeneous model with 100 servers

consistently for all four workloads. The computational time in the heterogeneous case is in the order of a few of ten times more than that for the homogeneous case. It also shows that a lesser number of clusters results in less computational time.

For the same workload distribution, all experiments end up with the same optimum, which is summarized in Table 4. In Table 4, the *Global Optimal* represents the optimal solution obtained from our method; the *Static Configuration* means the solution obtained from statically keeping all servers running all the time; *Local Optimum* is solution from the optimization where switching on and off cost is not considered. For the cost of Local Optimum, we further decompose it into *Switch Cost* and *Energy Cost.* The energy cost is supposed to be the same since the average of the workload is the same. The energy cost of Hyper-2 distribution is slightly smaller than other three cases because the workload in Hyper-2 is more likely to surpass the maximum offered capacity and thus is truncated. Because sinusoidal workload is deterministic, there is no confidence interval for the solutions of this workload. Note that Switch Cost portion includes the energy cost of performing consolidation. Due the significance of Switch Cost, the result from Local Optimum are even worse than that from Static Configuration in Exp and Hyper-2 cases. The Switch Cost is correlated to the "regularity" of workload shape. By applying Global Optimum, the cost savings over Static Configuration and Local Optimum are significant. Compared with Static Configuration, we achieve approximately 78%, 51%, 40% and 27% savings for sinusoid, Erlang-2, Exponential and Hyper-exponential-2, respectively. Compared with Local Optimum, we achieve approximately 46%, 46% and 41% for Erlang-2, Exponential and Hyper-exponential-2, respectively. It is interesting that the local optimum is to the global optimum in sinusoid workload. It is because of the structure of this sinusoid workload. Although we report results for four distributions here, understanding the influence of different workload distributions on the optimal cost is outside the scope of this paper. To the best of our knowledge, [71] is the first effort to decompose workload into certain substructures and compute optimal solution by the substructures. A more theoretical study are presented in [55].

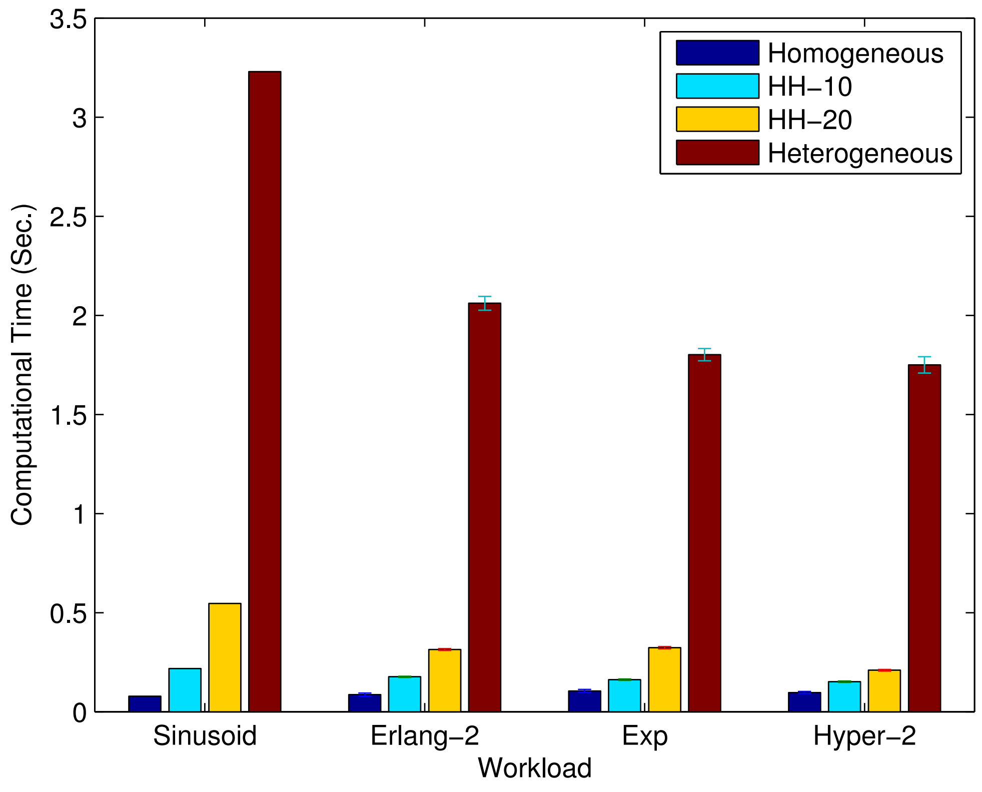


Figure 12: Computational time comparison of different models under different workload

Table 4: Optimal cost of different cases

Workload Type	Sinusoid	Erlang-2	Exp	Hyper-2
Global Optimum	123.5017	274.2851±4.3093	337.0227±6.8391	409.0129±8.5715
Static Configuration	561.1667	561.2562±0.1661	561.0864±0.1657	561.0731±0.2559
Local Optimum	123.5017	507.2604±8.1048	633.0925±12.8482	699.5993±19.5957
Switch Cost of Local Optimum	8.76	391.8281±7.1364	518.0794±11.4349	596.9251± 17.2464
Energy Cost of Local Optimum	114.7417	115.4359±1.4381	115.0131±1.7569	102.6741±2.5808

2.4.3 Insights of Aggregation

Before going to the results of aggregation, we discuss how aggregation influences the optimal. We define the degree of aggregation as the ratio of the number of original demands and aggregated demands. Denoting the degree of aggregation by α, we have $\alpha = \lceil T/\hat{T} \rceil$. Aggregation by maximum always causes over-provisioning. Energy to keep more than necessary servers running is wasted due to over-provisioning. On the other hand, the aggregation by maximum smoothes out the regularity of the workload, *i.e.*, the fluctuation of workload is alleviated which loosens the switching requirements. Moreover, the higher the degree of aggregation is, the more smooth the workload becomes. *Strict* dynamic aggregation is better than static aggregation only in terms of avoiding as much over-provisioning as possible. As to the second aspect of the cost: switching cost, we can not tell whether dynamic aggregation is better than static aggregation or not. Consequently, when considering the total of two cost components, it is possible that the total cost of static aggregation is even less than dynamic aggregation. This happens in the case that static aggregation gain more switching cost saving than dynamic aggregation and this difference is larger than the gain of over-provisioning energy consumption of dynamic aggregation over static aggregation. For the same reason, the optimal of high degree of aggregation maybe better than that of low degree of aggregation. This is called Insight 1 for ease of reference.

It is also noted our local smooth implementation may *not* be better than static aggregation in terms of avoid as much over-provisioning as possible since the implemented algorithm is an approximation based on local information. However, this local smooth implementation always favors smoothing workload and thus tends reduce switching cost. We call this Insight 2.

In aggregation by mean, the static aggregation and dynamic aggregation end up with the same level of average offered capacity since "mean" is used. That is, aggregation by mean does not cause over-provisioning and thus the energy consumption of static and dynamic aggregation by mean is equivalent. Thus which aggregation method costs less is solely decided by the switch cost which is determined by the fluctuation of the workload. This is called Insight 3.

2.4.4 Aggregation by Maximum

We now study the pros and cons of aggregating the workload by maximum. For this study, we consider an example with 5,000 identical servers in a data center and the servers are clustered into 50 100-homogenous-server groups for the purpose of management. Thus this falls into Model-HH. We run the optimization for this system with different degrees of aggregation i.e., $\alpha = 1, 2, 3, 6, 8, 12$. Note that $\alpha = 1$ means that there is no aggregation.

We first do static aggregation. As shown in Fig. 13, the optimum rises while the computational time decreases when the degree of aggregation augments in all workload cases. The gradient of the computational time with regard to the degree of aggregation decreases as the degree of aggregation goes up in all four cases. That means that the degree of aggregation in a smaller range (*e.g.*, 1-3 in this experiment) has more significant affect than that in a larger range (*e.g.*, 6-12 in this experiment). Compared with computational time for $\alpha = 12$, the computational time for $\alpha = 1$ increases more than 100 times for all four workload cases. This also confirms the computational complexity increases exponentially with respect to the number of time slots. On the other hand, the gradient of the optimum, with regard to the degree of aggregation, does not change too much in the whole observed range in all workloads. Compared with the optimal cost for $\alpha = 1$, the optimal cost for $\alpha = 12$ only increases approximately 5%, 16%, 19%, 24% for Sinusoid, Erlang-2, exponential and hyper-exponential-2 cases, respectively. It is observed that a small degree of aggregation reduces the computational time noticeably without significantly increasing the overall cost. Computing the optimal solution by decomposing workload is studied in [55, 71].

Now we do dynamic aggregation for the same example. The results are shown in Fig. 14. The pattern of change and the order of magnitude of computational time is very similar to that in the case of static aggregation since the number of variables and the number of constraints in dynamic aggregation are both the same as their counterparts in the static aggregation case. The optimum of static aggregation consistently increases when the degree of aggregation augments. However, this is not the case for some workloads in dynamic aggregation. It does not always follow this pattern in three random workload

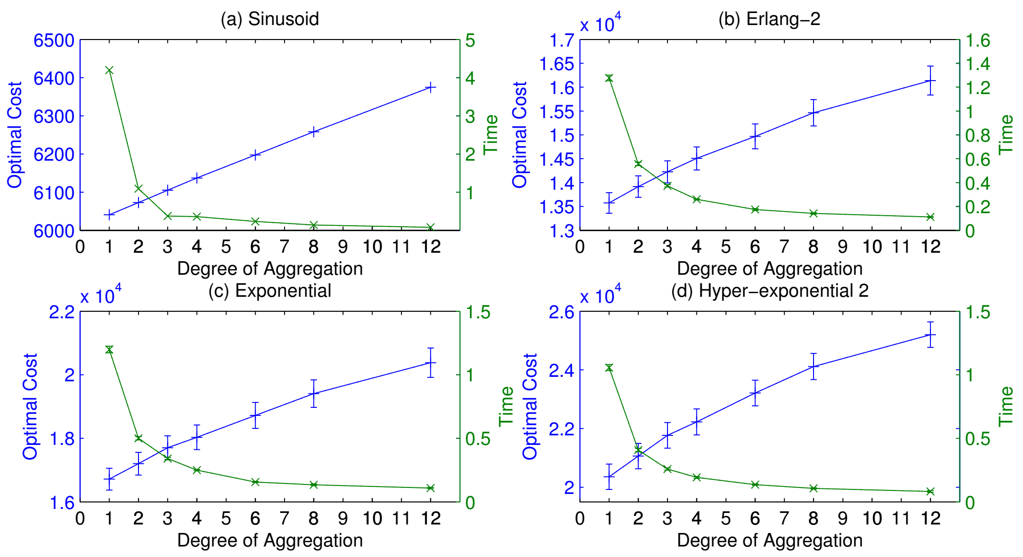


Figure 13: The optimal cost and computational time in static aggregation by maximum

cases. It is because our local smooth implementation of dynamic aggregation relies on local information and this may not be able to achieve the global optimum of minimizing energy cost but favors reducing switching costs as we have mentioned in Insight 1. It is worthy to note (a) the violation of the pattern is relatively slight; (b) there is no statistical difference due to the confidence interval overlap in most pattern violation events. More importantly, as we can see in Fig. 15, the dynamic aggregation outperforms static aggregation in all cases excerpt for $\alpha = 48$ in sinusoid, $\alpha = 2, 3, 4$ in Hyper-exponential-2 (there is no statical difference due to the confidence interval overlap). The energy consumption of over-provisioning due to aggregation by maximum is presented in Fig. 2.4.4. For random distributed workload cases, the implemented dynamic aggregation only reduces-over provisioning when degree of aggregation is small (≤ 4 for Erlang-2 and Exponential, ≤ 6 for Hyper-exponential. This also confirms that our implemented dynamic aggregation helps reducing workload fluctuation. As we have shown early the computational time is not related to how to aggregate. Therefore, our implementation of dynamic aggregation shows that it is a good choice over static aggregation in most cases. As we can see in Fig. 15(b),(c),(d), the discrepancy between static and dynamic aggregation forms an elliptical shape: it starts from 0 (no aggregation need to performed when $\alpha = 1$), then increases, then decreases and converges back to 0 (When $\alpha = 96$, the number of time slots is 1. Thus the aggregated workload demand becomes the largest workload demand of all original time slots).

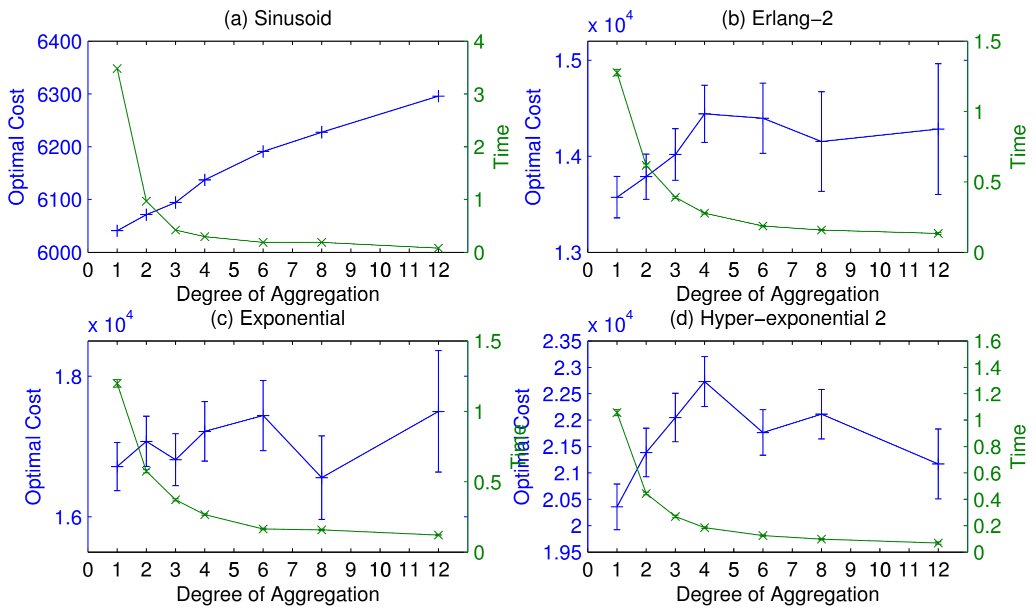


Figure 14: The optimal cost and computational time in dynamic aggregation by maximum

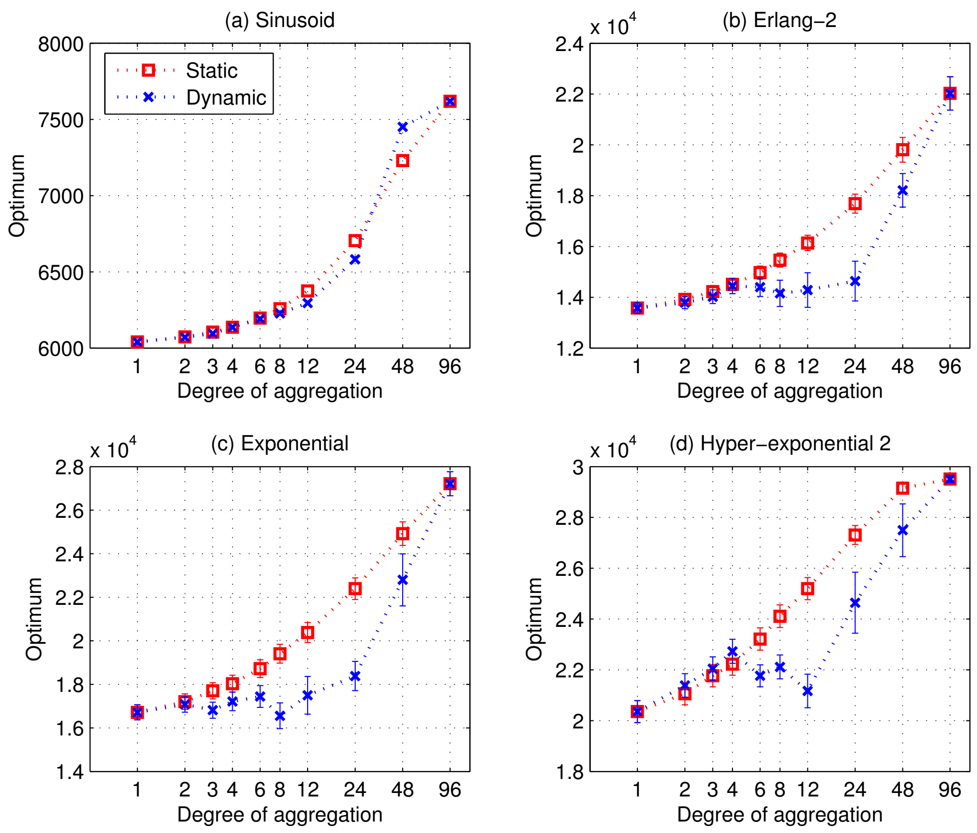


Figure 15: Comparison between static and dynamic aggregation

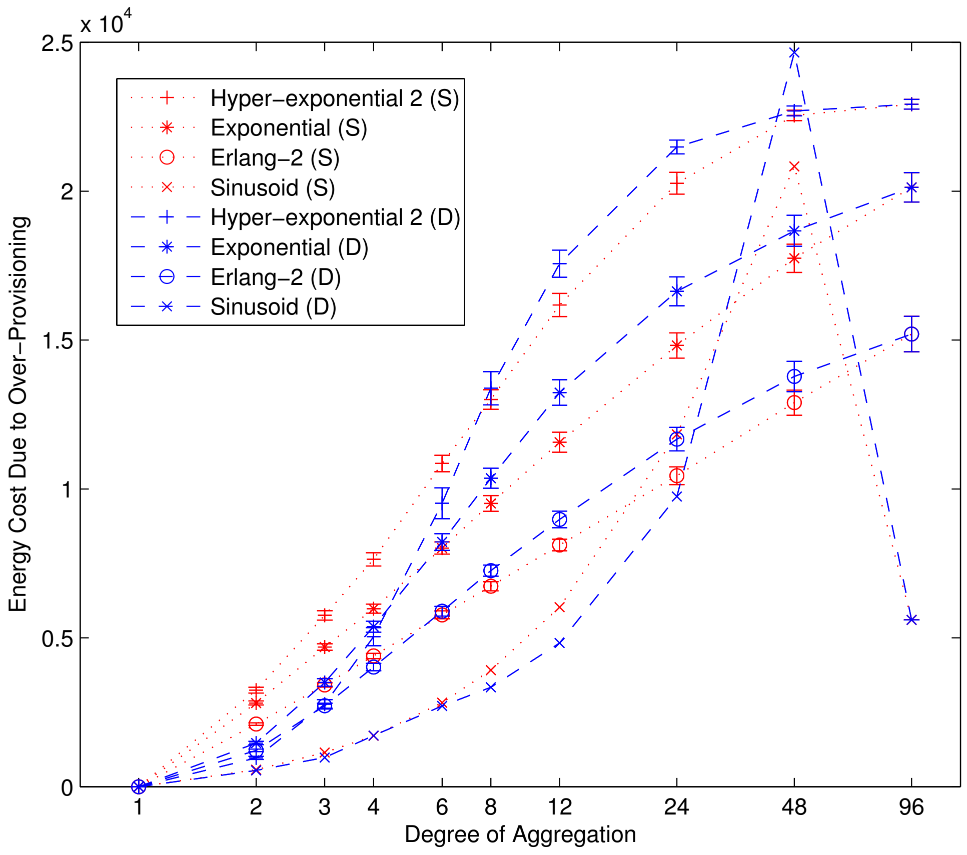


Figure 16: The energy costs due to over-provisioning in static (S) and dynamic (D) aggregation in dynamic aggregation by maximum

2.4.5 Aggregation by Mean

Fig. 17 shows the optimal costs and computational time when we do static aggregation by mean. In all three random distributed workloads, the optimal cost and computational time are both monotonically increasing with respect to the degree of aggregation as we have explained in Insight 3. The gain for both optimum and time complexity comes with the price of reallocating of workload demand.

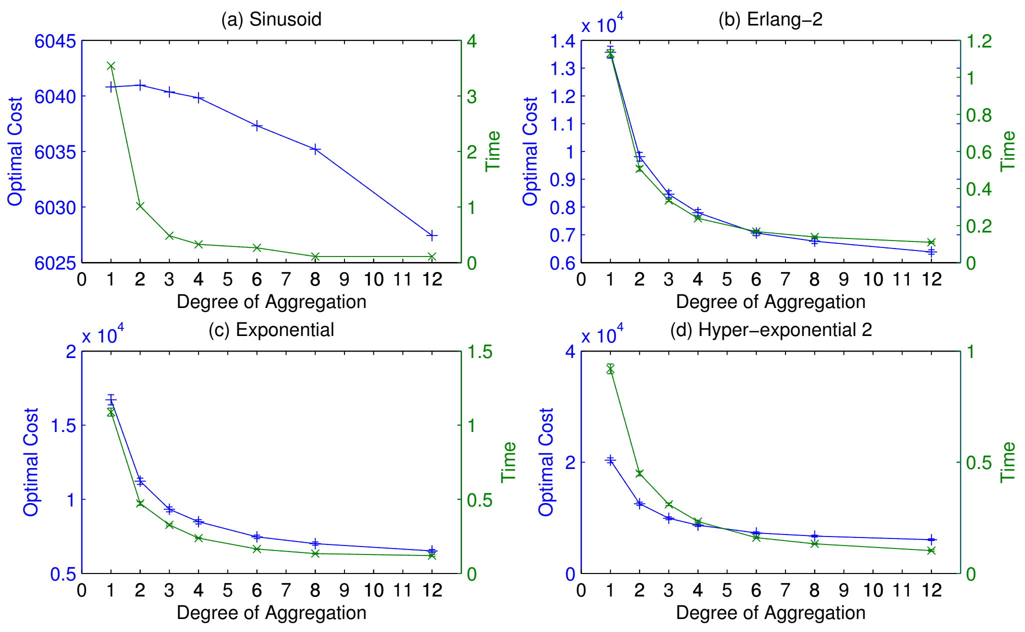


Figure 17: The optimal costs and computational time in static aggregation by mean

We now present the optimal costs and computational time when applying dynamic aggregation by mean in Fig. 18. We compare the dynamic aggregation with static aggregation. For the computational time, the conclusions are very similar to aggregation by maximum case. As to the optimal costs, the conclusions are totally different form aggregation by maximum case. As shown in Fig. 19, the static aggregation outperforms dynamic aggregation in all three random distributed workload cases. We have explained in Insight 3, the target of dynamic aggregation in the aggregation by mean is to reduce the reallocating workloads. Therefore, we also present comparative results of the amount of workload rearrangement in both static and dynamic aggregation by mean in Fig. 20. F. For all three random distributed workload cases, the amount of workload rearrangement of dynamic aggregation is smaller than that of static aggregation while that of dynamic aggregation is bigger than that of static aggregation in Sinusoid workload. This suggests that the implemented dynamic aggregation is suitable to be used in the highly fluctuated workload cases. As the degree of aggregation changes from 1 to 96, the gap starts from 0, then it increases to the largest value, then converges back to 0.

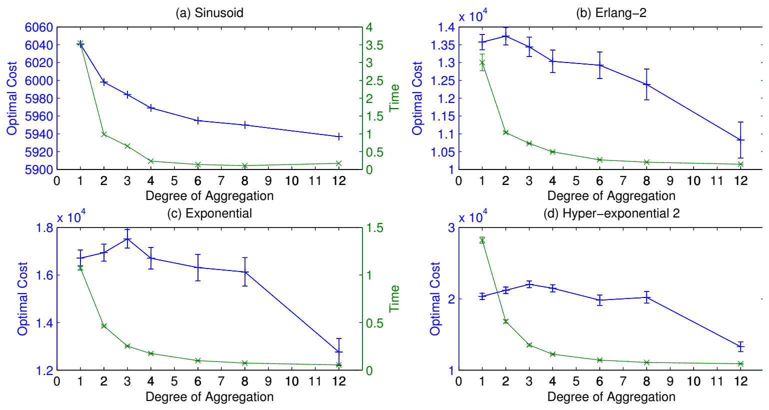


Figure 18: The optimal cost and computational time in dynamic aggregation by mean

2.4.6 Impacts of Varying Cost Components

In addition, we want to understand the impact of cost components change. We still use the 5000 server scenario used for previous aggregation study. Note that three models

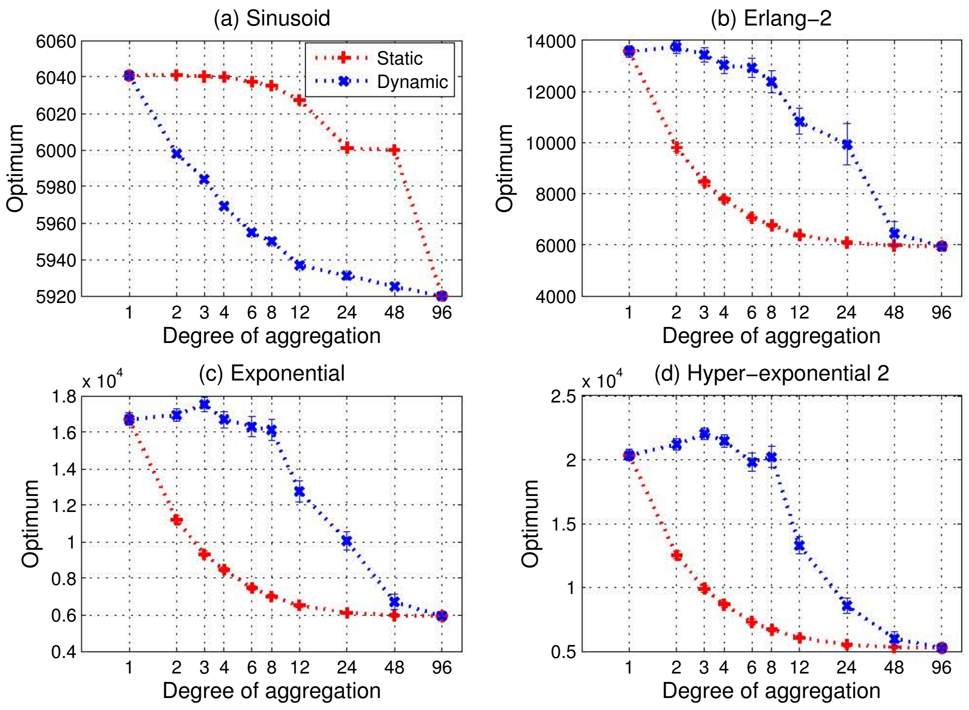


Figure 19: Comparison between static and dynamic aggregation by mean

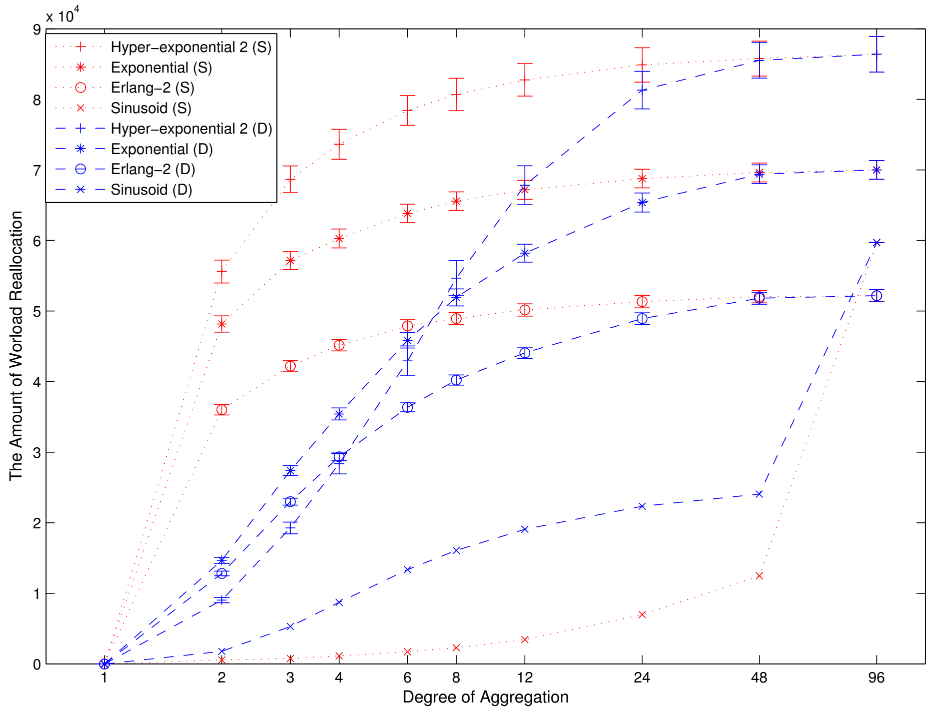


Figure 20: The amount of workload rearrangement in static (S) and dynamic (D) aggregation by mean

yields the same optimum (but different computational time), there is no difference for these three models when only optimum is considered. We vary the weight (β) of utility price from 0 to 1. That is, we consider the range from that the utility price is negligible to that the wear-and-tear cost is negligible.

