Resource Allocation for Multimedia Services over Cloud Computing

by

Xiaoming Nan

M. Sc., Beijing University of Posts and Telecommunications, Beijing, China, 2010

B. Sc., Xidian University, Xi'an, China, 2007

A Dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2015

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Resource Allocation for Multimedia Services over Cloud Computing

Doctor of Philosophy 2015

Xiaoming Nan

Electrical and Computer Engineering

Ryerson University

Abstract

Cloud-based multimedia application has emerged as a popular service, delivering on-demand media computing and storage to millions of users. Though widely deployed, the quality of service (QoS) in current cloud-based multimedia service is not satisfying, due to the varying user demands and strict response time requirements. This thesis investigates resource allocation approaches to improve QoS for cloud-based multimedia services.

A service model is desired to quantify the user demands and resource allocation. To meet this need, we propose a queueing model to characterize the cloud service process, based on which we investigate the response time minimization problem and the resource cost minimization problem in single-service scenario, multi-service scenario, and priority-service scenario, respectively.

Dynamic workload causes the unbalanced resource utilization and local congestion in multimedia cloud. To address this issue, we propose a two-time-scale resource configuration (TRC) scheme to dynamically allocate virtual machines (VMs) to adapt to varying

workload. Based on the TRC scheme, we solve the optimal VM configuration problems to minimize the resource cost or minimize the average response time for the single-site cloud scenario and the multi-site cloud scenario, respectively.

We propose optimal workload scheduling schemes at user level and task level, respectively. At user level, we optimize the workload assignment to minimize the response time or minimize the resource cost. At task level, we introduce a directed acyclic graph to model the precedence constraints among tasks, and then solve the execution time minimization problems for sequential structure, parallel structure, and mixed structure, respectively.

Cloud gaming is an emerging interactive multimedia service. However, current cloud gaming suffers from a high bandwidth consumption and a large response delay. We propose a hybrid streaming framework to provide a high quality cloud gaming experience. We solve the delay-rate-distortion (d-R-D) optimization problem to minimize the overall distortion under the bandwidth and response delay constraints.

Acknowledgements

I would like to first express my sincere gratitude to my supervisor, Dr. Ling Guan, for his invaluable guidance and support throughout my Ph.D. study. I deeply appreciate his insightful discussions, precious suggestions, and strict training in all aspects of my academic development, which will be my life long treasure. I am very grateful to my co-supervisor, Dr. Yifeng He, for his rigorous attitude toward research and great attentions on details. Dr. He has taught me various research approaches and given me insightful comments. I feel very fortunate to have Dr. Guan and Dr. He as my Ph.D. supervisors. Their dedication, diligence, and enthusiasm will continue motivating me in my future career.

I would also like to thank Dr. Baining Guo, Dr. Yan Lu, and Dr. Xun Guo, for their insightful ideas and discussions on cloud gaming project during my internship at Microsoft Research Asia. I have benefited immensely from their mentorship.

It is a blessing for me to join Ryerson Multimedia Research Laboratory (RML). I have gained both knowledge and friendship during my study at RML. I am grateful to all my colleagues at RML for their valuable support and help.

Last but not least, I would like to thank my parents for their unconditional support and encouragement. I am especially grateful to my beloved wife, Yi Liu, and my daughter, Stephanie Nan. Without their relentless love and support, I would never have been able to finish my thesis.

Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

Introduction

1.1 Motivation

Recent years have witnessed the rapid advances in cloud computing. According to the forecast from IDC [1], the worldwide revenue on public cloud services is expected to rise from $56.6 billion in 2014 to $127 billion in 2018, enjoying a five-year compound annual growth rate of 22.8%, which is about six times that of traditional IT industry. Due to the elastic and on-demand resource provisioning in cloud computing, numerous IT enterprises have moved their services to cloud side. Today, the most popular social network service, like Facebook [2], email service, like Gmail [3], online office service, like Microsoft Office Online [4], and on-demand gaming service, like OnLive [5], are all hosted on cloud data centers. Even the U.S. federal government has employed Amazon Web Service (AWS) cloud solutions in the Civilian Agencies and the Department of Defense to reduce IT maintenance cost and boost efficiencies [6]. The emergence of cloud computing has introduced a significant change to the traditional IT industry by making hardware and software as accessible services in daily life.

Cloud computing has been envisioned as the next generation computing paradigm. Compared with the conventional computing paradigms, a unique feature of cloud computing is to enable the rapid and elastic resource provisioning. In cloud data centers, a shared pool of servers are managed to provide on-demand computation, communication, and storage resources as utilities in a scalable manner [7]. To efficiently provide resources, virtualization techniques have been applied in cloud to package computation resources

into virtual machines (VMs). By managing VMs, cloud computing is able to provision or release resources in a fine granularity to meet service demands [8].

As the popularity of mobile devices, online multimedia services have been widely used in recent years, like photo and video sharing, online gaming, and media retrieval. Compared with the general Internet services, multimedia services typically demand intensive computation and high bandwidth, which impose burdens to client devices, especially to the resource-constrained thin devices. On the other hand, the workload intensity varies quickly in online multimedia service. In order to accommodate the burst of workload, multimedia service providers (MSPs) traditionally need to over-provision computation and bandwidth resources to guarantee the quality of service (QoS). However, over-provisioning is costly and ineffective, leading to low resource utilization in non-peaked period and a high resource cost.

The emergence of cloud computing provides an effective solution for multimedia services. Running multimedia applications on cloud eases users from the continuous hardware upgrade and software installation. In cloud-based multimedia services, the intensive computation tasks are executed on cloud servers, greatly reducing the hardware requirements on the user side. Furthermore, migrating multimedia services to cloud environment can benefit MSPs. The elastic and on-demand resource provisioning in cloud enables MSPs to serve millions of users at the same time with the quality assurance. In addition, leasing cloud resources can effectively reduce the overhead cost on dedicated servers. By dynamically configuring cloud resources, MSPs can enhance the resource utilization. As a result, an increasing number of MSPs have deployed services on cloud. For example, Netflix, one of the major internet media streaming providers, has moved its streaming services, retrieval engines, and media data to AWS public cloud to handle the ever-growing users [9].

Multimedia services bring new issues to cloud computing. First of all, multimedia services are delay-sensitive. Response time is one of the important QoS factors in cloud-based multimedia services, especially in the interactive multimedia services, like cloud gaming. But the workload in cloud varies rapidly. The occasional workload bursts will cause the suddenly peaked computation demands and accordingly extend the response time. Thus, it is necessary to develop a dynamic resource configuration scheme to meet response time requirements under the varying workload. Besides response time, another important concern for MSPs is the resource cost. MSPs lease a certain amount of VMs

from cloud computing providers. If the leased VMs are more than the real demands, resources cannot be fully utilized, causing the unnecessary waste. Conversely, if the allocated resources are less than the real demands, the response time will increase, which will degrade QoS and may even result in the loss of user access. Therefore, MSPs desire an economic resource allocation scheme to provide services at a low resource cost. Furthermore, the diversified multimedia services make the resource allocation more complicated. There exist different types of multimedia services, which have heterogeneous resource demands and QoS requirements. VMs are configured with different resource capacities. It is a challenge to satisfy all the QoS requirements by allocating VMs to each service. Therefore, MSPs require an efficient resource allocation scheme to optimally configure computation resources.

Considering the above mentioned issues, an effective resource allocation scheme is urgently needed. MSPs can benefit from the resource allocation scheme, which enables MSPs to provide satisfactory services at a low resource cost. Users can also benefit from the resource allocation scheme, which enables them to receive services with the high quality of experience (QoE). Driven by these demands, we, therefore, study the resource allocation for multimedia services over cloud computing in this thesis.

1.2 Objective

Cloud-based multimedia services involve three roles: *users*, *MSPs*, and *cloud computing providers*, which are illustrated in Figure 1.1. As customers, users send requests to cloud data center for the interested services. By leasing VMs from cloud computing providers, MSPs offer on-demand services. Once the requests enter cloud data center, a dispatcher will distribute the requests to the corresponding service. By monitoring the workload, a service model will quantitatively analyze the user demands, the QoS requirements, and the required cloud resources. Through the service modeling, the resource allocation module will allocate VMs to each service and dynamically adjust VMs to adapt to the varying workload. If the leased VMs cannot satisfy user demands, the resource allocation module will request on-demand VMs from cloud computing providers to guarantee the QoS. A workload scheduler distributes the incoming requests to VMs inside a specific service, so that all the allocated resources can be utilized in a balanced manner.

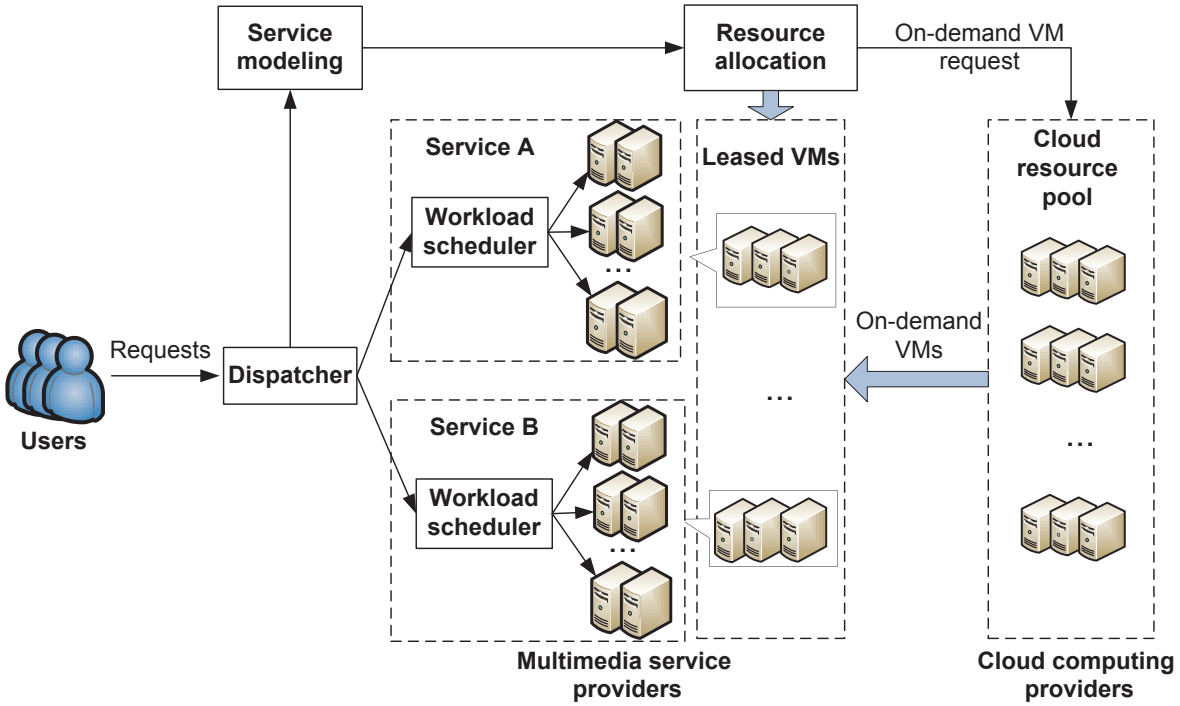


Figure 1.1: Illustration of cloud-based multimedia service.

In this thesis, we aim to optimally allocate cloud resources for multimedia services to maximize the QoE from the users' perspective or to minimize the total resource cost from the MSPs' perspective. In order to achieve this objective, we need to resolve a number of challenges, which will be presented in the following section.

1.3 Challenges in Multimedia Cloud Computing

Multimedia cloud computing not only presents new opportunities to users and MSPs, but also brings new challenges. Despite the fact that tremendous interests have been drawn from research communities, many challenges have not been fully addressed yet.

First of all, it is challenging to quantify the relationship among the allocated resources, user demands, and response time. In multimedia cloud, there is a trade-off between the response time and the resource cost. The under-provisioned resources would slow down services and deteriorate QoS, while the over-provisioned resources would reduce response time but lead to a high resource cost. Thus, it is a challenge for MSPs to precisely model

the dynamic user demands and effectively allocate resources. Many research work [10–12] apply utility functions to formulate the QoS performance and the allocated resources. However, the utility function cannot represent the service process. Moreover, multimedia services often have different priorities. For example, the real-time health monitoring service demands a higher priority than the on-demand video service. However, current resource management schemes [13, 14] are mainly designed for the first-come first-served (FCFS) service discipline. Therefore, an effective modeling method is needed to represent the cloud service process under different service disciplines.

Secondly, it is challenging to dynamically configure cloud resources to accommodate the varying workload. Most cloud computing providers [15–17] offer two different pricing plans, the *reservation plan* and the *on-demand plan*. In the reservation plan, MSPs make a onetime upfront payment for a VM instance and in turn receive a discount on the hourly rate. Compared to the reservation plan, the on-demand plan is more flexible. In the on-demand plan, a VM instance can be paid by usage without long-term commitment. MSPs need to determine the required number and class of VMs for each service. However, it is difficult to achieve an optimal VM allocation, since multimedia services demand different resources, while VMs are configured with heterogeneous capacities. In addition, even if the resources are initially allocated, the varying workload may cause the unbalanced resource utilization and the local congestion, where some services in cloud suffer from a burst of requests, while others may be under-loaded. Thus, an efficient resource allocation scheme should not only determine VM allocation at the initial stage, but also dynamically reconfigure VMs such that all resources can be fully utilized. Currently, the state-of-the-art resource allocation approach, used in Amazon Elastic Beanstalk [18], is designed based on the utilization. This approach can guarantee that the resource utilization is always lower than the threshold, but it cannot balance the resource utilizations among services.

Thirdly, it is challenging to optimally schedule the workload among the allocated cloud resources. There are two levels of workload scheduling in multimedia cloud. The first level is the user level scheduling, in which users' requests are distributed to different VMs according to the current load intensity at each VM. Since different VMs have different rates for processing user requests. It is difficult to schedule workload to each VM to achieve the best service performance. Compared to the user level scheduling, task level scheduling performs in a finer granularity. A complex multimedia service can be

regarded as a work flow, composed of a set of tasks. The objective of task level scheduling is to minimize the total execution time by optimally assigning tasks to different VMs. However, it is a challenge to determine the task assignment, since there are precedence constraints among tasks. Some tasks can run in parallel, while some other tasks must be processed sequentially. Some previous research work [19–21] employ the best-effort scheduling scheme. Workload in cloud will be assigned to the virtual server, which can provide the fastest or the earliest service. However, this scheme cannot guarantee the required QoS performance. Some researchers [22, 23] studied the scheduling problem by considering the CPU utilization at each cloud server to reduce the total power consumption. But these studies do not consider the heterogeneous resource demands.

Finally, it is challenging to guarantee the QoS for interactive multimedia services, like cloud gaming [5, 24, 25]. In cloud gaming, cloud servers fully or partially render game scenes and stream the encoded game frames to clients, while user's control events are captured and transmitted back to cloud servers. Most existing cloud gaming systems [5, 24, 25] transmit the encoded game frames using video streaming. However, video streaming demands a large bandwidth capacity to deliver the high-definition game frames. Some researchers used the attention model [26, 27] to allocate different rates to each content in proportional to its importance. However, the game attention model cannot precisely track player's attentions. The so-called less important object may be the focus of players. Simply encoding these contents with low rates may degrade the user experience. Therefore, it is still a challenge to provide users with a high quality cloud gaming experience under the bandwidth and delay constraints.

1.4 Main Contributions

In this thesis, we study the resource allocation for multimedia services over cloud computing. The main contributions in this thesis are illustrated in Figure 1.2.

- We propose a queueing model to represent cloud service process. Based on the queueing model, we theoretically study the equilibrium demands for scheduling, computation, and bandwidth resources in multimedia services and derive the relationship among the allocated resources, user demands, and response time. Moreover, we investigate the resource optimization in three different scenarios: the

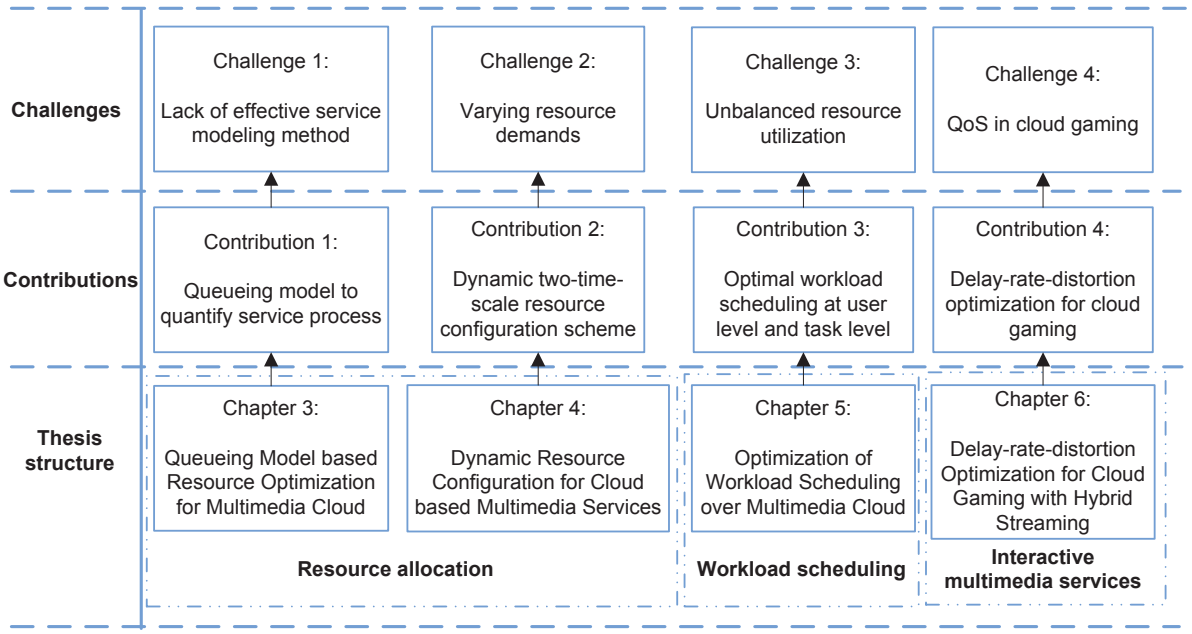


Figure 1.2: Illustration of main contributions in this thesis.

single-service scenario, the multi-service scenario, and the priority-service scenario. In each scenario, we formulate and solve the response time minimization problem and the resource cost minimization problem, respectively, to determine the optimal resource allocation.

- We propose a two-time-scale resource configuration (TRC) scheme to dynamically adjust VMs to deal with varying workload. In the TRC scheme, we first allocate resources in a mid-long time scale to determine the optimal number and class of VMs, and then reconfigure resources in a fine-grained time scale to address the dynamic demands. By considering different time scales, the proposed TRC scheme is able to flexibly adjust resources according to the real demands. We also investigate the resource cost minimization problem and the response time minimization problem in the single-site scenario and the multi-site scenario, respectively. Since the formulated problems are nonlinear integer programming, we develop heuristics to allocate resources in an efficient way.

- We study the workload scheduling in cloud to balance the resource utilization. Specifically, we investigate the user level workload scheduling and the task level

workload scheduling. At the user level, we optimize the workload assignment for the allocated VMs to achieve the minimal response time or the minimal resource cost. At the task level, we introduce a directed acyclic graph (DAG) to characterize the precedence constraints among tasks. Based on the DAG, we investigate the task scheduling for the sequential structure, parallel structure, and mixed structure services, respectively, and determine the optimal task assignment to minimize the total execution time.

- We propose a hybrid streaming framework to provide high quality cloud gaming experience under the bandwidth and response delay constraints. In the proposed framework, cloud server not only transmits the encoded game frames, but also progressively transmits graphics data, including geometry meshes and textures. The received graphics data are used to render a game frame, which provides extra reference candidate for encoding the following frames. As the accumulation of graphics data, the residual between the captured frame and the rendered graphics frame is decreasing, leading to the low encoding rate. Based on the proposed framework, we study the delay-rate-distortion optimization problem, where we optimize the rate allocation between video stream and graphics stream to achieve the minimal distortion under the given bandwidth and response delay constraints.

In this thesis, we formulate the resource allocation in cloud based multimedia services into constrained optimization problems, with an objective to minimize the response time or minimize the resource cost, subject to the resource constraints or QoS constraints. The summary of optimization problems is shown in Table 1.1.

1.5 Thesis Organization

The rest of this thesis is organized as follows.

Chapter 2 first gives an introduction of cloud computing, and then reviews the recent advances in resource allocation, workload scheduling, cloud gaming, and queueing theory.

Chapter 3 presents the queueing model based resource allocation. We investigate the resource allocation problems in single-service scenario, multi-service scenario, and priority-service scenario.

Optimization problem	**Problem type**	**Solution**
Minimize response time by optimizing resource allocation in single-service scenario in Eq. (3.4)	Convex optimization	Lagrange multiplier
Minimize resource cost by optimizing resource allocation in single-service scenario in Eq. (3.7)	Convex optimization	Lagrange multiplier
Minimize response time by optimizing resource allocation in multi-service scenario in Eq. (3.12)	Convex optimization	Interior point method
Minimize resource cost by optimizing resource allocation in multi-service scenario in Eq. (3.13)	Convex optimization	Interior point method
Minimize response time by optimizing resource allocation in priority-service scenario in Eq. (3.17)	Convex optimization	Interior point method
Minimize resource cost by optimizing resource allocation in priority-service scenario in Eq. (3.18)	Convex optimization	Interior point method
Minimize resource cost by optimizing VM allocation in single-site cloud in Eq. (4.4)	Nonlinear integer programming	Heuristic
Minimize average response time by optimizing VM reconfiguration in single-site cloud in Eq. (4.6)	Nonlinear integer programming	Heuristic
Minimize resource cost by optimizing VM allocation in multi-site cloud in Eq. (4.9)	Nonlinear integer programming	Heuristic
Minimize average response time by jointly optimizing workload assignment and VM reconfiguration in multi-site cloud in Eq. (4.13)	Mixed integer nonlinear programming	Heuristic
Minimize response time by optimizing workload scheduling weights in Eq. (5.1)	Convex optimization	Lagrange multiplier
Minimize resource cost by jointly optimizing workload scheduling weights and VM allocation in Eq. (5.3)	Mixed integer nonlinear programming	Heuristic
Minimize execution time by optimizing task-level scheduling for mixed structure in Eq. (5.6)	0-1 integer programming	Heuristic
Minimize total distortion by optimizing rates for video stream and graphics stream in Eq. (6.11)	Convex optimization	Lagrange multiplier

Table 1.1: Summary of optimization problems

Chapter 4 studies the dynamic resource configuration based on the proposed TRC scheme. The response time minimization problem and the resource cost minimization problem are examined in single-site cloud scenario and multi-site cloud scenario, respectively.

Chapter 5 investigates the workload scheduling problem. At the user level, the workload scheduling weights are optimized to minimize the response time or minimize

the overall cost. At the task level, the task assignments are determined to achieve the minimal execution time.

Chapter 6 proposes a hybrid streaming framework for cloud gaming. With the framework, we study the delay-rate-distortion optimization problem, where we optimize the rate allocation to achieve the minimal distortion under the bandwidth and response delay constraints.

Chapter 7 concludes the whole thesis and suggests the future research directions.

Chapter 2

Background

This chapter begins with a general overview of cloud computing, including the definition of cloud computing, service models, and QoS in multimedia cloud. Following that, we review the recent advances in the areas of cloud resource allocation, workload scheduling, cloud gaming, and queueing theory in resource allocation.

2.1 Cloud Computing Overview

2.1.1 Cloud Computing Definition

Cloud computing is becoming a popular term in various IT magazines, websites, and TV programs. The widely used analogy to explain cloud computing is the public utilities, like electricity and water. Just as the centralized utilities free individuals from the burdens of generating electricity or pumping water, cloud computing frees users from purchasing, installing, and maintaining hardware and software resources. By using the centralized computing resources on cloud, the cost on each individual user will be greatly reduced.

National Institute of Standards and Technology (NIST) gives a more concise and specific definition of cloud computing [7]: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The definition from NIST not only presents an objective concept of cloud computing,

but also describes a series of essential characteristics of cloud computing. First of all, cloud computing enables *on-demand* self-service. Users are able to freely request a variety of cloud resources, such as storage, bandwidth, and CPU. Secondly, *resource pooling* is used inside cloud data center to serve multiple users with a multi-tenant architecture. From users' perspective, they have no idea about the location of the provided resources, which is referred to as *resource location transparency*. Thirdly, *rapid elasticity* is enabled in cloud. Thus, capacities of allocated resources can be dynamically adjusted to meet the varying user demands. Fourthly, cloud service is *measurable*. Since cloud adopts the pay-as-you-go model to charge services by the resource usage, computing resources in cloud need to be monitored, controlled, and measured for both the providers and users. Lastly, cloud computing requires *broad network* to enable the access of users from heterogeneous client devices.

Cloud computing shares the similar features with *utility computing* and *grid computing*. Thus, people may confuse cloud computing with utility or grid computing. In the following, we will compare these related techniques to give a clear view of cloud computing. Utility computing is a service provisioning model, which is initialized in 1961 by John McCarthy on a speech at MIT centennial [28]. In utility computing, hardware and software resources are provided to users as needed, and the fees are charged based on the real usage. Utility computing utilizes the principle of consolidation, in which physical resources are shared by multiple users. For efficiency, computing resources are packaged as metered services, and users just pay a low initial cost to acquire the resources. The major concern in utility computing is the business model in resource metering. Besides utility computing, another similar technique is grid computing. Grid computing is defined as "a type of parallel and distributed system, which enables the sharing, selection, and aggregation of geographically distributed computing resources at runtime depending on their availability, capability, performance, and users' QoS requirements" [29]. Grid computing works as a super virtual mainframe composed of a cluster of loosely coupled computers, which are used to process computationally intensive applications. In grid, a task is generally divided into subtasks to be distributed to the grid computers. The major concern in grid computing is how to better utilize the parallel computing resources to process the large-scale task in a short time.

Based on the above descriptions, we can get the relationship among cloud computing, utility computing, and grid computing. Utility computing focuses on how computing

resources are packed as metered services to satisfy users' demands, but it has no specification about resource management and allocation. On the other hand, grid computing concerns the task division and parallel computing using the loosely coupled grid computers, but does not take the service charge into consideration. Currently, most grid platforms are sponsored by government and education organizations for research purposes. Compared to the two alternative paradigms, cloud computing is actually an evolution of utility and grid computing. Cloud uses the similar parallel processing technology as grid and the service business model as utility computing. Moreover, cloud computing develops its own features, such as virtualization techniques, centralized resource management, rapid elasticity, and high degree of scalability.

2.1.2 Service Models

According to the service provision, cloud computing can be categorized into three models, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [30]. The three models correspond to three levels, i.e. hardware level, system level, and application level [31]. As shown in Figure 2.1, IaaS is at the lowest hardware level, overlaid by PaaS at system level and SaaS at application level.

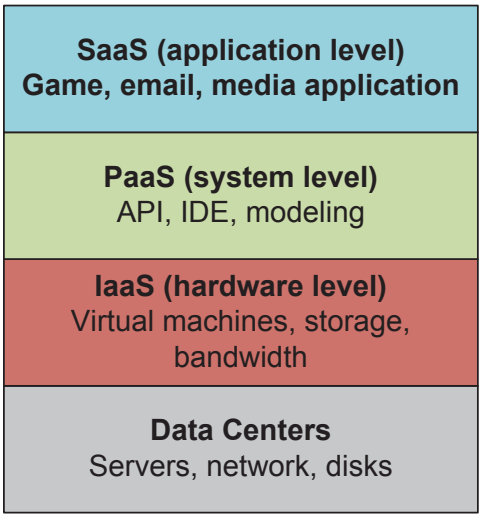


Figure 2.1: Illustration of cloud computing service models.

- IaaS delivers computing resources such as virtual machines, storage, connections,

and bandwidth to users. By renting the resources, users are able to set up any operating system or application environment. IaaS provides users with a high degree of flexibility, but users have to build the required environment from scratch. The provided infrastructure is virtualized, flexible, and scalable to meet the varying user demands. Classical IaaS examples include Amazon EC2 [15], , GoGrid [32], etc.

- PaaS delivers the high level integrated development environment to users. Users are able to utilize the provided application programming interface (API) and development environment to build, test, or host the customized applications. Compared to IaaS, PaaS users are free from the set up and maintenance of network, storage, and operating systems. Prime PaaS examples include Google App Engine [33] and Engine Yard [34].

- SaaS delivers software applications as services. In SaaS, users are able to access the provided software, which are executed on cloud servers, through a thin client interface, like web browser. The major benefit to users is that SaaS eliminates the investment in expensive local computers and the burdens of software installation at client side. As consumers, users do not concern about the hardware and application environment. They can send requests to providers and enjoy the interested services as they were performed on the local machine. Classical SaaS application include social media service, like Facebook [2], online office software service, like GMail [3] and Google Docs [35], and cloud gaming service, like OnLive [5] and CiiNow [25].

2.1.3 Cloud Computing Issues

Cloud computing not only brings benefits, but also introduces new issues, like data security, privacy, and energy consumption.

- **Data security**: In cloud environment, data are centralized on data center. From user's perspective, storing data remotely on cloud enables the flexible and on-demand access, but it also brings security threats towards the personal data. Despite the cloud data center is more powerful and reliable than personal computers, it still faces up with many attacks. For example, a collection of around 500 private

photos of various celebrities were posted on web site in August 2014 [36], and these images were obtained by hackers via a breach of Apple's iCloud service [37]. Besides data breach, data loss in cloud is another serious threat, which may occur when a hard drive died without a backup. For instance, Amazon Web Service suffered from a hardware failure on Eastern weekend in 2011 and lost some customers data due to operation errors [38]. To protect data security, researchers have developed some effective methods [39–41], including data encryption, third party monitoring mechanism, user data security assurance, etc.

- **Privacy**: As the increasing use of cloud services, like email, online photo sharing, and video broadcast, the privacy issue is becoming an important public concern. Once uploading the personal data to cloud data center, users shift the data control and privacy to service providers, who are able to monitor user's communications and access the personal data. To pursue commercial benefits, some cloud service providers may disclose user profile to third parties. Many countries have laws and regulations to protect cloud user privacy. For example, Canadian privacy laws mandates that any organization collecting personal information has a legal obligation to ensure that the privacy of their clients is protected. Some researchers [42, 43] proposed the privacy preserved digital identity management to address the privacy issue from the technical perspective.

- **Energy consumption**: With the expanded scale, energy consumption in cloud data centers is taken as a critical issue of both economic importance and environmental urgency. According to the study in [44], data centers in U.S. consumed electricity up to 100 billion kilowatt hours (kWh) in 2011, leading to a significant cost and emission of carbon dioxide. Many research work [45, 46] have been developed on the efficient energy use in cloud computing.

2.1.4 Multimedia Cloud Computing

Multimedia processing imposes new challenges to cloud computing, due to the special characteristics, like delay sensitivity, high computation intensity, and large bandwidth demands. Compared to the general services, multimedia processing not only demands computation resources from cloud, but also requires QoS guarantees. Therefore, simply

migrating multimedia services to cloud without considering QoS requirements may degrade the user experience. To better utilize cloud computing for multimedia services, Zhu *et al.* [47] proposed the concept of multimedia cloud computing, where the key concern was how cloud can effectively provide QoS-satisfied multimedia services for all users. Multimedia cloud computing is a new and open area, which attracts considerable attentions from researchers.

Some researchers started from architecture perspective to investigate how to support multimedia services on cloud. Zhu *et al.* [47] proposed a media edge cloud computing architecture, which physically placed computing servers at the edge of cloud in order to reduce transmission delays. Ferretti *et al.* [48] presented a cross-layer cloud architecture, which enabled wireless devices to seamlessly access multimedia services. Hui *et al.* [49] developed a layered media cloud architecture. The QoS control was implemented across three layers, namely media service layer, media overlay layer, and resource management layer. Specifically, the media service layer negotiated QoS requests with users, the media overlay layer ensured the security of data transmission, and the resource management layer adjusted resources to support QoS provision.

As the popularity of smart phones and tablets, multimedia cloud provides an ideal platform for mobile services. Miao *et al.* [50] proposed a collaborative rendering framework for cloud based free viewpoint video (FVV). In the framework, the cloud rendering was conducted during the stationary viewing time to achieve a high visual quality, while the local rendering was applied during the switch viewing time in order to conceal the interaction delay. Wang *et al.* [51] investigated cloud mobile gaming, an approach that enabled multi-player games on mobile devices. They proposed a rendering adaption technique to dynamically vary the complexity of graphics rendering according to the workload and computation constraints.

Multimedia cloud has also been widely used in social media field. Wu *et al.* [52] presented a mobile social TV framework, which provided transcoding services for different platforms and supported co-viewing experiences through chat exchanges among viewing users. Zhang *et al.* [53] presented a cloud-based interactive mobile visual search framework for social activities. In their work, mobile users captured a photo and circled the region-of-interest in the photo to identify the users' visual intent. Searching results were mapped to the sensory context, and related entities would be recommended to users.

2.2 Resource Allocation and Workload Scheduling

Cloud computing changes the traditional way we use computing resources. Users send requests to cloud, and virtual servers will be allocated to offer corresponding services. Since millions of requests are sent to cloud at the same time, resource allocation and workload scheduling become the critical issues. Resource allocation research focuses on how to effectively and economically manage cloud resources to serve all requests, while workload scheduling research investigates how to schedule workload to balance the load intensity at each server and pursue an efficiently distributed processing. In the following, we will summarize the recent advances in cloud resource allocation and workload scheduling.

2.2.1 Resource Allocation

The basis of cloud computing is virtualization techniques. Many researchers focus on the VM consolidation problem to exploit how to efficiently utilize available resources to consolidate multiple VMs. Nathuji *et al.* [54] proposed a hierarchical resource management system, where cloud resources are divided according to the local and global policies. On the local level, the system leveraged guest operating system resource management strategies. On the global level, the consolidation of VMs was handled by applying live migration. Verma *et al.* [55] formulated the problem of dynamic placement of applications in virtualized heterogeneous systems as a continuous optimization: at each time frame the placement of VMs was optimized to minimize resource consumption or maximize performance. Chaisiri *et al.* [56] studied the trade-off between the advance reservation and the on-demand resource allocation, and proposed a VM placement algorithm based on stochastic integer programming. The proposed algorithm minimized the total resource cost on an IaaS cloud.

Besides VM consolidation problem, the dynamic resource allocation is a challenging topic. Padala *et al.* [57] modeled and designed a feedback-driven resource control system based on the control theory. The system had a two-layered controller architecture, in which a utilization controller controlled the resource allocation for a single application and a decision controller controlled the resource allocation across multiple applications. Song *et al.* [58] presented a dynamic resource allocation approach according to the application

priorities. The proposed approach used machine learning to obtain utility functions for applications and defined priorities in advance. Cherkasova *et al.* [59] classified the application requests into disk- and CPU-bound ones and then constructed the dynamic scaling functions for the system capacity according to the prediction of the required resources. Lin *et al.* [60] developed a self-organizing model to adaptively manage cloud resources in the absence of centralized management control.

As an increasing number of multimedia services are deployed on cloud, there is an upsurge of research interests in resource allocation for cloud-based multimedia services. Miao *et al.* [50] studied the resource allocation for cloud-based FVV system. By jointly considering rate allocation among texture, depth, and channel rate, they investigated the bandwidth allocation problem to minimize the distortion under the given rate constraints. Wu *et al.* [61] presented a cloud-based video on demand (VoD) system, and investigated the relationship between the dynamic viewing behavior and the allocated cloud resources. Wang *et al.* [62] proposed a cloud-based live media streaming framework, which adaptively leased or adjusted cloud resources according to the temporal or spatial dynamics of user demands. Niu *et al.* [63] explored the cloud bandwidth allocation and pricing strategies. They stated that the utility of a tenant depended not only on the bandwidth usage, but also on the demand that was satisfied with performance guarantee. Thus, they formulated the cloud bandwidth pricing problem as the social welfare maximization problem, which was solved by distributed cutting-plane method. Yu *et al.* [64] proposed a video streaming workload prediction technique. With this method, multimedia workload can be inspected from the user access perspective.

2.2.2 Workload Scheduling

Workload scheduling, also known as load balancing, balances workload among all the available servers and enhances service quality. Hui *et al.* [65] presented a cloud load balancing technique to allocate the required resources to different applications in the shortest time. To achieve this goal, they considered not only the computation capacity of each server, but also the bandwidth capacity by assigning traffic through a set of parallel links. Srikantaiah *et al.* [23] studied the problem of workload scheduling for multi-tiered web applications in virtualized heterogeneous systems to minimize resource consumption, while meeting the performance requirements. To handle the optimization over multiple

resources, they proposed a heuristic to solve the multidimensional bin packing problem. The proposed heuristic was workload type and application dependent. Zhang *et al.* [66] explored the scheduling policy for the collaborative processing in mobile cloud computing. In this study, a mobile application was represented by a sequence of fine-grained tasks, and each of them was executed either on mobile device or on cloud side. They formulated the minimum energy consumption problem into a constrained shortest path problem on a directed acyclic graph. Chang *et al.* [67] applied an ant colony optimization algorithm for workload scheduling in distributed heterogeneous systems. They assumed each job as an ant and sent ants to search for resources. By using the ant algorithm, they minimized the makespan of a given set of jobs by balancing the entire system load. Tai *et al.* [68] proposed a burst workload balancer, which predicted the variations in user demands and accordingly adjusted the workload scheduling strategy.

Besides the user level workload scheduling, some researchers explore the internal relationship among tasks and study the task level scheduling. Zhu *et al.* proposed the cloud-based PhotoSynth service [69], where they explored the parallelization on the user level and the task level, respectively. On the user level parallelization, all tasks of a user were allocated to one server to process, but users can be served in parallel on cloud. On the task level parallelization, tasks of a user were distributed to different servers to compute in parallel. Yassa *et al.* [70] proposed a multi-objective scheduling approach for cloud. In this work, they formulated the workload scheduling problem as a constraint optimization problem with multiple objectives, including the execution time, resource cost, and energy consumption. The proposed technique allowed servers to work under different voltage supply levels by sacrificing the computation speed. Silberstein *et al.* [71] presented a scheduling algorithm by adapting the multi-level feedback queue approach in the distributed grid environment.

2.3 Cloud Gaming

Cloud gaming, also known as real-time game streaming, runs the computer games at cloud servers and streams the rendered frames back to users. Cloud gaming relieves users from endless hardware upgrades and solves the incompatibility issues between games and client machines. Currently, two different approaches are used in cloud gaming services: the

video streaming and the *graphics streaming*. The difference between these two approaches is if the rendering is executed on cloud servers or on clients. With video streaming, the cloud server renders the game scenes and streams the encoded game frames to client devices. This approach enables smooth game experience on thin devices. OnLive [5] and GaiKai [24] are typical cloud gaming providers using video streaming technique. Huang *et al.* [72,73] presented the first open source cloud gaming system, named *GamingAnywhere*. Cai *et al.* [74] presented the remote rendering game-as-a-service architecture and analyzed the system benefits and related research issues. Semsarzadeh *et al.* [75] utilized the game engine's information to accelerate the video encoding for cloud gaming service. In the proposed mechanism, the cloud server directly acquired object motion information from game engine and thus removed the motion estimation procedure in conventional video encoding. Cai *et al.* [76] explored the correlation of game videos for different players in the same game scene and proposed a cooperative video sharing scheme.

On the other hand, as the hardware upgrade on mobile devices, graphics streaming becomes an alternative approach. In graphics streaming, the required game models and textures are downloaded to client devices. After receiving the necessary graphics data, users can start playing the game, and the remaining graphics data will be gradually transmitted while users are playing the game. Since the actual games are running on local devices, graphics streaming enables a shorter interaction delay. Kalydo [77] adopts this approach for their online game service. Jurgelionis *et al.* [78] presented a distributed gaming project, which realized the video streaming protocol and 3D streaming protocol independently. Eisert *et al.* [79] proposed a graphics streaming framework for remote gaming in a local area network. They used a game server to intercept all OpenGL or DirectX commands and streamed the commands and graphics data to clients. With the local graphics chips, client devices can reconstruct the game scenes.

Besides streaming approaches, adaptive rate control is another challenge in cloud gaming. Wang *et al.* [51, 80] proposed a rendering adaption scheme, in which they adjusted bit rates to be transmitted by dynamically varying the rendering parameters. Shirmohammadi [81] presented a context aware approach to determine how important a game object was by building an importance matrix. Priority was assigned to each game object according to its importance to the player. Hemmati *et al.* [26] studied the video content adaption schemes for game scenes. In this work, they excluded the less important game objects from the game scene to reduce the processing time and rendering complexity

on cloud servers. Ahmadi *et al.* presented a game attention model in [27] for content adaption in encoding game frames.

Existing cloud gaming systems are mainly proprietary and closed, which imposes challenges for researchers to measure the QoS. Chen *et al.* [82, 83] studied the QoS in commercial cloud gaming systems and presented a suite of measurement techniques. By utilizing these techniques, they compared QoS in OnLive [5] and StreamMyGame [84]. Najaran *et al.* [85] studied the scalability of cloud-based multi-player first person shooter (FPS) gaming system. In this work, they employed a publish-subscribe system to reduce the aggregate bandwidth requirement. Hariri *et al.* [86] developed a network traffic model in FPS games. In this study, the network traffic was represented by the packet size and inter-packet interval, and a hierarchical hidden Markov model was constructed to provide packet level statistics.

2.4 Queueing Theory in Resource Allocation

Queueing theory is an important mathematical tool to model, analyze, and design service systems. A generic queueing process consists of the following steps: 1) customers enter a system, 2) customers wait in a line, 3) customers are served by a server, and then 4) customers leave the system. The performance metrics of a queueing system can be derived analytically. In queueing theory, a model is constructed to analyze the expected queue lengths and waiting time. Generally, a queueing system can be described by Kendall's Notation [87] $A/B/m/K/n/D$, where A is the distribution function of the inter arrival time, B is the distribution function of the server time, m is the number of servers, K is the capacity of the queueing system, n is the number of customer sources, and D is the service discipline, like FCFS or priority service discipline. Queueing theory can effectively model the service behavior in a system and evaluate the key performance metrics. As a result, it has been widely applied in communication networks, traffic engineering, airplane scheduling, computing, and design of shops or hospitals.

Queueing theory has also been used to resolve problems in resource allocation. Wu *et al.* [61] used a queueing network model to characterize the dynamic viewing behaviors in cloud VoD and investigated the equilibrium demands for streaming server capacity. Parandehgheibi *et al.* [88] investigated the rate allocation problem in a fading Gaussian

multiple-access channel with fixed transmission power. The objective in [88] was to maximize a concave utility function of expected rates for users. They first applied information theory and queueing theory to solve the problem, respectively. After that, they made a connection between the two approaches by finding that the information theoretic capacity region and queueing theoretic stability region were equivalent. Wu *et al.* [89] studied the latency problem in cloud gaming, and proposed an online access control algorithm to dispatch requests and allocate servers. Their objective was to reduce the operational cost while ensuring the user quality of experience (QoE) requirements. Towards this objective, they employed Lyapunov drift [90] for queueing networks to derive the online algorithm with proved upper bounds. Suresh Varma *et al.* [91] analyzed the performance of provided cloud services with queueing theory, and they derived the key parameters, like mean waiting time, throughput, and utilization in the cloud.

2.5 Chapter Summary

This chapter introduced cloud computing, from the cloud definition, service models, to multimedia cloud computing. After that, we reviewed the recent advances in cloud resource allocation and workload scheduling, and presented the previous research work for cloud gaming. Lastly, we discussed the utilization of queueing theory in solving resource allocation problems.

Existing research work is the foundation of our study. By reviewing the related literatures, we can find that some critical challenges, such as cloud service modeling, dynamic resource allocation, and delay-rate-distortion optimized cloud gaming, have not been completely addressed yet. These gaps inspire our research interests. In the following chapters, these problems will be studied.

Chapter 3

Queueing Model based Resource Optimization for Multimedia Cloud

In multimedia cloud, it is challenging to quantify the dynamic user demands and then allocate appropriate cloud resources to meet these demands. In this chapter, we introduce a queueing model to characterize the service process in multimedia cloud. Based on the queueing model, we investigate the response time minimization problem and the resource cost minimization problem in single-service scenario, multi-service scenario, and priority-service scenario, respectively.

3.1 Introduction

For multimedia service providers (MSPs), there are two major concerns: the response time and the resource cost. Generally, different multimedia services have different quality of service (QoS) requirements. Delay is an important QoS metric for many multimedia services such as image/video retrieval, video streaming, and cloud gaming. We focus on delay-sensitive cloud-based multimedia services, in which the response time is the dominant component in end-to-end service delay. The response time in cloud is defined as the duration from the time when the request arrives at the data center to the time when the service result completely departs from the data center. A lower response time means a faster service for the user's request. Therefore, it is important for the MSP to meet response time requirements for all users. Besides the response time, another

concern is the resource cost. In order to provide services, the MSP needs to rent required resources according to user demands. If the allocated resources are far more than the real demands, resources cannot be fully utilized, leading to the resource waste. Conversely, if the allocated resources are less than the demands, the QoS requirements cannot be guaranteed. Therefore, the MSP needs to quantify the user demands, and accordingly determine the optimal resource allocation.

However, it is challenging to satisfy these two concerns. Firstly, there exist different types of multimedia services, which have heterogenous resource demands. Secondly, different multimedia services have different requirements on response time. For example, compared to the image/video retrieval, the cloud gaming needs a lower response time such that users can enjoy a smooth and real-time interactive experience. The MSP should meet different response time requirements for various services. Thirdly, there is a trade-off between the response time and the resource cost. The under-provisioned resources would slow down the service and deteriorate QoS, while the over-provisioned resources would result in the unnecessary waste. Lastly, multimedia services may demand different priorities in the practical cloud. For example, some urgent applications, like real-time health monitoring, demand a higher priority to process abnormal events. However, current resource management schemes typically use the first-come first-served (FCFS) service discipline, which cannot adapt to services with different priorities. Therefore, a challenge for the MSP is how to optimize resources for the differentiated services.

To address above mentioned challenges, we propose the queueing model based resource allocation in this chapter. Our contributions can be summarized as follows.

1. We investigate the cloud resource allocation problem using queueing theory. Specifically, we introduce a queueing model to characterize the service process in multimedia cloud. We theoretically analyze the equilibrium demands for the schedule, computation, and bandwidth resources and derive the relationship between the allocated resource capacity and the response time.

2. Based on the proposed queueing model, we study the resource optimization in three different scenarios: the *single-service scenario*, the *multi-service scenario*, and the *priority-service scenario*. In each scenario, we formulate and solve the response time minimization problem and the resource cost minimization problem, respectively. Simulation results indicate that the proposed resource allocation schemes

can effectively allocate resources to achieve the minimal response time under a certain budget or provide satisfactory services at the minimal resource cost.

The remainder of this chapter is organized as follows. Section 3.2 presents the proposed system models. Based on the proposed models, we optimize resources for multimedia cloud in Section 3.3 under the single-service scenario, the multi-service scenario, and the priority-service scenario, respectively. Section 3.4 presents extensive performance evaluations. Finally, we summarize this work in Section 3.5.

3.2 System Models

In this section, we present the system models, including the data center architecture, the queueing model, and the cost model.

3.2.1 Data Center Architecture

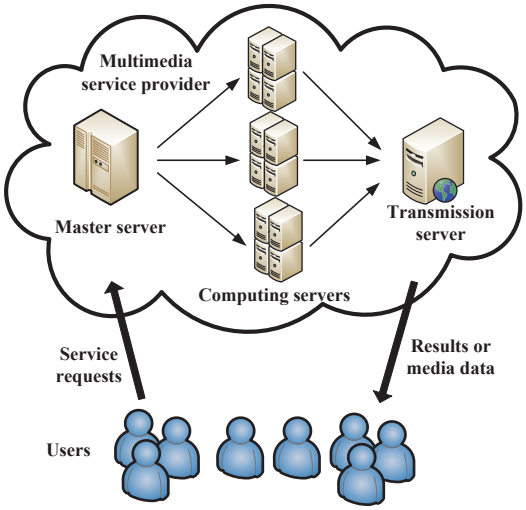


Figure 3.1: Illustration of multimedia cloud data center architecture.

Currently, most of clouds are built in the form of data centers [92]. Figure 3.1 illustrates the multimedia cloud data center architecture, which consists of the *master server*, a number of *computing servers*, and the *transmission server*. All servers are virtual clusters [93] composed by multiple VMs in order to provide more powerful resource

capacity. When requests arrive at the data center, the master server works as a scheduler, receiving all the requests and then distributing them to computing servers. The waiting time at the master server is dependent on the scheduling rate of the master server. The computing servers utilize the allocated computation resources to serve each request. The master server and computing servers are connected with high-speed communication links. Therefore, the internal latency for transferring requests is assumed to be negligible. The resource capacity at each computing server directly determines the processing speed. After processing, the service results or the requested media data will be sent back to users. The transmission server acts as a gateway node, which controls the overall traffic and directs the given packets to the specific destination. The allocated bandwidth at the transmission server determines how fast the results can be transmitted back to users.

The allocated cloud resources include the schedule resource, the computation resource, and the bandwidth resource. Owing to the varying workload, the resources have to be dynamically adjusted. Therefore, we divide the time domain into time slots with a fixed length, and the cloud resources will be dynamically allocated in each time slot. Suppose that there are N computing servers in the data center. At time slot t, we will denote by $S^{(t)}$ the allocated schedule resource in terms of the requests scheduled per second, $C_i^{(t)}$ the allocated computation resource at computing server i in terms of the instructions executed per second, and $B^{(t)}$ the allocated bandwidth resource in terms of the number of bits transmitted per second. Thus, the goal of resource allocation is to determine the optimal values of $S^{(t)}$, $C_i^{(t)}$ $(\forall i = 1, 2, \ldots, N)$, and $B^{(t)}$.

For each service in cloud, there are four parameters related to the resource demands: the average request arrival rate $\lambda^{(t)}$, the average task size F, the average result size D, and the upper bound of response time τ. Specifically, $\lambda^{(t)}$ represents the workload intensity of the service. When a service suffers from simultaneous bursts of requests, the workload intensity is heavy, and thus the MSP needs to allocate more resources. F indicates the computation complexity of the service. D represents the demands on bandwidth. In order to avoid the congestion in transmission, the bandwidth demands of each service should be satisfied. τ indicates the QoS requirement of the service. We will introduce a queueing model to analyze the relationship between dynamic demands and QoS requirements.

3.2.2 Queueing Model

We introduce a queueing model to analyze the service of requests. The queueing model is shown in Figure 3.2. The model consists of three concatenated queueing systems, which are the schedule queue, the computation queue, and the transmission queue. The master server maintains the schedule queue for all requests. Since two consecutive arriving requests may be sent from different users, the numbers of requests occur in non-overlapping intervals are independent random variables. According to Arlitt's study [94], the requests on a web server can be modeled by a Poisson process. Thus, we assume that the arrival of requests follows a Poisson process with an average of $\lambda^{(t)}$. The requests are scheduled to computing servers at the rate $S^{(t)}$ by the master server. Each computing server manages a corresponding computation queue to process requests. According to [95], subflows resulting from stochastically splitting a Poisson flow are still Poisson flows. Therefore, the arrivals of requests in the i-th computation queue can be modeled as a Poisson process with an average of $\lambda_i^{(t)}$. After processing, the service results are sent back to users through the transmission server at the rate B^t. During the cloud service, the resource availability is guaranteed by the service level agreement (SLA) and no request is dropped during the process. Therefore, the number of results is equal to the number of received requests.

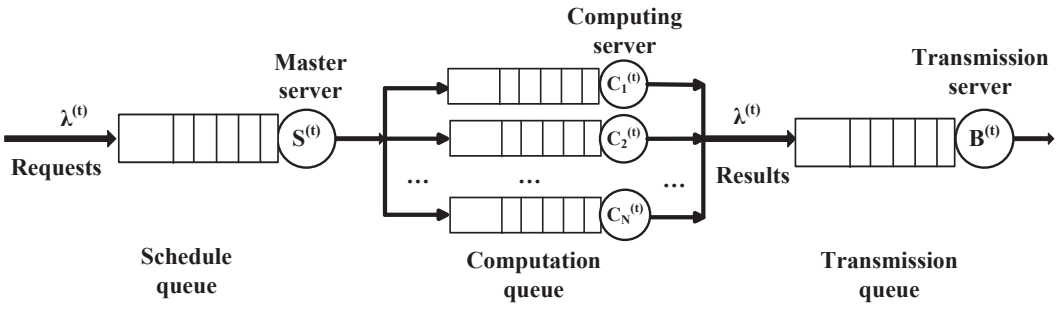


Figure 3.2: Queueing model of the data center in multimedia cloud.

We apply the proposed queueing model to represent the service process in multimedia cloud. The allocated cloud resources are represented by the capacity of the servers in each queueing system. For example, the allocated transmission resource is represented by the outbound bandwidth of the transmission server. It should be noted that the service rate in each queue must be larger than the request arrival rate to avoid overflow. This

constraint means that the actual allocated resources should satisfy the resource demands, otherwise the local congestion will occur in the service. In each queueing system, the waiting time of a job in the queue corresponds to the waiting time of the request for available resources, while the service time of the job represents the real processing time of the request. The total response time for a request in multimedia cloud is the sum of the waiting time and service time in the three concatenated queueing systems.

3.2.3 Cost Model

In order to deploy services, the MSP needs to rent required resources from the cloud computing provider. The resource cost in cloud is determined by the resource capacity and the occupied time. The more powerful resources will lead to a higher resource cost. The allocated cloud resources in our study include the schedule resource, the computation resource, and the bandwidth resource. We employ a linear function to model the relationship of the resource cost and the actual resource allocation. The total resource cost $\mathcal{C}^{tot(t)}$ at time slot t can be formulated as

$$\mathcal{C}^{tot(t)} = \left(\alpha S^{(t)} + \beta \sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)} \right) \bar{t}, \tag{3.1}$$

where $\bar{t}$ is the time slot length, α, β, and γ are price rates for the schedule resource, computation resource, and bandwidth resource, respectively. The linear cost model in Equation (3.1) has been justified by the numerical analysis on Amazon EC2 virtual machine (VM) instances [15].

3.3 Queueing Model based Resource Optimization

In this section, we will use the proposed queueing model to study the resource optimization problems for multimedia cloud. We consider three different scenarios, namely the single-service scenario, the multi-service scenario, and the priority-service scenario. In each scenario, we optimize cloud resources to minimize the response time or minimize the resource cost, respectively.

3.3.1 Single-service Scenario

We first study the single-service scenario, in which there is only one multimedia service provided by the MSP. All users send requests for the same service. Requests enter into the schedule queue first. The schedule queue can be modeled as an $M/M/1$ queueing system with a mean service rate $S^{(t)}$. In order to maintain a stable queue, $\lambda^{(t)} < S^{(t)}$ is required. The response time of a request in the schedule queue is given by [96] $T_{sing}^{sche(t)} = \frac{1/S^{(t)}}{1-\lambda^{(t)}/S^{(t)}}$. We denote by p_i the probability of a request assigned to the i-th computation queue for service. Thus, $\lambda_i^{(t)} = p_i\lambda_{(t)}$ and $\sum_{i=1}^{N} p_i = 1$. The service time at the computation queue depends on the allocated computation resource $C_i^{(t)}$ and the task size F. According to [97, 98], the execution time can be approximated as the exponential distribution with an average of $F/C_i^{(t)}$. The probability density function (PDF) of execution time can be given by $f(x) = \frac{C_i^{(t)}}{F} e^{-\frac{C_i^{(t)}}{F}x}$, $(x \geq 0)$. To avoid a congestion in the queue, the constraint $p_i\lambda^{(t)}F < C_i^{(t)}$ should be satisfied. Therefore, the response time in computation queue i is given by $T_{sing}^{comp(i)(t)} = \frac{F/C_i^{(t)}}{1-p_i\lambda^{(t)}F/C_i^{(t)}}$. Considering that a request may be assigned to any of the computation queues, the equilibrium response time is formulated by $T_{sing}^{comp(t)} = \sum_{i=1}^{N} p_i T_{sing}^{comp(i)(t)} = \sum_{i=1}^{N} \frac{p_i F/C_i^{(t)}}{1-p_i\lambda^{(t)}F/C_i^{(t)}}$. As no request is dropped in multimedia cloud, the average arrival rate of results at the transmission queue is $\lambda^{(t)}$. The transmission time is a random variable which depends on the result size D and the bandwidth resource $B^{(t)}$. From [61,99], the PDF of the transmission time can be approximately formulated as $f(x) = \frac{B^{(t)}}{D} e^{-\frac{B^{(t)}}{D}x}$, $(x \geq 0)$. To maintain the transmission queue stable, the transmission demands should be no more than the bandwidth capacity. Thus, $\lambda^{(t)}D < B^{(t)}$ is required. The response time in the transmission queue is given by $T_{sing}^{tran(t)} = \frac{D/B^{(t)}}{1-\lambda^{(t)}D/B^{(t)}}$.

Based on the above analysis, the equilibrium response time in the single-service scenario is the sum of response time in the three phases, which can be formulated as

$$\begin{aligned} T_{sing}^{tot(t)} &= T_{sing}^{sche(t)} + T_{sing}^{comp(t)} + T_{sing}^{tran(t)} \\ &= \frac{1/S^{(t)}}{1-\lambda^{(t)}/S^{(t)}} + \sum_{i=1}^{N} \frac{p_i F/C_i^{(t)}}{1-p_i\lambda^{(t)}F/C_i^{(t)}} + \frac{D/B^{(t)}}{1-\lambda^{(t)}D/B^{(t)}}. \end{aligned} \tag{3.2}$$

The total resource cost in the single-service scenario can be formulated as

$$\mathcal{C}_{sing}^{tot(t)} = \left(\alpha S^{(t)} + \beta \sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)} \right) \bar{t}. \tag{3.3}$$

Response Time Minimization Problem

For cloud based multimedia services, reducing the response time can greatly reduce the end-to-end delay and improve the quality of user experience. However, there is a trade-off between the response time and the resource allocation. We therefore formulate the *response time minimization problem*, which is stated as: to minimize the response time in the single-service scenario by jointly optimizing the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraint in each queueing system and the resource cost constraint. Mathematically, the problem can be formulated as

$$\begin{aligned} \underset{\{S^{(t)}, C_1^{(t)}, \cdots, C_N^{(t)}, B^{(t)}\}}{\text{Minimize}} \quad & T_{sing}^{tot(t)} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & p_i \lambda^{(t)} F < C_i^{(t)}, \quad \forall i = 1, \ldots, N, \\ & \lambda^{(t)} D < B^{(t)}, \\ & \left(\alpha S^{(t)} + \beta \textstyle\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)} \right) \bar{t} \leq \mathcal{C}_{max}, \end{aligned} \tag{3.4}$$

where $T_{sing}^{tot(t)}$, given by Equation (3.2), is the total response time, and $\mathcal{C}_{max}$ is the upper bound of resource cost. The objective function in Equation (3.4) is the equilibrium response time. The first, second, and third constraints represent the queueing stability requirements for each queue, and the constraint $\left(\alpha S^{(t)} + \beta \sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)} \right) \bar{t} \leq \mathcal{C}_{max}$ represents the budget constraint.