Fig. 21 presents the optimum obtained under aggregation by maximum. As we can see in Fig. 21(a),(e), the optimum cost is almost linear to β for all degrees of aggregation and both static aggregation and dynamic aggregation . The reason is that the switching cost in the sinusoid workload is very small compared with the energy cost since the sinusoid workload is very "smooth" . This is not the case for other three workload cases since their wear-and-tear cost is comparable to the energy cost . As shown in Fig. 21(b),(c),(d),(f),(g),(h), the plots of the optimum with respect to β are concave curve. The degree of concavity is negatively proportional to the degree of aggregation (α). As the degree of aggregation goes bigger, the workload becomes more smoother and thus causes the switching cost smaller. Therefore when the degree of aggregation does not exceed certain threshold (the threshold depends on the workload distribution), the optimum increases first then decreases with respect to the increasing of β. This important observation suggests that if the weight of switching cost is high, we can use high degree of aggregation to save computational time without compromising the optimum. The gaps among different degrees of aggregation increases as β increases.

In aggregation by mean, we see similar pattern regarding the concavity (the optimum with respect to beta) and the degree of concavity with respect to the degree of aggregation. Fig. 22 presents the same set of plots for aggregation by mean.

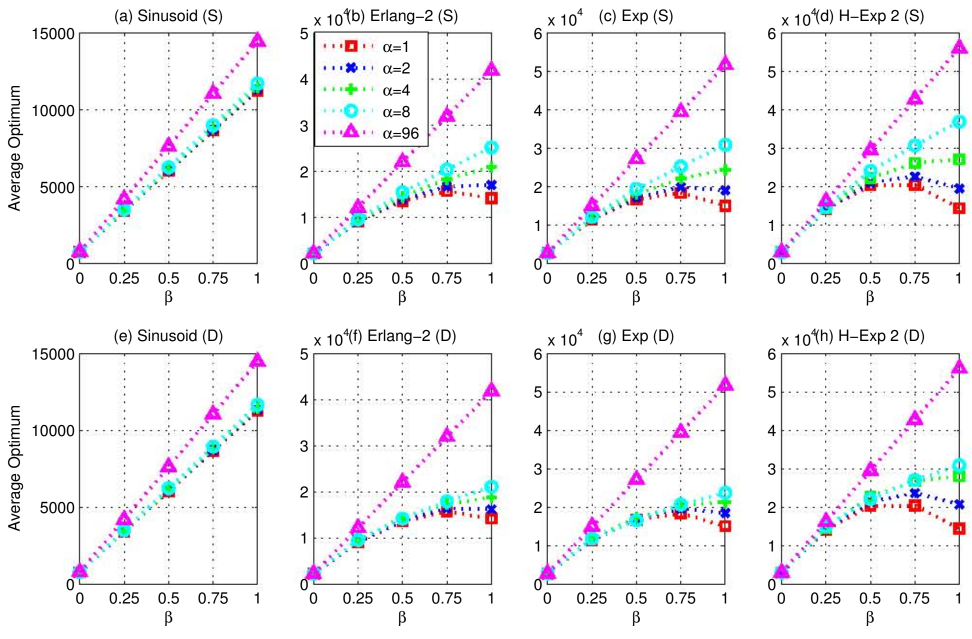


Figure 21: Varying cost component weight in 5000 server scenario adopting aggregation by maximum

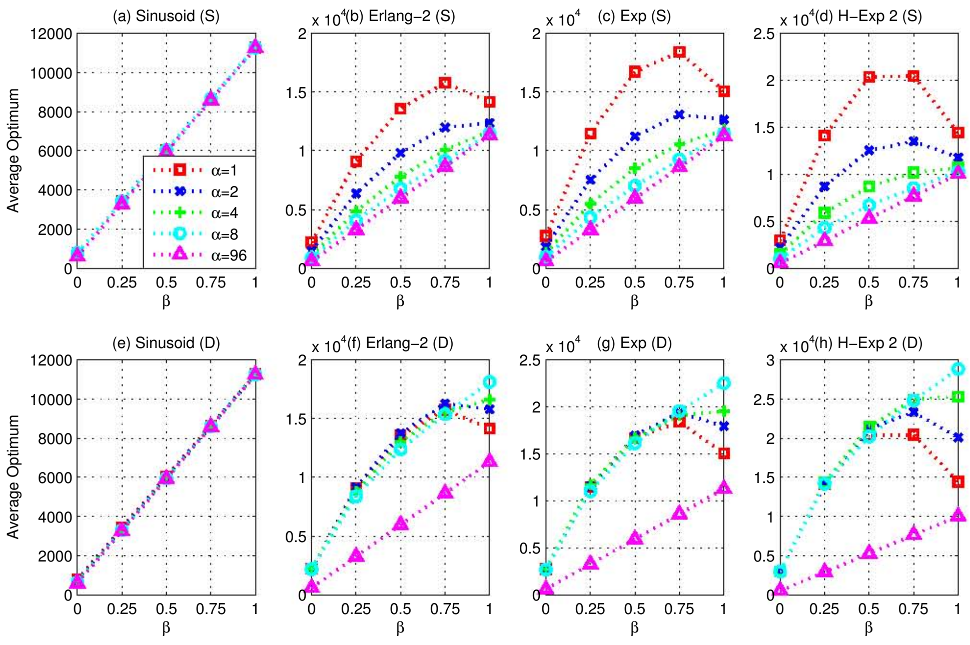


Figure 22: Varying the cost component weight in 5000 server scenario adopting aggregation by mean

2.5 Conclusion and Future Works

In this paper, we first introduce three formulations for different data center environments: homogeneous data center, heterogenous data center, and heterogenous homogenous-server-cluster data center. The computational time to obtain the optimum varies significantly in these three cases. In order to achieve on-line (or close to on-line) computation for large scale data centers, we propose to aggregate workload to fewer time slots. Depending on the requirements of applications, there are two types of aggregation modes. Aggregation by maximum guarantees the workload demand of every time slot is satisfied while aggregation by mean needs to delay or advance the workload demand. On the other hand, aggregation by maximum causes over-provisioning while aggregation by mean does not.

For each of aggregate modes, we propose two aggregation methods: static and dynamic aggregation. Aggregating fixed number of time slots into one is called static aggregation while aggregating with certain objective is named dynamic aggregation. In aggregation by maximum mode, the objective of dynamic aggregation is to minimize the over-provisioned capacity. In the aggregation by mean mode, the objective of dynamic aggregation is to minimize the delay and advance workload demands. An approximation implementation of dynamic aggregation is introduced to alleviate the computational overhead of implementing the exact algorithm.

Our numerical results show that aggregation is a efficient method to reduce the computational time. Choosing appropriate degree of aggregation is a tradeoff between the cost and the computational time. We observe that the dynamic aggregating method in both modes can achieve significant gain compared with the static aggregation approach in terms of their individual objective function. The study on varying of the cost component weights shows that appropriate degree of aggregation also depends on the weights. Although our current study is based on artificially generated workload using a number of random distributions as well as a deterministic sinusoidal shape, the proposed methods are general to apply to realistic workload.

We are working on decomposing workload and using the decomposed substructures of workload to determine optimal solutions. The other important direction we need to pursue in the future is to explore the solutions for unpredictable, partially predictable workloads. The partially predictability refers to: (a) we can only accurately predict certain length of time *not* the entire time horizon; (b) The predicted workload demand is not accurate (for example, in certain range); (c) combination of point (a) and (b). The first direction is actually the key to solve the second direction.

CHAPTER 3

DATA CENTER RESOURCE ALLOCATION WITH DVFS

In this chapter, we present an optimization model over a time horizon by consider it as a multi-time period problem with demand changing over time, where we address optimizing the energy consumption and server cost based on both switching on and off and DVFS methods in data centers. The aggregation by maximum discussed in Chapter 2 is applied. We show the impact of the degree of aggregation and CPU frequency options on optimal solutions. The dependence of the degree of aggregation on cost parameters sheds a light on how to choose the degree of aggregation. We show that it is indispensable to put the problem in the multi-time period framework. Although we use hypothetical demand distribution in our numeric study, our approach is applicable to arbitrary demand distribution.

The rest of this chapter is organized as follows. We present the optimization formulation in Section 3.1. In Section 3.2, the evaluation environment setup and results are presented. Section 3.3 discusses conclusion and further work.

3.1 Problem Formulation

For sake of clarity, we introduce our notations used in this chapter. The CSP data center has I servers. Let $\mathcal{I}$ denote the set of the servers. Let $\mathcal{J}(i)$ be the frequency option set for server i. In a homogeneous server cluster case, $\mathcal{J}(i) = \mathcal{J}, \forall i$. There are J

frequency options in $\mathcal{J}$. Server i running at j-th frequency option can offer a capacity of v_{ij} while satisfying the SLA. The power consumption of running server i at j-th frequency is denoted by c^p_{ij} per time unit. The wear-and-tear cost of turning server i on and off is denoted by c^{s+}_i, c^{s-}_i, respectively. Let $\mathcal{K}$ be the set of the services that is using the data center infrastructure consisting of K services. We divide the time horizon Υ hours into T equal time slots and the *duration of a time slot (slot size)* is then $\tau = \Upsilon/T$ hours (usually, we refer to the slot size in minutes). Let $\mathcal{T}$ be the set of these slots. At the review point, the number of active servers and their frequencies is configured. Let binary decision variables y_{ijt} denote if server i is running at frequency option j at time slot t. The continuous variables x_{ijkt} ($0 \leq x_{ijk}(t) \leq 1$) represent the proportion of service k hosted by server i that is running at frequency option j at time slot t.

There are a number of constraints in this problem. Usually, a server can only be operated at a specific frequency in a time slot—this can be represented as follows:

$$\sum_{j\in\mathcal{J}} y_{ijt} \leq 1,\ i = 1, \cdots, I;\ t = 1, \cdots, T. \tag{3.1}$$

The second constraint is that the total utilization of any running server must not be more than 1 and that of any “off” server must be 0.

$$\sum_{k\in\mathcal{K}} x_{ijk}(t) \leq y_{ij}(t),\ i = 1, \cdots, I,\ j \in \mathcal{J},\ t = 1, \cdots, T. \tag{3.2}$$

The third constraint is the demand requirement of each service over the entire planning horizon:

$$\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} v_{ij} \cdot x_{ijk}(t) \geq d_{kt},\ k = 1, \cdots, K;\ t = 1, \cdots, T. \tag{3.3}$$

The constraint (3.3) is a combined constraint of assigning servers to services and satisfying the demand of each service. We do not consider the assignment/allocation problem in this chapter although the service assignment problem is an important one [59]. Enabled by virtualization and consolidation, (3.2) and (3.3) can be combined as the aggregated demand constraint:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} v_{ij}\cdot y_{ijt} \ge \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} v_{ij}\cdot x_{ijkt} \ge \sum_{k\in\mathcal{K}} d_{kt}. \tag{3.4}$$

Let $\sum_{k\in\mathcal{K}} d_{kt} = d_t$; then, the demand constraint reduces to:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} v_{ij}\cdot y_{ijt} \ge d_t,\ t = 1,\cdots,T. \tag{3.5}$$

We next consider the objection function. The objective is to minimize the operational cost of running servers over the entire horizon that can be represented by

$$\begin{aligned}&\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c^p_{ij}\cdot y_{ijt} + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}\Big(c^{s+}_i\cdot\sum_{j\in\mathcal{J}} y_{ijt}\cdot\big(\sum_{j\in\mathcal{J}} y_{ijt} - \sum_{j\in\mathcal{J}} y_{ij(t-1)}\big) + \\ &c^{s-}_i\cdot\sum_{j\in\mathcal{J}} y_{ij(t-1)}\cdot\big(\sum_{j\in\mathcal{J}} y_{ij(t-1)} - \sum_{j\in\mathcal{J}} y_{ijt}\big)\Big).\end{aligned} \tag{3.6}$$

This objective function is a quadratic function. Given that y_{ijt} are decision variables, we can reduce the quadratic function to a linear function (by introducing additional variables and constraints) without resorting to any approximation. To do this, we introduce two binary variables y^+_{it} and y^-_{it} to represent turning on/off at review point of time slot t. $y^+_{it} = 1$ means server i is turned on at time t. Conversely, $y^-_{it} = 1$ means server i is turned off at time t. 0 indicates no change from time slot $t-1$ to t. Thus, we have:

$$\sum_{j\in\mathcal{J}} y_{ijt} - \sum_{j\in\mathcal{J}} y_{ij(t-1)} - y^+_{it} + y^-_{it} = 0,\ i = 1,\cdots,I;\ t = 1,\cdots,T. \tag{3.7}$$

Since y_{it}^{+} and y_{it}^{-} cannot be both 1 at the same review point, we use the following inequalities to force this requirement:

$$y_{it}^{+} + y_{it}^{-} \leq 1,\ i = 1, \cdots, I;\ t = 1, \cdots, T. \tag{3.8}$$

With the aid of y_{it}^{+} and y_{it}^{-}, the original quadratic objective function can be transformed to the following linear function:

$$\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij}^{p} \cdot y_{ijt} + \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} (c_i^{s+} y_{it}^{+} + c_i^{s-} y_{it}^{-}). \tag{3.9}$$

Since we consider the planning horizon to be the entire time period Υ, we make the assumption that at the beginning of this period, a reshuffling is done. In other words, all servers are reset at the beginning of the time horizon. In our case, we use:

$$y_{ij0} = 0,\ i = 1, \cdots, I;\ \forall j \in \mathcal{J}. \tag{3.10}$$

This then forces the new binary variables in the following way:

$$y_{i1}^{+} = \sum_{j \in J} y_{ij1},\ i = 1, \cdots, I, \quad y_{i1}^{-} = 0,\ i = 1, \cdots, I. \tag{3.11}$$

For sake of clarity, we rewrite the entire problem formulation in Fig. 3.1

In this formulation, the number of decision variables is $(J+2) \cdot I \cdot T$. The number of constraints is $(3I+1) \cdot T + 2I$, where $J = \#(\mathcal{J}), I = \#(\mathcal{I}), T = \#(\mathcal{T})$ ($\#(\cdot)$ denotes cardinality of the set).

3.2 Evaluation

In this section, using the optimization model, we focus on understanding: (a) the impact of the degree of aggregation on optimal solutions, (b) to what extent DVFS can

constants

c^p_{ij} energy consumption of server i running at frequency option j in a time slot
c^{s+}_i cost of switching server i on
c^{s-}_i cost of switching server j off
v_{ij} capacity of server i running at frequency j
d_t demand at t
$y_{i0} = 0$ initial (time slot 0) state of server i

variables

$y_{ijt} = 1$ if server i is running at frequency option at time slot t; 0, otherwise
$y^+_{it} = 1$ if server i is turned on at time slot t; 0, otherwise
$y^-_{it} = 1$ if server i is turned off at time slot t; 0, otherwise

objective

minimize $\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c^p_{ij}\cdot y_{ijt} + \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}(c^{s+}_i y^+_{it} + c^{s-}_i y^-_{it})$

constraints

$\sum_{j\in\mathcal{J}} y_{ijt} \leq 1,\ i = 1,\cdots,I;\ t = 1, cdots, T$
$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} v_{ij}\cdot y_{ijt} \geq d_t,\ t = 1,\cdots,T$
$\sum_{j\in\mathcal{J}} y_{ijt} - \sum_{j\in\mathcal{J}} y_{ij(t-1)} - y^+_{it} + y^-_{it} = 0,\ i = 1,\cdots,I;\ t = 1,\cdots,T$
$y^+_{it} + y^-_{it} \leq 1,\ i = 1,\cdots,I;\ t = 1,\cdots,T$
$y^+_{i1} = \sum_{j\in J} y_{ij1},\ i = 1,\cdots,I$
$y^-_{i1} = 0,\ i = 1,\cdots,I$

Figure 23: Binary IP: resource allocation with DVFS

improve the optimum, and (c) how the relative change in the wear-and-tear cost compared to the power consumption cost impact on the optimal solution. In order to do these studies, we start with our evaluation setup and parameters values considered.

3.2.1 Evaluation Setup

In our study, we consider a server cluster of 100 identical servers. The CPU frequency set and power consumptions are adopted from [29]. The capacity of a server (v'_j) is assumed to be a linear function of frequencies (F_j) with a fixed cost that is calculated as follows:

$$v'_j = \alpha + F_j, \forall j \in \mathcal{J}, \text{ where } \alpha \in \mathbb{R}, \alpha > -F_j. \tag{3.12}$$

There are different units that can be used for server capacity such as processors requested, HP computon, or SAPS. Without loss of generality, we can simply use numeric values to convey this information while assuming that the demand is using the same unit. We capitalize on α to make our model represent different measurements. For ease of computation, we normalize the capacity at j to be the capacity at the maximum frequency. That is,

$$v_j = v'_j/(\alpha + F_J). \tag{3.13}$$

Note that to keep the problem consistent, we scale the demand by the same normalization factor:

$$d_t = d'_t/(\alpha + F_J). \tag{3.14}$$

Like what we adopt in Chapter 2, We use \$.07 per kWh as the utility price. The energy cost in a time slot is the product of the power consumption, the utility price and the slot size. Google reported [14] that the personnel cost for each repair is \$100 and the replacement cost is 10% of the server cost (\$2000). We assume the lifetime for a disk to be 60,000 turning-on-and-off cycles. Using this, we arrive at 0.5 cents for the wear-and-tear cost per switching-on-off cycle. Out of 0.5 cents, we assume turn on to be a higher cost than turning off; thus, we split this value to 0.3 cents and 0.2 cents for switching on and switching off, respectively. We also assume that the power cost for switching on and turning off is 0.02 cents and 0.005 cents, respectively, since switching on draws much more

Frequency Option	j	1	2	3	4	5	6	7	8
Frequency (GHz)	F_j	1.4	1.57	1.74	1.91	2.08	2.25	2.42	2.6
Normalized Capacity	v_j	.5385	.6038	.6692	.7346	.8	.8645	.9308	1
Power Consumption (watts)	P_j	60	63	66.8	71.3	76.8	83.2	90.7	100
Power Cost (cents)	c_j	$.42\tau$	$.441\tau$	$.4676\tau$	$.4991\tau$	$.5376\tau$	$.5824\tau$	$.6349\tau$	$.7\tau$

Table 5: CPU frequencies, capacities and costs

power than switching off in most cases. This analysis of the switching on/off cost is similar to that in Chapter 2 except that the consolidation cost is not considered. We summarize the cost parameters in Table 5. The normalized capacity shown in Table 5 is an instance based on (3.13) given $\alpha = 0$ that makes the relative difference of capacities for different frequencies the largest. Therefore, this is the case that yields the upper limit of the the benefits that DVFS can contribute. Since the maximum capacity per server is normalized to 1, the maximum cluster capacity is equal to the number of servers. We assume that the demand profile is forecasted and profiled every 5 minutes based on traces of demand on the CPU. Due to the diurnal behavior associated with human beings' working cycles, we chose the 8 hour work time as the planning horizon where the dynamically changing demand from one time slot to another is generated for our study. We consider 5 different degree of aggregation: $\alpha = 1$ (5 minutes), $\alpha = 3$ (15 minutes), $\alpha = 6$ (30 minutes), and $\alpha = 12$ (60 minutes). We statically aggregate the demand by maximum. Refer to (2.41) in Chapter 2 for more information about static aggregation by maximum.

To study the benefits of DVFS on the optimal solutions and provide some insights for the CPU frequent management, we evaluate three schemes: 1) The CPU does not have DVFS capability and always runs at maximum frequency; this scheme serves as the baseline study and will be denoted by "*Max*"; 2) The CPU can scale only at the minimum and the maximum frequencies — this is denoted by "*PingPong*"; 3) The CPU can scale in the full spectrum of 8 frequency options — this is denoted by "*Full*". To show the benefits of our approach, we also present two baseline cases: 1) All servers are always on and run at maximum frequency, that is, no optimization and cost management is employed — this is referred to as "*Baseline-I*", 2) Optimization is done independently at each time slot — this is referred to as "*Baseline-II*".

We ran the optimization model using CPLEX through its integer programming solver on an Intel(R) Pentium(R)IV 3.00GHz with 2GB memory. Through preliminary runs, we observed that the overall cost does not improve if we allow 2,000 branch and cut nodes in CPLEX; thus, in our study, we set the branch and cut nodes limit to 2,000.

3.2.2 Results and Discussions

Fig. 24 depicts the minimum cost in a 100-server cluster with 20% utilization where the demand distribution is assumed to be exponential. The upper three curves show the minimum cost when the switching on/off cost is considered while the lower three curves show the minimum cost when the turning on/off cost is *not* considered.

We start our discussion for the case that assumes no switching on/off cost, i.e., when $c_i^{s+} = c_i^{s-} = 0$. From Fig. 24, we can see that the degree of aggregation has significant impact on the optimum. By increasing the time slot size from 5 minutes to 60 minutes, the minimum cost increases by more than 200% for all three schemes. This is not surprising since aggregations result in over-provisioning. Please refer to Chapter 2 for detail analysis on how the degree of aggregation impacts optimal solutions.

In the case when the switching on/off cost is considered, the Full scheme results in the minimum cost but only slightly less than the PingPong scheme while both of these schemes outperform the Max scheme significantly. The intrinsic reason for performing optimization periodically, instead of continuously, is the overhead caused by optimization and resulting operations. This overhead of DVFS is much smaller than that of switching on/off.

Consider next the relative differences between the case with the switching on/off cost and the case without. The degree of aggregation has a similar impact on optimum as the case that assumes no switching on/off cost. But the increase in the slope of the minimum cost is much less than in the case without switching on/off cost. By taking this factor into consideration, lower degree of aggregation requires more frequent optimization operations, which cause more switching on/off operations. This cost cancels out part of the savings brought by less degree of aggregation. Essentially, our approach intends to seek an optimal operating point for the accurate provisioning to the demand profile to reduce the power consumption cost and to minimize switching on/off operations (to reduce the wear-and-tear cost).

Fig. 24 also includes the optimal solutions for Baseline-I and Baseline-II. Note that the switching on/off cost is considered in these two baseline cases. Recall that, in Baseline-I, the configuration is static where servers are always operated at the maximum frequency and only the turning on cost incurs at the beginning of the first time slot. We define the ratio of relative difference between Baseline-I and our approach, considering

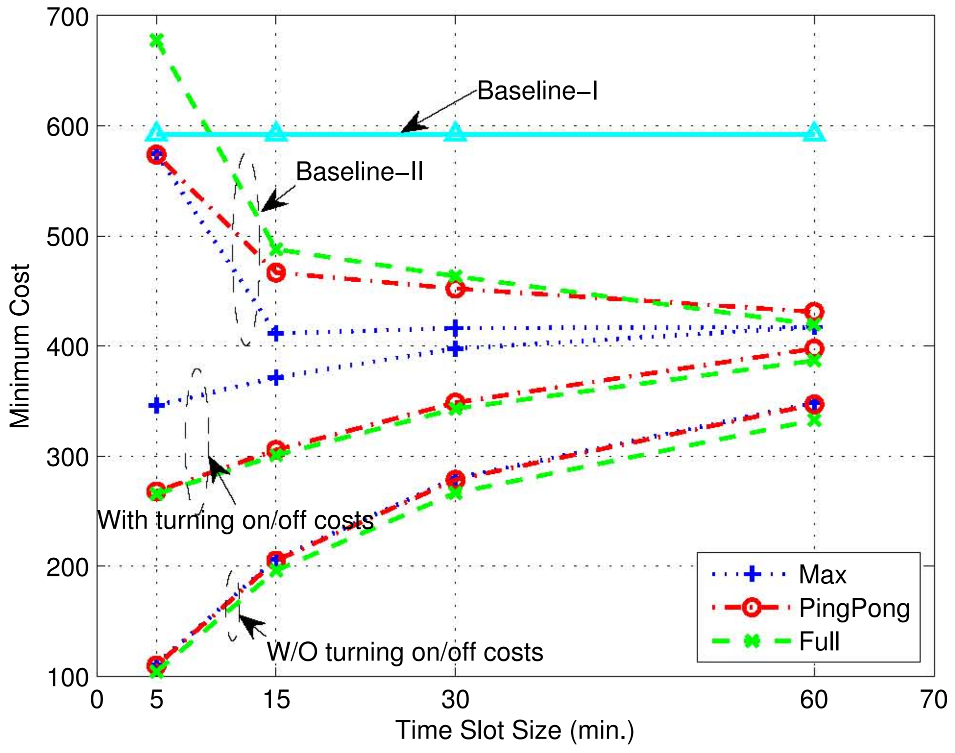


Figure 24: Minimum cost in a 100 server cluster

switching on/off cost, as the *relative improvement*. As shown in Fig. 25, the relative improvement is significant; clearly, the lower degree of aggregation of the time slot size, more than the frequency options, helps to reduce the minimum. In Baseline-II, optimization is done independently for each time slot. In the presence of the turning on/off cost, this local optimization can mislead the global optimum. As we can see from Fig. 24, in the case of 5 minute time slot size using Full scheme, the optimal solution obtained in Baseline-II is even worse than Baseline-I. When the time slot size is small, Baseline-II minimizes the server power consumption in each time slot independently and causes too many turning on/off operations that outweigh the savings in power consumption. The relative improvement of our approach compared with Baseline-II is shown in Fig. 25. As the degree of aggregation becomes higher, the improvement diminishes gradually. For large time granularity, the frequency of turning on/off operations is low; thus, the difference of optimizing for each time slot independently and multiple time slots considered together is small. This also explains that the optimum becomes lower as the time slot size granularity becomes larger, as shown in Fig. 24 as the switching on/off cost in Baseline-II cancels out the gain of the savings in the power cost by finer time slot size granularity. In essence, this also shows the value of the multi-time period framework.

The relative difference between the cost of switching on/off and the server power consumption cost also can have a different impact. To see this, we scale the switching on/off cost by factor R and study the impacts of relative difference in cost. Fig. 26 presents the optimal solutions for the cases whose switching on/off cost is scaled from 1 to 9 compared to the server power consumption cost. As we can see, the discrepancy of different granularity of time slot size becomes smaller along with the scaling factor increasing from 1 to 9. This trend is consistent for all 3 frequency schemes. Depending on the overhead, lower degree of aggregation does *not* always gives the optimal solution. If the switching on/off cost is negligible compared with the energy consumptions for all frequencies, this case can be regarded as equivalent to the case without considering the switching on/off cost; this would suggest that the lower degree of aggregation the best option. However, if the power consumption cost is negligible, we may avoid switching on and off operations and, instead, choose higher degree of aggregation.

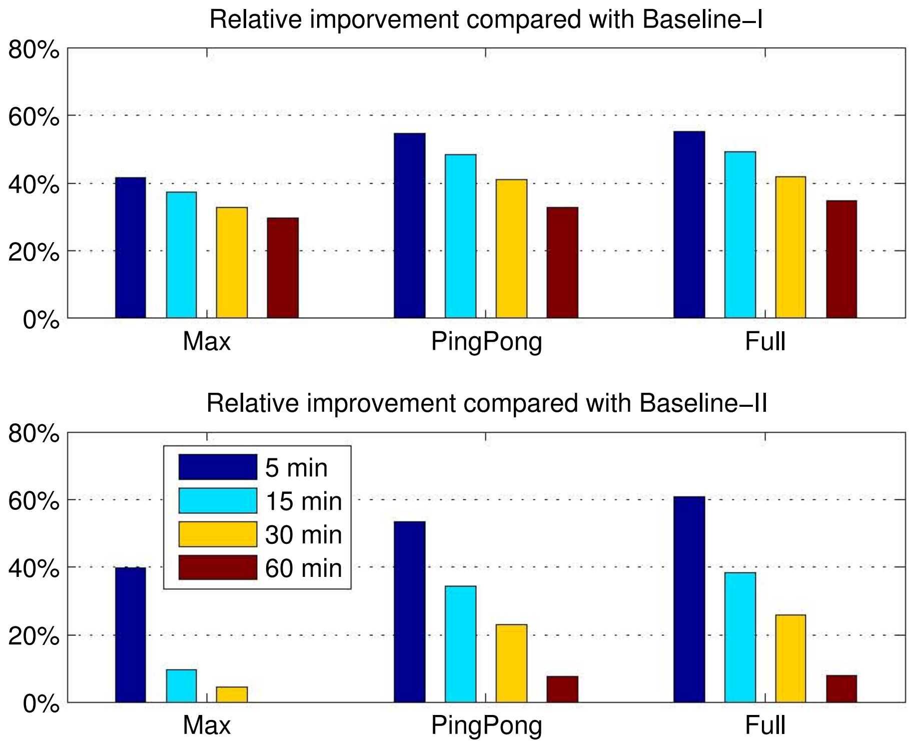


Figure 25: Relative improvement

3.3 Conclusion and Further work

Similar to Chapter 2, we use the wear-and-tear cost and power consumption to capture the server cost in data centers. The demand is considered to be dynamically changing over a time horizon. Data center operators desire to offer capacity just as needed to avoid over-provisioning; however, capacity re-assignment incurs the wear-and-tear cost. Leveraging two well-known methods, switching servers on/off and DVFS in a synchronous manner, we presented a multi-time period mathematical programming model

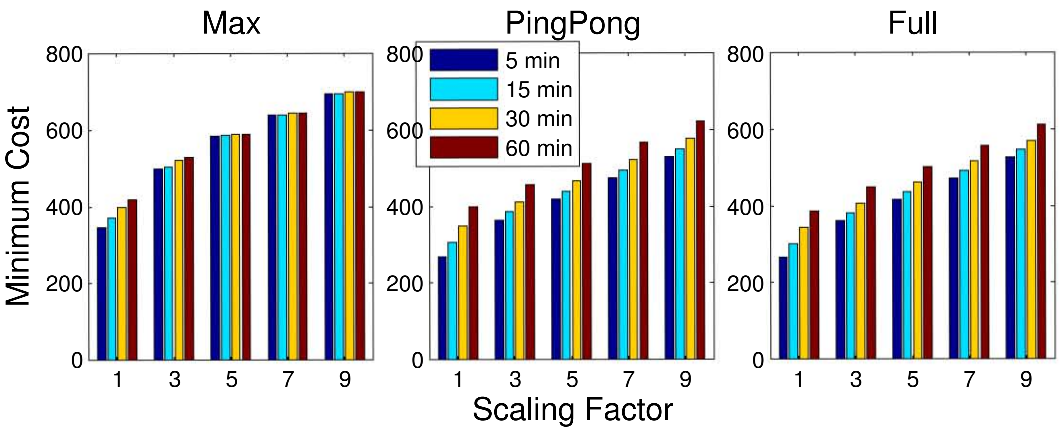


Figure 26: Scaling Factor Versus Minimum cost

to optimize the server. The evaluation study shows that our approach can significantly reduce the server operational cost compared with static capacity allocation (baseline-I) and when optimized locally (baseline-II); we found that the degree of aggregation has pronounced effects on the optimal solutions. To make our cost assumptions general, we study how the optimal time slot granularity is impacted by the relationship between two kinds of costs. Our aggregation methods introduced in Chapter 2 can be applied to this case with DVFS as to reduce the computational complexity. Actually aggregation is a more keen need in the case with DVFS since the computational time of case with DVFS is much more compared with the case without DVFS given other settings are the same.

The intrinsic reason for performing optimization periodically, instead of continuously, is the overhead caused by optimization and resulting operations. This overhead of DVFS is much smaller than that of turning on/off. In our current model, the two methods operated in a synchronous manner. It would be worthwhile to consider a model that is able to asynchronously use these two methods. We consider the problem for a known set of forecasted demand over a time horizon. A model that addresses uncertainty in demand over the time horizon would reflect a more realistic situation.

CHAPTER 4

A HIERARCHICAL MODEL TO EVALUATE QUALITY OF EXPERIENCE OF ONLINE SERVICES HOSTED BY DATA CENTERS

As Internet services utilize data centers to host their service via cloud computing model, they are challenged by evaluating the quality of experience and designing redirection strategies in this complicated environment. We propose a hierarchical modeling approach that can easily combine all components of this environment. Identifying interactions among the components is the key to construct such models. In this particular environment, we first construct four sub-models: an outbound bandwidth model, a data center availability model, a latency model and a data center (cloud computing) response time model. Then we use a redirection strategy graph to glue them together. We also introduce an all-in-one barometer to ease the evaluation. The numeric results show that our model serves as a very useful analytical tool for Internet service providers to evaluate data centers and design redirection strategies.

The main scope of the chapter is how to evaluate Quality of Experience (QoE) in a cloud computing environment where Internet service providers (ISePs) are the customers of the cloud computing service providers (CSPs) while CSPs use data centers as their infrastructure. As it turns out, it is extremely hard to build a monolithic model for this purpose; furthermore, such a model becomes intractable even for a small problem. The main contribution of our work is a unique and non-standard hierarchical model to analytically evaluate QoE that solves an otherwise intractable model. In addition, our model includes failure/repair as well as contention for resources (performability), response time distribution, and retry attempts. Our model provides flexibility to allow arbitrary changes or to add model components. Tackling each small piece (sub-model) of the model separately is much easier without ignoring the interactions among sub-models.

We present a model that helps ISePs analytically evaluate the QoE of their users and provides design insights, such as the selection of data center, and redirection strategies. The contributions of this chapter are as follows: (a) We identify various causes of user request failures in this environment; (b) We use a single metric with a secondary metric as supplement to evaluate QoE; and (c) We propose a hierarchical model for this environment.

The remainder of the chapter is organized as follows. We introduce background in Section 4.1. In Section 4.2, we present the details of the hierarchical analytical model. Numeric results are presented in Section 4.3. This chapter then ends with a conclusion and future work.

4.1 Background

4.1.1 Success Probability as an All-in-One Barometer

The performance of most Internet services have constraints on latency, such as VoIP, video streaming, podcast and search. For most Internet services, the outbound traffic to the data center (the service traffic) is dominant compared with inbound traffic (the request traffic) [91]. Thus, we concentrate mainly on the outbound bandwidth and SLAs usually bound on the latency from the data center to the users. Typically, when a user requests service, the ISeP serves from the data center then delivers to the user via Internet. If a QoE is within the bound of the SLA between users and the ISeP (that is, the user request is accepted and served within the promised latency in SLA), the request is deemed as a success. In this particular environment, there are four causes that can possibly make the request unsuccessful:

1. All virtual machines (VMs) assigned to the service fail in the data center of cloud computing provider.

2. If the service requires a constant bit rate (CBR) and the outbound bandwidth allocated to this service is used up, the data center will not be able to serve any more users. This is a classic circuit switch style loss model. The loss model affects the success probability by directly adding one more possibility of rejecting a request for service. If the service shares the bandwidth instead, a processor sharing (PS) model should be adopted. Therefore, latency perceived by users is a monotonically increasing function of the number of users sharing the link. The PS model affects the success probability by introducing more user experienced latency.

3. The buffer spaces allocated to the ISeP are used up.

4. A user request, though being accepted, cannot be served within promised latency. Based on the observation of a congested system, it is likely to become worse. In such a situation, a user should be pre-maturely directed to the next available data center with service replication when the waiting time exceeds a certain threshold.

We can categorize the causes of failure into two causes: one is "hard" rejection failure, the other one is latency-unsatisfactory ("soft") failure. The redirection strategy graph that will be discussed in Section 4.2.5 is the glue to combine these two kinds of failures into the all-in-one barometer: success probability. In the case that the success probabilities are the same or very close, the *average time to complete* is used to measure the QoE. Thus, we use the average time to complete as a secondary performance metric here.

4.1.2 Cloud Computing Availability

Cloud computing availability is comprehensively decided by the reliability of all the elements in the data center. In a data center, the causes of failure span software related errors (including Internet service software, cloud computing level management software), configuration faults, human and networking related errors, and hardware errors (in decreasing order in terms of percentage) [14]. On the other hand, the nature of large scale server clusters in a data center causes the mean time between failure (MTBF) to be much less than that of a single server. In an example given in [14], it was pointed out that a cluster of 10,000 servers sees approximately 100,000 times less MTBF than a single server does. A mega data center has on the order of tens of thousands or more severs and a micro data center has on the order of thousands of servers [39]. Thus, hardware failure is the norm rather than an exception [14]. Unavailability caused by these failures degrades QoE. Therefore, it is indispensable to use redundancy to improve availability of cloud services.

We are interested in the relationship between the number of redundant VMs and availability. To model availability, we first need to fit failure and repair characteristics to distributions in such large scale computer clusters and then apply modeling techniques. However, this requires considering complex combinations of heterogeneous components, complicated softwares, and onerous data mining. Recently, Vishwannath and Nagappan took the first step to characterize and understand cloud computing hardware faults that cause server failure [87]; rather, developing reliability/availaibility models for the whole data center and its servers remains an ongoing effort [87] and is outside the scope of our work. Instead, we will use a simplified availability model and a number of empirical assumptions to illustrate our idea. In this dissertaion, we consider the failure of service regardless of causes.

Another significant difference between a server cluster (cloud computing) and a single server is the ubiquitous dependency in server clusters. Because of this dependency, a server that is down can bring the entire cluster down. We will take this factor into consideration in our model. The ISePs usually use more VMs than necessary to prevent single-point-of-failure. Thus, we use redundancy to improve availability in this dissertation.

4.1.3 Redirection as an Enabling Mechanism

Serving users from a nearby data center can significantly reduce the latency. Therefore, the cloud provider desires to replicate the service across multiple geographically dispersed data centers. User requests can be directed to any one of these data centers as needed using parallel serving techniques e.g., DNS redirection. This idea has been commonly used in Content Distribution Networks (CDNs) [45]. OSPs also need to implement a function similar to the *distributor* in CDN. A user sends a DNS query to the authoritative DNS server of the ISeP, the ISeP replies with an IP address of a replicated service hosted by a data center, such as RTT between the user and data centers. We now consider one more step. What if a user is turned down by the given replicated service? The ISePs should render a list of locations (IP addresses) of all replicated services. This behavior will be further analyzed by the redirection strategy model in Section 4.2.5. Redirection is an enabling mechanism here. Mining user location patterns is the crux to decide where and the amount of resource in terms of the number of VMs. Agarwal *et. al.* revealed statistical information of client/user location information by studying the workload traces of two Internet services, Live Mesh and Live Message [9]. We assume that the service provider has done a similar study so that the data center location and required resource have been determined. Note that the overhead of redirection itself is not considered in this thesis since it is negligible compared with gains of using it, which will be discussed in 4.3.3.

4.2 The Hierarchical Model

We assume that an ISeP wants to migrate its service into cloud computing. The ISeP asks the CSP to replicate their services across multiple geographically dispersed data centers. Which user should request service from which data center is decided by the redirection strategies of the ISeP. User requests can be directed to any one of these data centers as needed using parallel serving techniques e.g., location based DNS redirection. The ISeP also wants to use more VMs than necessary to prevent the single-point-of-failure, that is, to improve availability. We present below the modeling approaches for each subsystem. At the end of this section, we summarize our approach and its limitations.

4.2.1 Cloud Computing Availability Model

We assume that data center j has M_j VMs. For brevity, we will discuss the availability model by dropping the subscript j for now, thus denoting the number of serving nodes as M. We assume that the time to failure of a VM is exponentially distributed with rate γ and the repair time of each serving node is also exponentially distributed with rate τ. Denote the ratio of failure rate and repair rate by $\varrho = \gamma/\tau$. Let c be the coverage probability that a VM going down does not bring all other VMs in the data center down. Let $\pi_k^{(a)}$ denote the steady state probability when this data center has k VMs assigned to the service are in operation. The Markov chain of this availability model and the equilibrium equations, which are omitted here, are described in [83]. The cloud computing availability, A is the probability that at least one VM in the data center is in service. The unavailability of the cloud computing, denoted by $\pi_0^{(a)}$ is the complementary part of A. That is, $\pi_0^{(a)} = 1 - A$.

4.2.2 Outbound Bandwidth Loss/PS Model

Let binary variable $\tilde{a}$ indicate either for loss model ($\tilde{a} = 1$) or PS model ($\tilde{a} = 0$) is used for outbound bandwidth. We show the loss model in detail then briefly discuss the PS model. We then show how these models can be plugged in the redirection strategy graph. The idea of the redirection strategy graph is originated from the user behavior graph [25, 88].

We denote the bandwidth of the bottleneck link by X. We assume that users demand the same CBR; that is, we consider the homogeneous case. For the homogeneous case where the demanded bandwidth of each request is fixed at Y, the effective bandwidth of the outbound bandwidth allocated to this service has the capacity of $H = \lfloor X/Y \rfloor$. We assume that the arrival process of the user requests is Poisson with rate λ and the time to transmit the content is exponentially distributed with rate μ. The system then becomes the classical Erlang loss model. Note that the arrival processes and content size distributions are dependent on the types of services; see, for example, [26]. The techniques to calculate the blocking probability of the loss model with different traffic characteristics, i.e., the heterogeneous case, are well known. We use the simplest model here for the purpose of illustration as part of the overall hierarchical model. Specifically, in the homogeneous case, the probability of outbound bandwidth being used is given by Erlang-B loss formula:

$$B = \pi_H^{(b)} = \frac{(\lambda/\mu)^H/H!}{\sum_{i=0}^{H}((\lambda/\mu)^i/i!)}. \tag{4.1}$$

The result can be extended to the case when user demanded CBR requests have different bandwidth requirements (the heterogeneous case) by using the Kaufman-Roberts formula [46, 77]. If the PS model is used, instead of blocking probability, we need to calculate the transmission latency that will be fed into the latency model.

4.2.3 Cloud Computing Response Time Model

We use the same assumptions that we made in Section 4.2.2. That is, the data size is exponentially distributed and the user request arrival process is Poisson with rate λ. We assume that the processing speed of all serving nodes is constant. Thus, the service time of a server site, that is the ratio of data size and processing speed, follows the exponential distribution. The justifications of these assumptions have been discussed in Section 4.2.2. The effective rate of arrivals, λ', to the service is given by:

$$\lambda' = (1 - \tilde{a}B)\lambda. \tag{4.2}$$

Let ν be the service rate of a data center. We model a server site as an $M/M/m/K$ system, where m is the number of VMs assigned to the service provider and K is the configured capacity for the service provider; this means that $b = K - m$ is the waiting room capacity (buffer size). Solving for the steady state probability,

$$\pi_k^{(c)} = \begin{cases} \rho^k \frac{m^k}{k!} \pi_0^{(c)}, & k = 0, \cdots, m-1, \\ \\ \rho^k \frac{m^m}{m!} \pi_0^{(c)}, & k = m, \cdots, K, \end{cases} \tag{4.3}$$

where $\rho = \lambda'/(m\nu)$ and

$$\pi_0^{(c)} = \left[\sum_{i=0}^{m-1} \rho^i \frac{m^i}{i!} + \sum_{i=m}^{K} \rho^i \frac{m^m}{m!} \right]^{-1}. \tag{4.4}$$

The probability of all configured capacity used up is:

$$C = \pi_K^{(c)}. \tag{4.5}$$

The cumulative distribution function (CDF) of the response time of $M/M/m/K$ system is given by [40]:

$$
\begin{aligned}
F^{(R)}(t) = \; & \sum_{i_1=0}^{m-1} \pi_{i_1}^{(c)} (1 - \exp(-\nu t)) \\
& + \sum_{i_1=m}^{K} \pi_{i_1}^{(c)} \Big\{ (\tfrac{m}{m-1})^{i_1-m+1} (1 - \exp(-\nu t)) \\
& - \tfrac{1}{m} \sum_{i_2=0}^{i_1-m} (\tfrac{m}{m-1})^{i_1-m-i_2+1} \cdot \\
& [1 - \exp(-m\nu t) \sum_{i_3=0}^{i_2} \tfrac{(m\nu t)^{i_3}}{(i_3)!}] \Big\} .
\end{aligned} \tag{4.6}
$$

4.2.4 Latency Model

The latency experienced by users consists of the data center (cloud) response time, the Internet delay, and the user computer processing time. The time of handing over a task from a failed VM to an available one, in case of VM failure, is taken care of by the CSP and considered as seamless. The service response time is the time used by a VM to process the requested data to be ready to transmit such as for packetization and scheduling. The network delay is the time consumed by the network transmission process. The user node processing time is considered too small and hence, negligible.

The network delay is comprised of transmission delay, queueing delay, propagation delay and nodal processing delay. We consider the delay occurring at the outbound link of the data center since the outbound bandwidth is part of the SLA between the ISeP and the CSP. The delay occurring at the rest of links in the path from the data center to users is dependent on the SLA between the CSP and ISePs, which is not considered in this thesis. The transmission delay of a particular link is the ratio of the data size and the bandwidth of this link. The queueing delay is a random variable that can be modeled as the response time of an $M/M/1$ system [83]. As mentioned in Section 4.2.2, there is no queueing delay if the loss model is applicable. If the PS model is used, the queueing delay of the PS model needs to be taken into account; however, the response time distribution of the PS model is a non-trivial exercise. The propagation delay is proportional to the path length. The routing processing delay is comparatively much smaller and can be ignored in a network where routers do not sniffer the packet content. Out of the non-negligible components, some are factors that remain constant such as the transmission delay and the propagation delay for a particular link speed; others are random such as, the service response time and the queueing delay. Let t_c and t_r be the constant factors and the random component, respectively. For the rest of the work, we use the loss model as illustration. That is, the service response time is the only random variable.

Waiting for a certain long time indicates a highly crowded system and users are more likely to wait for even longer time according to queueing observations. In order to optimize QoE, the ISeP needs to prematurely terminate such requests and redirect users to other available replications. This way, the users are more likely to get better QoE. So an ISeP can define a latency threshold, T_{th}, as the maximum time that the ISeP allows users waiting for the service from one data center. The probability that this bound is satisfied is $F^{(R)}(T_{th} - t_c)$, where $F^{(R)}$ is given by (4.6).

4.2.5 Redirection Strategy Model

We are now ready to present the success probability without considering redirection strategies. Let N data centers have replicated services. We will now put the subscript j back to refer to data center j, and use the subscript k to indicate VM k at a data center. Let q_j be the probability that a user request is accepted by data center j ($j \leq N$).

$F_{jk}^{(R)}(T_{th} - t_c)$ becomes the probability that the threshold is satisfied with k VMs assigned to the service at data center j.

We denote the four events that can cause the user requests to be denied by: 1) Φ_1, all assigned VMs fail. 2) Φ_2, the outbound bandwidth of the data center is used up. 3) Φ_3, the buffer assigned for the service in the data center is full. 4) Φ_4, user experienced latency exceeds the threshold. Therefore, the possibility of a user request being denied by the data center j is the summation of the following:

1. The probability that all assigned VMs fail:
   $P(\Phi_1) = \pi_{j0}^{(a)}$

2. The probability that at least a VM of data center j is available *but* the outbound bandwidth is used up:$P(\neg\Phi_1 \cdot \Phi_2) = (1 - \pi_{j0}^{(a)}) \cdot \tilde{a} \cdot B_j$.

3. The probability that at least a VM of data center j is available *and* the outbound bandwidth is sufficient, *but* the assigned buffer is full:$P(\neg\Phi_1 \cdot \neg\Phi_2 \cdot \Phi_3) = \sum_{i=1}^{M_j} \pi_{ji}^{(a)} \cdot (1 - \tilde{a} \cdot B_j) \cdot C_{ji}$. Note that we use subscript i in C_{ji} to indicate that C_{ji} is subject to the number of available VMs i.

4. The probability that at least a VM of data center j is available *and* the outbound bandwidth is sufficient *and* the assigned buffer is not full, *but* the user experienced latency exceeds the threshold: $P(\neg\Phi_1 \cdot \neg\Phi_2 \cdot \neg\Phi_3 \cdot \Phi_4) = \sum_{i=1}^{M_j} \pi_{ji}^{(a)} \cdot (1 - \tilde{a} \cdot B_j) \cdot (1 - C_{ji}) \cdot (1 - F_{ji}^{(R)}(T_{th} - \ell))$. Note that $F_{ji}^{(R)}$ is also subject to the number of available VMs at site j.

$$\mathbf{P} = \left[\begin{array}{ccccc|cc} 0 & 1-q_0 & 0 & \dots & 0 & q_0 & 0 \\ 0 & 0 & 1-q_1 & \dots & 0 & q_1 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ (1-q_{N-1})r & 0 & 0 & \dots & q_{N-1} & (1-q_{N-1})(1-r) & \\ \hline 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{array}\right] \tag{4.8}$$

The probability of an unsuccessful request to data center j, $1 - q_j$, is then equal to:

$$P(\Phi_1) + P(\neg\Phi_1 \cdot \Phi_2) + P(\neg\Phi_1 \cdot \neg\Phi_2 \cdot \Phi_3) + P(\neg\Phi_1 \cdot \neg\Phi_2 \cdot \neg\Phi_3 \cdot \Phi_4) \tag{4.7}$$

The ISeP can implement any redirection functionality. Therefore, the ISeP can utilize redirection strategies to increase success probability. In this chapter, we discuss three strategies for illustration: 1) Retry: retry probability, denoted by r, is an adjustable parameter to fine-tune this strategy. 2) Limited Try (LT) strategy, denoted by a binary variable $\tilde{t}$: $\tilde{t} = 1$ means a user is allowed to try N number of server sites, whilst $\tilde{t} = 0$ means a user can try any number of server sites. 3) Guided Hunting (GH) strategy, denoted by a binary variable $\tilde{h}$: $\tilde{h} = 1$: the ISeP provides the list of replicated service in the order of nearest to furthest ; $\tilde{h} = 0$: the ISeP provides the list of replicated service in a random order. Note that these are simple redirection strategies to illustrate how they can be incorporated in our hierarchical model.

The promised QoE decides how to apply these strategies. Such as the product of the number of maximum number of tries and the time threshold (without considering retry probability) should be bounded by the user tolerable latency. Optimizing the redirection strategies is out of the scope of this chapter. We use the redirection strategy graph to

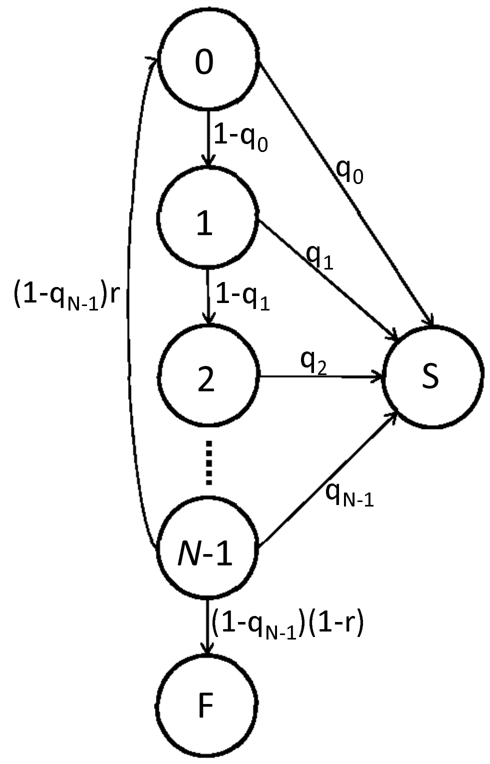


Figure 27: Redirection strategy graph

model these strategies. As shown in Fig. 27, an ISeP sends a user request to the data center (replicated service) 0 with probability q_0 of success and probability $1-q_0$ of failure. If the user is not successfully served, the ISeP then redirects the request to data center 1 with probability q_1 of success and so forth, until reaching the maximum number of data centers (if *LT* strategy is on). Then, the request can be redirected to the data centers that have been visited with the probability of $r \cdot (1-q_j)$ for $j = 0, 1, \cdots, N$, where $(1-q_j)$ is given by (4.7).