Lemma 3.1. *The optimization problem in Equation (3.4) is a convex optimization problem.*

Proof. The constraint functions in Equation (3.4) are all real-valued linear functions. Therefore, they are all convex [100]. The objective function in Equation (3.4) can be

expressed as

$$T_{sing}^{tot(t)} = \frac{1}{S^{(t)} - \lambda^{(t)}} + \sum_{i=1}^{N} \frac{p_i F}{C_i^{(t)} - p_i \lambda^{(t)} F} + \frac{D}{B^{(t)} - \lambda^{(t)} D}$$
$$= l\left(S^{(t)}\right) + \sum_{i=1}^{N} g_i\left(C_i^{(t)}\right) + h\left(B^{(t)}\right),$$

where $l(\cdot)$, $g_i(\cdot)$, and $h(\cdot)$ are single-variable functions. The second derivative of $l\left(S^{(t)}\right)$ is derived as $l''\left(S^{(t)}\right) = \frac{\mathrm{d}^2}{\mathrm{d}(S^{(t)})^2}\left(\frac{1}{S^{(t)}-\lambda^{(t)}}\right) = \frac{2}{\left(S^{(t)}-\lambda^{(t)}\right)^3}$, which is positive in the feasible region $S^{(t)} > \lambda^{(t)}$. Therefore, $l\left(S^{(t)}\right)$ is convex when $S^{(t)} > \lambda^{(t)}$. The second derivative of $g_i\left(C_i^{(t)}\right)$ is given by $g_i''\left(C_i^{(t)}\right) = \frac{\mathrm{d}^2}{\mathrm{d}(C_i^{(t)})^2}\left(\frac{p_i F}{C_i^{(t)}-p_i\lambda^{(t)}F}\right) = \frac{2p_i F}{\left(C_i^{(t)}-p_i\lambda^{(t)}F\right)^3}$, which is positive in the feasible region $C_i^{(t)} > p_i\lambda^{(t)}F$. Therefore, $g_i\left(C_i^{(t)}\right)$ is convex. The second derivative of $h\left(B^{(t)}\right)$ is given by $h''\left(B^{(t)}\right) = \frac{\mathrm{d}^2}{\mathrm{d}(B^{(t)})^2}\left(\frac{D}{B^{(t)}-\lambda^{(t)}D}\right) = \frac{2D}{\left(B^{(t)}-\lambda^{(t)}D\right)^3} > 0$ in feasible region $B^{(t)} > \lambda^{(t)}D$. Therefore, $h\left(B^{(t)}\right)$ is convex. Thus, $l\left(S^{(t)}\right)$, $g_i\left(C_i^{(t)}\right)$ and $h\left(B^{(t)}\right)$ are convex functions in feasible region. Since the objective function in Equation (3.4) is the summation of $l\left(S^{(t)}\right)$, $g_i\left(C_i^{(t)}\right)$ and $h\left(B^{(t)}\right)$, it is convex [100]. Therefore, the optimization problem in Equation (3.4) is a convex optimization problem since the objective function and all constraint functions are convex in feasible region. □

Since the optimization problem in Equation (3.4) is a convex optimization problem, Lagrange multiplier method can be used to find the optimal solution [100]. Observing Equation (3.4), we can find that all constraints are inequality constraints. We derive Equation (3.4) as following

$$\begin{aligned}
&\underset{\{S^{(t)},C_1^{(t)},\cdots,C_N^{(t)},B^{(t)}\}}{\text{Minimize}} && T_{sing}^{tot(t)} \\
&\text{subject to} && \\
& && \lambda^{(t)} - S^{(t)} < 0, \\
& && p_i\lambda^{(t)}F - C_i^{(t)} < 0, \quad \forall i = 1,\ldots,N, \\
& && \lambda^{(t)}D - B^{(t)} < 0, \\
& && \left(\alpha S^{(t)} + \beta\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)}\right)\bar{t} - \mathcal{C}_{max} \leq 0.
\end{aligned}$$

Thus, the Lagrange function can be formulated as

$$L\left(S^{(t)}, C_1^{(t)}, C_2^{(t)}, \ldots, C_N^{(t)}, B^{(t)}, \mu_1, \mu_{21}, \mu_{22}, \ldots, \mu_{2N}, \mu_3, \mu_4\right) = T_{sing}^{tot(t)} + \mu_1\left(\lambda^{(t)} - S^{(t)}\right)$$
$$+ \sum_{i=1}^{N} \mu_{2i}\left(p_i\lambda^{(t)}F - C_i^{(t)}\right) + \mu_3\left(\lambda^{(t)}D - B^{(t)}\right) + \mu_4\left((\alpha S^{(t)} + \beta\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)})\bar{t} - \mathcal{C}_{max}\right),$$

where $\mu_1,\ \mu_{2i}\ (\forall i = 1, \ldots, N),\ \mu_3$, and μ_4 are Lagrange multipliers. The optimal solution $S^{(t)*},\ C_i^{(t)*}\ (\forall i = 1, \ldots, N),\ B^{(t)*}$ exists such that the following conditions can be satisfied simultaneously.

$$\begin{cases} \dfrac{\partial}{\partial S^{(t)}} L\left(S^{(t)}, C_i^{(t)}, B^{(t)}, \mu_1, \mu_{2i}, \mu_3, \mu_4\right) = 0, \quad \forall i = 1, 2, \ldots, N \\ \dfrac{\partial}{\partial C_i^{(t)}} L\left(S^{(t)}, C_i^{(t)}, B^{(t)}, \mu_1, \mu_{2i}, \mu_3, \mu_4\right) = 0, \quad \forall i = 1, 2, \ldots, N \\ \dfrac{\partial}{\partial B^{(t)}} L\left(S^{(t)}, C_i^{(t)}, B^{(t)}, \mu_1, \mu_{2i}, \mu_3, \mu_4\right) = 0, \quad \forall i = 1, 2, \ldots, N \\ \lambda^{(t)} - S^{(t)} < 0, \\ p_i\lambda^{(t)}F - C_i^{(t)} < 0, \quad \forall i = 1, 2, \ldots, N \\ \lambda^{(t)}D - B^{(t)} < 0, \\ \left(\alpha S^{(t)} + \beta\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)}\right)\bar{t} - \mathcal{C}_{max} \leq 0, \\ \mu_1\left(\lambda^{(t)} - S^{(t)}\right) = 0, \\ \mu_{2i}\left(p_i\lambda^{(t)}F - C_i^{(t)}\right) = 0, \quad \forall i = 1, 2, \ldots, N \\ \mu_3\left(\lambda^{(t)}D - B^{(t)}\right) = 0, \\ \mu_4\left(\left(\alpha S^{(t)} + \beta\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)}\right)\bar{t} - \mathcal{C}_{max}\right) = 0, \\ \mu_1,\ \mu_{2i},\ \mu_3,\ \mu_4 \geq 0, \quad \forall i = 1, 2, \ldots, N. \end{cases} \tag{3.5}$$

By solving the above equations, we can get the optimal analytical solution to the optimization problem in Equation (3.4). The detailed derivation is given in Appendix A.1

and the optimal solution is given as follows.

$$
\begin{aligned}
S^{(t)*} &= \frac{\mathcal{C}_{max} - (\alpha + \beta F + \gamma D)\,\lambda^{(t)}\bar{t}}{\left(\sqrt{\alpha} + \sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i} + \sqrt{\gamma D}\right)\sqrt{\alpha}\bar{t}} + \lambda^{(t)}, \\
C_i^{(t)*} &= \frac{\left(\mathcal{C}_{max} - (\alpha + \beta F + \gamma D)\,\lambda^{(t)}\bar{t}\right)\sqrt{p_i F}}{\left(\sqrt{\alpha} + \sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i} + \sqrt{\gamma D}\right)\sqrt{\beta}\bar{t}} + p_i\lambda^{(t)}F, \quad \forall i = 1, 2, \ldots, N \\
B^{(t)*} &= \frac{\left(\mathcal{C}_{max} - (\alpha + \beta F + \gamma D)\lambda^{(t)}\bar{t}\right)\sqrt{D}}{\left(\sqrt{\alpha} + \sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i} + \sqrt{\gamma D}\right)\sqrt{\gamma}\bar{t}} + \lambda^{(t)}D.
\end{aligned}
\tag{3.6}
$$

It should be pointed out that the optimal analytical solution (3.6) is obtained under the schedule probability settings p_i, $(\forall i = 1, \ldots, N)$. The response time is dependent on the schedule probability settings. The following theorem gives the schedule probability setting to achieve the maximal response time or the minimal response time, respectively.

Theorem 3.1. *Given the response time minimization problem in Equation (3.4) based on the schedule probabilities p_i $(\forall i = 1, \ldots, N)$, we have the following two statements: 1) when p_i $(\forall i = 1, \ldots, N)$ is 1 and all other probabilities $p_{i'}$ $(i' \neq i,\ i' = 1, \ldots, N)$ are 0, we can achieve the minimal response time $T_{sing}^{tot(t)(min)} = \frac{\left(\sqrt{\alpha}+\sqrt{\beta F}+\sqrt{\gamma D}\right)^2}{\frac{\mathcal{C}_{max}}{\bar{t}} - \left(\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)}\right)}$; 2) when schedule probabilities are all equal, i.e. $p_1 = p_2 = \cdots = p_N = \frac{1}{N}$, we suffer from the maximal response time $T_{sing}^{tot(t)(max)} = \frac{\left(\sqrt{\alpha}+\sqrt{N}\sqrt{\beta F}+\sqrt{\gamma D}\right)^2}{\frac{\mathcal{C}_{max}}{\bar{t}} - (\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)})}$.*

The proof of Theorem 3.1 is given in Appendix A.2. Theorem 3.1 reveals that using one computation queue in the single-service scenario can achieve the lower response time than using multiple computation queues. The reason behind the theorem is that the single computation queue can fully utilize the available resources. For multiple computation queues, the scheduled demands must accurately match the allocated resources, otherwise it may result in the unbalanced resource utilization and deteriorate QoS. We verify Theorem 3.1 in the following example. The number of computing servers is set as 3, $\mathcal{C}_{max}$ is set as \$5000, and λ is 150 requests/s. Figure 3.3 shows the response time with different schedule probabilities. In Figure 3.3, p_1, p_2, and p_3 are all in the range of $[0, 1]$. If p_1 and p_2 are given, p_3 can be determined by $1 - p_1 - p_2$. From Figure 3.3, we can see that the response time reaches the minimal value 0.0183 seconds when (p_1, p_2, p_3) are at $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$, while it reaches the maximal value 0.0337 seconds

at $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, which justifies the statements in Theorem 3.1. Moreover, when p_2 is fixed, the maximal response time is acquired at $p_1 = p_3 = \frac{1-p_2}{2}$. Similar conclusions can be acquired when p_1 or p_3 is fixed.

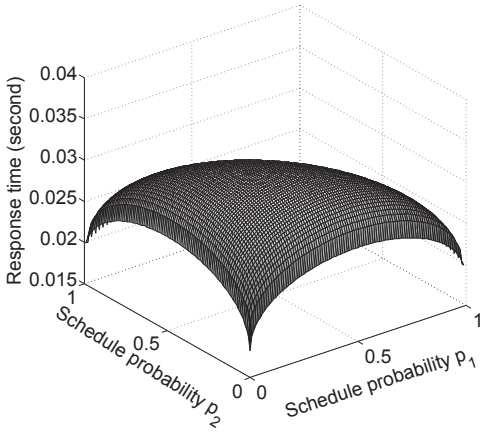


Figure 3.3: Response time with different schedule probability settings.

Resource Cost Minimization Problem

From the economic perspective, the MSP would always like to provide satisfactory services at the minimal resource cost. Therefore, we formulate the *resource cost minimization problem*, which can be stated as: to minimize the total resource cost in the single-service scenario by jointly optimizing the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraint in each queueing system and the response time constraint. Mathematically, the problem can be formulated as follows.

$$\begin{aligned} \underset{\{S^{(t)}, C_1^{(t)}, \cdots, C_N^{(t)}, B^{(t)}\}}{\text{Minimize}} \quad & \mathcal{C}_{sing}^{tot(t)} = \left(\alpha S^{(t)} + \beta \textstyle\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)}\right) \bar{t} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & p_i \lambda^{(t)} F < C_i^{(t)}, \quad \forall i = 1, \ldots, N, \\ & \lambda^{(t)} D < B^{(t)}, \\ & T_{sing}^{tot(t)} \leq \tau, \end{aligned} \tag{3.7}$$

where $T_{sing}^{tot(t)}$ is given by Equation (3.2) and τ is the upper bound of the response time. Similarly, we employ the Lagrange multiplier method [100] to solve the optimization problem in Equation (3.7), and get the optimal analytical solution as follows.

$$
\begin{aligned}
S^{(t)*} &= \frac{\sqrt{\alpha}+\sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i}+\sqrt{\gamma D}}{\sqrt{\alpha}\tau}+\lambda^{(t)}, \\
C_i^{(t)*} &= \frac{\left(\sqrt{\alpha}+\sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i}+\sqrt{\gamma D}\right)\sqrt{p_i F}}{\sqrt{\beta}\tau}+p_i\lambda^{(t)}F, \quad \forall i=1,\ldots,N, \\
B^{(t)*} &= \frac{\left(\sqrt{\alpha}+\sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i}+\sqrt{\gamma D}\right)\sqrt{D}}{\sqrt{\gamma}\tau}+\lambda^{(t)}D.
\end{aligned}
\tag{3.8}
$$

The optimal analytical solution (3.8) is also obtained under the schedule probability settings p_i ($\forall i = 1, 2, \ldots, N$). In order to investigate the relationship between the resource cost and the schedule probability settings, we derive the following theorem.

Theorem 3.2. *Given the resource cost optimization problem in Equation (3.7), we have the following two statements: 1) when schedule probability p_i ($\forall i = 1, \ldots, N$) is 1 and all other probabilities $p_{i'}$ ($i' \neq i, i' = 1, \ldots, N$) are 0, we can achieve the minimal resource cost $\mathcal{C}_{sing}^{tot(t)(min)} = \left(\frac{(\sqrt{\alpha}+\sqrt{\beta F}+\sqrt{\gamma D})^2}{\tau}+(\alpha+\beta F+\gamma D)\lambda^{(t)}\right)\bar{t}$; 2) when schedule probabilities are all equal, i.e. $p_1 = p_2 = \cdots = p_N = \frac{1}{N}$, we suffer from the maximal resource cost $\mathcal{C}_{sing}^{tot(t)(max)} = \left(\frac{(\sqrt{\alpha}+\sqrt{N}\sqrt{\beta F}+\sqrt{\gamma D})^2}{\tau}+(\alpha+\beta F+\gamma D)\lambda^{(t)}\right)\bar{t}$.*

We prove the Theorem 3.2 in Appendix A.3. Similar to Theorem 3.1, Theorem 3.2 indicates that under the same QoS constraint, the resource cost of using one computation queue in the single-service scenario is lower than that of using multiple computation queues. We verify Theorem 3.2 in a numerical example. The number of computing servers is 3, and the response time upper bound τ is set as 0.1 seconds. The other parameters are configured the same as those in the example for Theorem 3.1. Figure 3.4 shows the resource cost with different schedule probabilities. From Figure 3.4, we can find that the minimal resource cost \$2987.8 is achieved when (p_1, p_2, p_3) is equal to $(1, 0, 0)$, $(0, 1, 0)$ or $(0, 0, 1)$, while the maximal resource cost \$3368.3 is reached when (p_1, p_2, p_3) is $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The example verifies Theorem 3.2.

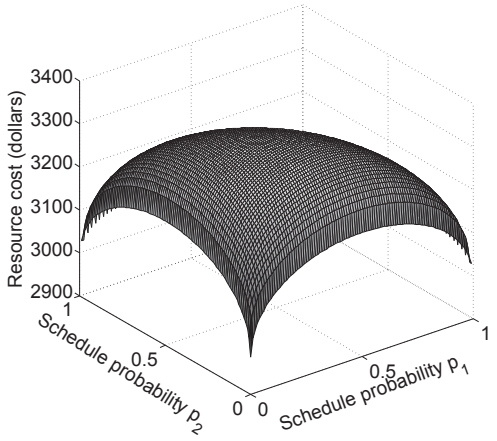


Figure 3.4: Total resource cost with different schedule probability settings.

3.3.2 Multi-service Scenario

In this subsection, we extend our study to the multi-service scenario, in which multiple services are provided in cloud. Specifically, different services have different processing procedures, different task sizes and result sizes, as well as different requirements on response time. Suppose that there are M classes of services deployed by the MSP. Each service requires a specific processing procedure. According to Theorem 3.1 and Theorem 3.2, using one computation queue to serve one class of requests performs better than using multiple computation queues. As a result, we employ M computation queues to provide M services in multi-service scenario, in which computation queue i is used to serve class-i requests. In cloud environment, resources can be on-demand provisioned or released. The number of computation queues can be dynamically changed according to the number of services.

Suppose that the request arrivals for each class follow a Poisson process with average arrival rates $\lambda_1^{(t)}, \lambda_2^{(t)}, \ldots, \lambda_M^{(t)}$, respectively. According to Poisson composition property, the total arrivals of the requests still follow a Poisson process with an average $\lambda^{(t)} = \sum_{i=1}^{M} \lambda_i^{(t)}$. Thus, the schedule queue is modeled as an $M/M/1$ queueing system. The response time in the schedule queue is $T_{mult}^{sche(t)} = \frac{1/S^{(t)}}{1-\lambda^{(t)}/S^{(t)}}$. For class-$i$ service, the average task size is denoted by F_i, and the execution time is a random variable, which follows exponential distribution [97,98] with an average of $F_i/C_i^{(t)}$. The response time in computation queue i is given by $T_{mult}^{comp(i)(t)} = \frac{F_i/C_i^{(t)}}{1-\lambda_i^{(t)}F_i/C_i^{(t)}}$. We also consider the average

response time for all requests in the computation phase, which is formulated as $T_{mult}^{comp(t)} = \sum_{i=1}^{M} \frac{\lambda_i^{(t)}}{\lambda^{(t)}} T_{mult}^{comp(i)(t)} = \sum_{i=1}^{M} \frac{\lambda_i^{(t)} F_i / C_i^{(t)}}{\lambda^{(t)}(1-\lambda_i^{(t)} F_i / C_i^{(t)})}$. After processing, all service results are sent to the transmission queue. Different services have different result sizes, leading to different transmission time. The transmission queue can be viewed as a queueing system in which customers are grouped into a single arrival stream and the service distribution is a mixture of M exponential distributions. Therefore, the service time follows the hyper-exponential distribution [96]. The transmission queue is modeled as an $M/H_M/1$ queueing system, where H_M represents the hyperexponential-M distribution. The response time of the $M/H_M/1$ queue can be derived from $M/G/1$ queue according to queueing theory [96]. The response time in the transmission queue is given by $T_{mult}^{tran(t)} = \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i^2}{(B^{(t)})^2 - B^{(t)} \sum_{i=1}^{M} \lambda_i^{(t)} D_i} + \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i}{\lambda^{(t)} B^{(t)}}$. To ensure that the transmission service can be completed, $\sum_{i=1}^{M} \lambda_i^{(t)} D_i < B^{(t)}$ is required.

Based on the above analysis, we can get the response time for class-i service as

$$\begin{aligned} T_{mult}^{tot(i)(t)} &= T_{mult}^{sche^{(t)}} + T_{mult}^{comp(i)(t)} + T_{mult}^{tran(t)} \\ &= \frac{1/S^{(t)}}{1 - \lambda^{(t)}/S^{(t)}} + \frac{F_i/C_i^{(t)}}{1 - \lambda_i^{(t)} F_i / C_i^{(t)}} + \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i^2}{(B^{(t)})^2 - B^{(t)} \sum_{i=1}^{M} \lambda_i^{(t)} D_i} \\ &\quad + \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i}{\lambda^{(t)} B^{(t)}}. \end{aligned} \tag{3.9}$$

Furthermore, the average response time in the multi-service scenario can be formulated as follows.

$$\begin{aligned} T_{mult}^{tot(t)} &= T_{mult}^{sche(t)} + T_{mult}^{comp(t)} + T_{mult}^{tran(t)} \\ &= \frac{1/S^{(t)}}{1 - \lambda^{(t)}/S^{(t)}} + \sum_{i=1}^{M} \frac{\lambda_i^{(t)} F_i / C_i^{(t)}}{\lambda^{(t)}(1 - \lambda_i^{(t)} F_i / C_i^{(t)})} + \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i^2}{(B^{(t)})^2 - B^{(t)} \sum_{i=1}^{M} \lambda_i^{(t)} D_i} \\ &\quad + \frac{\sum_{i=1}^{M} \lambda_i^{(t)} D_i}{\lambda^{(t)} B^{(t)}}. \end{aligned} \tag{3.10}$$

And the total resource cost in the multi-service scenario is given by

$$\mathcal{C}_{mult}^{tot(t)} = \left(\alpha S^{(t)} + \beta \sum_{i=1}^{M} C_i^{(t)} + \gamma B^{(t)} \right) \bar{t}. \tag{3.11}$$

Response Time Minimization Problem

Given a certain budget, it is a challenge for the MSP to satisfy response time requirement for diverse services. Therefore, we formulate the *response time minimization problem*, which can be stated as: to minimize the response time in multi-service scenario by jointly optimizing the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraint in each queueing system and the resource cost constraint. Mathematically, the problem can be formulated as follows.

$$\begin{aligned} \underset{\{S^{(t)},C_1^{(t)},\cdots,C_M^{(t)},B^{(t)}\}}{\text{Minimize}} \quad & T_{mult}^{tot(t)} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & \lambda_i^{(t)} F_i < C_i^{(t)}, \quad \forall i = 1, \ldots, M, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} D_i < B^{(t)}, \\ & (\alpha S^{(t)} + \beta \textstyle\sum_{i=1}^{M} C_i^{(t)} + \gamma B^{(t)})\bar{t} \leq \mathcal{C}_{max}, \end{aligned} \tag{3.12}$$

where $T_{mult}^{tot(t)}$, given by Equation (3.10), is the average response time for all services and $\mathcal{C}_{max}$ is the upper bound of the resource cost. The *response time minimization problem* in Equation (3.12) is a convex optimization problem [100]. Efficient solution methods for convex optimization problems are well developed. In this paper, we use the interior-point methods [100] to solve the formulated optimization problem.

Furthermore, when comparing the problem in Equation (3.4) in the single-service scenario and the problem in Equation (3.12) in the multi-service scenario, we have the following theorem.

Theorem 3.3. *The minimal response time obtained from the response time minimization problem in Equation (3.4) in the single-service scenario is equal to that obtained from the response time minimization problem in Equation (3.12) in the multi-service scenario, when the number of classes equal to 1.*

The proof of Theorem 3.3 is given in Appendix A.4. Theorem 3.3 reveals that if users in the multi-service scenario request for the same service, the MSP can efficiently configure resources according to the optimal analytical solution to problem in Equation (3.4) without applying primal-dual interior-point method to solve problem in Equation (3.12).

Resource Cost Minimization Problem

MSPs need to guarantee QoS provisioning for all services. However, it is challenging to achieve an economical resource allocation. Thus, we formulate the *resource cost minimization problem*, which can be stated as: to minimize the total resource cost in the multi-service scenario by jointly optimizing the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraint in each queueing system and the requirement on response time for each service. Mathematically, the problem can be formulated as follows.

$$\begin{aligned} \underset{\{S^{(t)},C_1^{(t)},\cdots,C_M^{(t)},B^{(t)}\}}{\text{Minimize}} \quad & \mathcal{C}_{mult}^{tot(t)} = (\alpha S^{(t)} + \beta \textstyle\sum_{i=1}^{M} C_i^{(t)} + \gamma B^{(t)})\bar{t} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & \lambda_i^{(t)} F_i < C_i^{(t)}, \ \forall i = 1, \ldots, M, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} D_i < B^{(t)}, \\ & T_{mult}^{tot(i)(t)} \leq \tau_i, \ \forall i = 1, \ldots, M, \end{aligned} \tag{3.13}$$

where $T_{mult}^{tot(i)(t)}$ is the response time of class-i service, which is given by Equation (3.9), and τ_i is the upper bound of response time for class-i service. The *resource cost minimization problem* in Equation (3.13) can be solved by using the primal-dual interior-point methods [100]. To examine the relationship between the resource cost optimization problem in Equation (3.7) in the single-service scenario and the problem in Equation (3.13) in the multi-service scenario, we derive the following theorem.

Theorem 3.4. *The minimal resource cost obtained from the optimization problem in Equation (3.7) for the single-service scenario is equal to the minimal resource cost obtained from the optimization problem in Equation (3.13) for the multi-service scenario with the number of classes equal to 1.*

The proof of Theorem 3.4 is given in Appendix A.5. Intuitively, when all users request for the same service in the multi-service scenario, resources will be allocated to the only requested service, which is the same as the single-service scenario. From Theorem 3.3 and Theorem 3.4, we can find that the proposed optimal resource allocation in the single-service scenario is a special case of that in the multi-service scenario.

3.3.3 Priority-service Scenario

Until now, we only discuss the non-priority service scheme, i.e. all services have the same priority in terms of response time. In practice, some urgent multimedia services, like the real-time health monitoring, need to be processed as soon as possible, and thus such requests should have a higher priority than other services. In this subsection, we extend our study to the priority-service scenario, in which requests for the higher-priority service should be processed before those for the lower-priority service. Specifically, we study the preemptive priority discipline.

Suppose that there are M services with different priorities. The service with a smaller index number has a higher priority. When requests arrive at data center, the master server always schedules requests with the highest priority first. Therefore, the schedule queue is modeled as an $M/M/1$ queue with preemptive priority. The response time for scheduling class-i requests is given by $T_{prio}^{sche(i)(t)} = \frac{1/S^{(t)}}{1-\sigma_{sche}^{(i-1)}} + \frac{\sum_{k=1}^{i}(\lambda_k^{(t)}/(S^{(t)})^2)}{(1-\sigma_{sche}^{(i-1)})(1-\sigma_{sche}^{(i)})}$, where $\sigma_{sche}^{(i)} = \sum_{k=1}^{i}\frac{\lambda_k^{(t)}}{S^{(t)}}$. To avoid the congestion in the schedule queue, $\sigma_{sche}^{(M)} = \sum_{i=1}^{M}\frac{\lambda_i^{(t)}}{S^{(t)}} < 1$ should be satisfied. Since schedule rates are the same for all requests, the average response time in the schedule queue is given by $T_{prio}^{sche(t)} = \frac{1/S^{(t)}}{1-\lambda^{(t)}/S^{(t)}}$. In computation phase, M computation queues are used to store requests with the corresponding priority. The total computation resources are aggregated to provide service. Requests with the highest priority have the preemptive right to obtain service immediately. The total computation resource is denoted as $C^{(t)}$, and the average task size of class-i application is F_i. Thus, the service process at the computing server can be modeled as a preemptive priority queue with unequal service rates. According to [101], the response time of class-i service in the computation queue is given by $T_{prio}^{comp(i)(t)} = \frac{F_i/C^{(t)}}{1-\sigma_{comp}^{(i-1)}} + \frac{\sum_{k=1}^{i}\left(\lambda_k^{(t)}F_k^2/(C^{(t)})^2\right)}{(1-\sigma_{comp}^{(i-1)})(1-\sigma_{comp}^{(i)})}$, where $\sigma_{comp}^{(i)} = \sum_{k=1}^{i}\frac{\lambda_k^{(t)}F_k}{C^{(t)}}$. Moreover, $\sigma_{comp}^{(M)} = \sum_{i=1}^{M}\frac{\lambda_i^{(t)}F_i}{C^{(t)}} < 1$ is required. Since the service rates are unequal for different services, the mean response time is given by [101] $T_{prio}^{comp(t)} = \sum_{i=1}^{M}\frac{\lambda_i^{(t)}T_{prio}^{comp(i)(t)}}{\lambda^{(t)}}$. After processing, all service results are sent through the transmission server. The results in the higher-priority classes are transmitted prior to those in the lower-priority classes. The allocated bandwidth resource is denoted by $B^{(t)}$. Thus, the response time for delivering class-i results is given by [101] $T_{prio}^{tran(i)(t)} = \frac{D_i/B^{(t)}}{1-\sigma_{tran}^{(i-1)}} + \frac{\sum_{k=1}^{i}(\lambda_k^{(t)}D_k^2/(B^{(t)})^2)}{(1-\sigma_{tran}^{(i-1)})(1-\sigma_{tran}^{(i)})}$, where $\sigma_{tran}^{(i)} = \sum_{k=1}^{i}\frac{\lambda_k^{(t)}D_k}{B^{(t)}}$. To ensure the transmission queue

stable, $\sigma_{tran}^{(M)} = \sum_{i=1}^{M} \frac{\lambda_i^{(t)} D_i}{B^{(t)}} < 1$ should be satisfied. Thus, the mean response time in the transmission phase is formulated as $T_{prio}^{tran(t)} = \sum_{i=1}^{M} \frac{\lambda_i^{(t)} T_{prio}^{tran(i)(t)}}{\lambda^{(t)}}$.

Based on the above analysis, the total response time in the priority-service scenario is the sum of response time in the three phases, which can be given by

$$T_{prio}^{tot(t)} = T_{prio}^{sche(t)} + T_{prio}^{comp(t)} + T_{prio}^{tran(t)}. \tag{3.14}$$

Furthermore, we can formulate the response time for the class-i service as

$$T_{prio}^{tot(i)(t)} = T_{prio}^{sche(i)(t)} + T_{prio}^{comp(i)(t)} + T_{prio}^{tran(i)(t)}, \quad \forall i = 1, 2, \ldots, M. \tag{3.15}$$

Additionally, the total resource cost in priority-service scenario is

$$\mathcal{C}_{prio}^{tot(t)} = \left(\alpha S^{(t)} + \beta C^{(t)} + \gamma B^{(t)}\right) \bar{t}. \tag{3.16}$$

Response Time Minimization Problem

In multimedia cloud, priority service discipline has been widely used. The MSP has to support different priority services and minimize the mean response time. Therefore, we formulate the *response time minimization problem* in the priority-service scenario, which can be stated as: to minimize the mean response time for all services by jointly optimizing the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraint in each queueing system and the resource cost constraint. Mathematically, the response time minimization problem can be formulated as

$$\begin{aligned} \underset{\{S^{(t)}, C^{(t)}, B^{(t)}\}}{\text{Minimize}} \quad & T_{prio}^{tot(t)} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} F_i < C^{(t)}, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} D_i < B^{(t)}, \\ & (\alpha S^{(t)} + \beta C^{(t)} + \gamma B^{(t)}) \bar{t} \leq \mathcal{C}_{max}, \end{aligned} \tag{3.17}$$

where $T_{prio}^{tot(t)}$ is given in Equation (3.14), and $\mathcal{C}_{max}$ is the upper bound of resource cost.