This redirection strategy graph is a discrete time Markov chain (DTMC) with two absorbing states, S and F. The transition probability matrix, $\mathbf{P}$, of the redirection strategy graph is given by (4.8), where q_j is given by (4.7). In a compact form, matrix, $\mathbf{P}$, in (4.8)

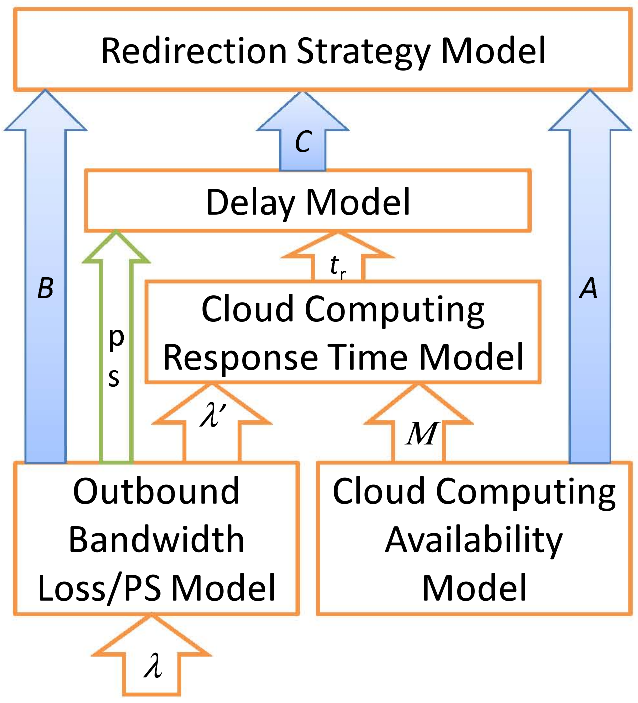

Redirection Strategy Model
C
Delay Model
t_r
B
A
ps
Cloud Computing Response Time Model
λ'
M
Outbound Bandwidth Loss/PS Model
Cloud Computing Availability Model
λ

Figure 28: Hierarchical model

can be partitioned so that:

$$\mathbf{P} = \left(\begin{array}{c|c} \mathbf{Q} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I} \end{array} \right), \tag{4.9}$$

where $\mathbf{Q}$ is an N by N matrix, $\mathbf{I}$ is a 2 by 2 identity matrix and $\mathbf{C}$ is an N by 2 rectangular matrix, where 2 indicates the number of absorbing states in the chain with $N+2$ states.

For an absorbing DTMC with 2 states, the expected number of visits V_k to state k before absorption (entering failure or success states) can be computed by solving the following system of linear equations [83]:

$$V_k = s'_k + \textstyle\sum_{i=0}^{N-1} q'_{ik} V_i, \quad k = 0, 1, \cdots, N-1, \tag{4.10}$$

where s'_k is the probability that a user tries to connect from node k and q'_{ik} is the transition probability from state i to state k. Where

$$q'_{ik} = \begin{cases} 1 - q_i, & k - i = 1. \\ (1 - q_{N-1})(1 - r), & i = N - 1, k = 0. \end{cases} \tag{4.11}$$

The average time for a request being accepted or denied is therefore given by:

$$\bar{t} = \sum_{i=0}^{N-1} V_i(\bar{t}_{ri} + t_{ci}), \tag{4.12}$$

where $\bar{t_{ri}}$ is the average response time of the service at data center i and t_{ci} is the rest latency component. Finally, let $\mathbf{A} = [a_{k_1,k_2}]$ be the matrix that a_{k_1,k_2} is the probability that a user request starting at data center k_1 ends at absorbing state k_2. It can be shown that $\mathbf{A} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{C}$ [58]. Then, the element (1,1) of $\mathbf{A}$ gives us the result for the user request failure probability.

4.2.6 Summary and Limitations

We have presented a hierarchical approach to model the environment that cloud computing hosts services for ISePs. Redirection strategies are the key to improve QoE. With this model, we can further calculate the success probability and average time to complete.

This model is built from the bottom up. As shown in Fig. 28, there are two models side to side at the bottom layer. The left side one is the outbound bandwidth loss model (*sub-model 1*). The right side one is the cloud computing availability model (*sub-model 2*). The cloud computing response time model is at the middle layer (*sub-model 3*). According to the arrival requests with rate λ' filtered by *sub-model 1* and the number of available serving nodes (M) which are subject to *sub-model 2*, *sub-model 3* calculates the probability that the response time of the cloud computing exceeds the threshold incorporated with the latency model (*sub-model 4*). At the top layer of this model, the redirection strategy graph (*sub-model*) which is constructed according to the redirection strategies used sums up all the possibilities that a request being denied (A, B, and C) from the sub-models under it.

As mentioned earlier, our assumptions of request content size, request arrival process, VM failure and repair distributions, may not be realistic. Instead, our focus here is showing how to construct a hierarchical model in this complicated environment to serve as an overall framework. For example, the PS model for the outbound bandwidth is much more common than the loss model for Internet service while we use the loss model here for illustration. The redirection strategies used here are also simplistic. Thus, there are a number of ways to include additional details or more complicated sub-models within the context of the overall model proposed here.

4.3 Numeric Results

In this section, we present a set of numeric results derived from our model to understand the different implications in this environment. This model is implemented in the SHARPE package [79].

4.3.1 Unavailability of Cloud Computing

We consider the unavailability of the cloud computing by considering both the coverage probability (c) and the number of VMs (M). Fig. 29 presents the optimum number of VMs in terms of unavailability for various values of c when ϱ is fixed at $1/5000$.

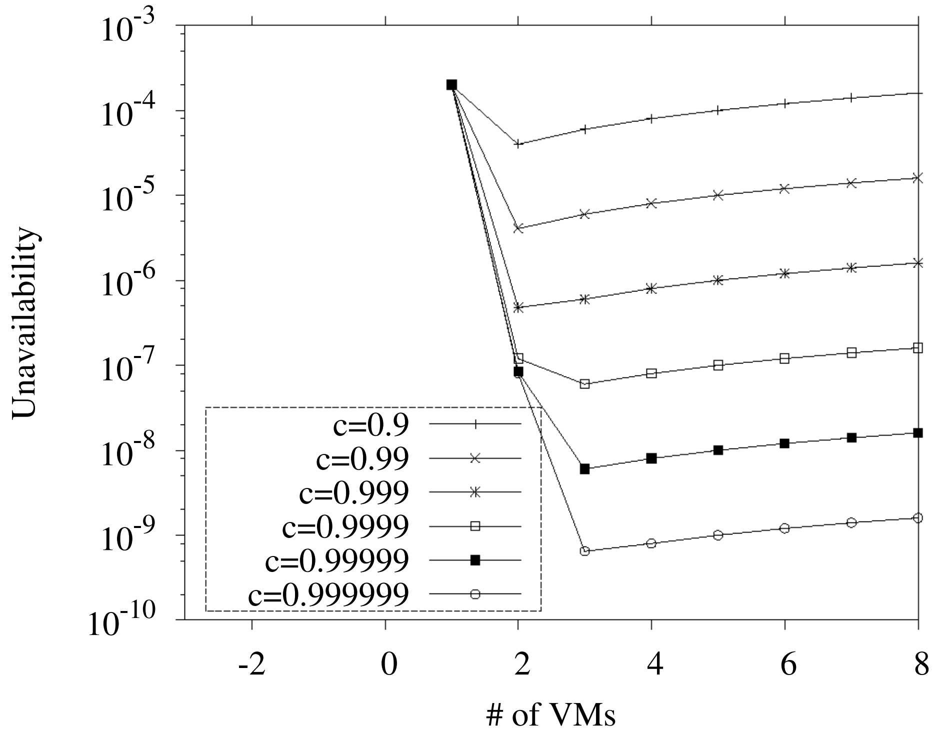


Figure 29: Unavailability ($\varrho = 2e - 4$)

The unavailability decreases at first, but increases when we continue to increase VMs. We can obtain the optimum number of VMs in terms of unavailability. The optimum is subject to coverage probability. Note that the optimum does not necessarily give the best experience for the users because more VMs always reduce the response time although the availability may decrease. As the coverage probability approaches 1, which means any VM down does not mean all VMs down, the optimal number of VMs converges to 3. We vary the ratio of failure rate and repair rate (ϱ) from $1/5000$ to $1/50000$. It is not a surprise that the unavailability increases as ϱ increases. We find that the basic pattern of unavailability remains the same, while the optimum is subject to ϱ.

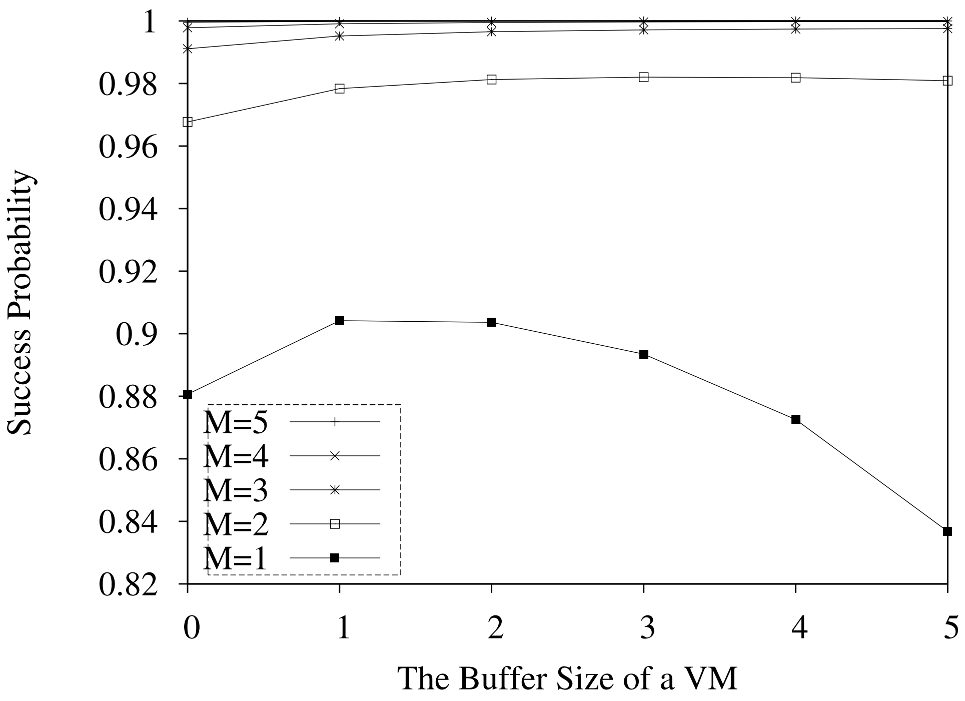


Figure 30: Success probability (c=0.9999)

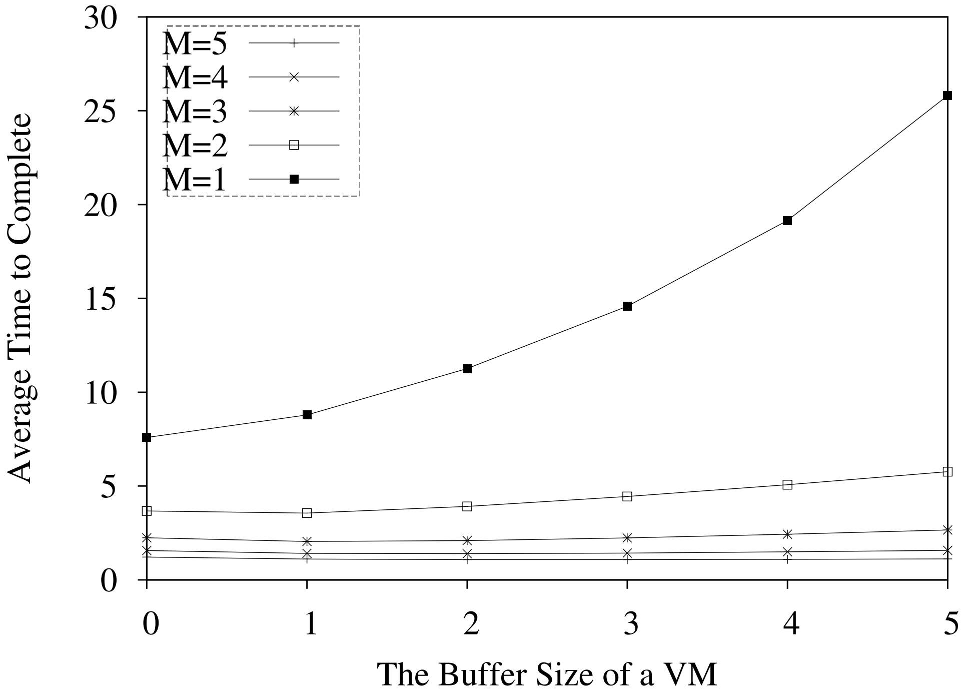


Figure 31: Average time (c=0.9999)

Table 6: Parameters of the hypothetical environment

sub-model1		sub-model2			sub-model3		sub-model4	sub-model5	
γ	τ	λ	μ	H	ν	N	t_c	r	T_{th}
2e-4	1	10	2	10	5	4	$0.1 + 0.1\Delta$	0.2	3

4.3.2 Data Center Configurations

Next, we study how data center hosting cloud computing configurations (the number of VMs, the buffer size, and the coverage probability) affect QoE. We use a hypothetical environment to illustrate how to use our model. For the same reason, we do not give units for the parameters. We assume that the latency of t_c is evenly incremented from the nearest data center (replicated service) to the furthest. Let Δ represent this order. The parameters we used are given in Table 6.

We find that the coverage probability within our considered range has no significant impact on the user perceived performance. So we only present the results given $c = 0.9999$. Fig. 30 and Fig. 31 depicts the QoE for a different number of VMs (M) and buffer size (b). It is clear that the larger number of VMs always improves both the success probability and the average time to complete. Because we only consider very small values of repair-to-failure rate ratio, the VMs are highly reliable. Note that we earlier observed that the optimum number of VMs in this scenario is the optimal for the cloud computing availability, but *not* for QoE. If we enlarge the buffer size, the success probability is improved at first but goes down after a turning point. The turning point increases as the number of VMs increases. As the buffer size is enlarged, acceptance probability of user requests become higher. But due to added congestion, the response time is more likely to exceed the threshold.

4.3.3 Redirection Strategies

To study how redirection strategies affect the QoE, we consider the same setup given in Section 4.3.2, except that the number of VMs and buffer size ($M = 3$, $b = 2$) are given while retry probability and the maximum number of allowed tries are variables. We consider normal and heavy traffic load cases. Fig. 32, Fig. 33, and Fig. 34 are for the normal traffic load case where $\lambda = 10$. Fig. 35, Fig. 36, and Fig. 37 are for the heavy traffic load case where $\lambda = 20$. Since OSPs need to guarantee more than 99% success probability in most cases, our study focuses on the range above 99%. We observed that increasing the maximum number of tries (N) always results in improving the success probability and the average time to complete and it becomes marginal after either the maximum number of tries or the retry probability reached a certain big number.

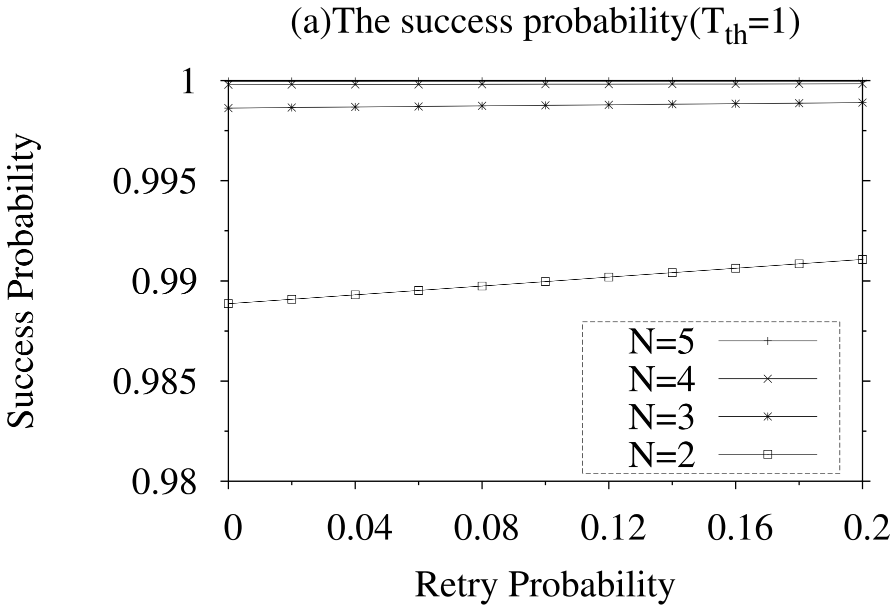
Note that for the average completion time, it is only true when the success probability is high. When the success probability is low, increasing N increases the average time to complete. Because when the success probability is low, more user requests are likely to be rejected. Increasing N does improve the success rate but prolongs the time to complete on average. It is reversed until N reaches a certain number, which causes the success probability to reach a certain level. Increasing retrial probability improves success probability while prolonging the average time to complete, which is more significant at a high retry probability and a low maximum number of tries. Setting T_{th} higher (i.e., at $T_{th} = 3$ and $T_{th} = 5$) yields a better performance than $T_{th} = 1$; the performance difference is more significant at a low retry probability and low maximum number of tries for the normal traffic load. In addition, we observed that the improvement of performance from $T_{th} = 3$ to $T_{th} = 5$ is not significant in both scenarios. We noticed that the success probability converges to 1 as the retry probability approaches 1 for the high traffic load case and the price paid for that is the significant increase of the average time to complete. On the other hand, for the normal traffic load case, the retry probability need not be high to achieve a high level of success probability. By comparing these figures, we need to choose from increasing the maximum number of trials (N), increasing retry probability (r) or increasing T_{th} to guarantee the success probability when the traffic load increases.

We use the following example to show the importance of being aware of the location of users. We still focus on the range that the success probabilities are above 99%. We follow the set up in Section 4.3.2 and we set the number of VMs and buffer size ($M = 4$, $b = 3$). We compare the redirection strategy that has guided hunting with that which has not. Fig. 38 shows that the success probabilities with guided hunting and without guided hunting are the same while the guided hunting mode shows its advantage in the average completion time. Increasing Δ increases the average completion time and does not affect the success probability. Increasing the arrival rate degrades both aspects of the user perceived performance. While these observations are not surprising, we show how our model can quantify the benefits using a better strategy which of course costs something else, such as, collecting and updating the $priori$ information in the guided hunting strategy.

4.4 Conclusion and Future Work

We introduced a hierarchical model to analytically evaluate QoE for ISePs which use data centers to host their services by cloud computing model. The nature of hierarchy is suitable for rich interactions in this environment. Another advantage of this model is any component or identified component is easy to change or added. The results of modeling can help data center operator efficiently quantify the resources (*i.e.*, higher reliability, more outbound bandwidth) need to acquire higher performance and implementing more complicated redirection strategies (*i.e.*, maximum number of tries, direction hunting,) while maintaining QoE.

There are several directions for future research efforts. The outbound bandwidth PS model can be used instead of the loss model. Secondly, we use the redirection strategy graph to quantify QoE. Analytically, optimizing the redirection strategies is another direction to pursue. Evaluation of the hierarchical model with non-Poisson arrival and non-exponential service time distribution would allow further understanding of the impact of distribution on the accurate determination of QoE. The availability model of data center is very rough. In the future, we want to develop more detailed availability model. Finally, in the long run, we wish to collect traces such as user arrivals, user locations, service time, and work load from ISePs and running environment set up from CSPs to identify the realistic distributions parameters in order to apply our approach on the same trace and compare with the logged QoE.



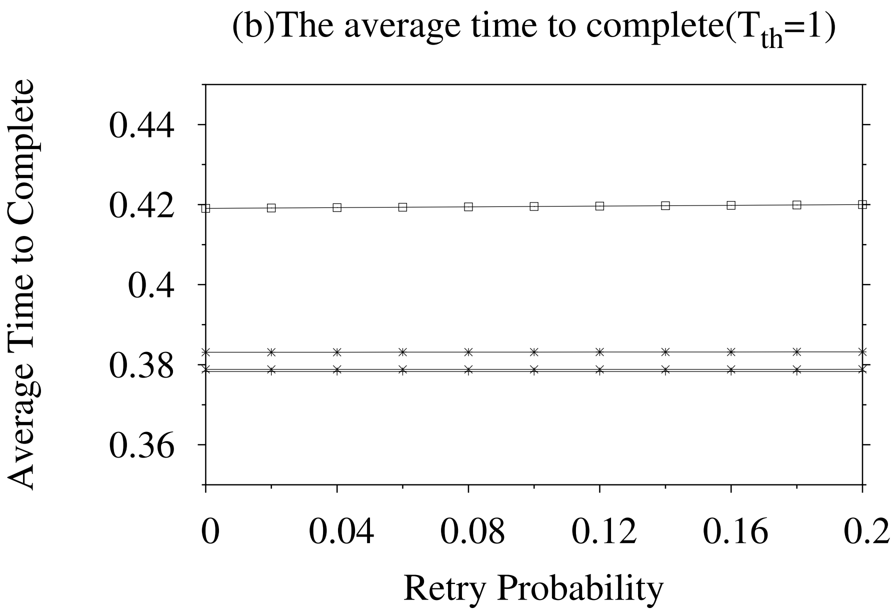
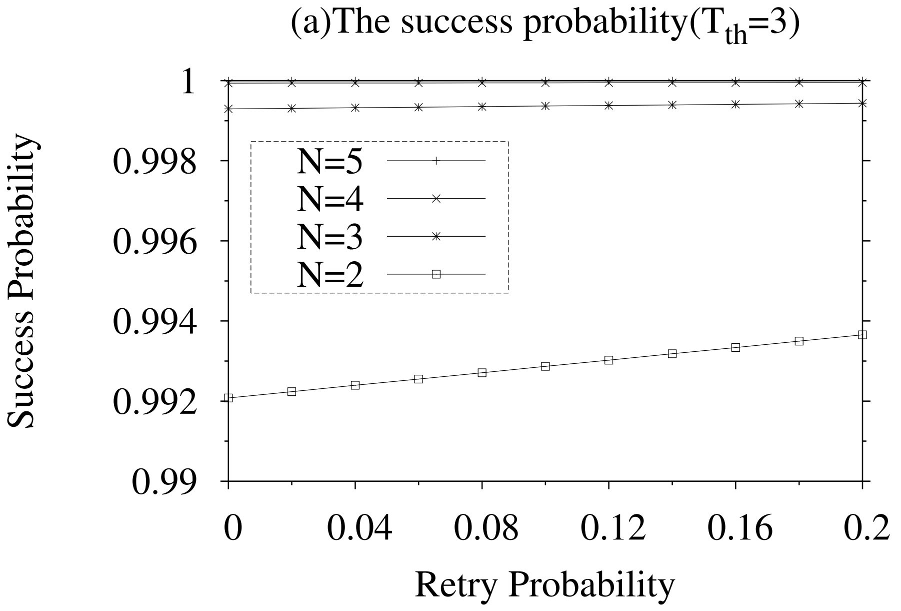


Figure 32: QoE ($T_{th} = 1$, $\lambda = 10$)



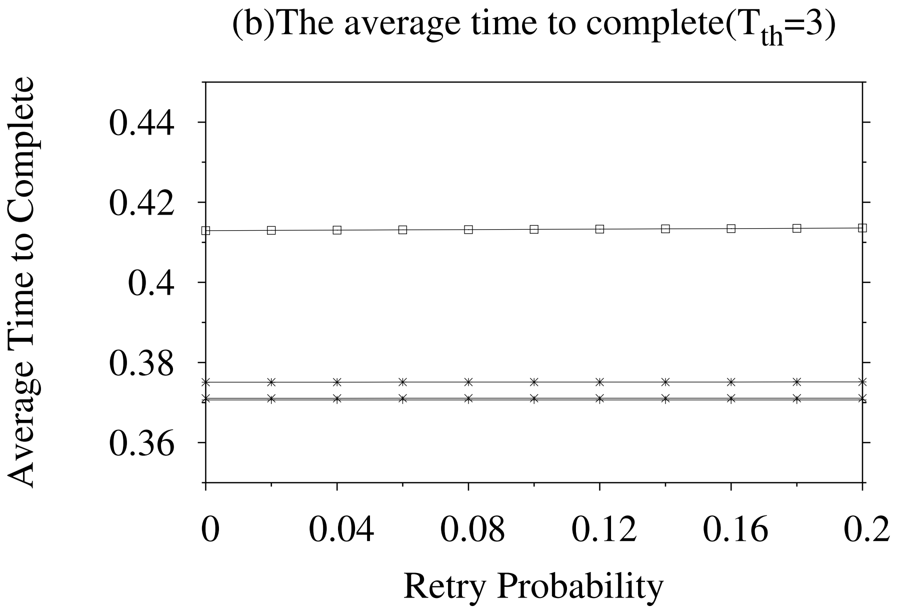
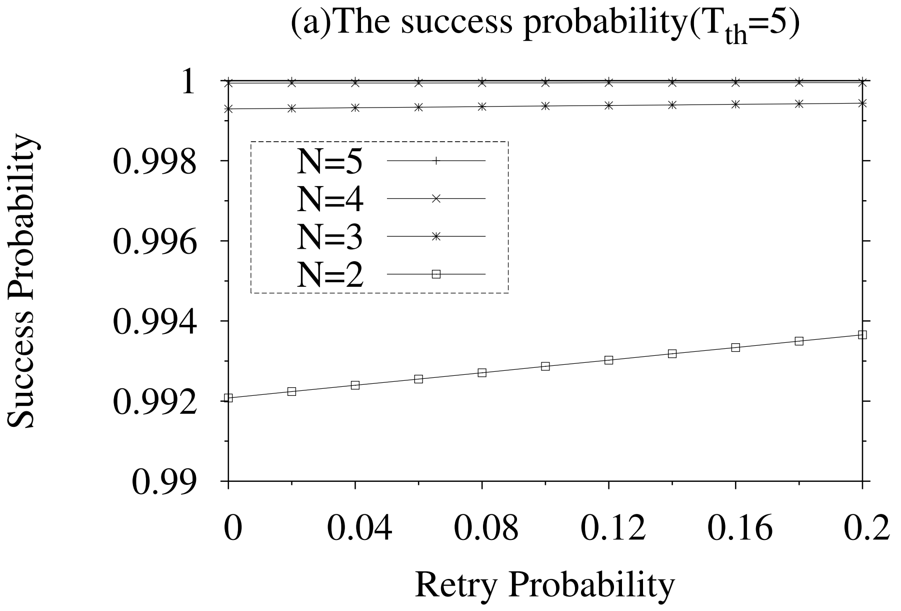


Figure 33: QoE ($T_{th} = 3$, $\lambda = 10$)



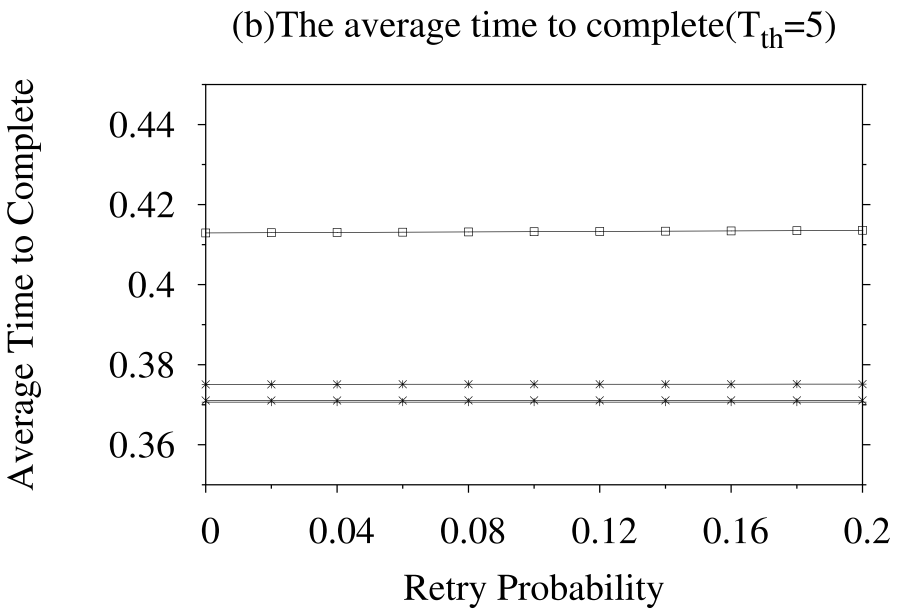


Figure 34: QoE ($T_{th} = 5$, $\lambda = 10$)

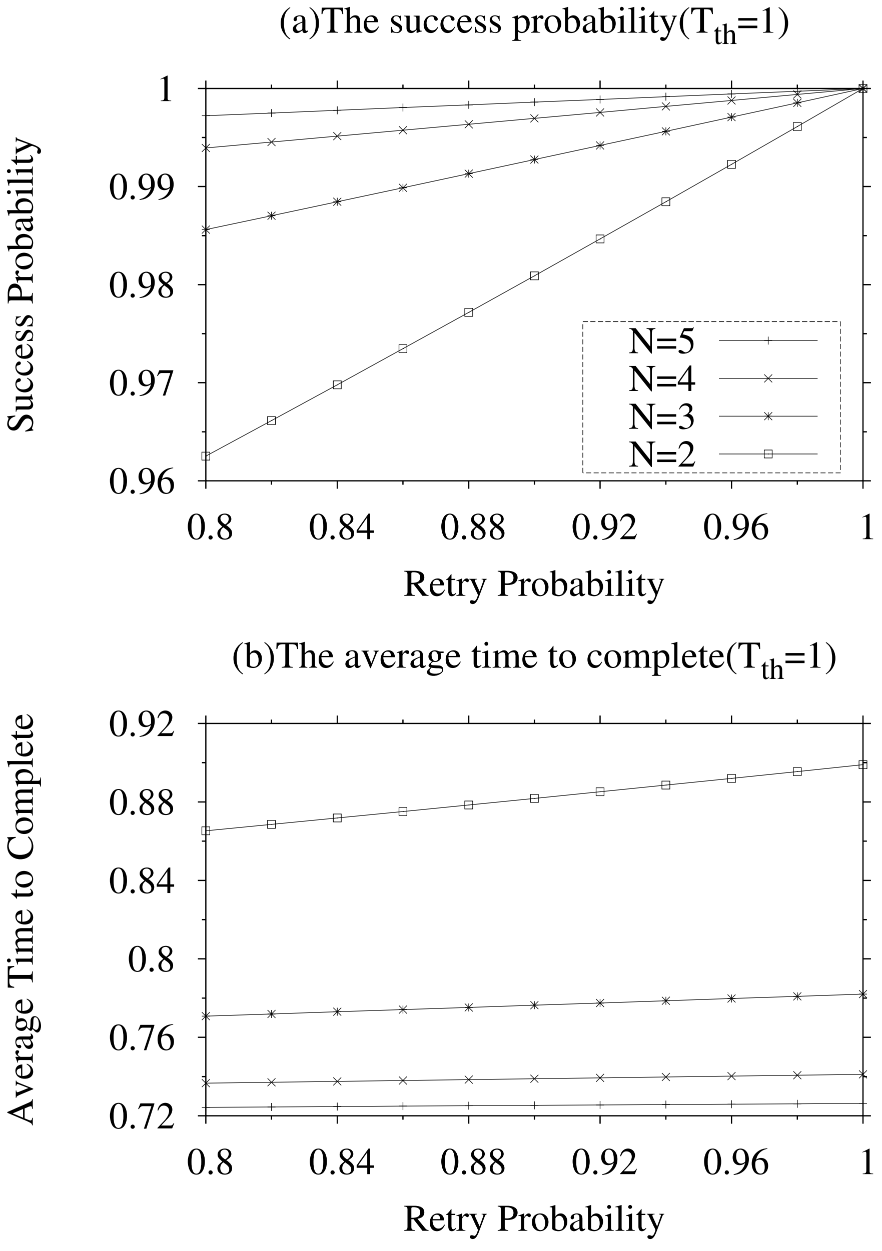
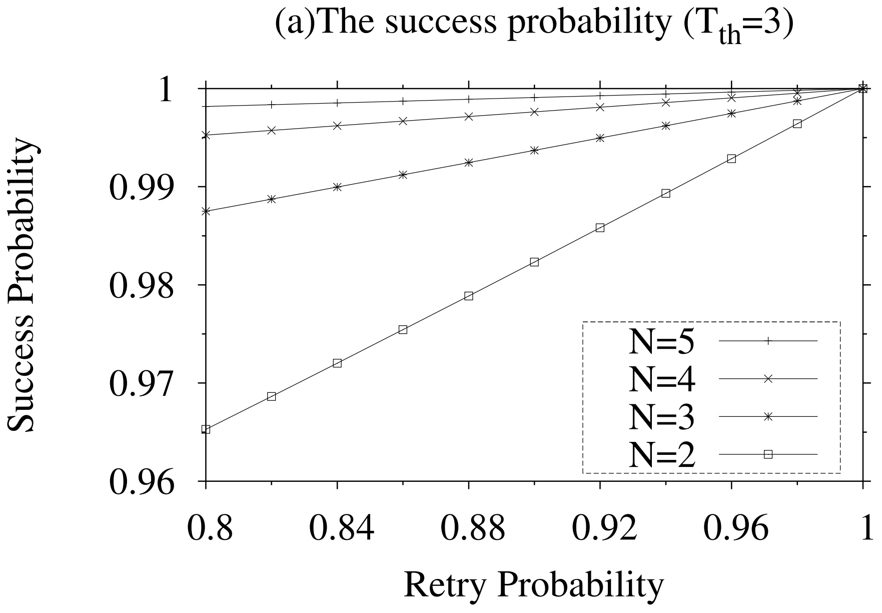


Figure 35: QoE ($T_{th} = 1$, $\lambda = 20$)



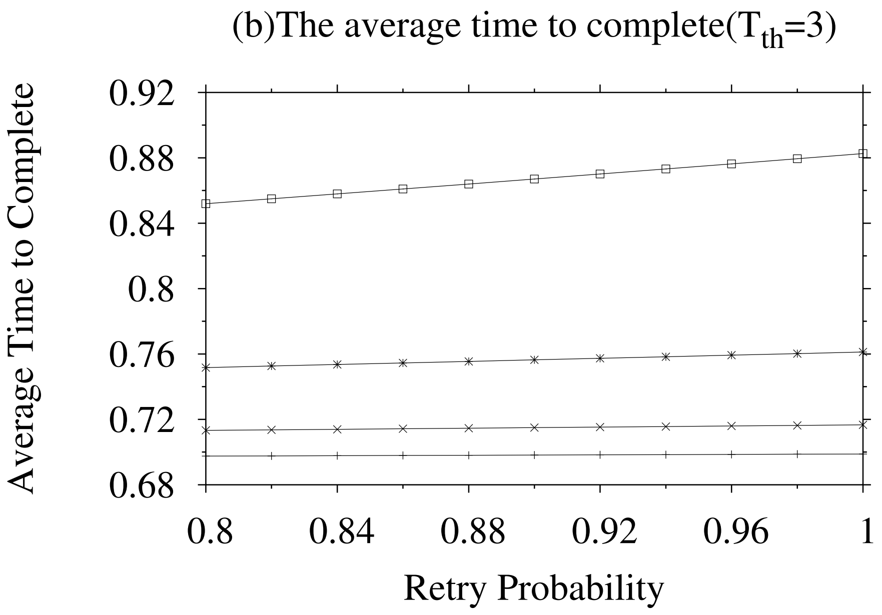


Figure 36: QoE ($T_{th} = 3$, $\lambda = 20$)

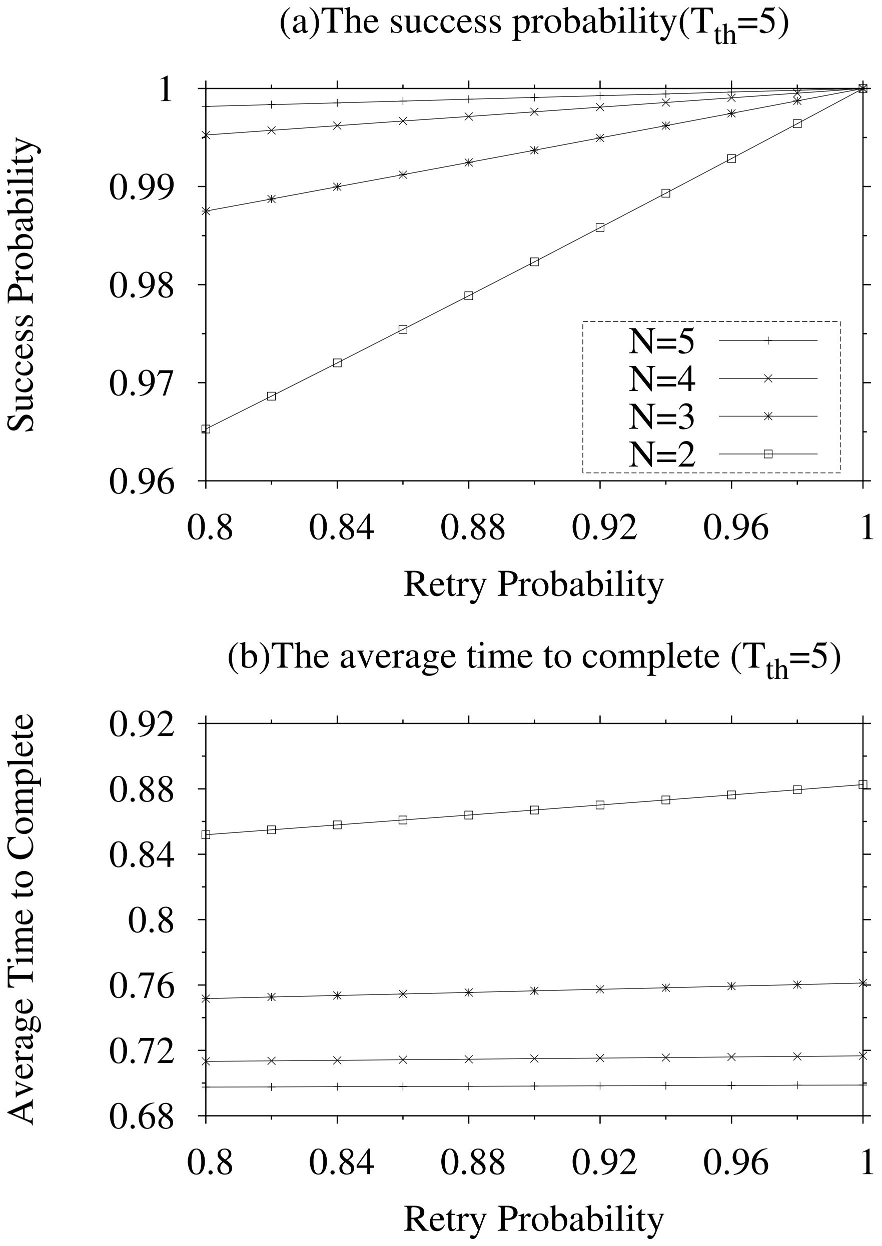


Figure 37: QoE ($T_{th} = 5$, $\lambda = 20$)

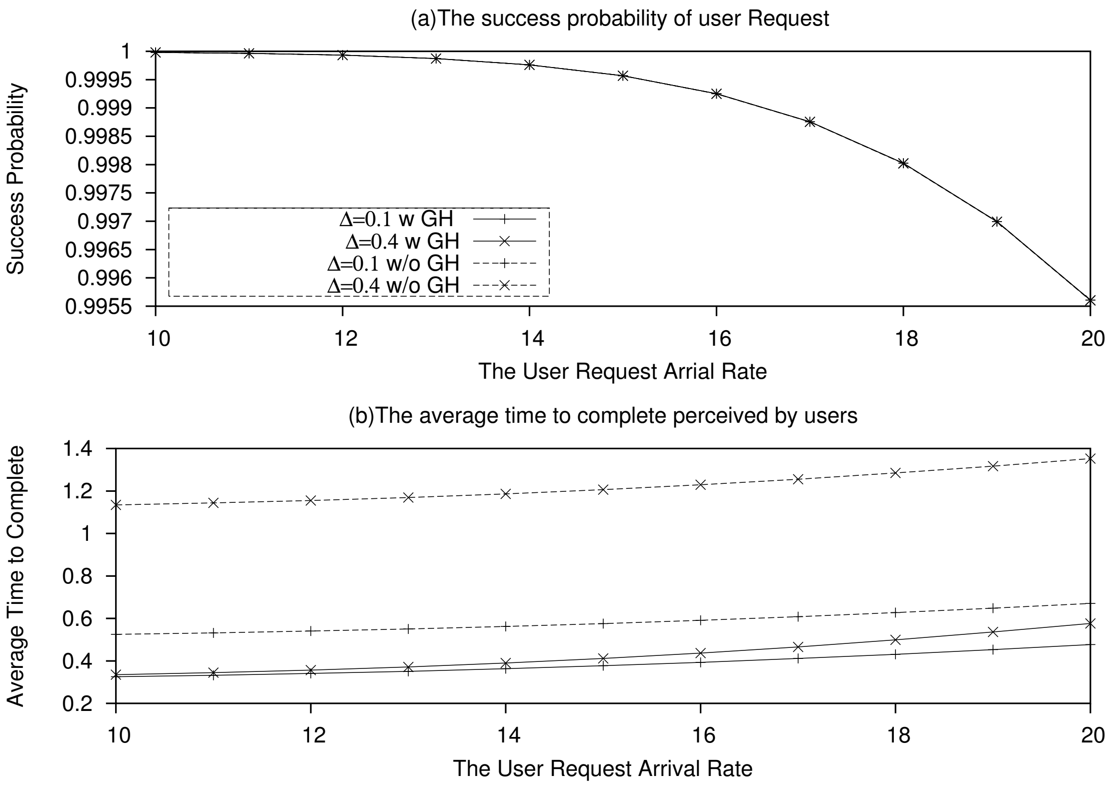


Figure 38: QoE w and w/o guided hunting against user request arrival rate

CHAPTER 5

BALANCING REQUEST DENIAL PROBABILITY AND LATENCY IN AN AGENT-BASED VPN ARCHITECTURE

Virtual Private Network (VPN) services have seen significant growth over the years. There are many forms of VPN services. A common one is the remote-access VPN in which an employee of an organization can use the VPN service to access the intranet of the organization from an off-site location through the public Internet by establishing a VPN tunnel. While such VPN services are often deployed with each organization installing a VPN server ("gateway") on its premise, this model does not serve well for organizations that do not often have the IT expertise to install and maintain VPN servers on a day-to-day basis. Many small organizations would want to have a hassle-free VPN service for their users if it can be provided and maintained by a third party and the service is cost-effective. Such a service is also appealing if it is flexible and adaptable to changing organizational and technological requirements. An *Agent-Based VPN Architecture* (ABVA) is a third-party approach that fits into a need. In this approach, one or more VPN agents (or brokers) are located outside the organizations that serve as *Rendezvous Points* (RP), where users and their associated organizations meet; the Rendezvous Point (RP) service is provided by a third party (ABVA provider) for multiple organizations while maintaining separate virtual tunnels for each organization. The agent-based remote access VPN service has been available from third-party providers for several years [67]; this service is sometimes also known as the *hosted VPN* service.

From the perspective of an ABVA provider, there are challenging design and management problems since customers want a certain quality-of-service as part of service-level agreements (SLAs), for example, a user's connectivity latency, bandwidth guarantee, and acceptable rate of successful connectivity. In order to address this, a question arisesas to how many different RPs are attempted if the user cannot connect to the first RP attempted? When the arrival of users seeking this service is stochastic, *and* the bandwidth guarantee during the session and the tolerable acceptance rate of service are the primary factors, then this problem looks similar to a loss system. On the other hand, the geographic diversity of RP locations as well as the latency perception of users make this problem fundamentally different than a pure loss system. It is further complicated by the fact that a user might travel from one city to another around the globe and logon; thus, the proximity of an RP for any user is not easily known. For instance, if a user is connected to a far-off RP, then it might be able to provide the bandwidth guarantee while the latency perceived is beyond the SLA-specified value. On the other hand, a user might locate a nearby RP, but this RP might be already heavily loaded and unable to accommodate this user, thus denying this user. Therefore, a service management problem for an ABVA provider is a trade-off between request denial probability and latency, in order to connect to an optimal RP location. A software client can be loaded to a user's device (such as a laptop computer) that is configurable to contain a proximity of RPs from the user's location; distributed protocols can be used to push additional information that can be cached to the devices based on previous logon location.

Our work focuses on balancing user request denial probability and latency in a capacitated ABVA environment (the server location is already given while the user has flexibility to choose which one to connect to). To the best of our knowledge, how to effectively provide user connectivity and minimize the latency in ABVA over IP-based networks has not received attention in the literature (see Section 1.7 for related work). In an earlier work, the overall latency minimization problem was formulated as an assignment problem assuming user arrivals and service time are *static* while providing bandwidth guarantees when the location and the number of the users are known and fixed [57]. Using this formulation, the lower bound on the overall system latency was obtained and it was observed that bandwidth guarantee was the dominant factor in a highly loaded environment, while connection to the closest RP was important in a lightly to moderately loaded environment. [68] relaxed the static user arrival process to Poisson assuming a finite population model while a limited study was conducted. This work extends [68] by presenting a comprehensive study that considers a number of different arrival processes under both finite and infinite population models for a number of different topologies.

The remainder of this chapter is organized as follows: in Section 5.1, the system model and a number of schemes are presented. In Section 5.2, numerical results are presented and studied. Concluding remarks along with future work and limitations are presented in Section 5.3.

5.1 System Models and Schemes

5.1.1 System Models

Consider an ABVA provider with N RPs and an overall capacity of c. Assume that each RP has the capacity $c_n, n = 1, 2, \cdots, N$, for providing bandwidth to users; for simplicity, we assume that all servers have the same capacity, i.e., $c_n = c/N$. We also assume that all users belong to a single organization. Note that in the multiple organization case, the capacity at RPs can be partitioned for each organization according to their SLAs; thus, each organization can be managed independently, which is preferable, since each can have separate SLAs with the ABVA provider. In other words, without loss of generality, we consider a single organization for the study of ABVA. A user connection request is accepted if the VPN server at an RP attempted has the capacity to accommodate this connection at the instant of the request arrival; otherwise, it is a candidate for rejection. We do not rule out the possibility that a user may try one or more VPN servers at other RPs if she cannot get through the first one. This will be further discussed later on.

Without loss of generality, we assume that the duration of a session is exponentially distributed with unit mean, i.e., $\mu = 1$. We consider two user population models:

1. The population of users is large enough to be treated as infinite. For brevity, this will be referred to as the infinite model.

2. The population of users is not large enough to be considered as infinite. For brevity, this will be referred to as the finite model where the RPs support u users.

Note that in the infinite model, we define the user inter-arrival time as the time interval between two continuous arrivals of arbitrary users, while for the finite model, we define this as the time interval between two continuous arrivals of the same user. Let λ be the arrival rate in the infinite population model and γ be the arrival rate of each user (which is referred to as arrival rate for brevity) when the user is idle in the finite model. The arrival rate in the infinite model is the limit case of the finite model:

$$\lim_{u \to \infty\, \gamma \to 0} u \cdot \gamma = \lambda. \tag{5.1}$$

In general, each user's arrival request may have different data rate requirements. For simplicity, we assume all users have a unit data rate requirement, i.e., we consider the homogeneous case here.

5.1.2 User Connection Schemes

We assume that the user arrival process is Poisson. The Poisson arrival assumption is reasonable because the connection request is initiated by human beings, which is similar to call arrivals in the telephone network. However, we cannot rule out the possibility of traffic showing non-Poisson behavior. Thus, we also wish to see how the results are impacted when arrival processes are non-Poissonian through simulation; this will be elaborated further in Section 5.2.

5.1.2.1 The Centralized-All Scheme–The Baseline

An agent-based VPN provider can deploy all agent servers at a monolithically single RP. This case will be referred to as the *centralized-all scheme* (CAS). The latency is static and there is no possibility to optimize latency for any user. The request denial probability in CAS can be analytically calculated. In the infinite model with Poisson arrival, the request denial probability for the CAS, $B_{\mathcal{M}}$, is given by the Erlang-B loss formula:

$$B_{\mathcal{M}}(\lambda, c) = \frac{\frac{\lambda^c}{c!}}{\sum_{\ell=0}^{c} \frac{\lambda^\ell}{\ell!}}. \tag{5.2}$$

In the finite model, the request denial probability, denoted by $\overline{B}_{\mathcal{M}}$, is given by Engset's formula: [43,47]

$$\overline{B}_{\mathcal{M}}(\gamma, u, c) = \frac{\binom{u-1}{c} \cdot \gamma^c}{\sum_{\ell=0}^{c} \binom{u-1}{\ell} \cdot \gamma^\ell}. \tag{5.3}$$

5.1.2.2 The Clustering Scheme

An agent-based VPN provider can deploy VPN servers over multiple RPs that are geographically distributed. Since the client software associated with a user's computer is programmable, a distributed caching concept can be used. For instance, if a subset of users is likely to stay within a region (referred to as a cluster), they can then be assigned to the RP in that region. For a user, the RP assigned to her is referred as her home RP and all others are referred as guest RPs. Thus, we can consider a cluster-based approach in which each user is associated with a home RP and users are only allowed to access their home RPs. By this way, the latency is minimized because providers are aware of the proximity information of users and locate VPN servers on the premise of clusters. Of course, the cluster assignment is subject to the change of user locations. Planning VPN server locations deliberately can achieve uniform distribution of all users over the clusters.

In the finite model, each cluster has u/N users. That is, for the cluster based approach, the population and VPN server capacity is reduced by factor N and becomes

u/N and c/N, respectively. We use superscript "$^{(1)}$" to denote this cluster-based scheme. Therefore, the request denial probability for the Poisson case in the finite model becomes

$$\overline{B}_{\mathcal{M}}^{(1)}(\gamma, u, c, N) = \overline{B}_{\mathcal{M}}(\gamma, \frac{u}{N}, \frac{c}{N}) = \frac{\binom{\frac{u}{N}-1}{\frac{c}{N}} \cdot \gamma^c}{\sum_{\ell=0}^{\frac{c}{N}} \binom{\frac{u}{N}-1}{\ell} \cdot \gamma^\ell}. \tag{5.4}$$

In the infinite model, the population of each cluster is still infinite because the number of clusters is finite. If we create a cluster for each RP, the user arrival rate for each cluster becomes λ/N. The request denial probability of the Poisson case in the infinite model is given by

$$B_{\mathcal{M}}^{(1)}(\lambda, c, N) = B_{\mathcal{M}}(\frac{\lambda}{N}, \frac{c}{N}) = \frac{\frac{\frac{\lambda}{N}^{\frac{c}{N}}}{\frac{c}{N}!}}{\sum_{\ell=0}^{\frac{c}{N}} \frac{\frac{\lambda}{N}^{\ell}}{\ell!}}. \tag{5.5}$$

Due to multiplexing gain, the CAS has a lower request denial probability than the cluster-based scheme does, i.e.,

$$B_{\mathcal{M}}(\lambda, c) < B_{\mathcal{M}}^{(1)}(\lambda, c, N); \tag{5.6}$$

note that (5.6) also applies to the infinite model. For CAS and the aforementioned cluster-based scheme, all requests only visit one RP, thus, the mean step for these schemes is always one. As we have mentioned before, the average latency of CAS is clearly larger than that of the cluster-based scheme.

5.1.2.3 The Hunting Scheme

Expanding on the clustering notion, and in order to reduce request denial probability, a hunting feature can be added. This means that if a user's request is denied by the home RP, then it is automatically forwarded to a guest RP; if denied again, the user's request is forwarded to another guest RP until a maximum allowed number $k, 1 \leq k \leq N$

of RPs are attempted. In order to do that, the user's client software would need to store the location information of possible guest RPs; out of these possibilities, one RP is randomly chosen and attempted. Note that this hunting feature introduces additional delay during set up. However, this delay can be neglected and is usually very small. For brevity, this *clustering with random hunting* scheme for attempting up to k RPs will be denoted as CRH-k. A variation of CRH-k is hunting for the next RPs based on the distance information. Once a request is denied by its home RP, the selection of the second RP can be the next nearest server. This distance (locality) information can be provided to the user either by the home RP at the time of request (as part of the denial response message), or *a priori* information through a system configuration update message. Obtaining this information certainly burdens the system. Ideally, the cost of obtaining this information should be counted as another contributor besides request denial probability and latency to the total cost. However, this cost is not included in our work. The *clustering with directional hunting* scheme is called CDH-k for short. We can see that the original cluster-based scheme is actually CD(R)H-1. Increasing the maximum number of steps can always decrease the request blocking probability and increase the mean step and latency for both CDH and CRH. Thus, k, the maximum number of allowed RPs serves as the functions to balance the request denial probability and the latency. Since the CDH-k for $k > 1$ scheme takes the location information into account when choosing the second server, CDH-k results in less average latency than CRH-k does while CDH-k requires a locality knowledge of servers.

5.1.3 Comments on the Analytical Formulae and Approximations

In the context of clustering, we assume that the arrival process of each cluster is independent and identically distributed. Only the first k elements (RPs) in the hunting sequence are active in the CRH-k or CDH-k schemes. A special case is that the users and all active RPs can form an *autonomous system* (no traffic from or to the outside of this system). It means that all the users only attempt the RPs within the autonomous system, and all the RPs are attempted at least once by one user and are attempted only by the users within the autonomous system. Thus, the users take the superposition of the capacity of all the RPs as the capacity of this autonomous system. On the other hand, the RPs take the superposition of all the user arrivals within the autonomous system as the arrivals of the system. We can see that if the CDH or CRH scheme can form an autonomous system when it comes to request denial probability, this system is equivalent to an aggregated system with the sum of all RPs in the autonomous system as its capacity and the sum of all user arrivals as its arrivals. For example, when a CDH/CRH-k scheme forms an autonomous system, the arrival rate becomes $k\lambda/N$ and the capacity becomes kc/N in the infinite model, while the arrival rate for each user is still γ and the capacity becomes kc/N in the finite model.

Theorem 5.1.1. *If a CDH-k or CRH-k scheme forms an autonomous system and the arrival process is Poisson, the request denial probability in the infinite population model is given by*

$$B^{(k)}_{\mathcal{M}}(\lambda, c, N) = B^{(\underline{k})}_{\mathcal{M}}(\lambda, c, N) = B_{\mathcal{M}}(\frac{k\lambda}{N}, \frac{kc}{N}), \tag{5.7}$$

and in the finite population model, it is given by

$$\overline{B}_{\mathcal{M}}^{(k)}(\gamma, u, c, N) = \overline{B}_{\mathcal{M}}^{(k)}(\gamma, u, c, N) = \overline{B}_{\mathcal{M}}(\gamma, \frac{ku}{N}, \frac{kc}{N}). \tag{5.8}$$

Note that CRH/CDH-1, CRH-N, and CDH-N naturally form autonomous systems. Therefore, the analytical request denial probabilities in these cases can be given by (5.7) or (5.8). However, for CRH-k, CDH-k for $k \neq 1, N$, whether an autonomous system can be formed or not, depends on the topology of the network. Most often, an autonomous system cannot be formed in CRH-k and CDH-k. In this chapter, we use (5.7) and (5.8) as approximations of analytical request denial probabilities for the infinite and finite models, respectively. The accuracy of the approximations will be assessed in Section 5.2.