Intuitively, the priority service scheme reduces the waiting time for the higher priority

services at the expense of a longer waiting time for the lower priority services. There are two questions in priority-service scenario: 1) how the introduction of priorities affects the overall response time, and 2) how the priority can be assigned appropriately? To answer the two questions, we do theoretical analysis and get the following theorem.

Theorem 3.5. *With the same resource cost constraint and the same services, the response time of the highest priority service (class-1 service) in the priority-service scenario is always no larger than the response time of the same service (class-1 service) in the multi-service scenario.*

The proof of Theorem 3.5 is given in Appendix A.6. Theorem 3.5 shows that the highest priority service in the priority-service scenario has a lower response time than the same service in the multi-service scenario. In the priority discipline, the highest priority requests have the preemptive right to obtain service. But in the multi-service scenario, different services take the same privilege and have to wait for the completion of previous requests. As expected, the imposition of priorities decreases the response time for the higher priority service.

We next examine the impact of priority assignment to the response time. Schrage and Miller [102] propose the shortest processing time (SPT) rule, which is described as follows. If the objective of a queue is to reduce the overall mean delay, the priority should be given to the group of customers that has the faster service rate. Therefore, the priority assignment in the practical multimedia cloud should depend on the primary objective. Specifically, if the primary objective in multimedia cloud is to reduce the response time for one specific service, like the real-time health monitoring, this service should be given the highest priority. However, if the primary objective is to reduce the overall delay for all services, the SPT rule should be employed to give the higher priority to the service with a faster processing rate.

Resource Cost Minimization Problem

In multimedia cloud, different priority services require different response time. It is challenging for the MSP to support heterogeneous QoS provisioning at the minimal resource cost. Therefore, we formulate the *resource cost minimization problem*, which can be stated as: to minimize the total resource cost in priority-service scenario by jointly optimizing

the allocated schedule resource, the computation resource, and the bandwidth resource, subject to the queueing stability constraints and the response time constraint for each service. Mathematically, the *resource cost minimization problem* can be formulated as follows.

$$\begin{aligned} \underset{\{S^{(t)},C^{(t)},B^{(t)}\}}{\text{Minimize}} \quad & \mathcal{C}_{prio}^{tot(t)} = (\alpha S^{(t)} + \beta C^{(t)} + \gamma B^{(t)})\bar{t} \\ \text{subject to} \quad & \\ & \lambda^{(t)} < S^{(t)}, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} F_i < C^{(t)}, \\ & \textstyle\sum_{i=1}^{M} \lambda_i^{(t)} D_i < B^{(t)}, \\ & T_{prio}^{tot(i)(t)} \leq \tau_i, \quad \forall i = 1, \ldots, M, \end{aligned} \tag{3.18}$$

where $T_{prio}^{tot(i)(t)}$, given in Equation (3.15), is the response time for class-i service, and τ_i is the upper bound of response time. We use the primal-dual interior-point method [100] to solve the *resource cost minimization problem* in Equation (3.18).

3.4 Simulations

We evaluate the proposed resource allocation schemes by extensive simulations. In order to make our simulations more convincing, we use the practical parameters and pricing rates of Windows Azure. Windows Azure [16] is a cloud platform developed by Microsoft, which provides on-demand computation and bandwidth resources for services through Microsoft data centers.

3.4.1 Simulations in Single-service Scenario

We first implement simulations to evaluate the resource allocation in the single-service scenario. We employ one medium instance as the master server to schedule requests and four extra large instances as the computing cluster to process requests. The detailed configuration and price rates can be found from [16]. Since major cloud computing providers (e.g. Amazon EC2 [15], Microsoft Azure [16], etc.) commonly charge resource usage at an hourly rate, we assume that the proposed resource allocation schemes periodically run at the beginning of each time slot with a fixed length of 1 hour. Similar assumptions are made in [61]. In the single-service scenario, the upper bound of the response time $\tau = 50$

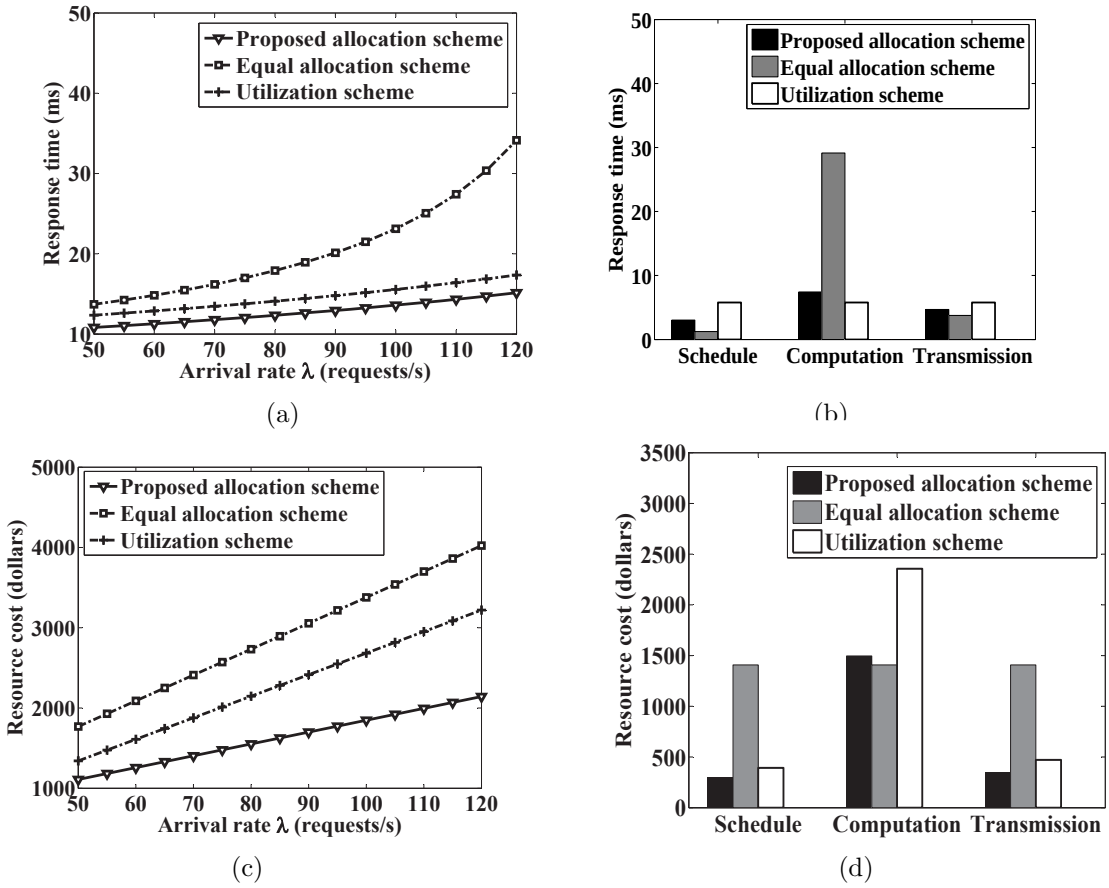


Figure 3.5: Simulation results of resource allocation in the single-service scenario: (a) comparison of response time, (b) comparison of response time in each phase when λ=120 requests/s, (c) comparison of resource cost, and (d) comparison of resource cost in each phase when λ=120 requests/s.

ms. The resource cost constraint is set to $\mathcal{C}_{max} = 5000$ dollars.

Currently, the state-of-the-art resource allocation scheme [103] is based on utilization threshold, which has been applied by Amazon AWS Elastic Beanstalk [18]. In the utilization scheme, the resource provisioning will be triggered when the utilization is higher than δ_h and stopped until utilization is lower or equal to δ_l ($\delta_l < \delta_h$). In our simulations, we compare the performance among the following three schemes: 1) the proposed allocation scheme, in which the cloud resources are optimally allocated by solving the optimization problem in Equation (3.4) or that in Equation (3.7); 2) the equal allocation scheme, in which the resource cost for schedule, computation, and transmission are

evenly allocated; and 3) the utilization scheme [103], in which δ_l and δ_h are set as 0.7 and 0.8.

Figure 3.5(a) shows the comparison of response time when λ varies from 50 requests/s to 120 requests/s. From Figure 3.5(a), we can see that the proposed allocation scheme achieves a much smaller response time compared to the other schemes under the same resource cost constraint. We analyze the reason behind Figure 3.5(a). The equal allocation scheme simply divides resource cost on the three phases, which fails to consider the real resource demands and thus takes the longest response time. The utilization scheme [103] only guarantees that the utilization in the service is lower than the upper bound. However, it cannot ensure that the budget is optimally allocated to the resources which are mostly needed. For the proposed resource allocation scheme, it searches the whole feasible region to find the optimal solution such that the response time can be minimized. Figure 3.5(b) gives a close look of the response time in the schedule, computation, and transmission phases when λ is 120 requests/s. As shown in Figure 3.5(b), the equal allocation scheme and the utilization scheme cannot optimally allocate resources among the three phases, leading to a higher response time. For instance, the equal allocation scheme allocates less resource in the computation phase and thus it takes longer response time to process requests. Next, we evaluate the resource cost among the three schemes, which is shown in Figure 3.5(c). From Figure 3.5(c), we can see that the proposed optimal allocation scheme achieves a lower resource cost compared to the other schemes under the same response time constraint. We can get the reason from the detailed resource cost shown in Figure 3.5(d), when arrival rate is 120 requests/s. Compared to the proposed optimal allocation scheme, the equal allocation scheme configures too much resources in the schedule and transmission phases. As a result, users' requests cannot be processed in time and all jobs will be congested in the computation queue, which deteriorates the system performance. The utilization scheme allocates excessive resources in each phase in order to ensure that the service utilization falls in a certain range, which leads to a higher total resource cost than the proposed allocation scheme.

3.4.2 Simulations in Multi-service Scenario

In this subsection, we evaluate the performance of the proposed resource allocation schemes in the multi-service scenario. Three classes of multimedia services are tested

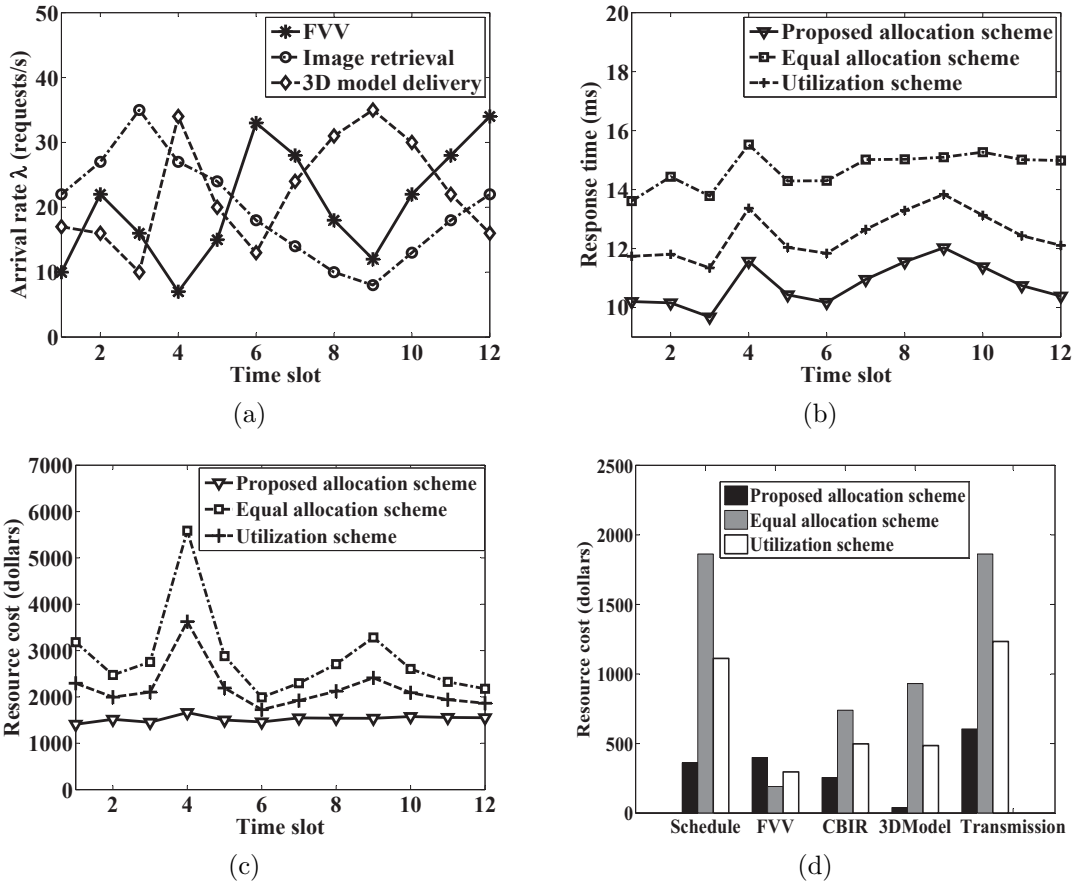


Figure 3.6: Simulation results of resource allocation in the multi-service scenario: (a) request arrival rate at each service in a 12-hour period, (b) comparison of response time, (c) comparison of resource cost, and (d) comparison of detailed resource cost at the 6th time slot.

in our simulations, including the free viewpoint video (FVV) service [50], content-based image retrieval (CBIR) service [53], and 3D model (3DModel) delivery service [104]. Different services have different request arrival rates, different task sizes, different result sizes, and different requirements on the response time. Specifically, the FVV service needs a low response time to guarantee the user interactive experience, the CBIR service demands intensive computation resource due to the high computation complexity, and the 3DModel service requires a large amount of bandwidth to deliver 3D models to users.

We compare the performance among: 1) the proposed optimal allocation scheme, in which the resources are allocated by solving the optimization problem in Equation (3.12)

or that in Equation (3.13); 2) the equal allocation scheme, in which the resource cost is evenly allocated; and 3) the utilization scheme [103] with $\delta_l = 0.7$ and $\delta_h = 0.8$. We record the arrivals of requests in a 12-hour period, which is shown in Figure 3.6(a). The workload intensity in each service varies over time. Given the resource cost constraint, we compare the average response time in the 12-hour period in Figure 3.6(b). From Figure 3.6(b), we can see that the proposed optimal allocation scheme takes the lower response time compared to the equal allocation scheme and the utilization scheme [103]. Furthermore, we compare the resource cost of the three schemes in Figure 3.6(c), from which we can find that the proposed optimal allocation scheme achieves the lowest resource cost among the three scheme. Figure 3.6(d) compares the detailed resource cost at the 6th time slot. As presented in Figure 3.6(a), the FVV service encounters a surge of requests at the 6th time slot. As a result, the proposed resource allocation scheme assigns more resources to the FVV service such that the local congestion can be effectively addressed. Compared with the proposed scheme, the equal allocation scheme and the utilization scheme assign excessive resources in the schedule and transmission phases and limited resources in the computation phase, leading to a higher resource cost than the proposed scheme, as shown in Figure 3.6(c).

3.4.3 Simulations in Priority-service Scenario

In this subsection, we evaluate the performance of the proposed resource allocation schemes in the priority-service scenario. We still employ the FVV, CBIR, and 3DModel services in our simulations. The FVV service enjoys the highest priority, while the 3D-Model service is assigned the lowest priority. The arrivals of requests for each service vary as in Figure 3.6(a). We compare the proposed optimal allocation scheme, the equal allocation scheme, and the utilization scheme [103].

Figure 3.7(a) shows the comparison of the response time. From Figure 3.7(a), we can see that the proposed resource allocation scheme takes a lower response time than the other two schemes. Moreover, we compare the response time of FVV service in the multi-service scenario and that in the priority-service scenario, which is shown in Figure 3.7(b). As stated in Theorem 3.5, priority service scheme reduces the response time for the highest priority service, which is the FVV service in our simulation. From Figure 3.7(b), we can find that the FVV service indeed enjoys a faster service in the priority-service

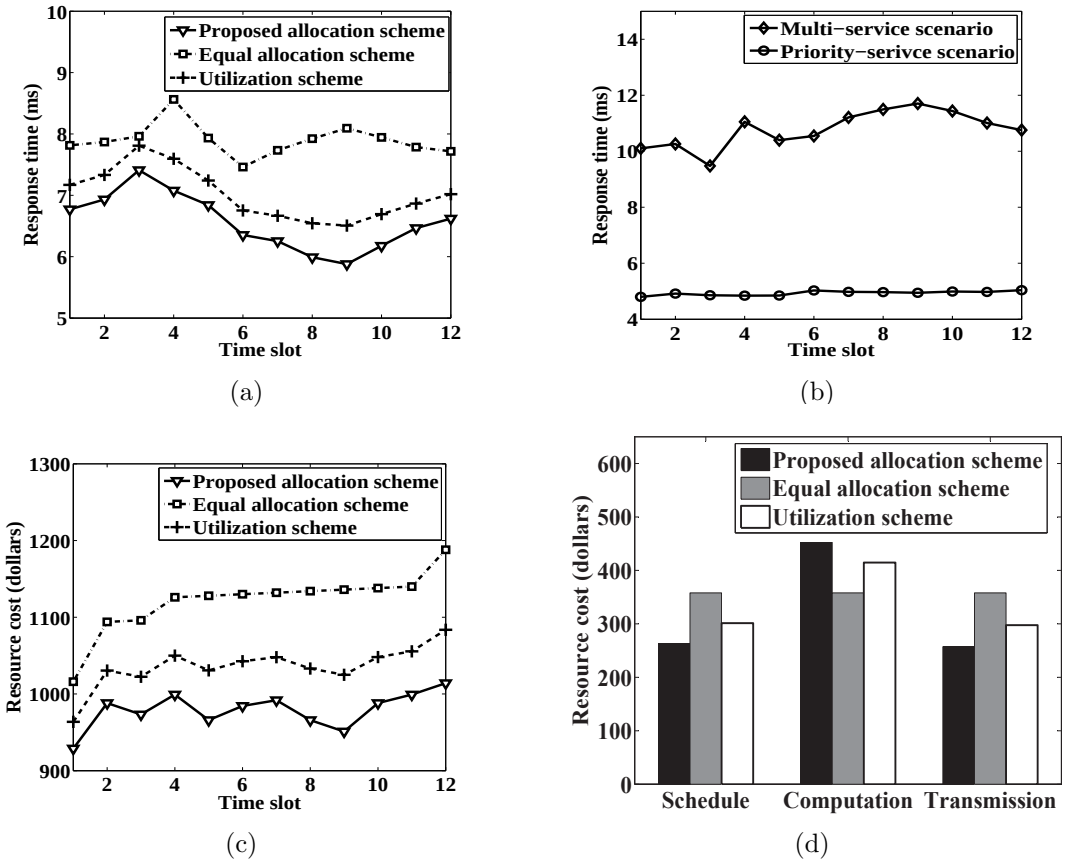


Figure 3.7: Simulation results of resource allocation in the priority-service scenario: (a) comparison of response time, (b) comparison of response time for FVV service in multi-service scenario and that in priority-service scenario, (c) comparison of resource cost, and (d) comparison of detailed resource cost at the 3rd time slot.

scenario compared to that in the multi-service scenario, which justifies the statement of Theorem 3.5. Next, we evaluate the proposed resource cost minimization scheme. As shown in Figure 3.7(c), the proposed optimal resource allocation scheme provides satisfactory services at a much lower resource cost than the equal allocation scheme and the utilization scheme. The detailed resource cost at the 3rd time slot is compared in Figure 3.7(d). As shown in Figure 3.6(a), there is a request peak for the CBIR service at the 3rd time slot. As a result, more computing resources are demanded at that time. Compared to the equal allocation scheme and the utilization scheme in Figure 3.7(d), the proposed optimal allocation scheme deploys more resources in computation such that

the surge of resource demands can be effectively satisfied.

3.5 Chapter Summary

In this chapter, we introduce a queueing model to characterize the service process in multimedia cloud. The proposed queueing model consists of three concatenated queueing systems, which are the schedule queue, the computation queue, and the transmission queue. We theoretically analyze the equilibrium demands and derive the relationships between the response time and the allocated cloud resources. Based on the proposed queueing model, we investigate resource optimization problems in three different scenarios: single-service scenario, multi-service scenario, and priority-service scenario. In each scenario, we formulate and solve the response time minimization problem and the resource cost minimization problem, respectively. Extensive simulation results demonstrate that the proposed resource allocation schemes can optimally utilize cloud resources to achieve the minimal response time under a certain budget or provide the satisfactory services at the minimal resource cost.

Chapter 4

Dynamic Resource Configuration for Cloud based Multimedia Services

In multimedia cloud, multimedia service providers (MSPs) rent virtual machines (VMs) and supply various media services. There are two key challenges for MSPs: 1) *how to allocate the required number and class of VMs at the minimal cost*; and 2) *how to dynamically adjust VMs to adapt to the varying workload*. In this chapter, we study the dynamic resource configuration. We propose a two-time-scale resource configuration (TRC) scheme, which allocates VMs in a mid-long time scale and dynamically adjusts VMs in a fine-grained time scale. Based on the TRC scheme, we investigate resource optimization problems in single-site cloud scenario and multi-site cloud scenario, respectively.

4.1 Introduction

In multimedia cloud, three major roles are involved in the service: *cloud computing providers*, *MSPs*, and *users*. As the computing resource suppliers, cloud computing providers operate the cloud infrastructure and provide VMs as service. MSPs deploy multimedia services on the leased VMs. As customers, users send requests for the interested services on cloud.

There are two challenges for MSPs. The first challenge is *how to allocate the optimal number and class of VMs for each service*. If too many VMs are allocated to one

service, the service speed can be increased, but the cost will also be increased. The under-provisioned resources would slow down the service, while the over-provisioned resources would lead to the unnecessary resource waste. MSPs have to avoid both under-provisioning and over-provisioning. Additionally, multimedia services have heterogeneous resource demands, while each type of VM has a different pre-configured resource capacity. An inappropriate VM allocation would cause the unnecessary waste. Thus, MSPs need the optimal resource allocation to satisfy resource demands at the minimal resource cost. Besides VM allocation, another challenge for MSPs is *how to dynamically adjust VMs to adapt to the varying workload*. Since thousands of users may request multimedia services from cloud at the same time, the instantaneous burst of requests will cause local congestion and unacceptable response time. Moreover, the varying workload leads to unbalanced resource utilization. For instance, the allocated VMs in one service cannot handle the request burst, while VMs in other services may be underloaded. Thus, an effective resource allocation scheme should be able to appropriately allocate VMs to avoid resource under-provisioning or over-provisioning and dynamically reconfigure VMs to satisfy time-varying workload.

To address aforementioned challenges, we study the dynamic resource configuration in this chapter. Our contributions are summarized as follows.

1. We propose a TRC scheme to allocate VMs and dynamically reconfigure VMs. Since major cloud computing providers, like Amazon EC2 [15], charge VMs on an hourly basis, MSPs need to determine the required VMs in each hour. Therefore, we focus on the resource allocation problem in a mid-long time scale, determining the optimal number and classes of VMs. On the other hand, we dynamically reallocate VMs among services in a fine-grained time scale to deal with the varying workload. By considering two different time scales, the proposed TRC scheme is able to flexibly allocate resources according to the real demands.

2. Based on the proposed TRC scheme, we study resource allocation problems in single-site cloud scenario and multi-site cloud scenario, respectively. In each scenario, we investigate the resource cost minimization problem and the response time minimization problem. Since the formulated problems are nonlinear integer programming, which are known as NP-hard [105, 106], we propose heuristics to allocate resources in an efficient way. The proposed heuristics are sub-optimal but

lightweight, which are demonstrated to perform close to the optimal solutions based on the simulation results.

The remainder of this chapter is organized as follows. Section 4.2 presents system models. Based on the proposed models, we study resource allocation and resource reconfiguration problems for single-site cloud scenario in Section 4.3 and for multi-site cloud scenario in Section 4.4. Section 4.5 presents extensive performance evaluations. Finally, we conclude the chapter in Section 4.6.

4.2 System Models

4.2.1 Multimedia Cloud Architecture

Currently, most of clouds are built in the form of data centers [92]. The cloud data center consists of a bunch of physical computing servers, which support the running and provisioning of VMs. Different classes of VM instances generally have different resource capacities in terms of CPU frequency, processor numbers, memory size, and I/O rates. VMs can form virtual cluster [61, 93] to provide faster services and more powerful resources.

The multimedia cloud architecture is illustrated in Figure 4.1. When a request arrives at data center, the dispatcher will firstly distribute the request to the corresponding service. The demand monitor performs the live monitoring on the type and number of requests, and forwards information to the service level agreement (SLA) negotiator. Based on the workload information, the SLA negotiator will determine QoS requirements for each service. With demand statistics and QoS requirements, the proposed TRC scheme will determine the class and the number of VMs for each service to achieve the minimal resource cost, while guaranteeing response time requirement. If the initially reserved VMs cannot satisfy resource demands, the proposed scheme will send requests to cloud computing providers for additional on-demand VMs. After resource allocation, the proposed scheme will inspect the varying resource demands and dynamically reconfigure VMs among services. The resource reconfiguration works in a fine-grained time scale. Thus, the varying resource demands in each service can be effectively satisfied.

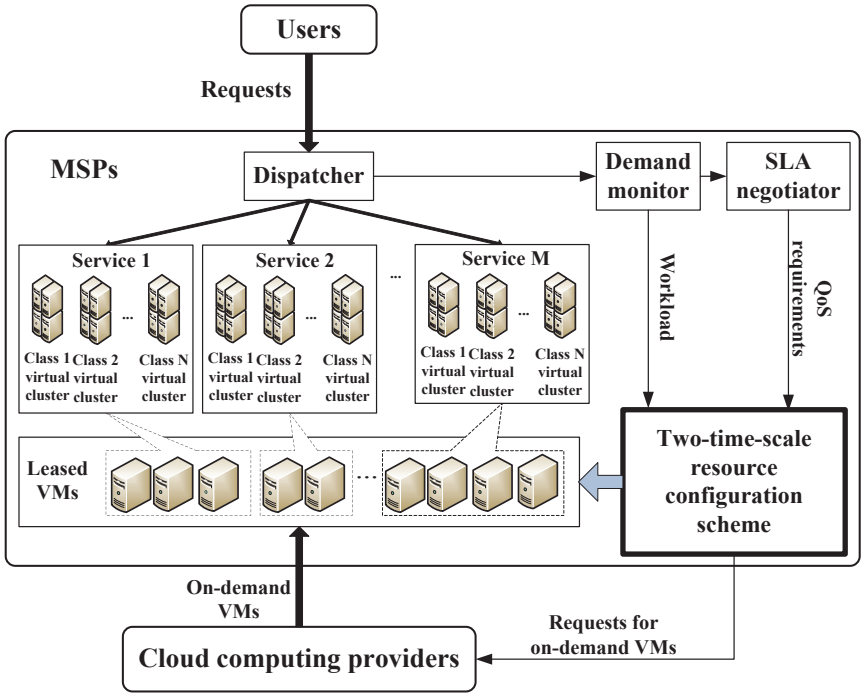


Figure 4.1: Illustration of multimedia cloud architecture.

4.2.2 VM Pricing Plan

Generally, major cloud computing providers [15, 16, 32] offer two different pricing plans, i.e. the reservation plan and the on-demand plan. In the reservation plan, MSPs make a one-time upfront payment for a VM instance and in turn receive a significant discount on the hourly rate for that instance. The reservation plan needs to be committed for a certain term. For example, the shortest term of reservation plan in Amazon EC2 [15] is 1 year. Besides the reservation plan, the on-demand plan provides a more flexible VM usage. In the on-demand plan, the VMs can be paid by the hourly usage without long-term commitment and upfront payment. But the price rates in the on-demand plan are higher than those in the reservation plan. In each plan, the hourly price rate is defined in dollars ($) per VM instance per hour, and VMs with different resource capacities have different price rates. Moreover, different sites generally have different price rates. For example, Amazon EC2 [15] currently has nine sites around the globe, in which the site at Sao Paulo charges the highest price rates [107].

4.2.3 Two-time-scale Resource Configuration (TRC) Scheme

We propose a TRC scheme to manage cloud resources. Two different time scales are considered in the scheme. We will denote by t_1 as the mid-long time scale for resource allocation and t_2 as the fine-grained time scale for resource reconfiguration. In scale t_1, our objective is to minimize the resource cost by optimizing VM allocation. On the other hand, the workload in cloud changes dynamically, leading to the variation in demands. The initially allocated VMs at t_1 scale cannot adapt to the varying demands. Therefore, we perform resource reconfiguration in a fine-grained time scale t_2. In resource reconfiguration, the initially configured VMs are reallocated among services according to real demands. We study the resource reconfiguration problem, in which the average response time is minimized by adjusting VMs among services. By considering two different time scales, the proposed TRC scheme is able to capture the macro trend and the local variation of demands. Specifically, the resource allocation in t_1 scale can neglect noises arising from instantaneous fluctuations, while the resource reconfiguration in time scale t_2 can flexibly adjust resources corresponding to the real workload variations.

Suppose that L data centers are available. The set of data centers can be denoted as $\mathcal{D} = \{D_1, D_2, \ldots, D_L\}$. In each data center, we will denote by $\mathcal{I}$ the set of VMs. Since different classes of VMs are charged at different price rates, let p_{li}^r and p_{li}^d denote the price rates of class-i ($\forall i \in \mathcal{I}$) VM instance at D_l in the reservation plan and the on-demand plan, respectively. The MSP may reserve a certain number of VMs at the initial stage, and let K_{li}^{ini} denote the number of initially reserved class-i VMs at D_l. We will denote by $\mathcal{J}$ the set of services. Due to the heterogeneous resource capacities, different classes of VMs have different service rates. We will denote by μ_{ij} the average service rate of class-i VM instance for processing type-j requests. At D_l, K_{lij}^r and K_{lij}^d are the numbers of allocated class-i VMs for type-j service in the reservation plan and on-demand plan, respectively. We will present the proposed TRC scheme to determine the optimal values of K_{lij}^r and K_{lij}^d at different time scales.