We next comment on the Erlang load equivalency. Given $\mu = 1$, the Erlang load in the infinite population model is

$$A = \frac{\lambda}{\mu} = \lambda \tag{5.9}$$

while in the finite population model, it is

$$\overline{A} = \frac{u\gamma}{\mu + \gamma} = \frac{u\gamma}{1 + \gamma}. \tag{5.10}$$

We deliberately align the Erlang load in the infinite and finite models to make the performance of these models comparable. In addition, if we vary the number of users in the finite model and still keep the Erlang load the same, we need to vary the arrival rate γ accordingly. If we change the number of users from u to nu, the arrival rate γ' is given by

$$\gamma' = \frac{\gamma}{\gamma(n-1) + n}. \tag{5.11}$$

We also explore both the smooth and the bursty traffic beside Poisson traffic due to the dynamic nature of the traffic in the agent-based VPN; specifically, we consider the inter-arrival time distribution being Erlang-k and hyper-exponential. For simplicity, they are referred to as the Erlang-k arrival and the hyper-exponential arrival. Here we choose Erlang-4 distribution ($C^2 = 1/4$) as the smooth traffic, and hyper-exponential distribution with $C^2 = 25$ as the bursty traffic. In the infinite model with a general arrival process under CAS, the system is actually a G/M/m/m loss system. The general arrival process can be represented by the matrix-exponential distribution [53]; then, the request denial probability can be analytically calculated using an algorithmic approach described in [56]. Calculating accurate analytical result sfor non-Poisson arrivals is outside the scope of this study. The following relation is known for the infinite model with Poisson arrival under the CAS scheme:

$$B_{\mathcal{E}} < B_{\mathcal{M}} < B_{\mathcal{H}} \tag{5.12}$$

where $B_{\mathcal{E}}$ and $B_{\mathcal{H}}$ represent the request denial probability under Erlang-k distribution and the hyper-exponential distribution, respectively. We seek to understand how arrival processes impact performance when schemes such as CDH and CRH are used, and how much they differ; this result will be reported in Section 5.2.

We have understood that the clustering approach causes the arrival traffic to be partitioned over VPN servers. We now comment on how the arrival process to each cluster changes by dividing the traffic in the infinite model. According to Raikov's decomposition theorem [76], if we split a Poisson arrival process over N clusters, the subprocess of each cluster is still Poisson. More generally, if we generate a sub-process by randomly splitting an arbitrary point process by choosing an event with probability p, then the sub-process has the squared coefficient of variation C_p^2 [43]:

$$C_p^2 = 1 + p \cdot (C^2 - 1), \tag{5.13}$$

where C^2 is the squared coefficient of variation of the original process. Based on (5.13), it is easy to show that

$$\begin{cases} 1 < C_p^2 < C^2 & \text{if } C^2 > 1 \\ C_p^2 = 1 & \text{if } C^2 = 1 \\ C^2 < C_p^2 < 1 & \text{if } C^2 < 1 \end{cases} . \tag{5.14}$$

(5.14) indicates that by randomly and uniformly splitting the traffic, the irregularity of the traffic does not change, but the degree of irregularity of the traffic changes toward the opposite direction of its original irregularity except for Poisson traffic (it remains as Poisson traffic where $C^2 \equiv 1$). In other words, the smooth traffic ($C^2 < 1$) becomes more bursty; the bursty traffic ($C^2 > 1$) becomes more smooth. Because we split the traffic uniformly over N cluster in the infinite population model, $p \equiv 1/N$. By plugging p and C^2 of each distribution model in (5.13), we obtain that $C_p^2 = 1$ for Poisson arrival; $C_p^2 = 1 - \frac{3}{4N}$ for Erlang-4 distribution with $C^2 = 0.25$, and $C_p^2 = 1 + \frac{24}{N}$ for the hyper-exponential distribution with $C^2 = 25$. Therefore, after splitting non-Poisson traffic in the infinite model, the traffic characteristics of each cluster are different from the original traffic. The traffic characteristics have influence on the performance, which will be presented in Section 5.2.

5.1.4 Composite Metric–Penalty

So far, we have discussed request denial probability. The second measure of interest in the agent-based VPN architecture is the latency perceived by users. We next elaborate how we measure the latency between users and RPs in this work. First, we introduce the mean number of RPs users needed to inquire in ABVA, which is referred to as the *mean step*. However, this metric cannot precisely reflect the latency because it only tells the number of steps and ignores the diversity of the distance from users to different RPs. Since these are logical connections over the complex underlying overlay networks, the actual latency is comprehensively decided by the status of the underlying networks and the chosen path. It is hard to model exact latency. Therefore, we introduce a more straightforward measure for *latency*. For simplicity, we define *latency* by the physical distance between the user and the RP, which is sufficient for understanding our proposed schemes and identifying the trade-off between the request denial probability and latency in ABVA. The latency can be simply adjusted based on the needs within our framework.

In order to understand the trade-off between the request denial probability and the average latency together, we introduce a composite metric called *penalty* as follows:

$$P = \eta \times \text{ request denial probability } + \text{ latency}, \tag{5.15}$$

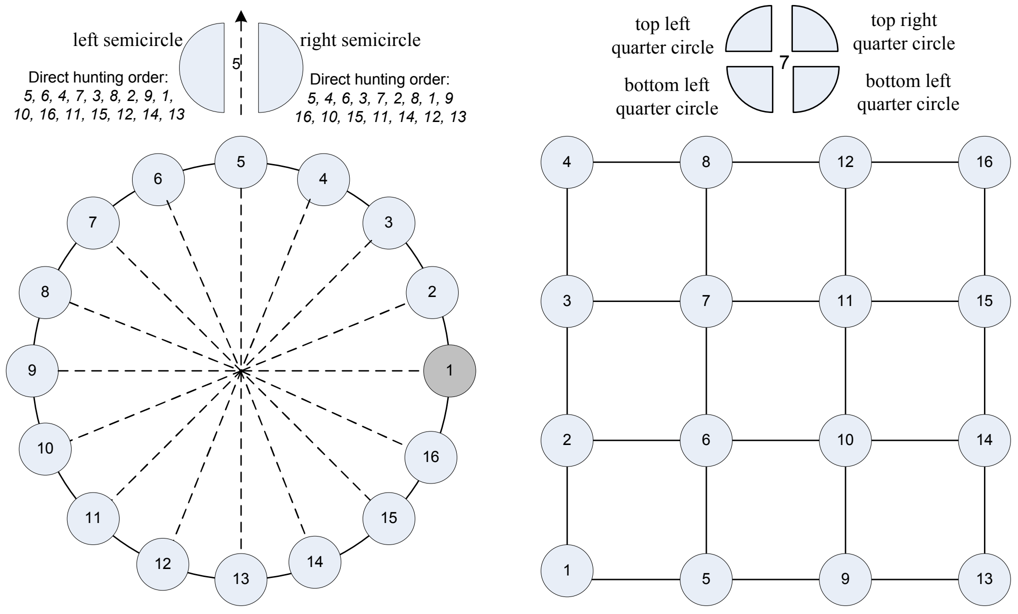
where η is the weight factor used to to balance the two factors. The low value of η implies that the dominant factor is latency while the high value of η implies that the dominant factor is the request denial probability.

5.2 Studies and Results

In the previous section, we presented the following different schemes: 1) centralized-all scheme (CAS), 2) clustering with randomized hunting (CRH-k), and 3) clustering with directional hunting (CDH-k). While we have presented analytical solutions or approximations for computing the request denial probability, there is no direct analytical method to determine mean step and latency. Thus, in this section, we present simulation results to verify the analytical solutions, to determine accuracy of the analytical approximation with regard to request denial probability. Through simulation, we quantify the average latency perceived as well as the average number of steps. We further seek to determine the optimal scheme balance between request denial probability and latency.

Since one of our goals is to understand the clustering approach in different topologies, we consider two topologies, ring and grid, for our studies. The ring topology is depicted in Fig. 39(a); it consists of 16 RPs ($N = 16$) that are evenly distributed on a circle with a radius of 100 units. Each VPN server has 25 units of capacity where the request data rate bandwidth requirement is assumed to be one unit; thus, the effective capacity of a VPN server is 25. Each VPN server is at the center of a smaller circle with a radius of one unit. In the infinite population model, user locations are randomly selected from 360 possible locations that are evenly distributed on the semicircle of the smaller circle on the left side of the directional line connecting the center of the big circle and that of the smaller circle. In the finite population model, users reside on n locations that are also evenly distributed on the smaller semicircle on the left side of the directional line connecting the center of the big circle and that of the smaller one. This specialized arrangement guarantees that users belong to the same cluster and have the same hunting sequence in the CDH-k schemes.

Now consider CDH in this topology. Because of the user location being in the



(a) The ring topology (b) The grid topology

Figure 39: The topologies

semicircle, in the case of multiple-step directional hunting, the attempts alternate between the nearest higher and lower numbered neighboring clusters. For example, if a user belongs to cluster i, then in CDH-2, it will try the server in cluster i followed by cluster $(i+1) \bmod N$. Note that server "0" which is resulted by above algorithm refers to server "N". In the case of multi-step directional hunting, a user, after being denied by its cluster server, will try servers in its neighboring clusters in the following order:

$$\begin{cases} (i+1, i-1, \cdots, i+N/2) \bmod N & \text{if } N = 2M \\ (i+1, i-1, \cdots, i-\lfloor N/2 \rfloor) \bmod N & \text{if } N = 2M+1 \end{cases} \tag{5.16}$$

where $M \in \mathbf{Z}^{+}$.

Finally, given this ring topology, we still want to consider centralized-all scheme (CAS) to serve as the base case for our study; in this case, all servers are assumed to be located centrally in the VPN server numbered 1.

At a first look, the ring topology and the centralized all case appear to be highly specialized; this is close to a situation that is often faced in reality. Suppose that an ABVA provider starts with a centralized RP location on the middle part of North America; now, its customers with users located around the globe continue to grow. Thus, the ABVA provider would want to expand by adding new server RP locations in the east and the west coasts in North America, and then expand eastward to Europe and westward to Asia, and eventually completing the circle around the globe. This then forms a "ring-like" topology. The specific distribution of users within their cluster is also very close to the reality. RPs are deployed in the center of the area with high density users and thus forms clusters; sparse users are assigned to nearest clusters. It is possible to design the clusters so that each cluster has about the same number of users. In addition, the distance of users within a cluster is much less than the distance of clusters. Therefore, the distance cost within a cluster can be neglected. Moreover, we can always manually set the hunting orders for users to form this specific ring topology.

It is still possible that a ring topology is not well planned and the provider does not want to manually set the hunting orders. This change results in the users within a cluster having different directional hunting orders. Note that it has no impact on random hunting orders. Therefore, the request denial probability and mean steps will not change for CRH and CAS schemes. In addition, the impact on the latency of changing user locations within their clusters is negligible as long as the directional/random hunting orders remain the same. Thus, the impact upon latency under CRH and CAS schemes is negligible because the hunting order for CRH is random and that there is only one server location for CAS.

Clearly, it is the directional hunting orders that may influence the performance of CDH schemes. Distributing users in the full circle instead of semicircle forms such a variation. This topology is referred to as *full ring*. The consequence of this variation is that the users in the same cluster have two different hunting orders for CDH ($1 < k \leq N$): $(i, i+1, i-1, \cdots, i+N/2)\text{mod}N$ for users at the left semicircle, $(i, i-1, i+1, \cdots, i-N/2)\text{mod}N$ for users at the right semicircle when N is even; $(i, i+1, i-1, \cdots, i-\lfloor N/2 \rfloor)\text{mod}N$ for users at the left semicircle, $(i, i-1, i+1, \cdots, i+\lfloor N/2 \rfloor)\text{mod}N$ for users at the right semicircle when N is odd. As an example, the hunting orders for users in cluster 5 are given in Fig. 39(a).

We next use the grid topology as an example to study the topology that is not well-planned in reality. It also consists of 16 servers that are distributed on a 4×4 grid where the grid distance is 100 units. Each server has 25 units of capacity where the request data rate bandwidth requirement is assumed to be 1 unit; thus, the effective capacity is 25. Each server is at the center of a smaller circle with a radius of 1 unit. The user location on the smaller circle is the same as that in the full ring topology. The topology is depicted in Fig. 39(b). We keep all other settings as same as those in the ring topology. In this topology, users have different directional hunting orders and the number of different directional hunting orders are different for different clusters. For example, users at cluster 7 have 4 different hunting orders as illustrated in Fig. 39(b).

Since we consider both the infinite and the finite models, we want their loads given by (5.9) and (5.10), respectively, to be comparable. In particular, in order to understand system performance and trade-offs for different loads, we vary the λ from a low

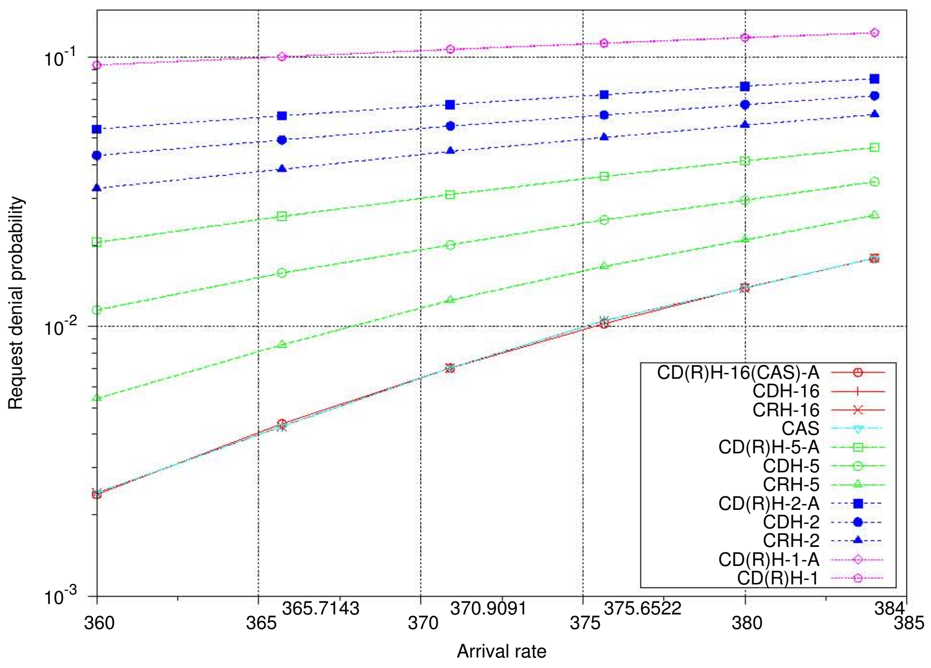


Figure 40: Request denial probability of analytic approximation and simulation in the infinite population model with Poisson arrival

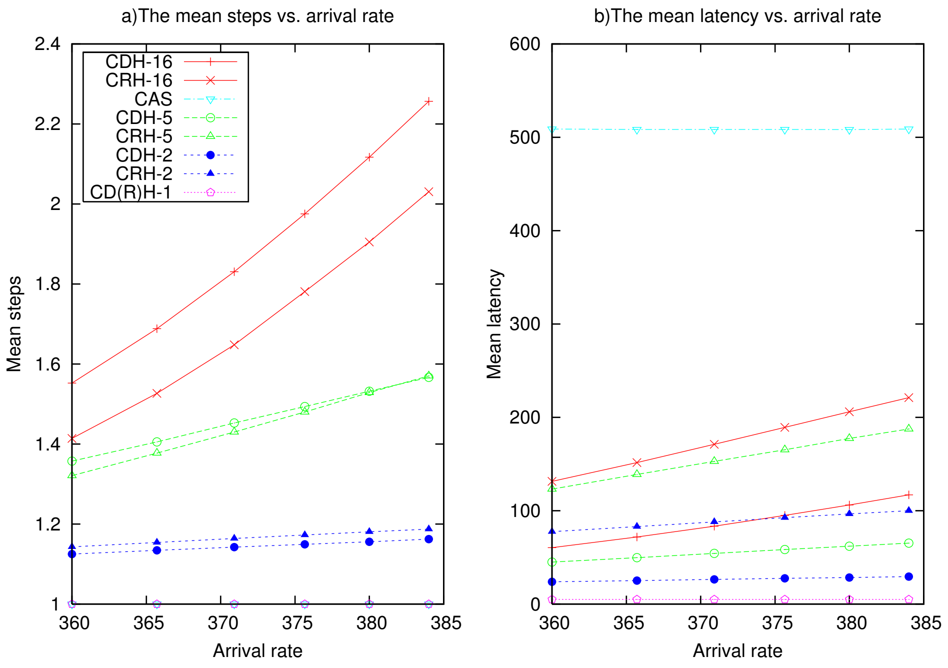


Figure 41: The mean step and latency in the simulated infinite population model with Poisson arrival

arrival rate, 360, to a high arrival rate, 384, in the infinite model. According to (5.9) and (5.10), arrival rate per user, γ, is set from 3 to 4 in the finite population model. For ease of following, we clarify the scenarios we have simulated. All performance metrics are plotted against the arrival rate, λ in the infinite population model and the arrival rate, γ in the finite population model. We consider the following strategies: centralized-all scheme(CAS), CD(R)H- 1, CRH- k,CDH- k. We study Poisson arrival as well as non-Poisson arrival under three different topologies for both infinite and finite models. We will consider the topologies in the following order: the ring topology, followed by the full ring topology, and finally the grid topology. For each topology, we study three different arrival processes as mentioned in Section 5.1; among these, we use Poisson arrival as a baseline study.

5.2.1 Infinite Model

Fig. 40 shows the request denial probability of simulation and the analytical results under different schemes for Poisson arrival; this is shown in log scale to avoid losing details at small values. We can see that the analytical results of request denial probability are accurate for CD(R)H-1, CRH-16 and CDH-16. These observations are consistent with Theorem 1 and form autonomous systems. With regard to CRH-k and CDH-k for $1 < k < 16$, we tested several different values. Here, we report results for $k = 2, 5$ to show the main differences. First, we note that the analytical approximation $B_{\mathcal{M}}(\frac{k\lambda}{N}, \frac{kc}{N})$ (for $1 < k < 16$), overestimates the request denial probability compared to simulation, although the gap is marginal at lower arrival rates; that is, the analytical approximation serves as an upper bound for the systems that cannot form autonomous systems. Furthermore, from simulation, we can see that the request denial probability of CRH-k is consistently smaller than that of CDH-k for $k = 2, 5$. It is not surprising that increasing the maximum allowed steps (k) reduces the request denial probability.

In Fig. 41, we present results on mean step and latency in the infinite model with Poisson arrival. Although, the mean step for both CD(R)H-1 and CAS is always 1, CD(R)H-1 is much better than CAS in terms of latency, because the clustering based scheme divides the server capacity into multiple RPs and the users only try the home RPs that are closest to them in CRH-1/CDH-1. In multiple step clustering based scheme cases, as the traffic load becomes heavier, the average number of steps increases because users (on average) need to go through more steps to find an available VPN server. As the maximum number of allowed steps k is increased, this trend becomes more pronounced because users are allowed to try more VPN servers. Likewise, the latency increases as traffic load becomes heavier and increasing k causes higher latency for the same clustering scheme. The advantage of CDH over CRH is significant in terms of latency. CDH-k always has less latency than CRH-k does when $k > 1$. CDH-k and CRH-k have very close mean step except for $k = 16$. The reason is that users randomly choose RPs instead following a static order in CDH to connect which gives the whole system more freedom and it is pronounced at large k.

It is noted that CD(R)H-1 has the least latency, but it sacrifices request denial probability, especially at high arrival rates. On the other hand, CDH, CRH, and CAS have the smallest request denial probability, but their latency is sacrificed. In Fig. 42, we plot

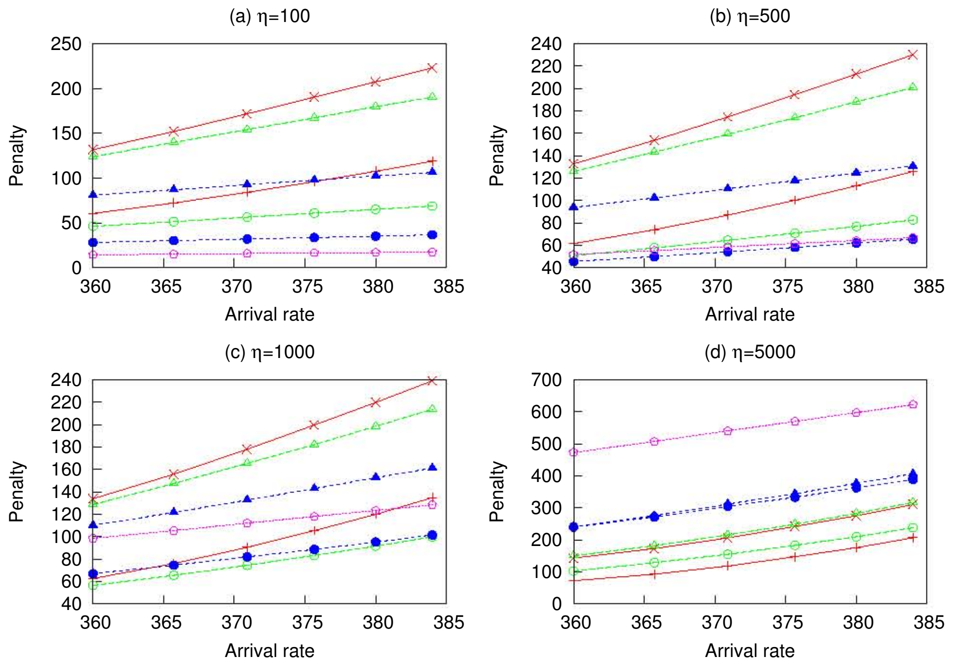


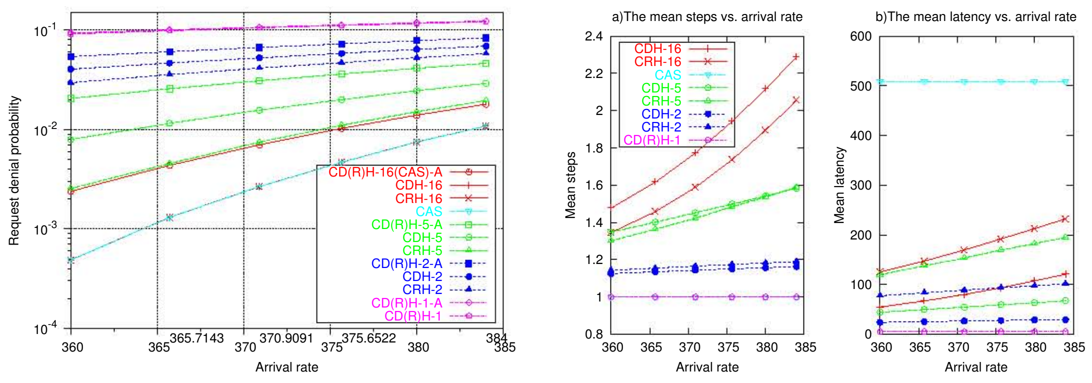
Figure 42: Penalty of different schemes in the infinite model with Poisson arrival (legends are the same as that in Fig. 41)

the penalty for four different values of η in (5.15). From the viewpoint of the penalty, CDH is always better than CRH because the advantage of latency of CDH outweighs the advantage of the request denial probability of CRH. We notice that four different schemes with the lowest penalty are shown for four different values of η: they are CD(R)H-1, CDH-2, CDH-5, CDH-16 for $\eta = 100, 500, 1000, 5000$, respectively. As we expect, the optimal scheme shifts towards the larger number of k as η is increased. Due to the non-linearity of request denial probability as well as latency with regard to the traffic load, the penalty is also subject to traffic load. For example, in the case of $\eta = 500$, the difference between the optimal scheme, CDH-2 , and the suboptimal scheme, CD(R)H-1 becomes marginal as the traffic load increases. The same convergence happens between the optimal scheme, CDH -5 and the suboptimal scheme, CDH-2 for $\eta = 1000$. The suboptimal solution becomes optimal at even higher traffic loads in these two cases.

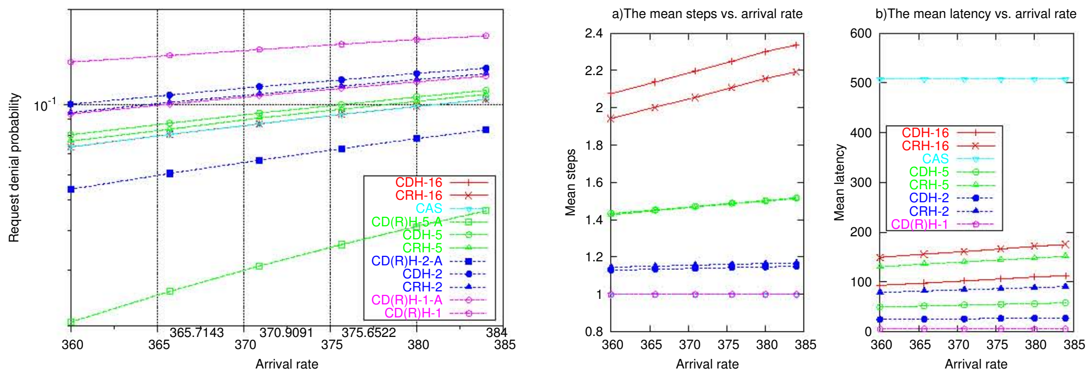
We consider the performance of these schemes with non-Poisson arrivals. Fig. 43 gives the performance of Erlang-4 ($C^2 = 1/4$) and hyper-exponential ($C^2 = 25$) inter-arrival time distribution. By comparing the analytical approximation and simulation results of the request denial probability on the two non-Poisson arrival cases, we find that the analytical model overestimates the request denial probability for the Erlang-4 case and this overestimation is more significant than that in the Poisson case. However, the analytical model for the Poisson case underestimates the request denial probability for the hyper-exponential cases. This phenomenon is explained by 1) the analytical approximations are made for the Poisson case, 2) (5.12) also holds for non-autonomous system, CDH/CRH- k for $k = 2, 5$ besides the autonomous system, CDH/CRH -k for $k = 2, 5$.

CDH-k results in higher request denial probability than CRH-k for $k = 2, 5$ in these non-Poisson cases. The mean steps for CDH and CRH scheme are very close, especially when the maximum allowed number of steps are small. We can see that CDH-16 has higher mean steps than CRH-16 has. The reason is the same as we have stated in the Poisson case. However, directional hunting schemes show significant advantages over random hunting schemes in terms of latency. In addition, we find that the simulation results for the non-Poisson case of CD(R)H-1 is closer to the analytical models made for the Poisson case than that of CDH-16/CRH-16; this is more clear in non-log plots, which are not shown here. This phenomena can be explained by (5.14). (The C^2 of CD(R)H-1 is closer to one (Poisson case) compared with that of CDH-16/CRH-16 in non-Poisson cases). Other observations that are consistent with those in the Poisson case are omitted. We use the same method to combine two performance metrics into one as we do for the Poisson arrival case.