4.2.4 Workload Prediction Model

Since two consecutive incoming requests may be sent from different users, the numbers of requests occur in non-overlapping intervals are independent random variables. We assume that the arrivals of requests for type-j service is a Poisson process with an average of λ_j. In

order to predict λ_j, we use the autoregressive integrated moving average (ARIMA) model, which has been applied to predict demands on P2P and cloud media system with high accuracy [108,109]. In ARIMA(p, d, q) model, a given time series Y is differenced d times to get a stationary series X, and each sample x_t can be estimated as a linear weighted sum of p previous samples $(x_{t-1}, \ldots, x_{t-p})$ and q previous random errors $(\epsilon_{t-1}, \ldots, \epsilon_{t-q})$. Since Poisson process has the property of stationary increments [110], we can obtain a stationary process when performing one order difference. We therefore have $d = 1$ in the ARIMA(p, d, q) model. Figure 4.2(a) shows the one order difference of request arrivals with an average rate of $\lambda = 300$ requests/s. In order to identify the orders p and q, the standard technique is to match the patterns of the autocorrelation and partial autocorrelation of the differenced time series with the theoretical patterns of known models. Figure 4.2(b) and Figure 4.2(c) plot the autocorrelation function $\hat{r}_k$ and the partial autocorrelation function $\hat{\varphi}_{kk}$, respectively. We can find that the partial autocorrelation cuts off after lag $k = 2$ and the autocorrelation tails off as damped sine wave, which identifies $p = 2$ and $q = 0$ [111]. Therefore, we derive an ARIMA(2,1,0) model to predict the workload at $t + 1$, which is given by

$$\lambda_j^{t+1} = \lambda_j^t + \varphi_1(\lambda_j^t - \lambda_j^{t-1}) + \varphi_2(\lambda_j^{t-1} - \lambda_j^{t-2}) + \epsilon^t, \tag{4.1}$$

where ϵ^t is the white noise, which can be treated as zero in practice [111], and $\varphi_1,\ \varphi_2$ are coefficients, which can be acquired by training the model with historical statistics.

4.3 Dynamic Resource Configuration for Single-site Cloud

In this section, we will study the optimal resource allocation and resource reconfiguration in single-site cloud scenario, in which MSP deploys services at one data center and all users send requests to the data center for service.

4.3.1 Resource Allocation for Single-site Cloud

The objective for resource allocation in t_1 time scale is to minimize the total resource cost. We use τ_{t_1} to denote the beginning time instant of t_1 period. The cost of allocated

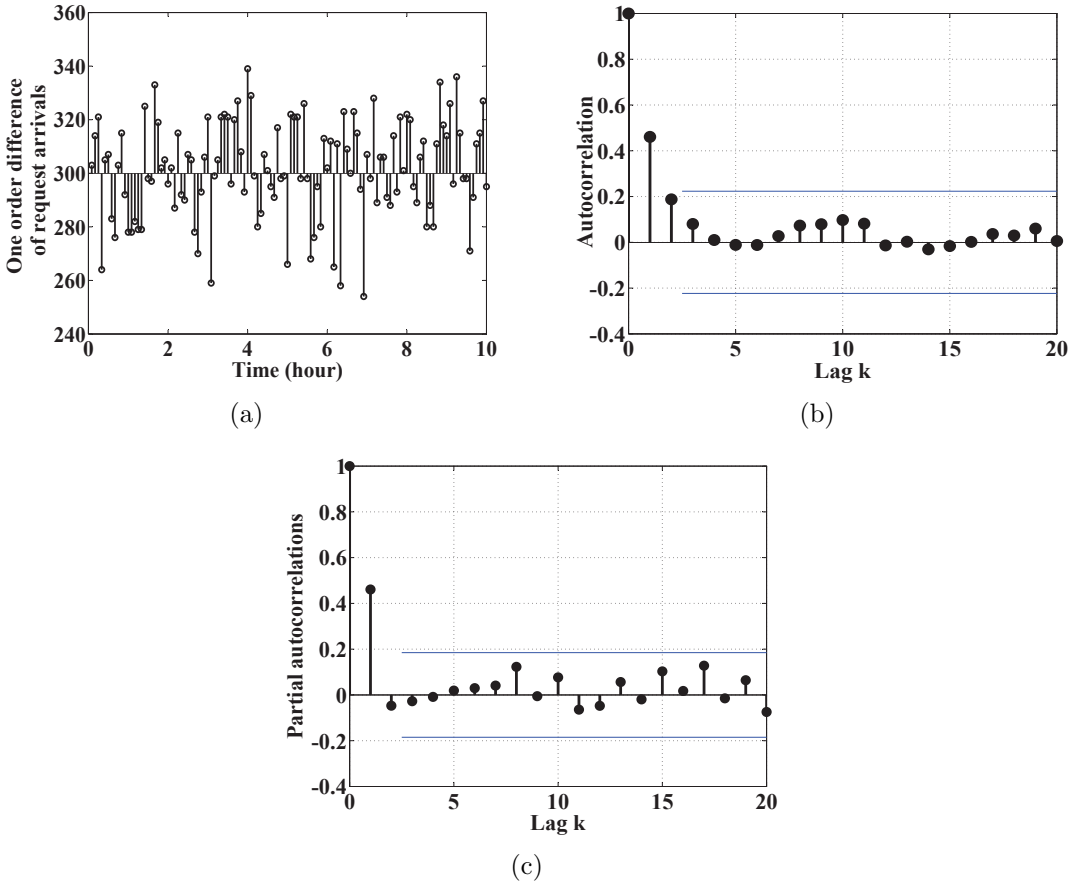


Figure 4.2: Determination of ARIMA model parameters: (a) one order difference of request arrivals with an average rate of 300 requests/s, (b) autocorrelation of series in (a), and (c) partial autocorrelations of series in (a).

VMs can be formulated as

$$C_{sgl} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^r K_{ij}^{r(\tau_{t_1})} + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^d K_{ij}^{d(\tau_{t_1})}, \tag{4.2}$$

where $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^r K_{ij}^{r(\tau_{t_1})}$ is the cost in the reservation plan and $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^d K_{ij}^{d(\tau_{t_1})}$ is the cost in the on-demand plan.

In type-j service, a number of $\left(K_{ij}^{r(\tau_{t_1})} + K_{ij}^{d(\tau_{t_1})}\right)$ class-i VMs are allocated. To balance workload among VMs, we schedule requests to each cluster in proportion to its service rate [112, 113], i.e. the cluster with a higher service rate will be assigned more requests to process. Thus, a type-j request is scheduled to class-i virtual cluster with

a probability of $\omega_{ij}^{(\tau_{t_1})} = \frac{(K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij}}{\sum_{i\in\mathcal{I}}\left((K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij}\right)}$. The performance of this scheduling scheme has been evaluated in our previous work [112]. The arrival requests are split into independent sub-processes. With the decomposition property of Poisson process [114], the requests scheduled to class-i virtual cluster still follow a Poisson process with an average rate of $\omega_{ij}^{(\tau_{t_1})}\lambda_j^{(\tau_{t_1})}$. Similar to assumptions commonly used in multimedia cloud literatures [61,115], we assume the service time at each cluster is exponential distribution with rate $(K_{ij}^{r(t)}+K_{ij}^{d(t)})\mu_{ij}$. The service process at class-i cluster is therefore modeled as an $M/M/1$ queuing system [114]. To make the queue stable, the request incoming rate must be less than the service rate, i.e. the constraint $\omega_{ij}^{(\tau_{t_1})}\lambda_j^{(\tau_{t_1})} < (K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij}$ is required. Thus, the mean response time of type-j service can be formulated as

$$T_j^{R(\tau_{t_1})} = \sum_{i\in\mathcal{I}} \frac{\omega_{ij}^{(\tau_{t_1})}}{(K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij} - \omega_{ij}^{(\tau_{t_1})}\lambda_j^{(\tau_{t_1})}}. \tag{4.3}$$

In addition, the amount of allocated class-i VMs from the reservation plan should be no more than the number of initially reserved class-i VMs. Thus, constraints $\sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_1})} \leq K_i^{ini},\ (\forall i \in \mathcal{I})$ need to be satisfied.

Based on the above analysis, we can formulate the optimal resource allocation problem for single-site cloud as follows.

$$\begin{aligned}
&\underset{\{K_{ij}^{r(\tau_{t_1})},K_{ij}^{d(\tau_{t_1})}\}}{\text{Minimize}} && \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^r K_{ij}^{r(\tau_{t_1})} + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_i^d K_{ij}^{d(\tau_{t_1})} \\
&\text{subject to} \\
&&& \textstyle\sum_{i\in\mathcal{I}} \frac{\omega_{ij}^{(\tau_{t_1})}}{(K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij} - \omega_{ij}^{(\tau_{t_1})}\lambda_j^{(\tau_{t_1})}} \leq T_j^U, && \forall j \in \mathcal{J}, \\
&&& \omega_{ij}^{(\tau_{t_1})}\lambda_j^{(\tau_{t_1})} < (K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\
&&& \textstyle\sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_1})} \leq K_i^{ini}, && \forall i \in \mathcal{I}, \\
&&& K_{ij}^{r(\tau_{t_1})}, K_{ij}^{d(\tau_{t_1})} \geq 0, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\
&&& K_{ij}^{r(\tau_{t_1})}, K_{ij}^{d(\tau_{t_1})} \in \mathbb{Z}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J},
\end{aligned} \tag{4.4}$$

where T_j^U is the upper bound of the response time for type-j service.

The optimization problem in Equation (4.4) is an integer programming, which is known to be NP-hard [116]. The problem can be solved by the branch-and-bound method

[116]. However, the branch-and-bound method takes exponential time complexity, which cannot satisfy the real-time requirement for practical applications. Therefore, we propose a heuristic to efficiently allocate VMs. In the heuristic, we introduce the unit cost, which is the equivalent cost of using one VM instance to process one unit request. In a service, the VM instance with a lower unit cost is more desirable and should be allocated to the service first. The heuristic is presented in Algorithm 4.1. Observing the proposed heuristic, we can find that the time complexities of step 1 and steps 3-13 are linear and the time complexity of step 2 is determined by the adopted sorting algorithm. If we use heap sort in step 2, the time complexity in the worst case is $O(n \log n)$ [117], which is much smaller than the exponential time complexity of the branch-and-bound method [118].

Algorithm 4.1: Heuristic for Resource Allocation in Single-site Cloud

1: Compute $q_{ij}^r = \frac{p_i^r}{\mu_{ij}}$ and $q_{ij}^d = \frac{p_i^d}{\mu_{ij}}$, which are unit costs of using one reserved and one on-demand class-i VM instance to process one type-j request, respectively.
2: Let set $Q = \{q_{ij}^r, q_{ij}^d | \forall i \in \mathbb{I}, \forall j \in \mathbb{J}\}$ and sort set Q in an ascending order.
3: **repeat**
4: Select the smallest $q_{ij}^v (v = r \text{ or } d)$ from Q.
5: **if** q_{ij}^v is from reserved VMs (i.e. $v = r$) **then**
6: Allocate $K_{ij}^{r(\tau_{t_1})}$ $(K_{ij}^{r(\tau_{t_1})} \leq K_i^{ini})$ reserved VMs to process requests λ_j', $(\lambda_j' \leq \lambda_j^{(\tau_{t_1})})$ such that $\frac{1}{K_{ij}^{r(\tau_{t_1})}\mu_{ij} - \lambda_j'} = T_j^U$.
7: Update $K_i^{ini} = K_i^{ini} - K_{ij}^{r(\tau_{t_1})}$, $\lambda_j^{(\tau_{t_1})} = \lambda_j^{(\tau_{t_1})} - \lambda_j'$.
8: **else if** q_{ij}^v is from on-demand VMs (i.e. $v = d$) **then**
9: Allocate $K_{ij}^{d(\tau_{t_1})}$ on-demand VMs to process all remaining type-j requests $\lambda_j^{(\tau_{t_1})}$ such that $\frac{1}{K_{ij}^{d(\tau_{t_1})}\mu_{ij} - \lambda_j^{(\tau_{t_1})}} = T_j^U$.
10: Update $\lambda_j^{(\tau_{t_1})} = 0$.
11: **end if**
12: **until** all requests are processed (i.e. $\lambda_j^{(\tau_{t_1})} = 0, \ \forall j \in \mathbb{J}$).
13: **return** $K_{ij}^{r(\tau_{t_1})}$ and $K_{ij}^{d(\tau_{t_1})}$, $(\forall i \in \mathbb{I}, \forall j \in \mathbb{J})$.

4.3.2 Resource Reconfiguration for Single-site Cloud

The proposed resource allocation in Section 4.3.1 determines the number of VMs for each service in t_1 scale. However, the allocated VMs cannot adapt to the dynamic demands.

Some services are short of resources while some services still have idle VMs. In order to address this issue, we propose the optimal resource reconfiguration in t_2 scale, in which we reconfigure the VMs allocated in time scale t_1 to minimize the average response time. Let τ_{t_2} be the beginning time instant of each t_2 time slot.

Let $K_{ij}^{r(\tau_{t_2})}$ and $K_{ij}^{d(\tau_{t_2})}$ be the numbers of reconfigured class-i VMs in the reservation plan and on-demand plan, respectively. According to Equation (4.3), the response time of type-j service is formulated as $T_j^{R(\tau_{t_2})} = \sum_{i\in\mathcal{I}} \frac{\omega_{ij}^{(\tau_{t_2})}}{\left(K_{ij}^{r(\tau_{t_2})}+K_{ij}^{d(\tau_{t_2})}\right)\mu_{ij}-\omega_{ij}^{(\tau_{t_2})}\lambda_j^{(\tau_{t_2})}}$. The average response time in the single-site cloud is therefore given by

$$\begin{aligned} T_{sgl}^{R(\tau_{t_2})} &= \sum_{j\in\mathcal{J}} \left(\frac{\lambda_j^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} T_j^{R(\tau_{t_2})} \right) \\ &= \sum_{j\in\mathcal{J}} \left(\frac{\lambda_j^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} \sum_{i\in\mathcal{I}} \frac{\omega_{ij}^{(\tau_{t_2})}}{(K_{ij}^{r(\tau_{t_2})}+K_{ij}^{d(\tau_{t_2})})\mu_{ij}-\omega_{ij}^{(\tau_{t_2})}\lambda_j^{(\tau_{t_2})}} \right). \end{aligned} \tag{4.5}$$

where $\lambda^{(\tau_{t_2})} = \sum_{j\in\mathcal{J}} \lambda_j^{(\tau_{t_2})}$ is the total workload in the data center.

In resource reconfiguration, our objective is to best utilize the initially allocated VMs to adapt to the varying workload. Thus, there is no subscription for extra VMs, i.e. the number of reconfigured class-i VMs in t_2 scale should be no more than the number of allocated class-i VMs in t_1 scale. Constraints $\sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_2})} \le \sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_1})}$ and $\sum_{j\in\mathcal{J}} K_{ij}^{d(\tau_{t_2})} \le \sum_{j\in\mathcal{J}} K_{ij}^{d(\tau_{t_1})}$ $(\forall i \in \mathcal{I})$ are required.

Based on the above analysis, we can formulate the optimal resource reconfiguration problem as

$$\begin{aligned} \underset{\{K_{ij}^{r(\tau_{t_2})}, K_{ij}^{d(\tau_{t_2})}\}}{\text{Minimize}} \quad & \sum_{j\in\mathcal{J}} \left(\frac{\lambda_j^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} \sum_{i\in\mathcal{I}} \frac{\omega_{ij}^{(\tau_{t_2})}}{(K_{ij}^{r(\tau_{t_2})}+K_{ij}^{d(\tau_{t_2})})\mu_{ij}-\omega_{ij}^{(\tau_{t_2})}\lambda_j^{(\tau_{t_2})}} \right) \\ \text{subject to} \quad & \\ & \omega_{ij}^{(\tau_{t_2})}\lambda_j^{(\tau_{t_2})} < (K_{ij}^{r(\tau_{t_2})} + K_{ij}^{d(\tau_{t_2})})\mu_{ij}, && \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \\ & \textstyle\sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_2})} \le \sum_{j\in\mathcal{J}} K_{ij}^{r(\tau_{t_1})}, && \forall i \in \mathcal{I}, \\ & \textstyle\sum_{j\in\mathcal{J}} K_{ij}^{d(\tau_{t_2})} \le \sum_{j\in\mathcal{J}} K_{ij}^{d(\tau_{t_1})}, && \forall i \in \mathcal{I}, \\ & K_{ij}^{r(\tau_{t_2})}, K_{ij}^{d(\tau_{t_2})} \ge 0, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\ & K_{ij}^{r(\tau_{t_2})}, K_{ij}^{d(\tau_{t_2})} \in \mathbb{Z}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}. \end{aligned} \tag{4.6}$$

The resource reconfiguration problem in Equation (4.6) is a nonlinear integer pro-

gramming, which can be solved by the branch-and-bound method [116]. However, the branch-and-bound method has an intensive computation complexity, thus not suitable for real-time applications. In order to reconfigure resources in an efficient way, we propose a heuristic in Algorithm 4.2. The idea of the heuristic is to reallocate VMs according to the utilization at each service. In queueing theory [114, 119], the utilization $\rho = \lambda/\mu$ represents the fraction of time that the server is busy, in which λ is the average arrival rate and μ is the average service rate. A higher utilization ρ means a higher percentage of time when the server is busy. In the heuristic, we firstly compute the utilization ρ_j of type-j service and the average utilization $\bar{\rho}$ among all services. If the difference between ρ_j and $\bar{\rho}$ is larger than a threshold ϵ, we reallocate extra VMs to type-j service from the service with the lowest utilization. The threshold ϵ is the sensitivity for unbalanced resource utilization, which can be used to control the frequency of resource reconfiguration. In practice, reallocating VMs too frequently will introduce extra overhead for cloud. Therefore, the threshold ϵ should be determined based on the variance of the arriving requests.

Algorithm 4.2: Heuristic for Resource Reconfiguration in Single-site Cloud

1: Calculate $\rho_j^{(\tau_{t_2})} = \frac{\lambda_j^{(\tau_{t_2})}}{\sum_{i\in\mathcal{I}}(K_{ij}^{r(\tau_{t_1})}+K_{ij}^{d(\tau_{t_1})})\mu_{ij}}$ and insert it into a min heap $\mathbf{H}_\rho$ [120].
Calculate $\bar{\rho} = \sum_{j\in\mathcal{J}} \rho_j^{(\tau_{t_2})}/|\mathcal{J}|$.
2: **for** j in $\mathcal{J}$ **do**
3: **while** $\rho_j^{(\tau_{t_2})} - \bar{\rho} > \epsilon$ **do**
4: Get j' from the root of $\mathbf{H}_\rho$.
5: Reallocate VMs from type-j' service to type-j service such that $\rho_{j'}^{(\tau_{t_2})} - \bar{\rho} = \epsilon$. VMs with a higher service rate for type-j service will be reallocated first.
6: Update $\mathbf{H}_\rho$ to get the new lowest utilization.
7: **end while**
8: **end for**
9: **return** $K_{ij}^{r(\tau_{t_2})}$ and $K_{ij}^{d(\tau_{t_2})}$, $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J})$.

4.4 Dynamic Resource Configuration for Multi-site Cloud

In this section, we extend our study to multi-site cloud scenario, in which MSP can deploy services on geographically distributed data centers.

4.4.1 Resource Allocation for Multi-site Cloud

Multi-site cloud covers a larger geographical scale and requires an efficient resource allocation. Therefore, we study the distributed resource allocation approach in multi-site cloud, which can be performed in parallel at each site. Since services are deployed on data centers at geographically distributed locations, the total resource cost can be formulated as

$$C_{mul} = \sum_{D_l \in \mathcal{D}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} p_{li}^{r} K_{lij}^{r(\tau_{t_1})} + \sum_{D_l \in \mathcal{D}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} p_{li}^{d} K_{lij}^{d(\tau_{t_1})}, \tag{4.7}$$

in which $\sum_{D_l \in \mathcal{D}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} p_{li}^{r} K_{lij}^{r(\tau_{t_1})}$ represents the cost in the reservation plan and $\sum_{D_l \in \mathcal{D}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} p_{li}^{d} K_{lij}^{d(\tau_{t_1})}$ represents the cost in the on-demand plan. To maintain a stable queue, the service rate must be faster than the request arrival rate, i.e. $\omega_{lij}^{(\tau_{t_1})} \lambda_{lj}^{(\tau_{t_1})} < (K_{lij}^{r(\tau_{t_1})} + K_{lij}^{d(\tau_{t_1})}) \mu_{ij}$ is required. According to Equation (4.3), the response time of type-j service at D_l can be formulated as

$$T_{lj}^{R(\tau_{t_1})} = \sum_{i \in \mathcal{I}} \frac{\omega_{lij}^{(\tau_{t_1})}}{(K_{lij}^{r(\tau_{t_1})} + K_{lij}^{d(\tau_{t_1})}) \mu_{ij} - \omega_{lij}^{(\tau_{t_1})} \lambda_{lj}^{(\tau_{t_1})}}. \tag{4.8}$$

On the other hand, the number of allocated VMs at D_l from the reservation plan should be no more than the initially reserved VMs. The constraints can be represented by $\sum_{j \in \mathcal{J}} K_{lij}^{r(\tau_{t_1})} \leq K_{li}^{ini}$.

Based on the above analysis, the optimal resource allocation problem for multi-site cloud can be formulated as follows.

$$
\begin{aligned}
&\underset{\{K_{lij}^{r(\tau_{t_1})}, K_{lij}^{d(\tau_{t_1})}\}}{\text{Minimize}} \quad \sum_{D_l\in\mathcal{D}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_{li}^{r} K_{lij}^{r(\tau_{t_1})} + \sum_{D_l\in\mathcal{D}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_{li}^{d} K_{lij}^{d(\tau_{t_1})} \\
&\text{subject to} \\
&\qquad \textstyle\sum_{i\in\mathcal{I}} \frac{\omega_{lij}^{(\tau_{t_1})}}{(K_{lij}^{r(\tau_{t_1})}+K_{lij}^{d(\tau_{t_1})})\mu_{ij}-\omega_{lij}^{(\tau_{t_1})}\lambda_{lj}^{(\tau_{t_1})}} \le T_j^U, && \forall D_l\in\mathcal{D},\ \forall j\in\mathcal{J}, \\
&\qquad \omega_{lij}^{(\tau_{t_1})}\lambda_{lj}^{(\tau_{t_1})} < (K_{lij}^{r(\tau_{t_1})}+K_{lij}^{d(\tau_{t_1})})\mu_{ij}, && \forall D_l\in\mathcal{D},\ \forall i\in\mathcal{I},\ \forall j\in\mathcal{J}, \\
&\qquad \textstyle\sum_{j\in\mathcal{J}} K_{lij}^{r(\tau_{t_1})} \le K_{li}^{ini}, && \forall D_l\in\mathcal{D},\ \forall i\in\mathcal{I}, \\
&\qquad K_{lij}^{r(\tau_{t_1})}, K_{lij}^{d(\tau_{t_1})} \ge 0, && \forall D_l\in\mathcal{D},\ \forall i\in\mathcal{I},\ \forall j\in\mathcal{J}, \\
&\qquad K_{lij}^{r(\tau_{t_1})}, K_{lij}^{d(\tau_{t_1})} \in \mathbb{Z}, && \forall D_l\in\mathcal{D},\ \forall i\in\mathcal{I},\ \forall j\in\mathcal{J},
\end{aligned}
\tag{4.9}
$$

where T_j^U is the upper bound of response time for type-j service.

The optimization problem in Equation (4.9) is a nonlinear integer programming, which is known to be NP-hard [116]. From the objective function in Equation (4.9), we can find the resource allocation at D_l does not need information from any other sites. Therefore, if the minimal resource cost can be achieved at each single site, the total resource cost in multi-site cloud will be minimized. Based on the analysis, we propose a heuristic for multi-site cloud in Algorithm 4.3, which is based on our previous heuristic in single-site cloud. The proposed heuristic is a distributed resource allocation approach, which can be performed in parallel at each data center.

Algorithm 4.3: Heuristic for Resource Allocation in Multi-site Cloud

1: **for** each $D_l \in \mathcal{D}$ **do**
2: Collect request arrival rate $\lambda_{lj}^{(\tau_{t_1})}$ and the initially reserved VMs K_{li}^{ini} at D_l
3: Allocate VMs at D_l by using the *Heuristic for Resource Allocation in Single-site Cloud* in Algorithm 4.1.
4: **end for**

4.4.2 Resource Reconfiguration for Multi-site Cloud

In multi-site cloud, the local congestion may occur due to the time-varying workload. For example, the instantaneous surge of local requests may result in unacceptable service delay at one site, while at the same time, some other sites may have insufficient load

and underutilized VMs. To address this issue, we perform resource reconfiguration for multi-site cloud. Comparing with single-site cloud, the resource reconfiguration in multi-site not only reallocates VMs among services at each site, but also balances workload by transferring excessive requests to the underloaded sites for service. By jointly optimizing the global workload assignment and the local resource reconfiguration, we can achieve the minimal average response time for the whole system.

In the proposed resource reconfiguration, if local congestion appears at any site, excessive requests will be redirected to other sites for service. The transmission delay between D_l and $D_{l'}$ is denoted as $d_{ll'}^{(\tau_{t_2})}$. Specially, when $l' = l$, $d_{ll}^{(\tau_{t_2})} = 0$. Let $z_{ll'j}^{(\tau_{t_2})}$ denote the probability that type-j requests are transferred from D_l to $D_{l'}$. Because all requests have to be served, we have $\sum_{D_{l'} \in \mathcal{D}} z_{ll'j}^{(\tau_{t_2})} = 1$ and $0 \leq z_{ll'j}^{(\tau_{t_2})} \leq 1$. In practice, the workload redirection is transparent to users. Users just request services and they do not need to know which data center receives and processes their requests. We assume that users' requests will be first sent to the closest data center. The arrivals of type-j requests at D_l before load redirection is denoted as $\lambda_{lj}^{(\tau_{t_2})}$. If a request is transferred to $D_{l'}$, the response time is the sum of the execution time at $D_{l'}$ and the transmission delay, which is given by $T_{l'j}^{E(\tau_{t_2})} + d_{ll'}^{(\tau_{t_2})}$, where $T_{l'j}^{E(\tau_{t_2})}$ is the execution time at $D_{l'}$. Therefore, the average response time of type-j service at D_l can be formulated as

$$T_{lj}^{R(\tau_{t_2})} = \sum_{D_{l'} \in \mathcal{D}} \left(z_{ll'j}^{(\tau_{t_2})} (T_{l'j}^{E(\tau_{t_2})} + d_{ll'}^{(\tau_{t_2})}) \right). \tag{4.10}$$

At D_l, a number of $K_{lij}^{r(\tau_{t_2})}$ reserved and $K_{lij}^{d(\tau_{t_2})}$ on-demand class-i VMs are reallocated to the type-j service. In workload transfer mechanism, requests from other data centers may be transferred to D_l. After load redirection, the request arrivals of type-j service are $\tilde{\lambda}_{lj}^{(\tau_{t_2})} = \sum_{D_{l'} \in \mathcal{D}} z_{l'lj}^{(\tau_{t_2})} \lambda_{l'j}^{(\tau_{t_2})}$. According to properties of Poisson distribution [121], the sum of multiple independent Poisson variables is still Poisson distributed. Thus, the arrivals of type-j requests at D_l still follow Poisson distribution. To ensure a stable queue, $\omega_{lij}^{(\tau_{t_2})} \tilde{\lambda}_{lj}^{(\tau_{t_2})} < (K_{lij}^{r(\tau_{t_2})} + K_{lij}^{d(\tau_{t_2})}) \mu_{ij}$ is required. Therefore, the execution time for type-j service at D_l can be given by

$$T_{lj}^{E(\tau_{t_2})} = \sum_{i \in \mathcal{I}} \frac{\omega_{lij}^{(\tau_{t_2})}}{(K_{lij}^{r(\tau_{t_2})} + K_{lij}^{d(\tau_{t_2})}) \mu_{ij} - \omega_{lij}^{(\tau_{t_2})} \tilde{\lambda}_{lj}^{(\tau_{t_2})}}. \tag{4.11}$$

Based on the above analysis, the average response time in multi-site cloud can be formulated as

$$\begin{aligned} T_{mul}^{R(\tau_{t_2})} &= \sum_{D_l \in \mathcal{D}} \sum_{j \in \mathcal{J}} \left(\frac{\lambda_{lj}^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} T_{lj}^{R(\tau_{t_2})} \right) \\ &= \sum_{D_l \in \mathcal{D}} \sum_{j \in \mathcal{J}} \left(\frac{\lambda_{lj}^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} \sum_{D_{l'} \in \mathcal{D}} \left(z_{ll'j}^{(\tau_{t_2})} \left(T_{l'j}^{E(\tau_{t_2})} + d_{ll'}^{(\tau_{t_2})} \right) \right) \right), \end{aligned} \tag{4.12}$$

where $\lambda^{(\tau_{t_2})} = \sum_{D_l \in \mathcal{D}} \sum_{j \in \mathcal{J}} \lambda_{lj}^{(\tau_{t_2})}$ is the total workload in the multi-site cloud, and $T_{l'j}^{E(\tau_{t_2})}$ is given by Equation (4.11).

On the other hand, the goal of resource reconfiguration is to balance workload and adjust resources among different sites. There are no extra VMs subscribed during the reconfiguration. Thus, we need to satisfy VM amount constraints, which are given by $\sum_{j \in \mathcal{J}} K_{lij}^{r(\tau_{t_2})} \leq \sum_{j \in \mathcal{J}} K_{lij}^{r(\tau_{t_1})}$ and $\sum_{j \in \mathcal{J}} K_{lij}^{d(\tau_{t_2})} \leq \sum_{j \in \mathcal{J}} K_{lij}^{d(\tau_{t_1})}$.