Since the penalty of CDH schemes is always less than CRH schemes, we focus on CDH schemes, specifically CDH-2 and CDH-5, to compare the performance of different arrival processes. Fig. 44 and Fig. 45 re-plot the request denial probability, mean steps and latency under CDH-2 and CDH-5 with three different arrival processes. It is clear that the arrival processes have significant impact on the request denial probability. For CDH-5, the request denial probability of the hyper-exponential case is more than one time higher than that of the Poisson case, and for CDH-2, this diverge becomes more than seven times higher. Since mean step and latency discussed here does not include the requests being denied, higher request denial probability does not necessarily cause



The Erlang-4 case



The hyper-exponential ($C^2 = 25$) case

Figure 43: Performance in the infinite model with non-Poisson arrivals

η	100			500			1000			5000		
C^2	Ring	FullRing	Grid	Ring	FullRing	Grid	Ring	FullRing	Grid	Ring	FullRing	Grid
$1/4$	CDH-1	CDH-1	CDH-1	CDH-2	CDH-2	CDH-1	CDH-5	CDH-5	CDH-2 $\downarrow$ CDH-1	CDH-16	CDH-16	CDH-16
1	CDH-1	CDH-1	CDH-1	CDH-2	CDH-2	CDH-1	CDH-5	CDH-5	CDH-2 $\downarrow$ CDH-1	CDH-16	CDH-16	CDH-16 $\downarrow$ CDH-5
25	CDH-1	CDH-1	CDH-1	CDH-1	CDH-1	CDH-1	CDH-2	CDH-2	CDH-1	CDH-5	CDH-5	CDH-5

Table 7: Optimal schemes in the infinite model ("$a \to b$" vertically means that the optimal scheme changes from a to b as the arrival rate increases)

higher mean step and latency. Therefore, the order from highest to lowest of mean step and latency is ranked as hyper-exponential, Poisson, Erlang-4 at light traffic load, and the order changes to Erlang-4, Poisson, hyper-exponential at heavy traffic load. Namely, accepted Erlang-4 requests need the least mean step/latency to find an available server at low load while accepted hyper-exponential requests need the least mean step/latency at high load. However, the switch points for mean steps and latency are different because mean step ignores the distance information. As we can see from the summary of optimal solutions in Table 7, changing C^2 of arrival processes from high to low may change optimal solutions, because the variation of the optimal solution is dependent on whose change can dominate the change of the penalty—request denial probability or latency. The request denial probability consistently increases while the latency increases at low traffic load then decreases at high traffic load as C^2 reduces. Note that CD(R)H-1 is written as CDH-1 for short in Table 7. It is not surprising that CDH always outperforms than CRH in the absence of the cost of obtaining server location information in CDH.

Fig. 46 and Fig. 47 compare the performance of three topologies in the infinite model with Poisson arrival. Since the number of steps of CD(R)H-1 and CAS for any topology are always one, these known results are not shown in Fig. 47. We can see that

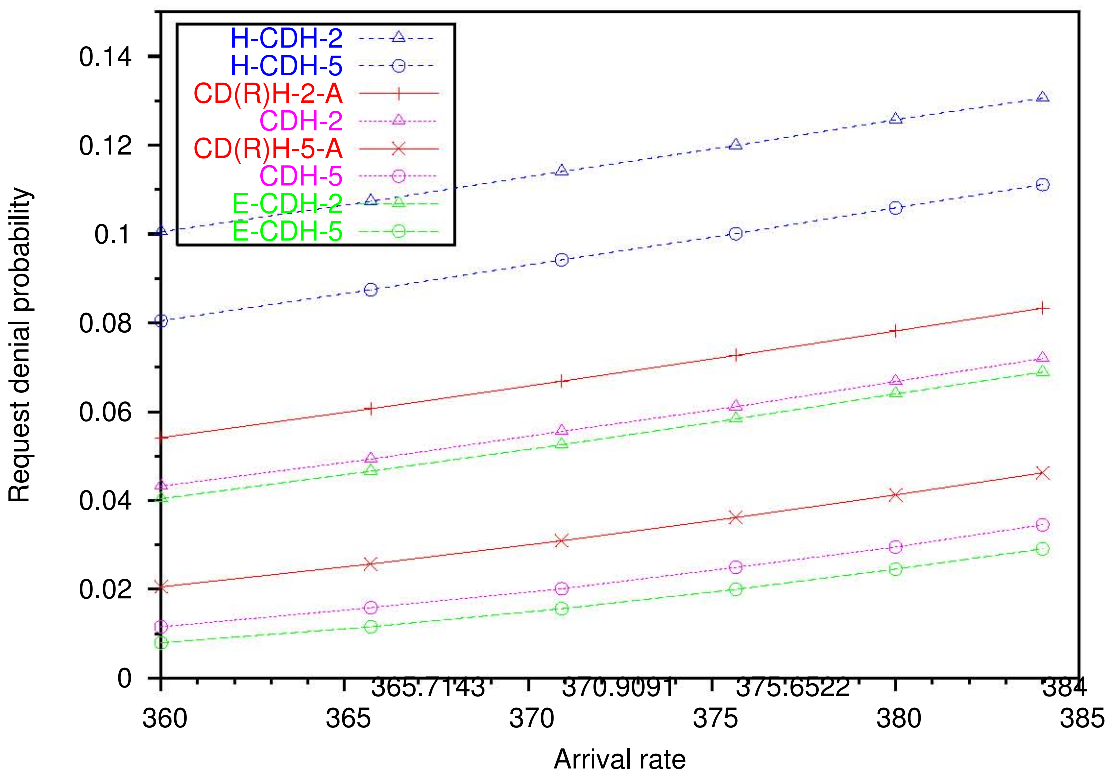


Figure 44: Request denial probability of CDH in the infinite model with different arrivals

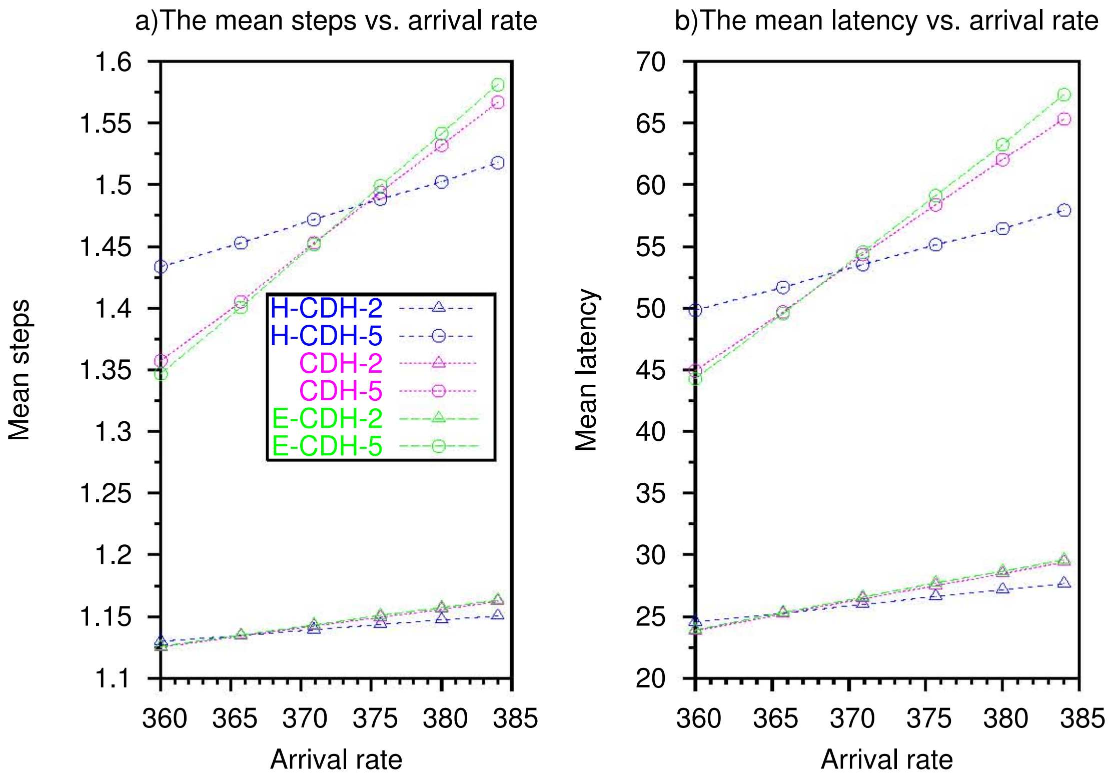


Figure 45: The mean step and latency of CDH in the infinite model with different arrivals

topology going from ring to full ring has no impact on any metrics. The topology going from ring to grid only significantly augment the latency, which is caused by the the distance difference of users to RPs between these two topologies. With regard to the mean step, going from the ring to the grid has some influence on CDH schemes while no influence on CRH scheme. It can be again explained by the more freedom that CRH has than CDH does. Changing topology from the ring to the grid does not change the random hunting orders but changes the directional hunting order. The reason why the mean step of CDH does not change when going from the ring to the full ring is that the hunting orders are evenly distributed among users in full ring topology which is not the case in the grid topology. Nevertheless, the difference for CDH schemes is so small that it can be ignored. Just like the ring topology, CRH schemes in other two topologies result in slightly higher request denial probability due to the freedom of random hunting. The advantage of CDH over CRH in terms of latency is significant for all three topologies. The clustering schemes can effectively balance the request denial probability and latency in these topologies. Therefore, as shown in Table 7, going from ring to full ring has no affect on penalty because changing locations within a cluster has little impact on the performance metrics, while going from ring to grid may cause the k of the optimal solution, CDH-k to decrease to alleviate the increase of latency.

The observation of comparing performance in different topologies for non-Poisson cases is consistent with the Poisson cases. The clustering schemes also play the role of balancing the request denial probability and latency. As shown in Table 7, we can see that the above conclusion regarding the influence of optimal schemes caused by changing arrival process in full ring and grid topologies and caused by changing topology for Poisson cases also holds for non-Poisson cases.

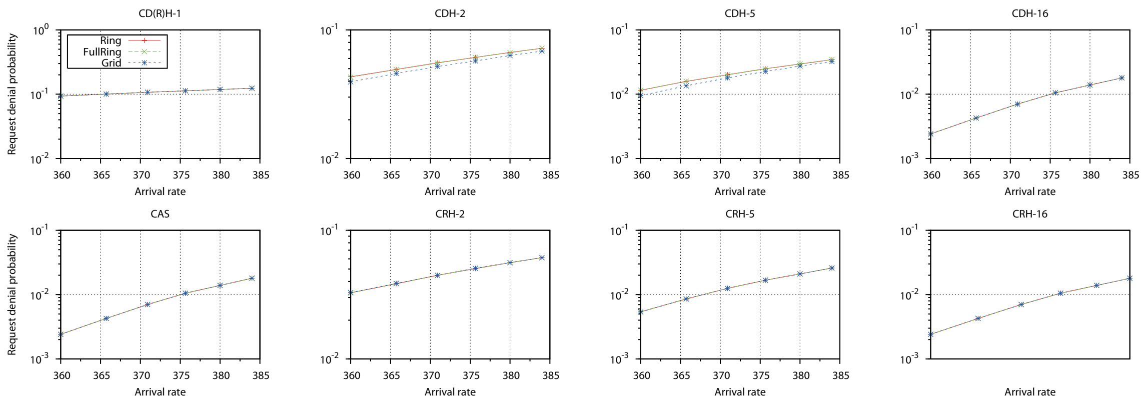


Figure 46: Request denial probability in the infinite model with Poisson arrival under different topologies

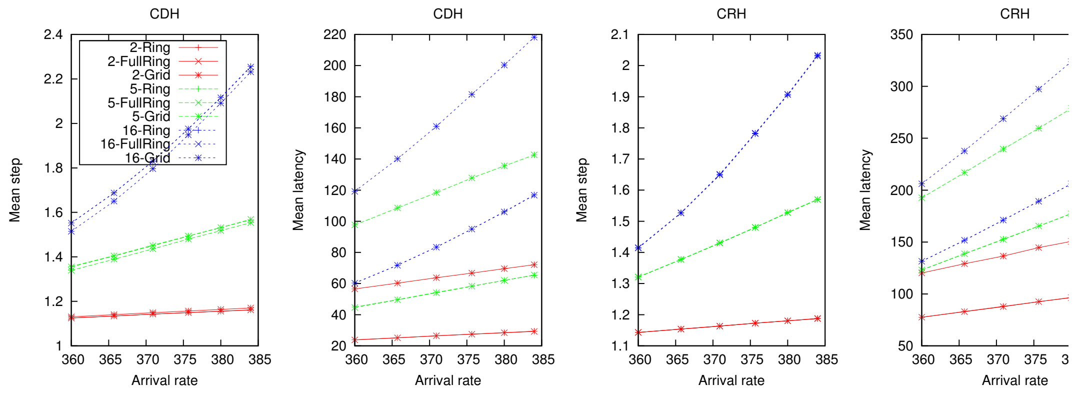


Figure 47: The mean step and latency in the infinite model with Poisson arrival under different topologies

η	100			500			1000			5000		
C^2	Ring	FullRing	Grid	Ring	FullRing	Grid	Ring	FullRing	Grid	Ring	FullRing	Grid
$1/4$	CDH-1	CDH-1	CDH-1	CDH-5 ↓ CDH-2	CDH-5 ↓ CDH-1	CDH-1	CDH-5	CDH-5	CDH-5 ↓ CDH-1	CDH-16	CDH-16	CDH-16
1	CDH-1	CDH-1	CDH-1	CDH-5 ↓ CDH-2	CDH-2	CDH-1	CDH-16 ↓ CDH-5	CDH-16	CDH-5 ↓ CDH-2 ↓ CDH-1	CDH-16	CDH-16	CDH-16
25	CDH-1	CDH-1	CDH-1	CDH-5 ↓ CDH-2	CDH-5 ↓ CDH-2	CDH-2 ↓ CDH-1	CDH-16 ↓ CDH-5	CDH-16 ↓ CDH-5	CDH-5 ↓ CDH-2	CDH-16	CDH-16	CDH-16

Table 8: Optimal schemes in the finite model ("$a \to b$" vertically means that the optimal scheme changes from a to b as the arrival rate increases)

5.2.2 Finite Model

We start studying the finite model with the case that the arrival process is Poissonian and the topology is ring. According to (5.9) and (5.10), we equate the Erlang load of the finite model to the infinite model by setting the user arrival rate in the range of 3 to 4 in the finite model. Fig. 48 shows the simulation and analytical results of request denial probability. Fig. 49 depicts the mean step and latency. The observations are consistent with the finite model. It is noted that the analytical models for the infinite case are also accurate for CD(R)H-1, CDH-16, CRH-16 and CAS, which is also consistent with Theorem 1. In regard to CDH-k and CRH-k, when $k = 2, 5$, we find that analytical approximation $\overline{B}_{\mathcal{M}}^{(k)}$, overestimates request denial probability compared to simulation. For sake of space, we do not repeat other observations that are similar in the infinite model. The penalty for different values of weight factor, η is depicted in Fig. 50. As shown in Table 8, the optimal schemes depend on the weight factor as well as traffic load (arrival rate).

Fig. 51 and Fig. 52 compare the performance among three different arrival processes in the ring topology. Unlike the finite model the changes of the request denial probability as arrival processes change have no consistent pattern in the finite model. As

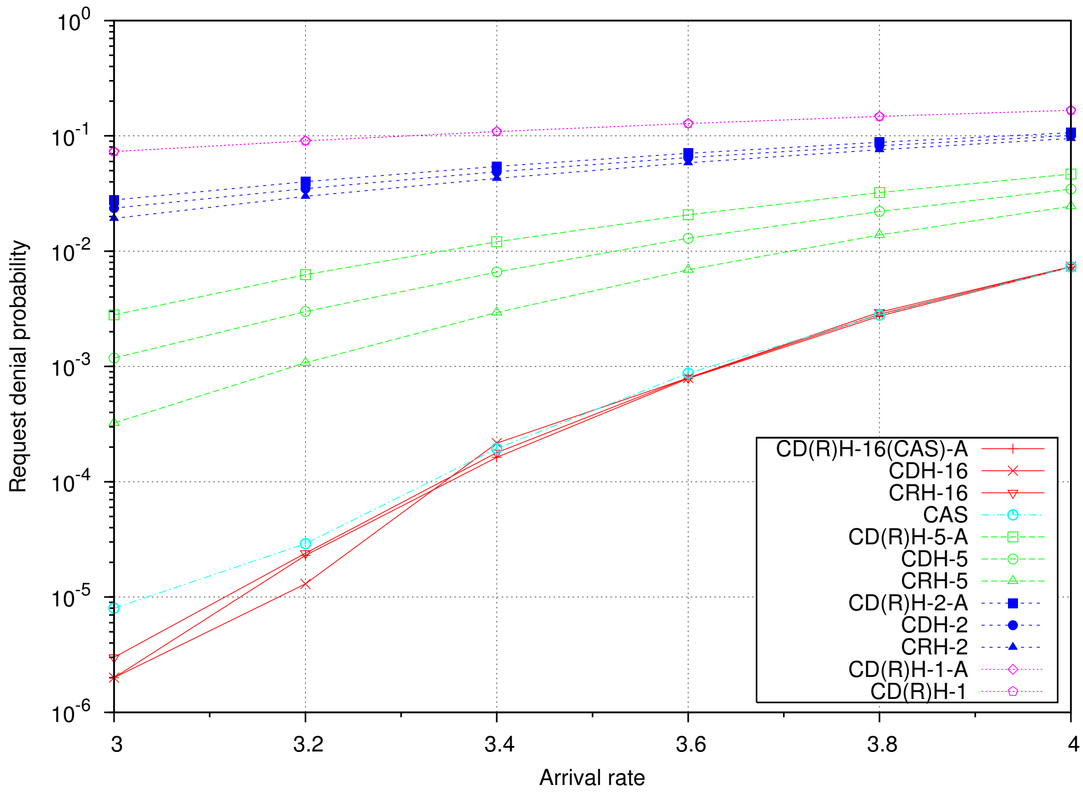


Figure 48: Request denial probability in the finite model with Poisson arrival

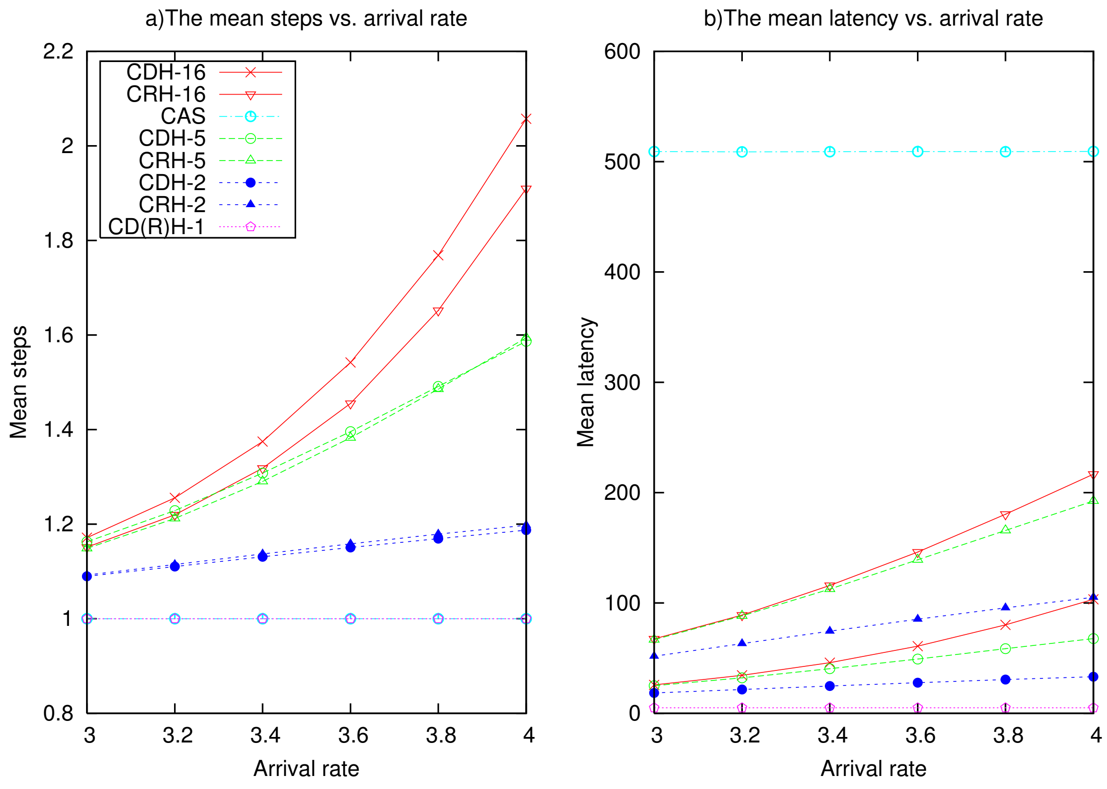


Figure 49: The mean step and latency in the finite model with Poisson arrival

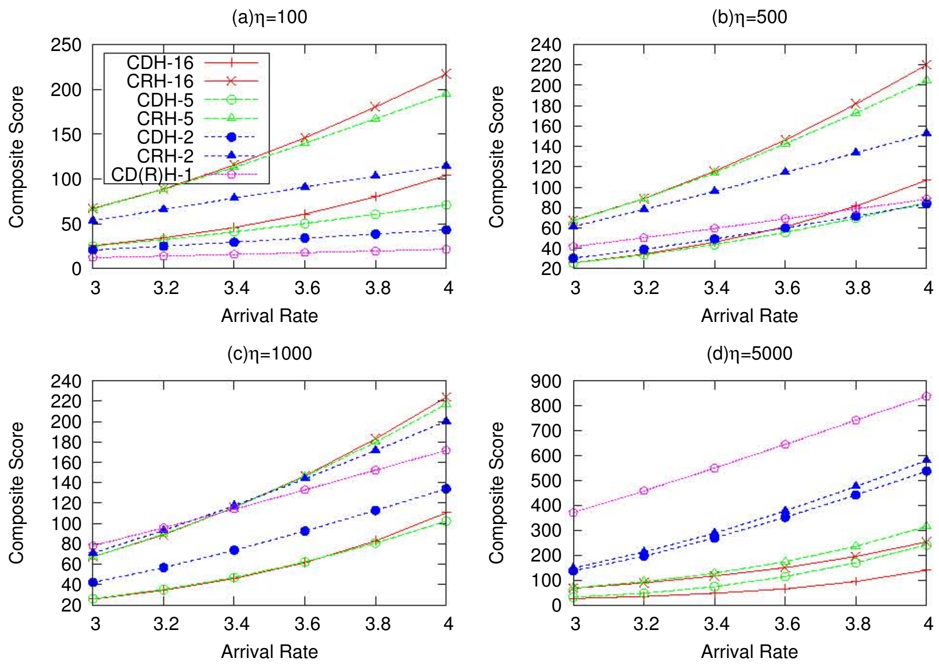


Figure 50: Penalty of different schemes in the finite model with Poisson arrival

to the mean step and latency, Erlang-4 case is consistently the largest, followed by Poisson case, then by the case of hyper-exponential($C^2 = 25$) for all multiple-step clustering schemes. The optimal schemes for these cases are summarized in Table 8.

Fig. 53 and Fig. 54 compare the performance of three topologies in the finite model with Poisson arrival. We find that the influence of changing topologies on optimal schemes in the finite model is very similar to that in the infinite model. For sake of space, the comparison plots of changing topology for non-Poisson cases that are consistent with the Poisson case are not shown in this thesis. We conclude that our schemes work for different arrival processes. The optimal solutions for these cases are summarized in Table 8.

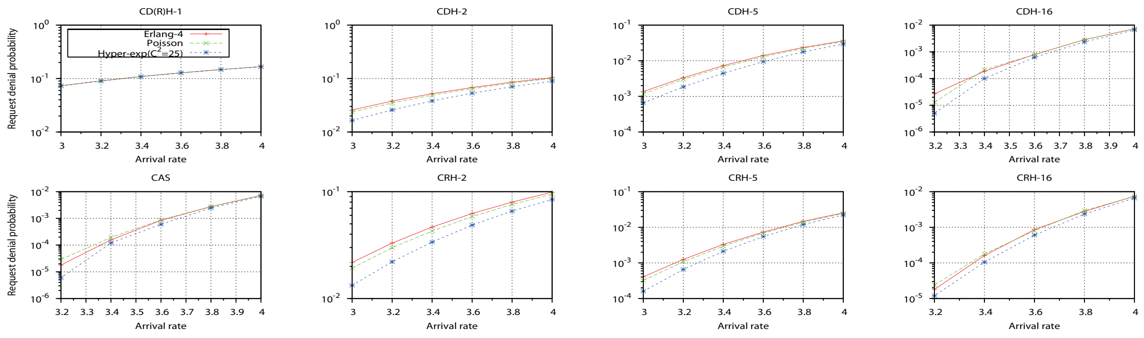


Figure 51: Request denial probability in the finite model with different arrivals in the ring topology

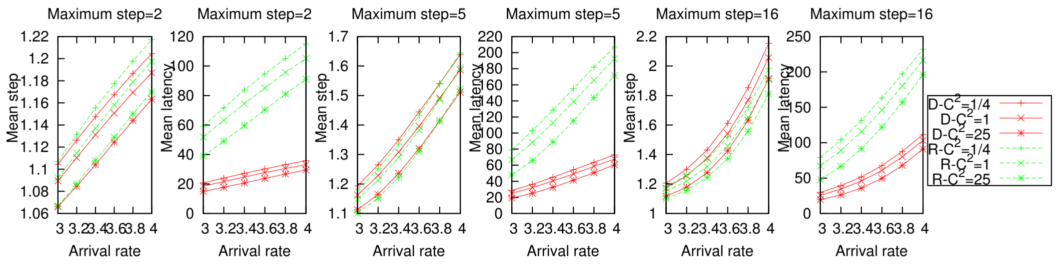


Figure 52: The mean step and latency in the finite model with different arrivals in the ring topology

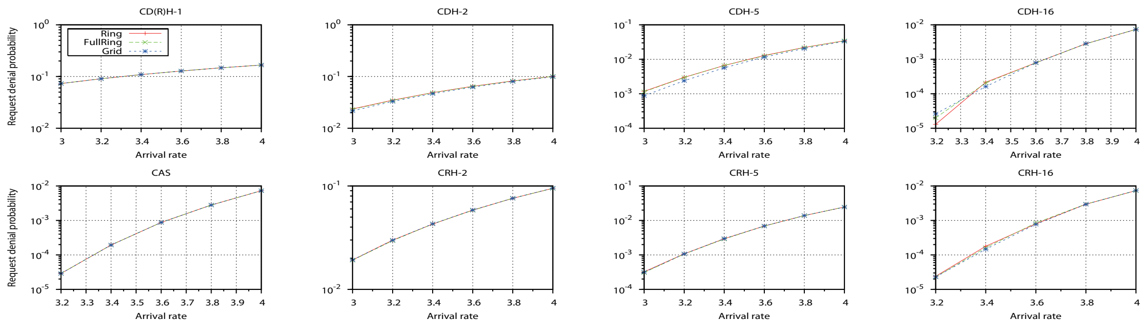


Figure 53: Request denial probability in the finite model with Poisson arrival under different topologies

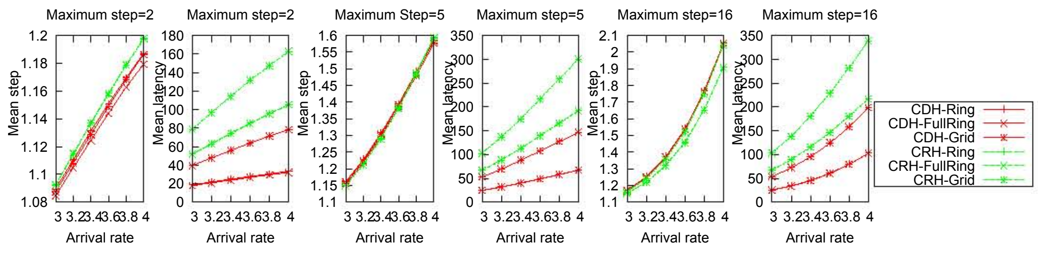


Figure 54: The mean step and latency in the finite model with Poisson arrival under different topologies

5.2.3 Comparison of Infinite and Finite Models

In order to understand how the number of users change the performance in the finite model and compare finite model with infinite model as the number of users increases while keeping the Erlang load the same, we set the number of users in each RP in the finite models to a number of different users ranging from 30 to 180. We calculate the arrival rate for different number of users based on (5.11) so that the Erlang load remains the same. Fig. 55 depicts the request denial probability of these three finite models and the infinite model with Poisson arrival in the ring topology for CD(R)H-1, CDH-2, CDH-5 and CDH-16. Note that the x-axis is the total Erlang load for ease of comparison. Fig. 56 presents the request denial probability of these three finite models and the infinite model with Poisson arrival in the ring topology for CRH-2, CRH-5 and CRH-16.