In multi-site cloud, we jointly optimize the workload transfer among sites and the resource reconfiguration in each site. The optimal resource reconfiguration problem can be formulated as

$$\begin{aligned} &\underset{\{z_{ll'j}^{(\tau_{t_2})}, K_{lij}^{r(\tau_{t_2})}, K_{lij}^{d(\tau_{t_2})}\}}{\text{Minimize}} \quad \sum_{D_l \in \mathcal{D}} \sum_{j \in \mathcal{J}} \left(\frac{\lambda_{lj}^{(\tau_{t_2})}}{\lambda^{(\tau_{t_2})}} \sum_{D_{l'} \in \mathcal{D}} \left(z_{ll'j}^{(\tau_{t_2})} \left(T_{l'j}^{E(\tau_{t_2})} + d_{ll'}^{(\tau_{t_2})} \right) \right) \right) \\ &\text{subject to} \\ &\omega_{lij}^{(\tau_{t_2})} \tilde{\lambda}_{lj}^{(\tau_{t_2})} < (K_{lij}^{r(\tau_{t_2})} + K_{lij}^{d(\tau_{t_2})}) \mu_{ij}, \quad \forall D_l \in \mathcal{D},\ \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\ &\sum_{j \in \mathcal{J}} K_{lij}^{r(\tau_{t_2})} \leq \sum_{j \in \mathcal{J}} K_{lij}^{r(\tau_{t_1})}, \quad \forall D_l \in \mathcal{D},\ \forall i \in \mathcal{I}, \\ &\sum_{j \in \mathcal{J}} K_{lij}^{d(\tau_{t_2})} \leq \sum_{j \in \mathcal{J}} K_{lij}^{d(\tau_{t_1})}, \quad \forall D_l \in \mathcal{D},\ \forall i \in \mathcal{I}, \\ &\sum_{D_{l'} \in \mathcal{D}} z_{ll'j}^{(\tau_{t_2})} = 1, \quad \forall D_l \in \mathcal{D},\ \forall j \in \mathcal{J}, \\ &0 \leq z_{ll'j}^{(\tau_{t_2})} \leq 1, \quad \forall D_l, D_{l'} \in \mathcal{D},\ \forall j \in \mathcal{J}, \\ &K_{lij}^{r(\tau_{t_2})}, K_{lij}^{d(\tau_{t_2})} \geq 0, \quad \forall D_l \in \mathcal{D},\ \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\ &z_{ll'j}^{(\tau_{t_2})} \in \mathbb{R},\ K_{lij}^{r(\tau_{t_2})}, K_{lij}^{d(\tau_{t_2})} \in \mathbb{Z}, \quad \forall D_l, D_{l'} \in \mathcal{D},\ \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \end{aligned} \tag{4.13}$$

The optimal resource reconfiguration problem in Equation (4.13) is a mixed integer nonlinear programming, which is known as NP-hard problem [122, 123]. The problem can be solved by the branch-and-bound method [116]. However, the branch-and-bound method needs to inspect all candidate solutions, which is not suitable for large scale cloud systems. For service providers, they require rapid and efficient resource reconfiguration and load balancing schemes. For this reason, we present a heuristic in Algorithm 4.4,

Algorithm 4.4: Heuristic for Resource Reconfiguration in Multi-site Cloud

1: Calculate utilization for each service before request redirection $\rho_{lj}^{(\tau_{t_2})} = \frac{\lambda_{lj}^{(\tau_{t_2})}}{\sum_{i\in\mathcal{I}}(K_{lij}^{r(\tau_{t_1})}+K_{lij}^{d(\tau_{t_1})})\mu_{ij}}$ and insert $\rho_{lj}^{(\tau_{t_2})}$ into a min heap $\mathbf{H}_{\rho_j}$ [120]. Calculate average utilization $\bar{\rho}_j = \sum_{D_l\in\mathcal{D}} \rho_{lj}^{(\tau_{t_2})}/L$, where L is the number of data centers.
2: **for** each site D_l, $(D_l \in \mathcal{D})$ **do**
3: **while** $\rho_{lj}^{(\tau_{t_2})} - \bar{\rho}_j > \epsilon_j$ **do**
4: Get site $D_{l'}$ which has the lowest utilization for service j from the root of $\mathbf{H}_{\rho_j}$.
5: Redirect type-j service requests from D_l to $D_{l'}$ such that $\rho_{l'j}^{(\tau_{t_2})} - \bar{\rho} = \epsilon_j$
6: Update $\mathbf{H}_{\rho_j}$ to get the new lowest utilization.
7: **end while**
8: **end for**
9: **for** each site D_l, $(D_l \in \mathcal{D})$ **do**
10: Reallocate VMs among services according to the *Heuristic for Resource Reconfiguration in Single-site Cloud* in Algorithm 4.2.
11: **end for**

which can reconfigure VMs and balance load in a practical way. The proposed heuristic firstly balance load among sites according to the difference of resource utilization and then reconfigure resources among services at each site. Moreover, the resource reconfiguration can be performed at each site in parallel after the global workload balancing.

4.5 Performance Evaluation

We verify the proposed schemes by numerical simulations. Amazon EC2 [15] is a popular cloud computing platform allowing service providers to lease VMs for various services. To make our simulations convincible, we apply the price rates and VM configurations of Amazon EC2 in our simulations. Four classes of VM instances are tested, including small, large, extra large standard instances, and extra large high-CPU instance. The detailed configuration and price rates can be found from [15]. As the hourly VM rental is used by Amazon EC2 [15], the resource allocation time scale t_1 is set as 1 hour.

4.5.1 Simulations for Single-site Cloud

In single-site cloud scenario, we select Amazon's data center at Ireland as the site to deploy services. We first verify the proposed resource allocation scheme. Currently, the state-of-the-art resource allocation scheme, adopted by Amazon Elastic Beanstalk [18], is based on utilization threshold, in which the VM provisioning will be triggered when utilization ρ is higher than δ_h and stopped until utilization ρ is lower than or equal to δ_l ($\delta_l < \delta_h$). We compare the resource cost among the following alternative schemes: 1) the optimal allocation scheme, in which VMs are allocated optimally by solving the optimization problem in Equation (4.2), 2) the proposed heuristic in Algorithm 4.1, and 3) the utilization scheme [18] with δ_l and δ_h set as 0.5 and 0.7, respectively.

Figure 4.3(a) presents the comparison of resource cost among the three schemes. We can see that the optimal resource allocation scheme can achieve the lowest resource cost in Figure 4.3(a). Compared with the utilization scheme [18], the proposed heuristic can provide satisfactory services at a lower resource cost. We analyze the reason. The proposed heuristic is based on a greedy idea, which only considers the current best choice without making a global inspection, while the optimal allocation scheme searches the whole feasible region to find the globally optimal solution. Thus, the proposed heuristic is a sub-optimal but lightweight solution. The utilization scheme [18] only guarantees that the utilization at each service is lower than the upper bound. However, it fails to assign the most suitable VMs to each service. Figure 4.3(b) gives a close look at the resource cost in each service when the total load reaches 15000 requests/s. Compared with the utilization scheme [18], the optimal allocation scheme and the proposed heuristic consume a lower resource cost. We also record the arrivals of requests in a 9-hour period. During the period, the mean request arrival rate is varied as in Figure 4.3(c). The corresponding resource cost among the three schemes is compared in Figure 4.3(d). From Figure 4.3(d), we find that the proposed heuristic performs close to the optimal allocation scheme.

Next, we evaluate the proposed optimal resource reconfiguration scheme for single-site cloud. In the simulation, we compare the average response time among the following schemes: 1) the optimal reconfiguration scheme, in which VMs are reconfigured by solving the optimization problem in Equation (4.5), 2) the proposed heuristic in Algorithm 4.2, and 3) the utilization scheme [18], in which the initially allocated VMs for each service remain unchanged during the one hour period. Figure 4.4(a) displays a close view of

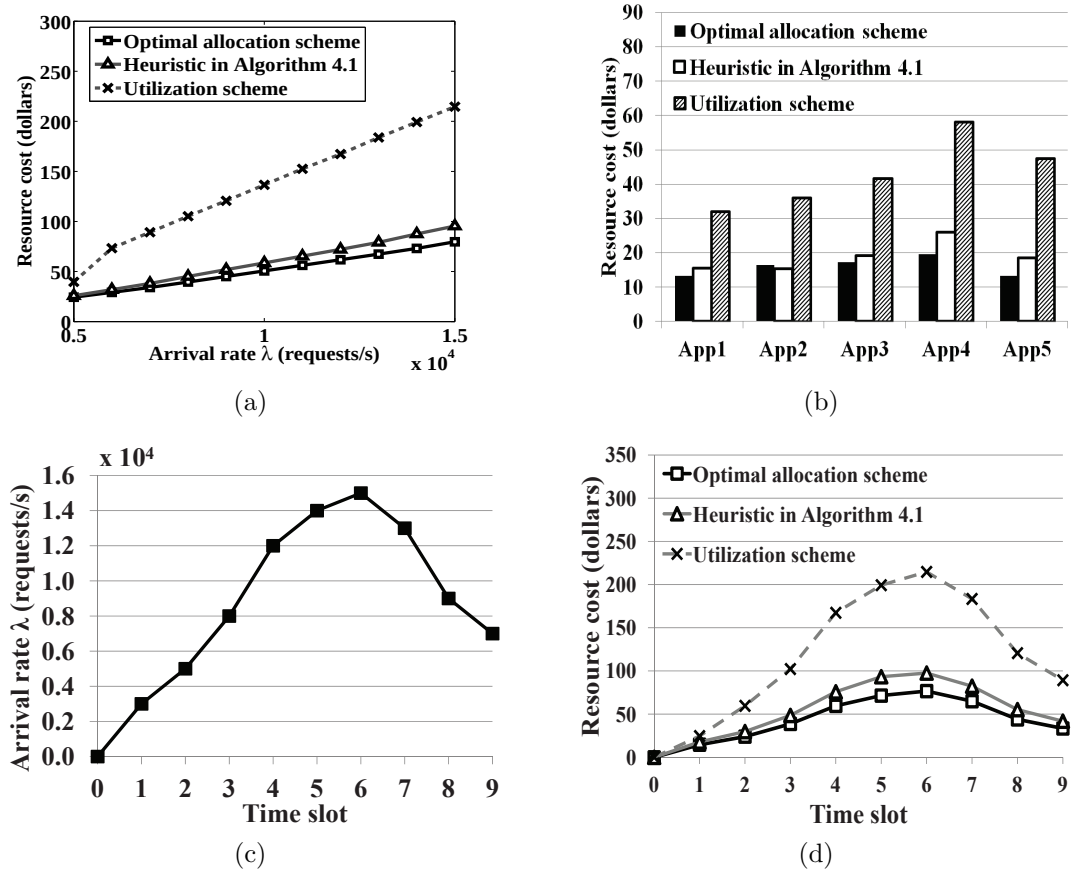


Figure 4.3: Simulation results of resource allocation for single-site cloud: (a) comparison of resource cost when λ varies from 5000 requests/s to 15000 requests/s, (b) comparison of resource cost in each service when λ=15000 requests/s, (c) mean request arrival rate in a 9-hour period, and (d) comparison of resource cost in the 9-hour period.

the time-varying request arrival rate during the 5th hour in Figure 4.3(c), during which the arrival rate increases from 12000 requests/s to 14000 requests/s. We verify the proposed resource reconfiguration scheme on the time-varying workload in Figure 4.4(a). The t_2 scale is set as 5 minutes. The comparison of the average response time among the three schemes is shown in Figure 4.4(b). From Figure 4.4(b), we can find that the optimal reconfiguration scheme reaches the lowest average response time compared to the other two schemes. The utilization scheme does not adjust VMs to adapt to the time-varying workload, leading to the unbalanced resource utilization. The proposed heuristic inspects the utilization in each service and correspondingly adjusts VMs to balance the

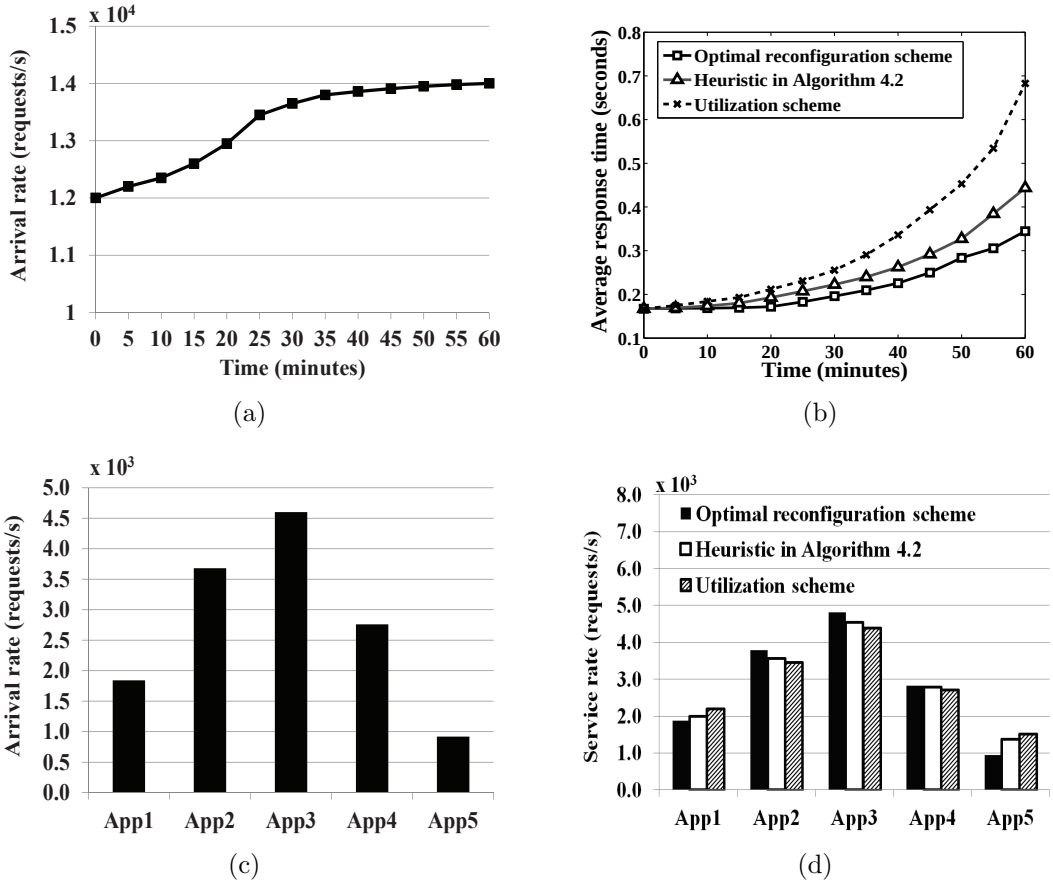
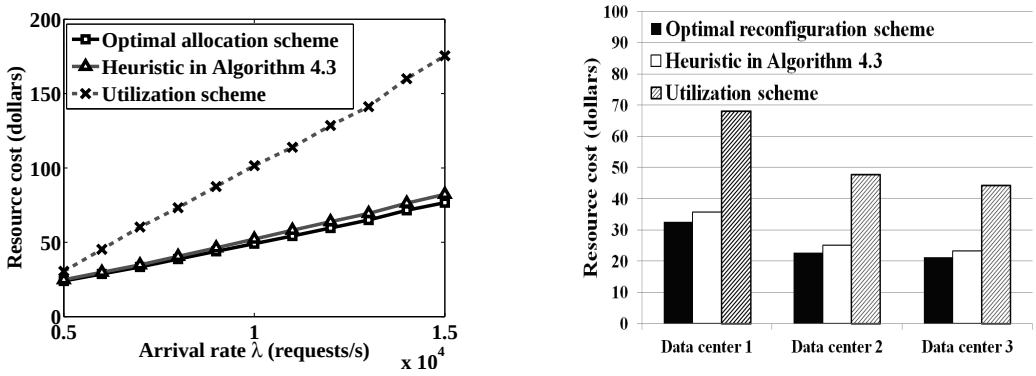


Figure 4.4: Simulation results of resource reconfiguration for single-site cloud: (a) request arrival rate during the 5th hour in Figure 4.3(c), (b) comparison of average response time among the three schemes, (c) request arrivals for each service between 50-55 minutes, and (d) total service rate of allocated VMs for each service.

workload. Thus, the proposed heuristic can reduce average response time compared to the utilization scheme. In addition, we show the arrivals of requests for each service between 50-55 minutes in Figure 4.4(c). To have a close view on the resource allocations at that time slot, we show the service rate for each service in Figure 4.4(d). Compared with the utilization scheme, the optimal reconfiguration scheme and the heuristic allocate more resources to the heavy services (e.g. App2 and App3) while less resources to the light services (e.g. App1 and App5), thus reducing the average response time.

4.5.2 Simulations for Multi-site Cloud

We perform simulations to evaluate the proposed schemes for multi-site cloud scenario. In our simulations, three Amazon data centers are employed, in which D_1 is located at Tokyo, D_2 is located at Ireland which is also used in Section 4.5.1, and D_3 is located at East Virginia. In terms of price rates of VM instances, D_1 is the most expensive site, while D_3 is the cheapest site. The detailed price rates at each site can be found from [15].



(a) (b)

Figure 4.5: Simulation results of resource allocation for multi-site cloud: (a) comparison of resource cost when λ varies from 5000 requests/s to 15000 requests/s, and (b) comparison of resource cost at each data center when $\lambda = 15000$ requests/s.

Firstly, we evaluate the proposed resource allocation scheme for multi-site cloud. We compare the total resource cost among: 1) the optimal allocation scheme, in which the VMs are determined by solving the optimization problem in Equation (4.9), 2) the proposed heuristic in Algorithm 4.3, and 3) the utilization scheme in [18]. Figure 4.5(a) shows the comparison of resource cost among the three schemes. From Figure 4.5(a), we can see that the proposed optimal resource allocation scheme can provide services at a lower resource cost than the heuristic and the utilization scheme [18]. Meanwhile, we also find that the proposed heuristic performs close to the optimal resource allocation scheme. The optimal allocation scheme provides the globally optimal benchmark but takes long time to converge, while the proposed heuristic is sub-optimal but lightweight and efficient. Figure 4.5(b) compares the resource cost at each data center when $\lambda = 15000$ requests/s. From Figure 4.5(b), we can find that the optimal allocation scheme reaches the lowest cost at each data center, while the proposed heuristic has a lower cost than the utilization

scheme.

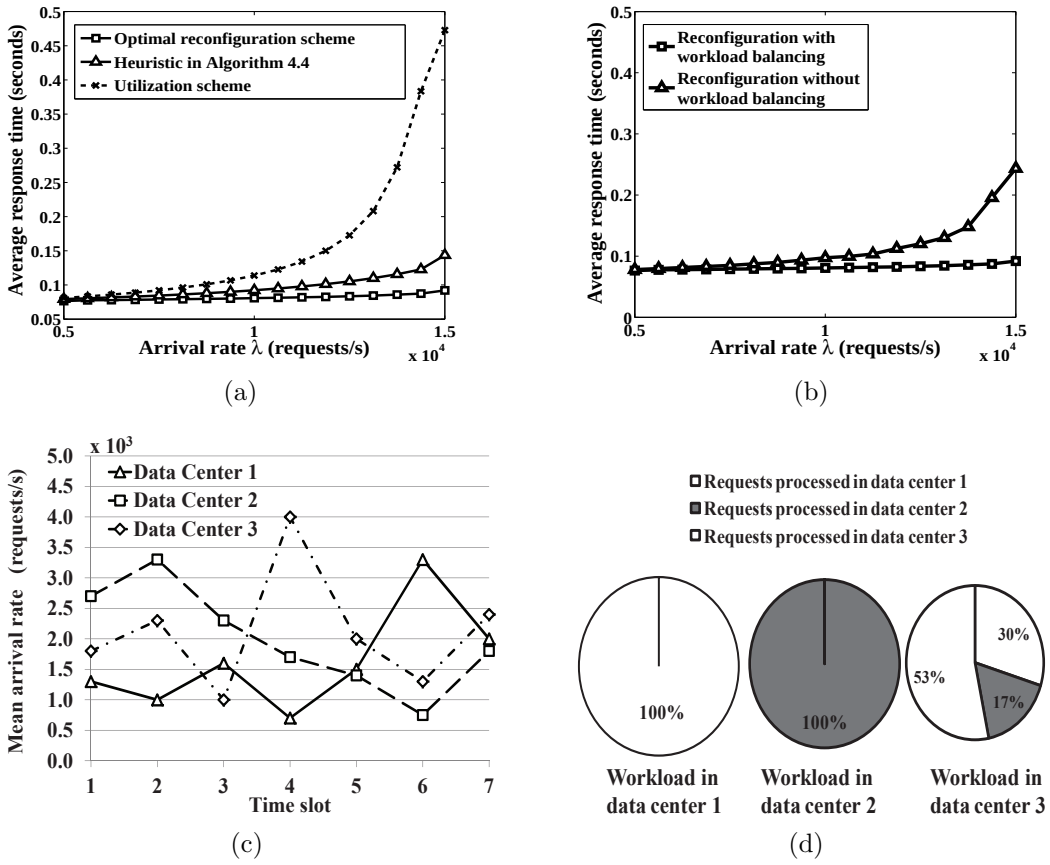


Figure 4.6: Simulation results of resource reconfiguration for multi-site cloud: (a) comparison of average response time among the three schemes, (b) comparison of average response time between the resource reconfiguration without workload balancing and that with workload balancing, (c) request arrivals in each data center, and (d) workload assignment in multi-site cloud at the 4th time slot.

Next, we evaluate the resource reconfiguration scheme for multi-site cloud. We compare the average response time among the following schemes: 1) the proposed optimal resource reconfiguration scheme, in which the workload assignment and the local VM allocation are determined by solving the optimization problem in Equation (4.12), 2) the proposed heuristic in Algorithm 4.4, and 3) the utilization scheme, in which the initial VM allocation at each site remains unchanged. The comparison results of the average response time among the three schemes are shown in Figure 4.6(a). From Figure 4.6(a),

we can see that the optimal reconfiguration scheme takes the lowest average response time. Since the utilization scheme keeps the allocated resources unchanged, resources among services cannot be dynamically adjusted, leading to a higher response time. The proposed heuristic reconfigures both workload assignment and local VM allocation according to the utilization. Thus, the proposed heuristic can achieve lower response time compared with the utilization scheme. We also compare the average response time between the optimal resource reconfiguration with workload balancing and that without workload balancing. The comparison results are shown in Figure 4.6(b). From Figure 4.6(b), we can find that the resource reconfiguration with workload balancing can achieve lower average response time. Since we jointly optimize the workload assignment and the resource reconfiguration, each request can be assigned to the most appropriate site for service and the allocated VMs in each site can be optimally utilized. Therefore, the proposed resource configuration with workload balancing can achieve the globally minimal average response time. We also evaluate the performance of workload balancing in the proposed resource reconfiguration scheme. The arrivals of requests at each data cent are shown in Figure 4.6(c). At the 4th time slot in Figure 4.6(c), data centers 1 and 2 have light workload, while data center 3 reaches a peak of 4000 requests/s. By applying the proposed resource reconfiguration scheme, the workload is assigned as shown in Figure 4.6(d). From Figure 4.6(d), we can find that data centers 1 and 2 process all their local requests and help data center 3 to serve partial workload such that data center 3 can effectively avoid the local congestion.

4.6 Chapter Summary

In this chapter, we study the dynamic resource configuration for cloud based multimedia services. We first propose a TRC scheme, which considers resource configuration in two different time scales. Specifically, the proposed scheme captures the pattern of workload and performs resource allocation in a mid-long time scale, while in a fine-grained time scale, the proposed scheme dynamically reallocates resources to cope with the varying workload. Based on the TRC scheme, we study resource allocation and resource reconfiguration problems in single-site cloud scenario and multi-site cloud scenario, respectively. In each scenario, we investigate the resource allocation problem, which optimally allo-

cates VMs to achieve the minimal resource cost, and the resource reconfiguration problem, which dynamically adjusts VMs according to varying demands to minimize the average response time. Since the formulated optimization problems are nonlinear integer programming, we propose heuristics to efficiently allocate resources in real time. Simulation results demonstrate that the proposed dynamic resource configuration schemes not only allocate resources effectively to achieve a low resource cost, but also reconfigure resources dynamically to reach low average response time.

Chapter 5

Optimization of Workload Scheduling over Multimedia Cloud

Given the allocated cloud resources, another challenge for multimedia service providers (MSPs) is how to effectively schedule workload to diverse virtual machines (VMs) for distributed processing. In this chapter, we study the optimization of workload scheduling at two different levels: the *user level scheduling* and the *task level scheduling*. Simulation results demonstrate that the proposed schemes can optimize workload scheduling to achieve the minimal response time or the minimal resource cost.

5.1 Introduction

Workload scheduling aims to enhance the overall system performance by balancing workload among VMs. Generally, there are two levels of workload scheduling in cloud. The first level is the user level scheduling, where requests for a service are distributed to different virtual clusters according to the load intensity at each cluster. By balancing user requests, the user level scheduling can effectively avoid local congestions. Compared to the user level scheduling, the task level scheduling is performed in a finer granularity. In general, a multimedia service can be decomposed into a set of tasks. Some tasks can run in parallel, while some tasks must be processed sequentially. The goal of task level scheduling is to assign tasks to VMs for the efficient execution. The two-level workload scheduling is performed in a hierarchical way.

However, there are challenges to achieve the optimal workload scheduling. First of all, VMs have heterogeneous resource capacities and thus process users' requests at different rates. It is difficult to schedule workload to match the service rate at each virtual server. Secondly, the workload varies rapidly. Accordingly, the workload scheduling scheme needs to adapt to the varying workload. Thirdly, there are diverse precedence constraints among tasks in a multimedia service. For instance, Figure 5.1 shows a video retrieval service [124], where the concept detection cannot be executed until visual features have been extracted. When assigning tasks to VMs, it is difficult to satisfy all these precedence constraints. Fourthly, multimedia services have different operation structures. Generally, there are three basic structures: *sequential*, *parallel*, and *mixed* structures. In the sequential structure, tasks are executed in a serial order, where a task cannot be started until all of its previous tasks have been completed. In the parallel structure, tasks can be performed concurrently. Mixed structure combines both sequential and parallel structures. It is a challenge to find an effective scheduling scheme for different structures.

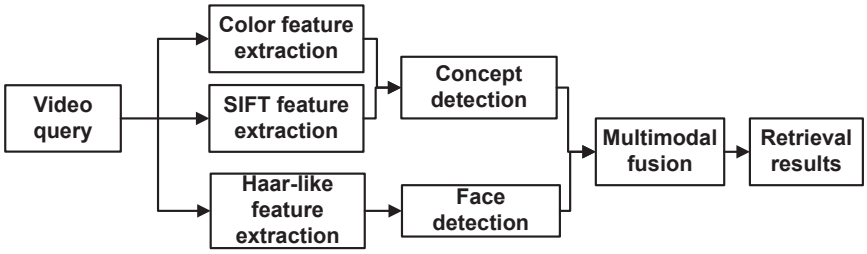


Figure 5.1: Illustration of video retrieval service.

To address the above mentioned challenges, we investigate the optimal workload scheduling problems at the user level and the task level, respectively. The major contributions can be summarized as follows.

- We examine two workload scheduling problems at user level. The first problem is the response time minimization problem, which optimizes the workload scheduling weights to minimize the response time. The formulated problem is a convex optimization. We derive the optimal analytical solution. The second problem is the resource cost minimization problem, which jointly optimizes the workload scheduling weights and the required VMs to minimize the cost. The resource cost minimization problem is a mixed integer non-linear programming. We propose a

greedy algorithm to efficiently schedule the workload.

- We also investigate the optimal workload scheduling problem at task level. A directed acyclic graph is introduced to characterize the precedence constraints among tasks. Based on the model, we optimize the task level scheduling for the sequential, parallel, and mixed structures, respectively. In each structure, we optimize the task assignment on different VMs to minimize the total execution time. Additionally, we propose a heuristic to efficiently achieve the optimal task level scheduling.

The remainder of this chapter is organized as follows. The optimization of workload scheduling at user level and task level are presented in Section 5.2 and 5.3, respectively. The chapter is summarized in Section 5.4

5.2 Optimization of User Level Workload Scheduling

5.2.1 User Level Workload Scheduling Model

Since the workload keeps varying, we divide the time domain into slots and denote by t the time slot for the workload scheduling in our model. Suppose that N classes of VMs are available in a service. A number of VMs in the same class work together as the virtual cluster [93] to provide a faster service.

Figure 5.2 shows the workload scheduling model. S is the workload scheduler and $C_1, \ldots, C_N$ are the virtual clusters. In time slot t, the numbers of allocated class-i VMs in the reservation scheme and the on-demand scheme are denoted by $K_i^{r(t)}$ and $K_i^{d(t)}$, respectively. Similar to assumptions in literatures [61, 115], we assume the service time at each cluster is exponential distribution with a mean service rate $(K_i^{r(t)} + K_i^{d(t)})\mu_i$. The arrivals of requests in time slot t are assumed to be a Poisson process with an average of $\lambda^{(t)}$. As shown in Figure 5.2, the incoming requests are distributed to the class-i virtual cluster with the corresponding scheduling weight $\omega_i^{(t)}$ $(\forall i = 1, 2, \ldots, N)$. Thus, the arrivals of scheduled requests to class-i virtual cluster also follow a Poisson Process with an average of $\omega_i^{(t)}\lambda^{(t)}$. The service process at class-i virtual cluster can be modeled as an $M/M/1$ queueing system [114]. To make the queueing system stable, $\omega_i^{(t)}\lambda^{(t)} < (K_i^{r(t)} + K_i^{d(t)})\mu_i$ is required. The response time T_i at the class-i virtual cluster

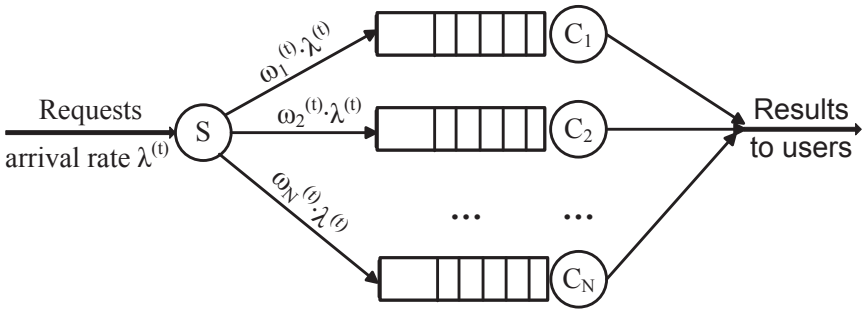


Figure 5.2: The workload scheduling model for cloud based multimedia service.

can be formulated as $T_i = \frac{1}{(K_i^{r(t)}+K_i^{d(t)})\mu_i - \omega_i^{(t)}\lambda^{(t)}}$, and thus the mean response time is given by $T = \sum_{i=1}^{N} \omega_i T_i = \sum_{i=1}^{N} \frac{\omega_i^{(t)}}{(K_i^{r(t)}+K_i^{d(t)})\mu_i - \omega_i^{(t)}\lambda^{(t)}}$.

5.2.2 Response Time Minimization Problem

In multimedia cloud, MSPs hope to balance the workload to provide services at a low delay. However, it is challenging to determine the optimal workload scheduling weights, due to the heterogeneous resource capacities and varying demands. Therefore, we formulate the response time minimization problem, which can be stated as: to minimize the mean response time by optimizing the workload scheduling weights, subject to the queueing stability constraint, the workload conservation constraint, and the scheduling weight constraints. Mathematically, the response time minimization problem can be formulated as follows.

$$\begin{aligned} \underset{\{\omega_1,\omega_2,\ldots,\omega_N\}}{\text{Minimize}} \quad & \sum_{i=1}^{N} \frac{\omega_i^{(t)}}{(K_i^{r(t)}+K_i^{d(t)})\mu_i - \omega_i^{(t)}\lambda^{(t)}} \\ \text{subject to} \quad & \\ & \omega_i^{(t)}\lambda^{(t)} < (K_i^{r(t)} + K_i^{d(t)})\mu_i, \\ & \forall i = 1, \ldots, N, \\ & \sum_{i=1}^{N} \omega_i = 1, \\ & \omega_i \geq 0, \ \forall i = 1, \ldots, N. \end{aligned} \tag{5.1}$$

In Equation (5.1), the objective function is the mean response time. The constraints $\omega_i^{(t)}\lambda^{(t)} < (K_i^{r(t)} + K_i^{d(t)})\mu_i$ $(\forall i = 1, \ldots, N)$ represent the queueing stability constraint at each virtual cluster. The constraint $\sum_{i=1}^{N} \omega_i = 1$ is the workload conservation constraint, i.e. all workloads should be scheduled. The constraints $\omega_i \geq 0$ $(\forall i = 1, \ldots, N)$ represent

the scheduling weight constraints.

The response time minimization problem in Equation (5.1) is a convex optimization problem [100]. We use the Lagrange multiplier method [100] to solve the problem and get the optimal analytical solution to the response time minimization problem in Equation (5.1) as follows.

$$\omega_i^{(t)} = \frac{\lambda^{(t)}\sqrt{\xi_i^{(t)}}+\xi_i^{(t)}\sum_{j=1}^{N}\sqrt{\xi_j^{(t)}}-\sqrt{\xi_i^{(t)}}\sum_{j=1}^{N}\xi_j^{(t)}}{\lambda^{(t)}\sum_{j=1}^{N}\sqrt{\xi_j^{(t)}}}, \quad \forall i = 1, 2, \ldots, N, \tag{5.2}$$

where $\xi_i^{(t)} = (K_i^{r(t)} + K_i^{d(t)})\mu_i$ represents the total service rate at the class-i virtual cluster.

Intuitively, the workload scheduling weight should be proportional to the service rate $\xi_i^{(t)}$ at the class-i virtual cluster. A cluster with a higher service rate should be assigned with more requests to process, while a cluster with a lower service rate should be assigned with less workload. Thus, the intuitive heuristic scheduling scheme uses the normalized service rate $\frac{\xi_i^{(t)}}{\sum_{j=1}^{N}\xi_j^{(t)}}$ as the scheduling weight. Compared to the optimal scheduling scheme, the heuristic scheduling scheme is a lightweight but sub-optimal solution. The performance comparison between the optimal scheduling scheme and the heuristic is presented in Section 5.2.4.

5.2.3 Resource Cost Minimization Problem

MSPs always want to provide satisfactory services at the minimal resource cost. Suppose that the price rates of one class-i VM instance in the reservation plan and the on-demand plan are p_i^r and p_i^d, respectively. Thus, the total resource cost at time slot t can be given by $\sum_{i=1}^{N}(p_i^r K_i^{r(t)} + p_i^d K_i^{d(t)})$. The resource cost minimization problem can be stated as: to minimize the total resource cost by jointly optimizing the workload scheduling weights and the required VMs, subject to the response time constraint, the queueing stability constraints, the VM reservation constraint, the workload conservation constraint, and the workload scheduling weight constraints. Mathematically, the resource cost minimization

problem can be formulated as follows.

$$\begin{aligned}
&\underset{\left\{\substack{K_1^{r(t)},\ldots,K_N^{r(t)},\\ K_1^{d(t)},\ldots,K_N^{d(t)},\\ \omega_1,\ldots,\omega_N}\right\}}{\text{Minimize}} && \sum_{i=1}^{N}\left(p_i^r K_i^{r(t)} + p_i^d K_i^{d(t)}\right) \\
&\text{subject to} && \\
& && \sum_{i=1}^{N} \frac{\omega_i^{(t)}}{(K_i^{r(t)}+K_i^{d(t)})\mu_i - \omega_i^{(t)}\lambda^{(t)}} \leq \tau, \\
& && \omega_i^{(t)}\lambda^{(t)} < (K_i^{r(t)} + K_i^{d(t)})\mu_i, \quad \forall i = 1,\ldots,N, \\
& && K_i^{r(t)} \leq K_i^{ini}, \quad \forall i = 1,\ldots,N, \\
& && \sum_{i=1}^{N} \omega_i = 1, \\
& && \omega_i \geq 0, \quad \forall i = 1,\ldots,N,
\end{aligned} \tag{5.3}$$

where τ is the upper bound of the application response time, and K_i^{ini} is the initially reserved class-i VMs.

In Equation (5.3), the objective function is the total resource cost. The constraint $\sum_{i=1}^{N} \frac{\omega_i^{(t)}}{(K_i^{r(t)}+K_i^{d(t)})\mu_i - \omega_i^{(t)}\lambda^{(t)}} \leq \tau$ represents the response time constraint. The constraints $\omega_i^{(t)}\lambda^{(t)} < (K_i^{r(t)} + K_i^{d(t)})\mu_i$ $(\forall i = 1,\ldots,N)$ are the queueing stability constraints. The constraint $K_i^{r(t)} \leq K_i^{ini}$ is the class-i VM reservation constraint, i.e. the utilized class-i VMs from the reservation scheme cannot exceed the initially reserved class-i VMs. The constraint $\sum_{i=1}^{N} \omega_i = 1$ is the workload conservation constraint. The constraints $\omega_i \geq 0$ $(\forall i = 1,\ldots,N)$ are the workload scheduling weight constraints.

The resource cost minimization problem in Equation (5.3) is a mixed integer nonlinear programming, which is known to be NP-hard [116]. The problem can be solved by the branch-and-bound method [116]. But MSPs require a rapid and efficient workload scheduling scheme, which can quickly adapt to the time-varying workload. Therefore, we propose a greedy algorithm, which is presented in Algorithm 5.1.

5.2.4 Simulations

In this section, we perform simulations to evaluate the proposed optimal workload scheduling schemes. Amazon EC2 [15] is Amazon's cloud computing platform. We employ the price rates and VM configurations of Amazon EC2 in the simulation. Three classes of VMs are used. The reservation and on-demand price rates for the three classes of VMs

Algorithm 5.1: Greedy Algorithm for Joint Workload Scheduling and VM Allocation

1: Compute $q_i^r = \frac{p_i^r}{\mu_i}$ and $q_i^d = \frac{p_i^d}{\mu_i}$, which are the cost rates of using one reserved and on-demand class-i VM to process one unit request, respectively. Let set $Q = \{q_1^r, q_1^d, \ldots, q_N^r, q_N^d\}$.
2: Sort set Q in an ascending order.
3: **repeat**
4: Select the smallest q_i^v ($\forall i = 1, 2, \ldots, N$, $v = r$ or d) from the set Q
5: **if** q_i^v from the reserved VMs (i.e. $v = r$) **then**
6: Schedule the user requests $\lambda_i^{r(t)}$ to the selected class-i reserved VMs as long as the requirements $\frac{1}{K_i^{r(t)}\mu_i - \lambda_i^{r(t)}} \leq \tau$ and $K_i^{r(t)} \leq K_i^{ini}$ are satisfied. Update $\lambda^{(t)} = \lambda^{(t)} - \lambda_i^{r(t)}$.
7: **else if** q_i^v from the on-demand VMs (i.e. $v = d$) **then**
8: Schedule all unscheduled user requests $\lambda^{(t)}$ as $\lambda_i^{d(t)}$ to the selected class-i on-demand VMs until the requirement $\frac{1}{K_i^{r(t)}\mu_i - \lambda_i^{(t)}} \leq \tau$ is satisfied. Update $\lambda^{(t)} = 0$.
9: **end if**
10: **until** all requests are processed (i.e. $\lambda^{(t)} = 0$).
11: Compute the scheduling weight $\omega_i = \frac{\lambda_i^{r(t)} + \lambda_i^{d(t)}}{\lambda^{(t)}}$, and the total resource cost $C^{(t)} = \sum_{i=1}^{N} \left(p_i^r K_i^{r(t)} + p_i^d K_i^{d(t)} \right)$.

are $p^r = \{0.05\$/\text{h}, 0.20\$/\text{h}, 0.40\$/\text{h}\}$ and $p^d = \{0.085\$/\text{h}, 0.34\$/\text{h}, 0.68\$/\text{h}\}$, respectively. The numbers of initially reserved VMs are $K^{ini} = \{60, 30, 20\}$, and the service rates for each class of VM instance are $\mu = \{25$ requests/s, 97 requests/s, 185 requests/s$\}$.

We first compare the response time between the proposed optimal scheduling scheme, which is the optimal solution in Equation (5.2), and the heuristic scheduling scheme, in which the scheduling weight is the normalized service rate $\frac{\xi_i}{\sum_{i=1}^{N} \xi_i}$. The optimal scheduling scheme is the globally optimal benchmark, while the heuristic scheduling scheme is a sub-optimal but lightweight solution. The comparison of the response time between the two schemes is shown in Figure 5.3. The mean request arrival rate increases from 4000 requests/s to 5000 requests/s. From Figure 5.3, we can see that the proposed optimal scheduling scheme achieves lower response time compared to the heuristic scheduling scheme under the same request arrival rate. Figure 5.3 also shows that the heuristic scheduling scheme using the normalized service rate as the scheduling weight performs

very close to the optimal scheduling scheme when the system workload is light. The difference of the response time between the two schemes is increased when the workload becomes heavier. When the arrival rate is 5000 requests/s, the difference of the response time between the two schemes is 0.007 seconds.

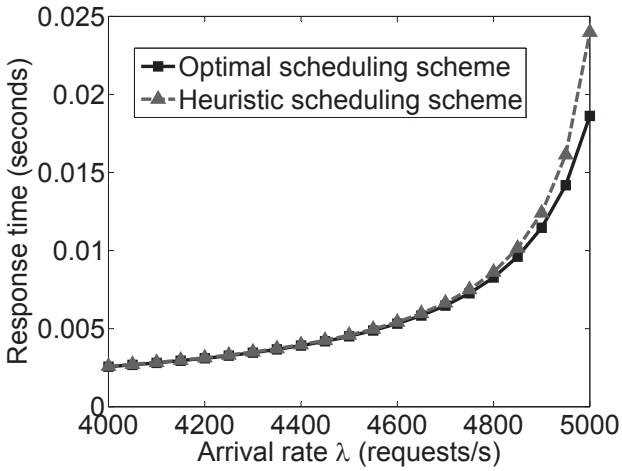


Figure 5.3: Comparison of response time between the optimal scheduling scheme and the heuristic scheduling scheme.

Next, we compare the resource cost between the proposed optimal scheduling scheme, in which the scheduling weights and the required VMs are optimally obtained by solving the optimization problem in Equation (5.3), and the proposed greedy algorithm. The optimal scheduling scheme is the globally optimal benchmark but not practical, while the greedy algorithm is sub-optimal but efficient and practical. The comparison of the resource cost between the proposed optimal scheduling scheme and the proposed greedy algorithm is shown in Figure 5.4. The mean request arrival rate increases from 1000 requests/s to 5000 requests/s. From Figure 5.4, we can see that the optimal scheduling scheme can achieve a lower resource cost compared to the greedy algorithm under the same request arrival rate. Moreover, Figure 5.4 shows that the proposed greedy algorithm has a close performance to the optimal benchmark. The greedy algorithm only considers the current best choice but fails to make a global inspection, while the optimal scheduling scheme searches the whole feasible region to find the globally optimal solution. Thus, the greedy algorithm achieves the sub-optimal solution with a lighter computation.

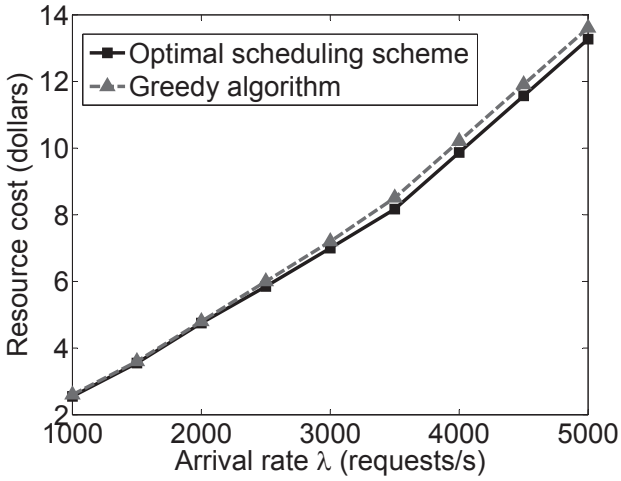


Figure 5.4: Comparison of resource cost between the optimal scheduling scheme and the greedy algorithm.

5.3 Optimization of Task Level Workload Scheduling

In this section, we study the task level scheduling problem. We propose an optimal workload scheduling scheme to assign each task to different VM for distributed processing

5.3.1 Task Level Workload Scheduling Model

To characterize precedence constraints among multimedia tasks, we introduce a directed acyclic graph (DAG), which is a directed graph with no path that starts and ends at the same vertex. The DAG can be denoted as: $DAG = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of vertices and $\mathbb{E}$ is the set of edges. Suppose that a multimedia service can be decomposed into K tasks. Each vertex in DAG represents a task. Thus, $\mathbb{V} = \{V_1, V_2, \ldots, V_K\}$ represents the set of tasks. Each edge in DAG characterizes a precedence constraint between two tasks. The edge $E_{k'k}$ represents that task V_k cannot be executed until task $V_{k'}$ has finished. Thus, the execution of task $V_{k'}$ is a precedence constraint for task V_k. Figure 5.5 illustrates the DAG of the video retrieval framework in Figure 5.1. There are 6 tasks in Figure 5.5, and the video query and the retrieval result are denoted as source and sink, which represent the start and the end of DAG.

Suppose that there are N types of VM instances. Each task can be served by any VM instance. Let t_j^k denote the execution time by using type-j VM instance to execute

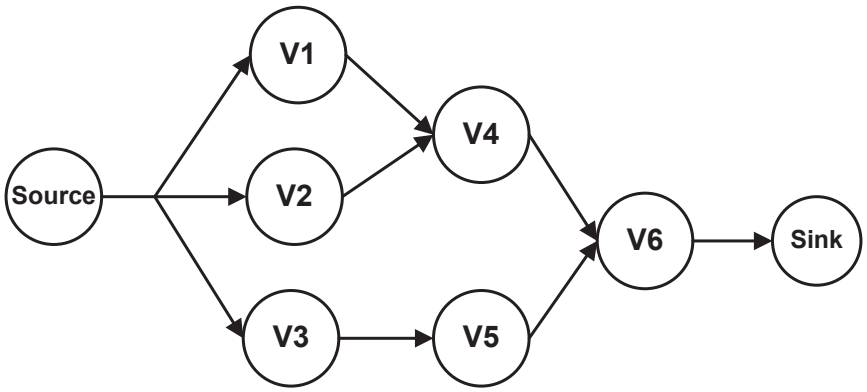


Figure 5.5: The DAG of the video retrieval framework in Figure 5.1.

task V_k, and let p_j be the resource cost for renting one type-j VM instance. We assume that each task can only be assigned to one VM instance for execution. Therefore, the purpose of task-level scheduling is to optimally assign tasks to VMs. To represent the task assignment, we introduce s_j^k, which is given by

$$s_j^k = \begin{cases} 1 & \text{if } V_k \text{ is assigned to a type-}j \text{ VM instance,} \\ 0 & \text{otherwise.} \end{cases}$$

We will propose the optimal task scheduling scheme to determine the optimal value of s_j^k $(V_k \in \mathbb{V}, \forall j = 1, 2, \ldots, N)$ to minimize the total execution time.

5.3.2 Optimal Task Level Scheduling for Sequential Structure

We first investigate the sequential structure, in which all tasks are executed serially. Our objective is to minimize the total execution time. As described in Section 5.3.1, one task can only be assigned to one VM instance. If the task V_k is scheduled to type-j VM instance, $s_j^k = 1$ and $s_{j'}^k = 0$ $(j' \neq j)$. Thus, the execution time of V_k is given by $\sum_{j=1}^{N} s_j^k t_j^k$, which represents that the type-j VM instance takes t_j^k to execute V_k, if V_k is scheduled to the VM instance. Since all tasks are executed serially in the sequential structure, the total execution time is the sum of the execution time for each task. Therefore, the total execution time is given by $T_{seq}^{tot} = \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k t_j^k$. The cost for processing V_k is denoted by $\sum_{j=1}^{N} s_j^k p_j$. Thus, the total resource cost for serving the multimedia application is $C_{seq}^{tot} = \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j$. In addition, every task should be assigned to one VM instance

for execution. Thus, constraints $\sum_{j=1}^{N} s_j^k = 1$ ($V_k \in \mathbb{V}$) have to be satisfied. Based on the above analysis, we can formulate the task schedule optimization problem for the sequential structure as follows.

$$\begin{aligned} \underset{\{s_j^k\}}{\text{Minimize}} \quad & \textstyle\sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k t_j^k \\ \text{subject to} \quad & \\ & \textstyle\sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j \leq \mathcal{C}_{max}, \\ & \textstyle\sum_{j=1}^{N} s_j^k = 1, \\ & s_j^k \in \{0,1\},\ V_k \in \mathbb{V},\ \forall j = 1, \ldots, N, \end{aligned} \tag{5.4}$$

where $\mathcal{C}_{max}$ is the upper bound of resource cost. The optimization problem in Equation (5.4) is a 0-1 integer programming, which can be solved by enumerating every possible s_j^k. But the time complexity of enumeration cannot satisfy the real-time requirement for the practical application. Therefore, we propose a heuristic in Section 5.3.5 to efficiently solve problem in Equation (5.4).

5.3.3 Optimal Task Level Scheduling for Parallel Structure

In this subsection, we study the optimal task-level scheduling for the parallel structure, in which all tasks can be executed concurrently. Thus, the application can deploy different tasks to different VMs for distributed processing. According to Section 5.3.1, the execution time of V_k is given by $\sum_{j=1}^{N} s_j^k t_j^k$. The total execution time in the parallel structure depends on the largest execution time among all tasks, which can be formulated as $T_{par}^{tot} = \max_{\{V_k \in \mathbb{V}\}} \{\sum_{j=1}^{N} s_j^k t_j^k\}$. The total resource cost is given by $C_{par}^{tot} = \sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j$. Additionally, constraints $\sum_{j=1}^{N} s_j^k = 1$ ($V_k \in \mathbb{V}$) are required to guarantee all tasks are executed. Therefore, the task schedule optimization problem for the parallel structure can be formulated as

$$\begin{aligned} \underset{\{s_j^k\}}{\text{Minimize}} \quad & \max_{\{V_k \in \mathbb{V}\}} \{\textstyle\sum_{j=1}^{N} s_j^k t_j^k\} \\ \text{subject to} \quad & \\ & \textstyle\sum_{k=1}^{K} \sum_{j=1}^{N} s_j^k p_j \leq \mathcal{C}_{max}, \\ & \textstyle\sum_{j=1}^{N} s_j^k = 1, \\ & s_j^k \in \{0,1\},\ V_k \in \mathbb{V},\ \forall j = 1, \ldots, N. \end{aligned} \tag{5.5}$$

The optimization problem in Equation (5.5) is a 0-1 integer programming, which can be solved by the enumeration method.

5.3.4 Optimal Task Level Scheduling for Mixed Structure

We extend our study to the mixed structure in this subsection. In the mixed structure, some tasks can be processed in parallel, while some tasks must be executed serially. The video retrieval framework in Figure 5.1 is an example of mixed structure. In the DAG of mixed structure, there are multiple paths from source to sink. The tasks in each path must be executed sequentially, while different paths can be taken as parallel structures. Suppose that there are W paths in a DAG, and let Ψ_w denote the set of vertices on the w^{th} path. The execution time of V_k is given by $\sum_{j=1}^{N} s_j^k t_j^k$, and thus the execution time for Ψ_w can be formulated as $\sum_{k\in\Psi_w}\sum_{j=1}^{N} s_j^k t_j^k$. Therefore, the total execution time is $T_{mix}^{tot} = \max_{\{w\in W\}}\left\{\sum_{V_k\in\Psi_w}\sum_{j=1}^{N} s_j^k t_j^k\right\}$, which represents the largest execution time among all paths. The total resource cost is $C_{mix}^{tot} = \sum_{k=1}^{K}\sum_{j=1}^{N} s_j^k p_j$. Moreover, we need to satisfy constraints $\sum_{j=1}^{N} s_j^k = 1$ ($V_k \in \mathbb{V}$) to process all tasks. Therefore, we can formulate the optimal task schedule problem for the mixed structure as follows.

$$
\begin{aligned}
\underset{\{s_j^k\}}{\text{Minimize}} \quad & \max_{\{w\in W\}}\left\{\sum_{V_k\in\Psi_w}\sum_{j=1}^{N} s_j^k t_j^k\right\} \\
\text{subject to} \quad & \\
& \textstyle\sum_{k=1}^{K}\sum_{j=1}^{N} s_j^k p_j \le \mathcal{C}_{max}, \\
& \textstyle\sum_{j=1}^{N} s_j^k = 1, \\
& s_j^k \in \{0,1\},\ V_k \in \mathbb{V}, \\
& \forall j = 1,2,\ldots,N,\ \forall w = 1,2,\ldots,W.
\end{aligned}
\tag{5.6}
$$

The optimization problem in Equation (5.6) is a 0-1 integer programming.

5.3.5 Heuristic for Optimal Task Level Scheduling

The task schedule optimization problems in Equations (5.4), (5.5), and (5.6) are all 0-1 integer programming, which is NP-complete [105]. By exhaustively searching along both task dimension and VM dimension, the optimal solutions can be achieved. However, the enumeration method is quite inefficient, especially when the number of tasks is huge.

Therefore, we propose a heuristic to efficiently approach the optimal task scheduling. The core of the heuristic is to find critical tasks [125], which are on the longest path and determine the total execution time. By speeding up the execution of critical tasks, we can reduce the total execution time. The proposed heuristic is presented in Algorithm 5.2.

Algorithm 5.2: Heuristic for Optimal Task Scheduling

1: Schedule each task to the cheapest VM and compute C^{tot} and T^{tot}. Let τ^k and C^k denote current execution time and resource cost for V_k, respectively.
2: If $C^{tot} > C^{upp}$, no solution exists.
3: **while** $C^{tot} < C^{upp}$, **do**
4: Calculate the earliest start time $e^k \leftarrow \max\{e^{k'} + \tau^{k'} | E_{k'k} \in \mathbb{E}\}$, and the latest start time $l^k \leftarrow \min\{l^{k''} - \tau^{k''} | E_{kk''} \in \mathbb{E}\}$. Moreover, initialize $e^1 \leftarrow 0$. If $e^k = l^k$, V_k is a critical task.
5: **for** each critical task V_k, **do**
6: If $t_j^k < \tau^k$, $\Delta t_j^k \leftarrow \tau^k - t_j^k$, $\Delta C_j^k \leftarrow p_j - C_k$, $\rho_j^k \leftarrow \frac{\Delta t_j^k}{\Delta C_j^k}$, which represents that reschedule V_k to type-j VM needs ΔC_j^k more cost but saves Δt_j^k time.
7: **end for**
8: Sort all ρ_j^k in descending order.
9: Select maximal ρ_j^k and reschedule V_k to type-j VM.
10: Update $T^{tot} \leftarrow T^{tot} - \Delta t_j^k$, $C^{tot} \leftarrow C^{tot} + \Delta C_j^k$, $\tau^k \leftarrow t_j^k$, $C^k \leftarrow p_j$.
11: **end while**
12: **return** T^{tot}.

5.3.6 Simulations

In this section, we perform experiments to evaluate the proposed optimal task-level scheduling. Our experiments consist of the numerical simulation and the practical multimedia application.

We first verify the proposed task scheduling scheme in the numerical simulation. Four types of VMs are provided, including small, large, extra large standard instances, and medium high-CPU instances. The detailed configuration and price rate can be found from [15]. The number of tasks varies from 4 to 20.

In the simulation, we compare the total execution time among the proposed optimal task schedule scheme, in which tasks are scheduled by solving optimization problems in

Equations (5.4), (5.5), and (5.6), the proposed heuristic in Algorithm 5.2, and the static execution scheme, in which all tasks are executed on the medium standard instance. The optimal task schedule scheme is achieved by enumeration, which is optimal benchmark but not efficient, while the proposed heuristic is near optimal but lightweight and practical. The comparison results of the total execution time for the sequential, the parallel, and the mixed structures are shown in Figure 5.6(a), 5.6(b), and 5.6(c), respectively. We can see that the proposed optimal task scheduling scheme can achieve lower execution time compared to the proposed heuristic and the static execution scheme under the same resource cost constraint. Additionally, we can find that the total execution time acquired by the proposed heuristic is quite close to the globally optimal solution. Thus, the proposed heuristic can approach the optimal task scheduling in an efficient way. The static execution scheme employs one type of VM instance to execute all tasks, thus the application cannot be effectively processed in a distributed manner, leading to a longer execution time. Comparing Figure 5.6(a), 5.6(b), and 5.6(c), we also find that the total execution time in the mixed structure is longer than that in the parallel structure but lower than that in the sequential structure, which conforms Amdahl's law [126] that the execution time for a program is determined by the sequential portion of the program. Since the sequential structure has the largest sequential portion, the execution time for the sequential structure is the longest.

Next, we evaluate the proposed heuristic for optimal task scheduling by applying it to the video retrieval framework [124]. All video queries and video database are from TRECVID 2009 search task [127]. Each video query is around 5 seconds. VMs are supported by two servers (Intel i7 CPU 3.07GHz, 8G RAM, and 1T hard drive). The comparison of the total execution time between the proposed heuristic and the static execution scheme is shown in Figure 5.6(d). From Figure 5.6(d), we can see that the proposed heuristic can process video queries with lower execution time than the static scheme. When the number of queries is 100, the proposed heuristic can reduce the total execution time by 28% compared to the static execution scheme.

At last, we compare the actual running time to achieve the numerical solution by the proposed heuristic and the enumeration method. The two approaches are implemented in C++ on a server, which is configured with Intel i7 CPU 3.07GHz and 8G RAM. We conduct 5 rounds test and calculate the mean running time for each configuration with different number of tasks. The comparison of the running time is presented in Table

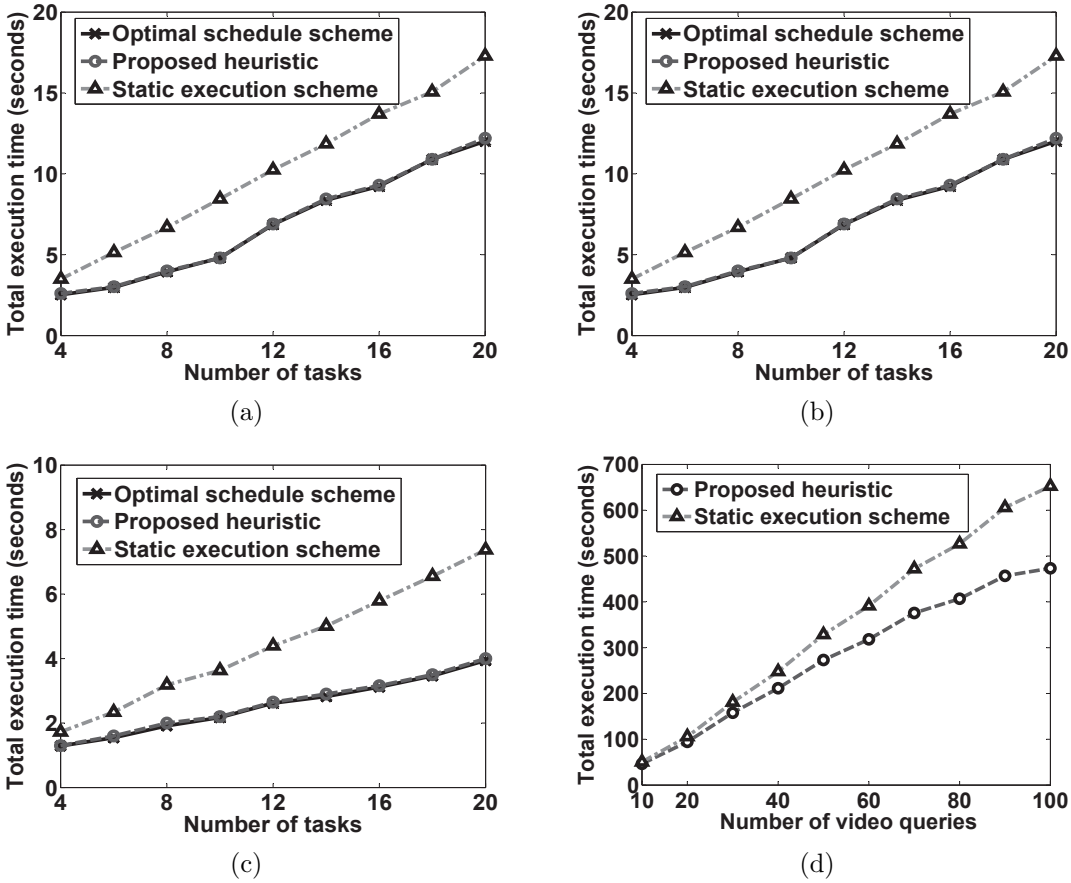


Figure 5.6: Comparison results of the total execution time (a) for sequential structure, (b) for parallel structure, (c) for mixed structure, (d) for video retrieval framework.