From Fig. 55(d) and Fig. 56(c), we note that there is a significant difference in performance between the infinite model and the finite model with up to 40 users per cluster. We found that, at high load, the relative difference between the infinite and the finite models is only about 2% when the number of users per cluster is 90 or higher; this means that the infinite model can be used to approximate the finite model at high load.

On the other hand, when the load is low, we found that the relative deference can be as high as 24% when comparing the infinite model and the finite model with 180 users per cluster. This suggests that at a low load, it is better to use the finite model to get more accurate understanding of the system performance.

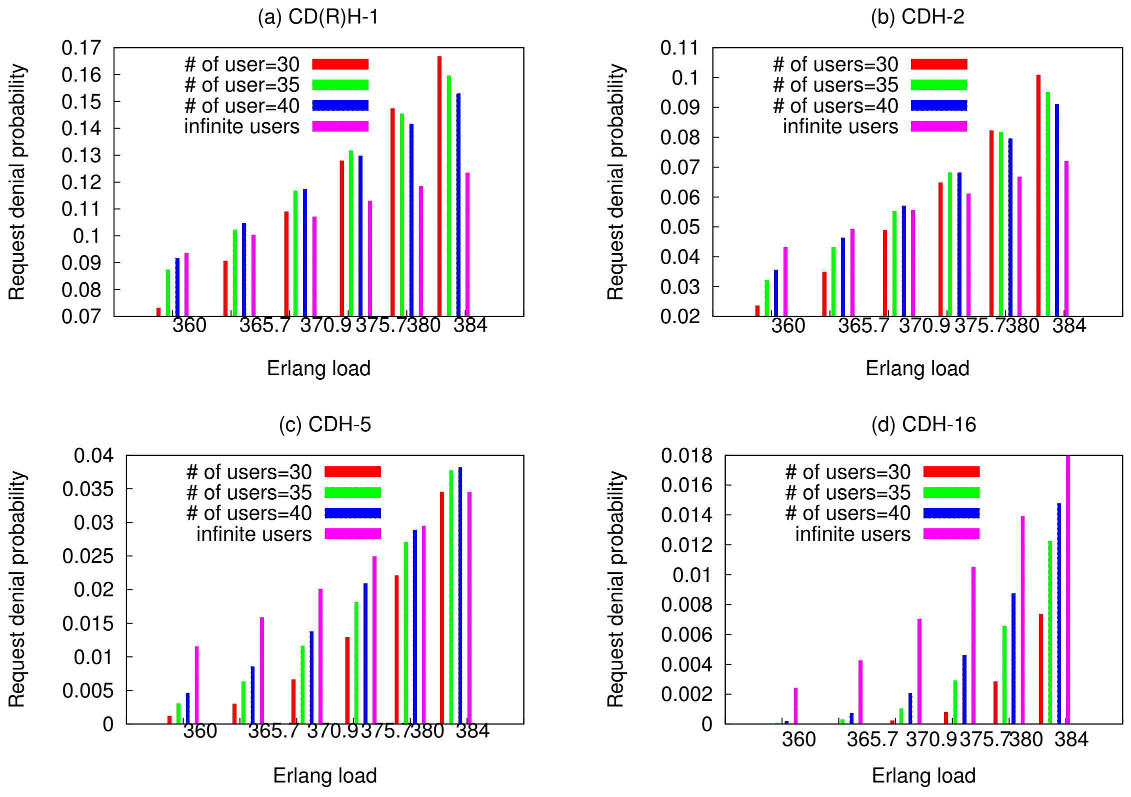


Figure 55: Request denial probability of CDH in the infinite model and finite model with different number of users with Poisson arrival in the ring topology

5.3 Summary

In this chapter, we consider the offering of an agent-based VPN service where organizations and their users utilize Rendezvous Points for remote-access VPN services. In

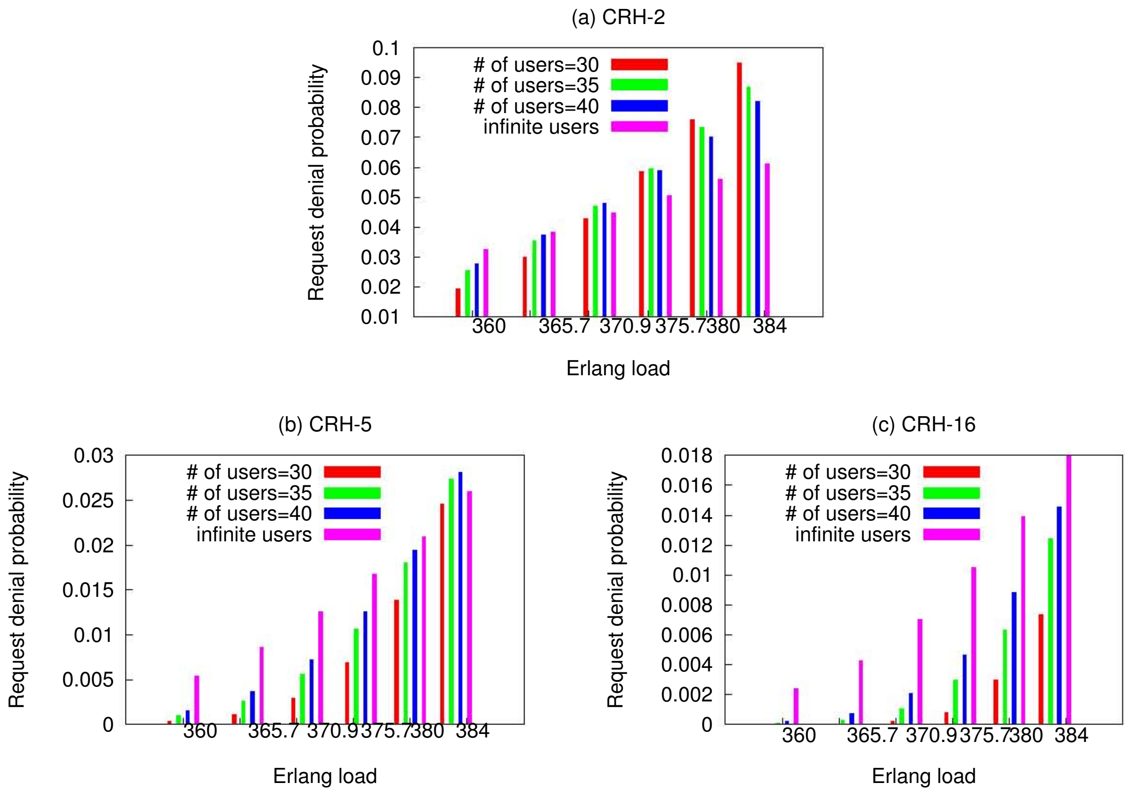


Figure 56: Request denial probability of CRH in the infinite model and finite model with different number of users with Poisson arrival in the ring topology

particular, we study the systems management problems of an ABVA provider to balance request denial probability and latency perceived by users. The models with assumption of both infinite number of users and finite number of users are discussed and compared in this chapter. Analytical results for request denial probability for Poisson arrival were discussed; results for non-Poisson arrival (with $C^2 = 1/4, 25$) are presented through simulation. Furthermore, the latency perceived is studied by considering ring, full ring, and grid topologies. In the infinite model, we found that the arrival processes have significant and consistent impact on the request denial probability and the impact on the latency is dependent on the traffic load. In the finite model, the arrival processes have inconsistent impact on the request denial probability. As to the latency in the finite model, Erlang-4 case is consistently largest, followed by Poisson case, then by hyper-exponential case.

To consider the case of VPN servers being located at multiple Rendezvous Points, we have identified a number of schemes for server selection that includes a concept based on clustering. In particular, we find that clustering with directional hunting is the best option if the goal is to balance request denial probability and latency in the condition that the cost of obtaining directional information is free. The optimal schemes are decided by multiple factors: arrival process, topologies, traffic load, and the weight factor. Tables 7 and 8 are included to provide a comprehensive summary in this regard.

Finally, a strength of this work is that we compared finite and the infinite models; such comparison is commonly ignored in the literature. We found that the request denial probability of the finite model converges to that of the infinite model at the high load as long as the number of users of each cluster is higher than 90 and the arrival process is the same under CAS, CDH-16 and CRH-16. However, at low load, there is a marked difference in the result between the infinite model and the finite model (even with up to 180 users per cluster). These observations serve as a guideline to use the finite model in the low load case and the infinite model as a good approximation in the high load case.

There are a number of limitations of this work. First, we consider arrival models that may not reflect reality. Due to the unavailability of a traffic data set, we could not consider an accurate arrival model; instead, we resorted to standard Poisson and non-Poisson models to gain some understanding of the problem. Secondly, our work is limited to considering ring, full ring, and grid topologies. While we point out later that the ring topology is fairly realistic, additional realistic topologies would be worthwhile to consider, especially to see how the hunting schemes perform. Thirdly, we considered a loss system for arriving requests if the requested bandwidth request cannot be provided; in reality, the system can allow some more requests to be admitted while somewhat downgrading the bandwidth for existing users; this brings up another trade-off factor, which is not addressed in this chapter. In future work, it would be worthwhile to pursue how to overcome these limitations.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This dissertation has solved two data center resource management problems. The first one is to answer how to minimize server operational cost by leveraging switching servers on/off and DVFS when resource requirements (workload demand) is given. The dynamics of the workload demand must be aware of in this particular problem to avoid local optimum. We formulate this problem in this dissertation. Our evaluation shows that our approaches can save significant energy as well as the data center CAPEX. To alleviate the computational complexity to solve the problem, this dissertation has proposed aggregation workload demand. The numerical results shows that aggregation workload over time slots is efficient with small overhead for dynamic aggregation method. We are working on decomposing workload demand into special substructures and developing algorithms to compute optimal solutions according to the substructures. The substructure based algorithms are more efficient than the mathematical programming based method.

The other one is to answer how many resources data center operators should allocate to ISePs such that user QoE is guaranteed. This dissertation has built a hierarchical model to evaluate QoE considering data center reliability, allocated outbound bandwidth, allocated buffer size, data center redirection strategies, request arrival and service pattern, *etc.* We plan to develop a more realistic data center availability model considering components with different availability with different request dynamics (*i.e.*, user arrival and service pattern). We propose ”heuristic style” redirection strategies for agent-based VPN service.

Ultimately, we want to combine these two problem into a uniform data center resource management framework.

REFERENCE LIST

[1] Amazon EC2. http://aws.amazon.com/ec2/.

[2] Bing. http://www.bing.com.

[3] Facebook. http://www.facebook.com.

[4] Google. http://www.google.com.

[5] Linkedin. http://www.linkedin.com.

[6] PhoneFactor. http://www.phonefactor.com.

[7] Twitter. http://www.twitter.com.

[8] Zynga. http://www.zynga.com/games.

[9] Agarwal, S., Dunagan, J., Jain, N., Saroiu, S., Wolman, A., and Bhogan, H. Volley: Automated Data Placement for Geo-Distributed Cloud Services. In *NSDI'2010* (2010).

[10] Arlitt, M., and Jin, T. Workload Characterization of the 1998 World Cup Web Site. Tech. rep., IEEE Network, 1999.

[11] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., and Zaharia, M. Above the Clouds: A Berkeley View of Cloud Computing. Tech. rep., EECS Department, University of California, Berkeley, 2009.

[12] Barroso, L. A. The Price of Performance. *Queue 3*, 7 (Sept. 2005), 48–53.

[13] Barroso, L. A., and Hölzle, U. The Case for Energy-Proportional Computing. *Computer 40*, 12 (2007), 33–37.

[14] Barroso, L. A., and Hölzle, U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[15] Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. A Taxonomy and Survery of Energy-Efficient Data Centers and Cloud Computing Systems. http://arxiv.org/abs/1007.0066, September 2010.

[16] Berral, J. L., Goiri, Í n., Nou, R., Julià, F., Guitart, J., Gavaldà, R., and Torres, J. Towards Energy-Aware Scheduling in Data Centers using Machine Learning. In *e-Energy '10: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking* (New York, NY, USA, 2010), ACM, pp. 215–224.

[17] Bertini, L., Leite, J. C. B., and Mossé, D. Power optimization for dynamic configuration in heterogeneous web server clusters. *J. Syst. Softw. 83*, 4 (2010), 585–598.

[18] Bianchini, R., and Rajamony, R. Power and Energy Management for Server Systems. *IEEE Computer 37* (2004), 2004.

[19] Bichler, M., Setzer, T., and Speitkamp, B. Capacity Planning for Virtualized Servers. In *Workshop on Information Technologies and Systems (WITS)* (Milwaukee, Wisconsin, 2006).

[20] Blackburn, M. Five Ways to Reduce Data Center Power Consumption, 2008. http://www.greenbiz.com/research/report/2009/03/10/five-ways-reduce-data-center-power-consumption.

[21] Bodik, P., Armbrust, M. P., Canini, K., Fox, A., Jordan, M., and Patterson, D. A. A Case for Adaptive Datacenters To Conserver Energy and Improve Reliability. Tech. rep., EECS Department, University of California, Berkeley, 9 2008.

[22] Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. *The case for power management in web servers*. Kluwer Academic Publishers, Norwell, MA, USA, 2002, pp. 261–289.

[23] Borst, S., Gupta, V., and Walid, A. Distributed Cache Algorithms for Conten Distribution Networks. In *Proc. of IEEE INFOCOM* (2010).

[24] Brey, T., and Lamers, L. Using Virtualization to Improve Data Center Efficiency. http://www.thegreengrid.org/ /media/WhitePapers/Whiteng2009.

[25] Calzarossa, M., and Ferrari, D. A sensitivity study of the clustering approach to work-load modeling. *Performance Evaluation 6* (1986), 25–33.

[26] Chandra, K., and Eckberg, A. E. Traffic Characteristics of On-line Services. In *2nd IEEE Symposium on Computers and Communications* (July 1997), pp. 17–21.

[27] Chase, J. S., Anderson, D. C., Thakar, P. N., and Vahdat, A. M. Managing Energy and Server Resources in Hosting Centers. In *In Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP* (2001), pp. 103–116.

[28] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., and Zhao, F. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the USENIX NSDI'08* (Berkeley, CA, USA, 2008), USENIX Association, pp. 337–350.

[29] Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., and Gautam, N. Managing server energy and operational costs in hosting centers. *SIGMETRICS Perform. Eval. Rev. 33*, 1 (2005), 303–314.

[30] Duan, Z., Zhang, Z.-L., and Hou, Y. T. Service overlay networks: SLAs, QoS, and bandwidth provisioning. *IEEE/ACM Trans. Networking 11*, 6 (2003), 870–883.

[31] Duffield, N. G., Goyal, P., Greenberg, A., Mishra, P., Ramakrishnan, K. K., and van der Merwe, J. E. Resource management with hoses: point-to-cloud services for virtual private networks. *IEEE/ACM Trans. Networking 10*, 5 (2002), 679–692.

[32] Elnozahy, E. M., Kistler, M., and Rajamony, R. Energy-Efficient Server Clusters. In *Proc. of the 2nd Workshop on Power-Aware Computing Systems* (2002), pp. 179–196.

[33] Erlang, A. K. The Theory of Probabilities and Telephone Conversations. *Nyt Tidsskrift for Matematik B 20* (1909).

[34] Erlang, A. K. Solution of some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges. *Elektrotkeknikeren 13* (1917).

[35] Fan, X., Weber, W.-D., and Barroso, L. A. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture* (New York, NY, USA, 2007), ISCA '07, ACM, pp. 13–23.

[36] Filani, D., He, J., Gao, S., Rajappa, M., Kumar, A., Shah, R., and Naappan, R. Dynamic Data Center Power Management: Trends, Issues and Solutions. *Intel Technology Journal* (2008).

[37] Gleeson, B., Lin, A., Heinanen, J., Armitage, G., and Malis, A. A Framework for IP Based Virtual Private Networks. RFC 2764, 2000.

[38] Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization* (Washington, DC, USA, 2007), IISWC '07, IEEE Computer Society, pp. 171–180.

[39] Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev. 39*, 1 (2009), 68–73.

[40] Grottke, M., Mainkar, V., Trivedi, K. S., and Woolet, S. Response time distributions in networks of queues. In *Queueing Networks: A Fundamental Approach (R. Boucherie, N.V. Dijk (Eds.))*, Springer-Verlag. to appear.

[41] Guyton, J. D., and Schwartz, M. F. Locating nearby copies of replicated Internet servers. In *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1995), ACM, pp. 288–298.

[42] Hayes, B. Cloud Computing. *Communications of the ACM 51* (2008), 9–11.

[43] Iversen, V. Teletraffic Engineering and Network Planning. *manuscript for course 34340, Technical University of Denmark* (2006).

[44] Kalyanakrishnan, M., Iyer, R., and Patel, J. Reliability of Internet Hosts - A Case Study from the End User's Perspective. *Computer Communications and Networks, International Conference on 0* (1997), 418.

[45] Kangasharju, J., Ross, K. W., and Roberts, J. W. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications 24*, 2 (2001), 207 – 214.

[46] Kaufman, J. S. Blocking in a shared resource environment. *IEEE Trans. on Communications COM-29* (1981), 1474–1481.

[47] Kleinrock, L. *Theory, Volume 1, Queueing Systems*. Wiley, 1975.

[48] Kohavi, R. Practical guide to controlled experiments on the web: Listen to your customers not to the HiPPO. In *In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2007* (2007), pp. 959–967.

[49] Koomey, J. G. Estimating total power consumption by servers in the U.S. and the world. Tech. rep., Lawrence Derkley National Laboratory, Feb. 2007.

[50] Krishnan, P., Raz, D., and Shavitt, Y. The Cache Location Problem. *IEEE/ACM Trans. Networking 8* (2000), 568–582.

[51] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing 12*, 1 (2009), 1–15.

[52] Li, A., Yang, X., Kandula, S., and Zhang, M. CloudCmp: Shopping for a Cloud Made Easy. In *Proc. of 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)* (June 2010).

[53] Lipsky, L. *Queueing Theory: A Linear Algebraic Approach.* Springer Publishing Company, Incorporated, 2008.

[54] Liu, K. Y. K., Lui, J. C. S., and Zhang, Z.-L. Distributed Algorithm for Service Replication in Service Overlay Network. In *Proc. of 3rd IFIP-TC6 Networking Conference (Networking 2004)* (2004), pp. 1156–1167.

[55] Lu, T., and Chen, M. Simple and Effective Dynamic Provisioning for Power-Proportional Data Centers. http://arxiv.org/abs/1112.0442, Dec. 2011.

[56] Medhi, D. Some Results for Renewal Arrival to a Communication Link. *A. C. Borthakur and H. Choudhury, New Age International Limited* (1996), 85–107.

[57] Medhi, D., Choi, B.-Y., Scoglio, C., Song, S., and Dispensa, S. Agent-Based VPN Architecture: A Framework and the Optimal User Connectivity Problem (Static

Case). *15th International Conference on Advanced Computing and Communications (ADCOM)* (2007), 138–143.

[58] Medhi, J. *Stochastic Processess (3rd Edition).* New Age Science, 2009.

[59] Meng, X., Papas, V., and Zhang, L. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *INFOCOM* (2010).

[60] Moore, G. E. Cramming more components onto integrated circuits (Reprinted from Electronics, pg 114-117, April 19, 1965). *Proceedings of the IEEE 86*, 1 (1998), 82–85.

[61] Nagarajan (Ed)., A. Generic Requirements for Provider Provisioned Virtual Private Networks (PPVPN). RFC 3809, June 2004.

[62] Petrucci, V., Loques, O., and Mossé, D. A dynamic configuration model for power-efficient virtualized server clusters. 11th Brazilian Workshop on Real-Time and Embeded Systems, May 2009.

[63] Petrucci, V., Loques, O., and Mossé, D. Dynamic configuration support for power-aware virtualized server clusters. Tech. rep., 2009. Technical Report.

[64] Petrucci, V., Loques, O., and Mossé, D. Dynamic optimization of power and performance for virtualized server clusters, Technical Report, 2009.

[65] Pinherio, E., Bianchini, R., Carrera, E. V., and Heath, T. *Compilers and Operating Systems for Low Power.* Kluwer Academic Publishers, 2003, ch. Dynamic Cluster Reconfiguration for Power and Performance.

[66] Plaxton, C. G., Rajaraman, R., and Richa, A. W. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA'97)* (Newport, Rhode Island, 1997), pp. 311–320.

[67] Positive Networks. http://www.anx.com/content/solutions/managed-remote-access/positive-networks.

[68] Qian, H., Dispensa, S., and Medhi, D. Optimizing request denial and latency in an agent-based VPN architecture. In *IEEE/IFIP Network Operations and Management Symposium (NOMS) 2008* (Salvador, Brazil, April 2008), pp. 248–255.

[69] Qian, H., Dispensa, S., and Medhi, D. Balancing Request Denial Probability and Latency in an Agent-Based VPN Architecture. *IEEE Transactions on Network and Service Management 7*, 4 (2010), 282–295.

[70] Qian, H., Li, F., and Medhi, D. On energy-aware aggregation of dynamic temporal demand in cloud computing. In *COMSNETS* (2012), pp. 1–6.

[71] Qian, H., and Medhi, D. Estimating optimal cost of allocating virtualized resources with dynamic demand. In *Proceedings of the 23rd International Teletraffic Congress* (2011), ITC '11, ITCP, pp. 320–321.

[72] Qian, H., and Medhi, D. Server Operational Cost Optimization for Cloud Computing Service Providers over a Time Horizon. In *UNSENIX Hot'ICE 2011* (2011).

[73] Qian, H., Medhi, D., and Trivedi, K. S. A hierarchical model to evaluate quality of experience of online services hosted by cloud computing. In *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management, IM 2011, Dublin, Ireland, 23-27 May 2011* (2011), N. Agoulmine, C. Bartolini, T. Pfeifer, and D. O. Sullivan, Eds., IEEE, pp. 105–112.

[74] Qian, H., Surapaneni, C. S., Dispensa, S., and Medhi, D. Service management architecture and system capacity design for PhoneFactor&#8482;: a two-factor authentication service. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management* (Piscataway, NJ, USA, 2009), IM'09, IEEE Press, pp. 73–80.

[75] Qian, H., Surapaneni, R. S., Ray, M., Dispensa, S., and Medhi, D. DReaM-Cache: Distributed Real-Time Transaction Memory Cache to Support Two-Factor Authentication Services and its Reliability. In *IEEE/IFIP Network Operations and Management Symposium 2010* (2010).

[76] Raikov, D. On the Decomposition of Poisson Laws. *C. R. (Doklady) Academy of Sciences of URSS 14* (1937).

[77] Roberts, J. W. A service system with heterogeneous user requirements: application to multi-services telecommunications systems. *Performance of Data Communication Systems, and Their Applications, G. Pujolle (Ed.), North-Holland* (1981).

[78] Rosen, E., and Rekhter, Y. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, February 2006. Updated by RFCs 4577, 4684, 5462.

[79] Sahner, R., Trivedi, K., and Puliafito, A. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1995.

[80] Shaikh, A., Tewari, R., and Agrawal, M. On the Effectiveness of DNS-based Server Selection. In *Proc. of IEEE INFOCOM* (2001).

[81] Srikantaiah, S., Kansal, A., and Zhao, F. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems* (Berkeley, CA, USA, 2008), HotPower'08, USENIX Association, pp. 10–10.

[82] Sripanidkulchai, K., Maggs, B. M., and Zhang, H. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proc. of IEEE INFOCOM* (2003).

[83] Trivedi, K. S. *Probability and Statistics with Reliability Queuing and Computer Science Applications (2nd Edition)*. John Wiley & Sons, 2002.

[84] _____. EPA Report on Server and Data Center Energy Efficiency. U.S. Enviromental Protetion Agency, 2007. ENERGY STAR Program.

[85] _____. VMware Distributed Power Management Concepts and Use. http://www.vmware.com/files/pdf/DPM.pdf, 2009.

[86] Verma, A., Dasgupta, G., Nayak, T. K., De, P., and Kothari, R. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 conference on USENIX Annual technical conference* (Berkeley, CA, USA, 2009), USENIX'09, USENIX Association, pp. 28–28.

[87] Vishwanath, K. V., and Nagappan, N. Characterizing Cloud Computing Hardware Reliability. In *Proc. of 1st ACM Symposium on Cloud Computing* (June 2010).

[88] Wang, D., and Trivedi, K. Modeling User-Perceived Reliability Based on User Behavior Graphs. *International Journal of Reliability, Quality & Safety Engineering 16*, 4 (August 2009), 303–330.

[89] Wustenhoff, E. Serice Level Agreement in the Data Center. http://www-it.desy.de/common/documentation/cd-docs/sun/blueprints/0402/sla.pdf.

[90] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications 1*, 1 (2010), 7–18.

[91] Zhang, Z., Zhang, M., Greenberg, A., Charlie, Y., Ratul, H., and Christian, M. B. Optimizing Cost and Performance in Online Service Provider Networks. In *NSDI* (2010).

VITA

Haiyang Qian was born on February 11, 1981 in Binhai, Jiangsu Province, China. He attended Nanjing University of Science & Technology in September 1998 and graduated in June 2002 with a Bachelor degree in Communication Engineering. In 2002-2004, Mr. Qian was an R&D engineer at Nanjing Nari Telecommunication Corp.

Mr. Qian attended Technical University of Denmark in 2004 and received his Master of Science in Telecommunications in 2006. In August 2006, Mr. Qian joined the Ph.D. program in University of Missouri–Kansas City (UMKC) with Telecommunications and Computer Networking (TCN) and Electrical Engineering as his coordinating discipline and Co-discipline, respectively. Mr. Qian was awarded UMKC Chancellor's Doctoral Fellowships in 2009–2010 and 2010–2011. Mr. Qian received Best Student Poster award in the 23rd International Teletraffic Congress (ITC 2011) in San Francisco, CA. Upon completion his Ph.D. requirements, Mr. Qian plans to continue to research in the data center and computer networking domains.

Mr. Qian is a member of Institute of Electrical and Electronic Engineers (IEEE) and Association of Computing Machinery (ACM).