5.1. As demonstrated in Table 5.1, the proposed heuristic can achieve the numerical solution at a much lower running time, compared to the enumeration method. Since the time complexity for the enumeration method is exponential, the running time increases significantly as the number of tasks increases.

Table 5.1: Comparison of running time

Number of VM types	Number of tasks	Running time (ms)	
		Proposed heuristic	Enumeration
4	4	0.001	0.001
4	8	0.001	2.800
4	12	0.001	609.000
4	16	0.002	198,694.000

5.4 Chapter Summary

In this chapter, we study the optimal workload scheduling schemes at the user level and the task level to best utilize the allocated cloud resources. At the user level, users' requests are distributed according to the current load intensity at each virtual cluster. We optimize the workload assignment to minimize the response time or minimize the resource cost. Heuristics are proposed to efficiently schedule workload. Furthermore, we study the task level scheduling, in which a multimedia service is decomposed into tasks and these tasks are assigned to appropriate VMs for efficient processing. We introduce a directed acyclic graph to model precedence constraints among tasks. Based on the model, we formulate and solve the execution time minimization problem for sequential, parallel, and mixed structures. Simulation results demonstrate that the proposed schemes can optimize workload scheduling to achieve a minimal response time or a minimal resource cost.

Chapter 6

Delay-rate-distortion Optimization for Cloud Gaming with Hybrid Streaming

Cloud gaming is an emerging interactive multimedia service, offering players a ubiquitous gaming experience. However, due to the huge data transmission, cloud gaming suffers from high bandwidth consumption and response delay. In this chapter, we propose a novel hybrid streaming framework, jointly applying video streaming and graphics streaming to provide users with a high quality cloud gaming experience. Based on the proposed framework, we investigate the delay-rate-distortion (d-R-D) optimization problem for cloud gaming, in which we optimize the rate allocation between the video stream and the graphics stream to minimize the overall distortion under the bandwidth and response delay constraints. Moreover, we analyze the relationship among distortion, source rate, and response delay in cloud gaming, and propose a practical rate allocation algorithm. Experimental results demonstrate that the proposed hybrid streaming can achieve the lowest distortion under the constraints of bandwidth and response delay, compared to the traditional video streaming and graphics streaming.

6.1 Introduction

Driven by the increasing shipments of smart phones and tablets, the worldwide game market enjoys an annual growth rate of 18.3% in 2013 and is expected to reach $128 billion by 2017 [128]. With the popularity of mobile devices, users strongly desire ubiquitous gaming experience, where users are able to play games anywhere anytime on any devices. The emergence of cloud gaming effectively meets this desire. In cloud gaming, cloud servers fully or partially render game scenes and stream compressed game frames to clients, while user's inputs from mice, keyboard, and touch screen are recorded and transmitted back to cloud servers to control the actions of game characters. Cloud gaming benefits both users and game developers. Due to the shift of computationally intensive tasks to cloud servers, cloud gaming reduces hardware requirements at user side and frees users from the unlimited hardware upgrade. Moreover, users' profiles and game progresses are stored on cloud servers, which enable continuous game playing when users switch to different devices. From game developers' perspectives, deploying games on cloud can support various end devices, which addresses the incompatibility issues and reduces the development cost. Due to the popularity of cloud gaming, an increasing number of corporations have been attracted by the potential market, like OnLive [5], GaiKai [24], Microsoft [129], Kalydo [77], StreamMyGame [84], CiiNOW [25], and so on.

Current cloud gaming systems mainly use two different approaches: the *video streaming* and the *graphics streaming*. The difference between these two approaches is which device (e.g., the server or the client) renders the game scenes. With video streaming, the cloud server renders the game scenes and streams the encoded game frames to client. This approach enables smooth game experience on thin devices. OnLive [5] and GaiKai [24] are typical cloud gaming providers using video streaming technique. Video streaming is ideal to resource constrained devices, since all workload is done by cloud servers. As the release of computationally powerful mobile devices in recent years, graphics streaming becomes an alternative approach for cloud gaming. In graphics streaming, the required game models and textures are downloaded to client devices. After receiving the necessary graphics data, users can start playing the game, and the remaining graphics data will be gradually transmitted while users are playing the game. Since the actual games are running on local devices, graphics streaming enables a shorter interaction delay. Kalydo [77] adopts this approach for their online game service.

However, both streaming approaches have limitations. In video streaming, the captured game scene is transmitted in a frame-by-frame manner, which consumes a large bandwidth. In OnLive [5], the minimum bandwidth requirement is 2 Megabits per second (Mbps) and a higher resolution performance, like 720P (1280×720), requires a bandwidth higher than 5 Mbps. Thus, users are required to have a high-speed network connection. Additionally, cloud gaming is a highly interactive application. Users expect a low response delay when playing games. It is challenging to meet the tight delay requirement in video streaming, since each high-definition frame has to go through the whole pipeline of rendering, capturing, encoding, transmission, decoding, and display. Compared to video streaming, cloud gaming with graphics streaming has a lower response delay, since game rendering is performed on local devices. But in graphics streaming, the client has to receive the required graphics data prior to rendering the game scene. When users start playing a new game or move into a new game scene, it takes a buffering period to download the required graphics data, which is unacceptable for real-time games.

Existing cloud gaming systems are built on video streaming or graphics streaming. The video streaming consumes a high bandwidth and is sensitive to delay, while the graphics streaming requires buffering delay when players move into a new scene. To fill this gap, we propose a novel hybrid streaming framework, which integrates the video and graphics streaming approaches. Based on the framework, we study the d-R-D optimization problem to minimize the overall distortion during game playing. Our main contributions are summarized as follows:

- We propose a hybrid streaming framework by jointly using video streaming and graphics streaming approaches. In the proposed framework, cloud server not only transmits the encoded video frame, but also sends the graphics data. The received graphics data are used to render a game frame, which provides an additional reference candidate for video encoding. When encoding a captured game frame, the encoder will choose the reference frame with a lower residual error, from the previous frame and the rendered frame. As the accumulation of graphics data, the rendered graphics frame has a lower residual than the previous frame, leading to a lower encoding bit rate. After transmitting all the game data, cloud servers can skip encoding. As a result, the bit rate and the response delay can be reduced.

- Based on the proposed framework, we formulate the rate allocation problem as the

d-R-D optimization problem. Specifically, we optimize the source rates for video stream and graphics stream to minimize the total distortion during game playing period under the bandwidth and the response delay constraints. Furthermore, we perform theoretical analysis on the relationship among distortion, source rate, and response delay in cloud gaming, and develop a practical rate allocation algorithm to efficiently allocate source rates for the proposed hybrid streaming framework.

The remainder of this chapter is organized as follows. Section 6.2 presents the proposed hybrid streaming framework. Based on the proposed framework, we study the source rate allocation problem in Section 6.3 and present extensive performance evaluations in Section 6.4. Finally, we summarize the chapter in Section 6.5.

6.2 Hybrid Streaming Framework for Cloud Gaming

Figure 6.1 illustrates the architecture of the proposed hybrid streaming framework. During gaming, user's inputs on keyboard, mouse, or touchpad, are encoded and transmitted to cloud. After decoding user inputs, cloud server will push them into a queue. By replaying the user's inputs, cloud server can acquire the user's current game status, like position, viewpoint, and movement. Once a game frame is rendered, the server will capture the game frame and compress it by video encoder. We introduce two sync buffers at both the cloud and the client sides. Cloud server not only transmits the encoded video frame, but also sends the graphics data, including geometry mesh and textures. The graphics data will be updated at the sync buffers and used to render a graphics frame. When encoding a captured game frame, the encoder will choose the reference frame from the previous frame and the rendered graphics frame. The reference frame with a lower residual will be used, in order to reduce the encoding bit rate. The proposed rate allocation algorithm dynamically allocates source rates to the video stream and the graphics stream. If all the required game data have been transmitted to client, cloud server will skip video encoding and notify the client by sending a skip signal. Once a skip signal is received at client, a switch will notify the video decoder to skip decoding, and directly display the rendered graphics frame. In such a case, the game rendering is transferred from cloud server to client device, which effectively reduces the bandwidth consumption and the encoding delay at cloud.

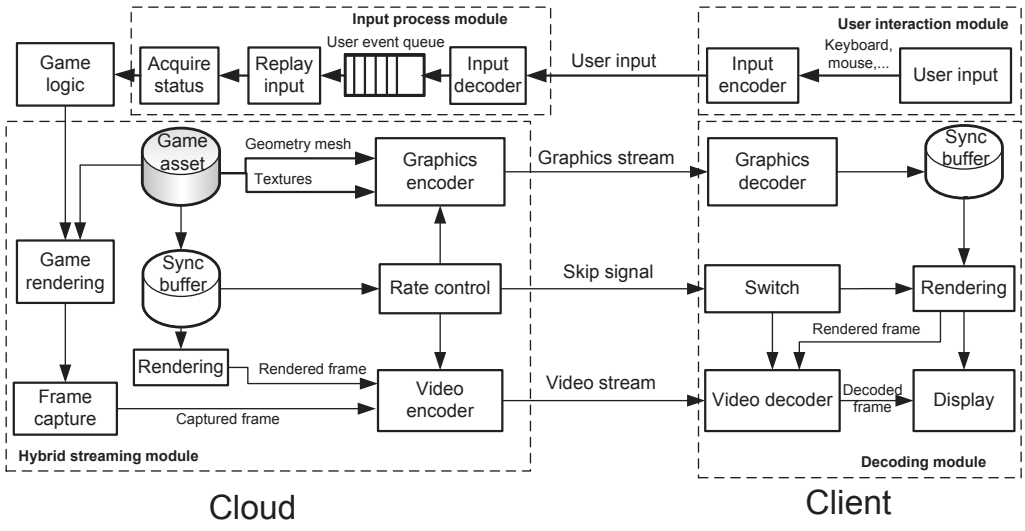


Figure 6.1: Illustration of proposed hybrid streaming framework for cloud gaming.

By jointly using video streaming and graphics streaming, the proposed framework has the following features. First of all, the proposed framework eliminates the initial buffering delay compared to the graphics streaming. When users start playing a new game or move into a new game scene, the sync buffers are empty. Cloud servers will use the previous video frame as reference, which is illustrated in Figure 6.2(a). By streaming the compressed video frames to client, users can directly play the game without any initial delay. In addition, the proposed framework effectively reduces the overall bit rates compared to the video streaming. During game playing, the graphics buffers accumulate more and more game data. Once received sufficient game data, the frame rendered from sync buffer has a lower residual than the previous video frame, which is illustrated in Figure 6.2(b). In video encoding, a lower residual leads to a less coding bit rate. Therefore, choosing reference frame with a lower residual can effectively reduce the overall coding bit rates.

To efficiently transmit graphics data to client, we apply a scalable delivery scheme in the proposed framework. The scalable delivery scheme requires the multi-resolution representation of graphics data and the progressive transmission. Hoppe *et al.* [130] proposed the progressive mesh representation to enable a multi-resolution rendering of 3D meshes. The core operations in progressive mesh are *edge collapse* and *vertex split*.

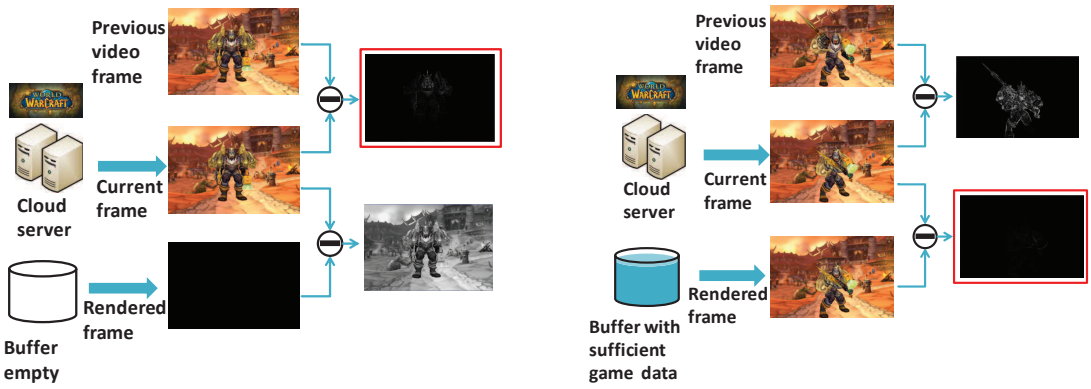


Figure 6.2: Illustration of reference frame selection: (a) when the synch buffer is empty, and (b) when the synch buffer collects sufficient game data.

As shown in Figure 6.3, the edge collapse operation will unify two adjacent vertices V_s and V_t into one single vertex V_s. After the edge collapse, the vertex V_t and two faces $\{V_s, V_t, V_l\}$ and $\{V_s, V_r, V_t\}$ are removed. The simplified 3D mesh by the edge collapse can be reversed by the vertex split. As illustrated in Figure 6.3, the vertex split operation adds a new vertex V_t near the vertex V_s and thus two new faces $\{V_s, V_t, V_l\}$ and $\{V_s, V_r, V_t\}$ can be constructed.

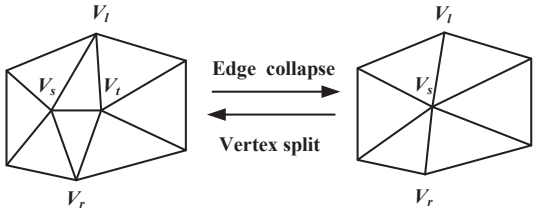
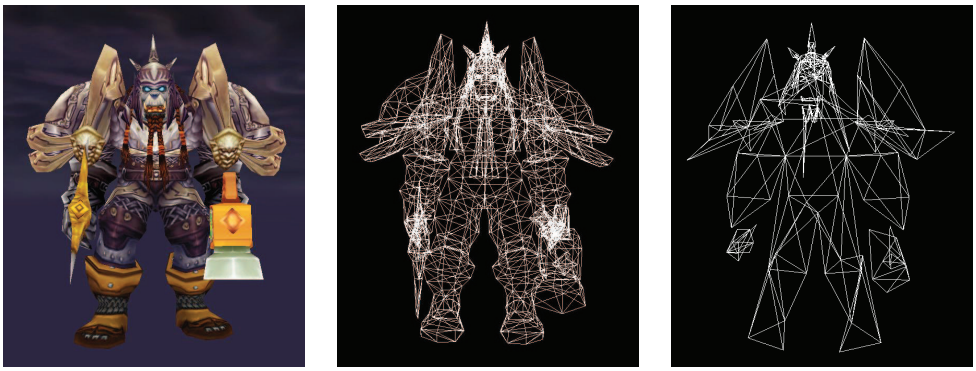


Figure 6.3: Edge collapse and vertex split in progressive mesh.

Based on the edge collapse and vertex split operations, a game model can be rendered progressively. Given a initial game model M_{ini}, we apply a series of edge collapse operations to simplify the model by reducing vertices and faces. This process can be denoted as $M_{ini} \xrightarrow{edge\ collapse} \cdots \xrightarrow{edge\ collapse} M_{base}$, where the finally simplified model M_{base} is called the base mesh. Given the base mesh, the original game model can be reconstructed by a series of vertex split operations, i.e. $M_{base} \xrightarrow{vertex\ split} \cdots \xrightarrow{vertex\ split} M_{ini}$. Therefore, a game model can be represented by the base mesh and a series of vertex splits informa-

tion. The client can render a simple and coarse quality model after receiving the base mesh, and then incrementally enhance the rendering quality by adding more vertices and faces. Figure 6.4(a) shows an equipped Orc warrior from WoW [131], and the initial mesh and the base mesh of the Orc warrior are shown in Figure 6.4(b) and Figure6.4(c), respectively.



(a) (b) (c)

Figure 6.4: Illustration of progressive mesh representation: (a) the Orc warrior from WoW, (b) the initial mesh of the warrior (3344 faces), and (c) the base mesh of the warrior (171 faces).

Besides the progressive mesh compression, we employ the adaptive scalable texture compression (ASTC) to compress the textures. ASTC is a block-based fixed-rate lossy texture compression algorithm [132]. In the proposed framework, we compress textures at two different bit rates, in which the *high rate texture* is compressed by ASTC at 8 bpp and the *low rate texture* is compressed at 2 bpp. If the client bandwidth is not sufficient, the low rate textures will be transmitted to client to adapt to the limited bandwidth. Figure 6.5(a) shows the ocular texture of the Orc warrior, and the corresponding textures compressed by ASTC at 8 bpp and 2 bpp are shown in Figure 6.5(b) and Figure 6.5(c), respectively.

(a) (b) (c)

Figure 6.5: Illustration of ASTC: (a) the original ocular texture of the Orc warrior from World of Warcraft, (b) the reconstructed texture compressed at 8 bpp, and (c) the reconstructed texture compressed at 2 bpp.

6.3 Optimal Rate Allocation for Hybrid Streaming Framework

In this section, we investigate the source rate allocation problem in the proposed hybrid streaming framework. Specifically, we formulate the problem as a d-R-D optimization, which aims to minimize the overall distortion under the bandwidth and the response delay constraints by optimizing the allocated rates for video stream and graphics stream.

6.3.1 Problem Formulation

In the proposed hybrid streaming framework, there is a trade-off in source rate allocation. If cloud servers allocate a higher source rate for video stream, it will take a longer time for the client to receive all the necessary game data. On the other hand, if a higher rate is allocated to graphics stream, the visual quality in the video stream will be degraded. Therefore, we need to determine the optimal rate allocation for video stream and graphics stream.

Suppose the current time is t, and R_V^t and R_G^t are the allocated rates for video stream and graphics stream at t. We formulate the rate allocation problem as a d-R-D optimization. Let $\mathcal{D}_{tot}$ be the total distortion at client for the whole game playing period, $\mathcal{R}_{tot}^t$ be the total bit rate allocation at t, d_{tot}^t be the total response delay at t. To minimize the total distortion under the bandwidth and the response delay constraints,

we can formulate the rate allocation problem as

$$\begin{aligned} \underset{\{R_V^t, R_G^t\}}{\text{Minimize}} \quad & \mathcal{D}_{tot} \\ \text{subject to} \quad & \\ & \mathcal{R}_{tot}^t \leq R_{max}, \\ & d_{tot}^t \leq d_{max}, \end{aligned} \tag{6.1}$$

where R_{max} is the maximum bandwidth capacity and d_{max} is the maximum tolerable response delay. In Equation (6.1), $\mathcal{D}_{tot}$ represents the total distortion during the gaming process, $\mathcal{R}_{tot}^t \leq R_{max}$ and $d_{tot}^t \leq d_{max}$ represent the bandwidth and response delay constraints, respectively.

6.3.2 Gaming Process

According to the current time t, the gaming process can be divided into: the past period before t, the current time instant at t, and the future period after t. The total distortion $\mathcal{D}_{tot}$ can be formulated as

$$\mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + \mathcal{D}_{cur}^{t} + \mathcal{D}_{fut}^{t+}, \tag{6.2}$$

where $\mathcal{D}_{pas}^{t-}$ is the past distortion, $\mathcal{D}_{cur}^{t}$ is the current distortion, and $\mathcal{D}_{fut}^{t+}$ is the future distortion. Among the three terms, $\mathcal{D}_{pas}^{t-}$ occurs before t. So, it is a constant. In the following, we will focus on the analysis of $\mathcal{D}_{cur}^{t}$ and $\mathcal{D}_{fut}^{t+}$.

During game playing, the cloud server progressively transmits game data, including geometry meshes and textures, to the client. Let Ω^t denote the set of received game data from the start time to the current time t. Specially, Ω^0 is the initial set, which represents the game data available at the start time. If the game is played for the first time, the buffer is empty at the beginning, i.e. $\Omega^0 = \emptyset$. As the game playing, game assets are received and accumulated in the sync buffer, and accordingly, the game data set keeps increasing, which can be represented as $\Omega^t = \Omega^{t-1} \cup \{x\}^t$, where x represents game data and $\{x\}^t$ represents the game data received in the interval between time instant $t-1$ and time instant t. As the more game data received at client, the quality of the rendered frame will be refined progressively.

Fig. 6.6 illustrates the gaming process. The residual of the rendered frame decreases with time. Suppose t_α is the threshold time instant after which the residual of the

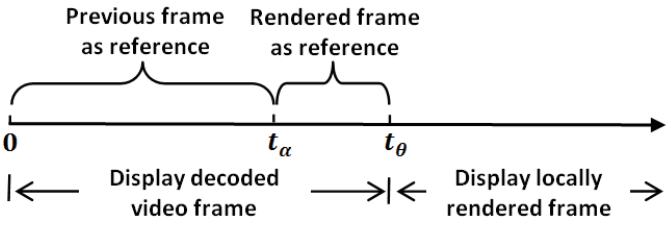


Figure 6.6: Illustration of gaming process.

rendered frame is lower than that of the previous frame. Therefore, the cloud server chooses the previous frame as reference from the beginning to t_α and chooses the rendered frame as reference after t_α. Let Ω_θ be the set of all the required game data, and t_θ be the expected time when Ω_θ is reached. Thus, the decoded video frame is displayed as game frame from the beginning to t_θ, and the locally rendered frame is displayed after t_θ. Given the current set Ω^t and the allocated rate R_G^t, the expected t_θ can be determined by $t_\theta = t + \frac{B_{\{\Omega_\theta \backslash \Omega^t\}}}{R_G^t}$, where $\Omega_\theta \backslash \Omega^t = \{x \in \Omega_\theta | x \notin \Omega^t\}$, representing the set of game data in Ω_θ but not in Ω^t yet, $B_{\{\Omega_\theta \backslash \Omega^t\}}$ is the data size of the set $\{\Omega_\theta \backslash \Omega^t\}$, and $\frac{B_{\{\Omega_\theta \backslash \Omega^t\}}}{R_G^t}$ represents the expected remaining time to reach Ω_θ under the current set Ω^t and the allocated rate R_G^t. Specifically, if $\Omega_\theta = \Omega^t$, i.e. the current game set contains the required game data, we will have $\Omega_\theta \backslash \Omega^t = \emptyset$ and thus $t_\theta = t + \frac{B_\emptyset}{R_G^t} = t$, indicating that the locally rendered frame is already displayed at client. Let D_V be the distortion of the decoded video frame, and D_G be the distortion of the locally rendered frame. Based on the analysis of gaming process, if $t < t_\theta$, the distortion is determined by D_V; if $t \geq t_\theta$, the distortion is caused by D_G. Therefore, the current distortion $\mathcal{D}_{cur}^t$ can be given by

$$\mathcal{D}_{cur}^t = \begin{cases} D_V, & \text{if } t < t_\theta, \\ D_G, & \text{if } t \geq t_\theta, \end{cases} \tag{6.3}$$

where the close form functions of D_V and D_G will be investigated in Section 6.3.3.

R_V^t and R_G^t affect not only the current game quality, but also the duration required to transmit all game data. Although the accurate value of $\mathcal{D}_{fut}^{t+}$ is unknown at current time t, the expectation of $\mathcal{D}_{fut}^{t+}$ can be computed. Suppose $\tilde{t}$ is the expected playing time in the current game scene. In practice, $\tilde{t}$ can be estimated from players' statistics. Suppose t_V and t_G are the remaining time lengths of using the video frame and the rendered frame

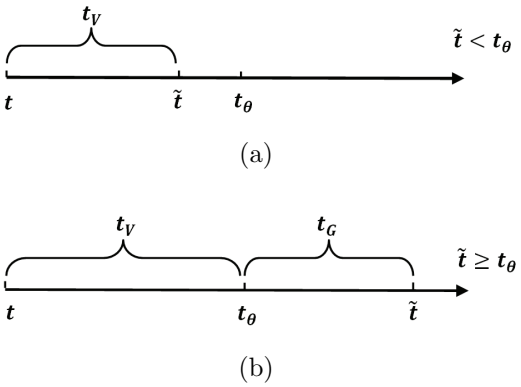


Figure 6.7: Illustration of t_V and t_G in two different situations: (a) when $\tilde{t} < t_\theta$, and (b) when $\tilde{t} \geq t_\theta$.

as game frame, respectively. According to the user's playing time $\tilde{t}$, t_V and t_G can be determined in two different situations. If $\tilde{t}$ is shorter than t_θ, as illustrated in Figure 6.7(a), the received game data until the time when the user leaves the current scene is not sufficient for rendering a high quality frame. Therefore, the video frame will be used during the gaming session. On the other hand, if $\tilde{t}$ is longer than t_θ, as illustrated in Figure 6.7(b), the video frame will be displayed from t to t_θ, and the rendered frame will be displayed from t_θ to $\tilde{t}$. Based on the above analysis, we can formulate t_V and t_G as

$$\begin{cases} t_V = \tilde{t} - t, & t_G = 0, & \text{if } \tilde{t} < t_\theta, \\ t_V = t_\theta - t, & t_G = \tilde{t} - t_\theta, & \text{if } \tilde{t} \geq t_\theta. \end{cases} \tag{6.4}$$

The expected distortion $\mathcal{D}_{fut}^{t+}$ can be represented as the sum of the expected distortions in both periods, the t_V period and the t_G period. $\mathcal{D}_{fut}^{t+}$ can therefore be formulated as

$$\begin{aligned} \mathcal{D}_{fut}^{t+} &= (t_V D_V \ + \ t_G D_G) \cdot f \\ &= \begin{cases} (\tilde{t} - t) D_V \cdot f, & \text{if } \tilde{t} < t_\theta, \\ \big((t_\theta - t) D_V \ + \ (\tilde{t} - t_\theta) D_G\big) \cdot f, & \text{if } \tilde{t} \geq t_\theta, \end{cases} \end{aligned} \tag{6.5}$$

where f is the frame rate.

6.3.3 Rate-Distortion (R-D) Analysis

In the proposed framework, we use H.264/AVC codec to compress the video frame. Cloud gaming demands the real-time service. The decoding of bi-direction (B) frames will introduce extra delays, since it depends on the decoding of bi-direction reference frames. Thus, in cloud gaming systems, like GamingAnyWhere [72], the B frame encoding is disabled. In our work, we utilize the IPPPP coding structure. When encoding the captured game frame, there are two reference frames to be chosen: one is the previous frame, and the other is the rendered frame from graphics buffer. The reference frame with a lower residual will be used in encoding, which is similar to the multiple reference frame selection in H.264/AVC encoding [133].

According to the previous research work on the rate-distortion analysis [134, 135], the rate and the average distortion can be formulated as the functions of quantization step size. Supposing the mean squared error (MSE) is used as the distortion measure. The relationship of the average distortion D_V and the quantization step size Q_{step} is given by [135] $D_V = MSE = \rho Q_{step} + \gamma$, where ρ and γ are the distortion model parameters. In addition, the rate R_V^t can be represented by [134] $R_V^t = \frac{\mu}{Q_{step}} + \nu$, where μ and ν are the rate model parameters. By eliminating Q_{step}, we can get the close form relationship between D_V and R_V^t as

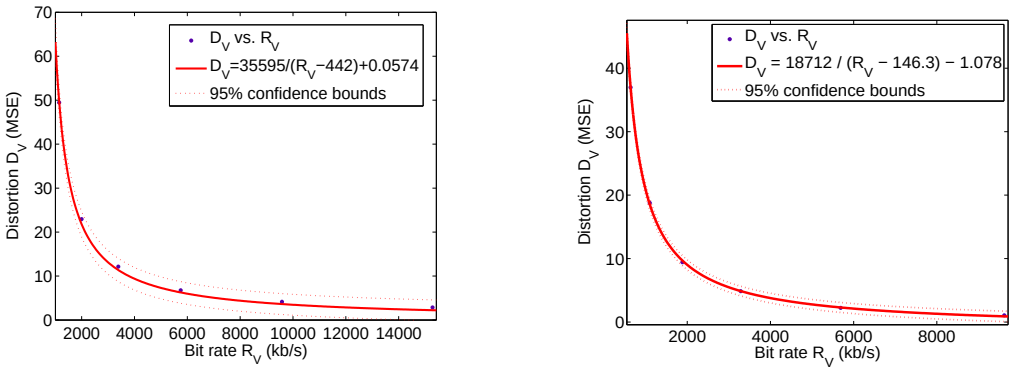
$$D_V = \frac{\rho\mu}{R_V^t - \nu} + \gamma. \tag{6.6}$$

The distortion D_G is determined by the received game data set. The progressive mesh representation is a continuous and lossless representation [130]. Therefore, once all the vertex splits of a game model have been received, the geometry mesh of the model at client is the same as that at cloud server. In contrast to progressive mesh, ASTC is a lossy texture compression [132]. According to the study in [132], the distortion of ASTC is affected by the bit rate. The higher bit rate used in the compressed texture, the lower distortion can be achieved. Using the fixed bit rate in texture compression, the distortion D_G can be assumed as a constant, which is given by

$$D_G = \sigma_{ASTC}. \tag{6.7}$$

where σ_{ASTC} is the distortion caused by texture compression.

We conduct experiments to verify the rate and distortion models in Equation (6.6)

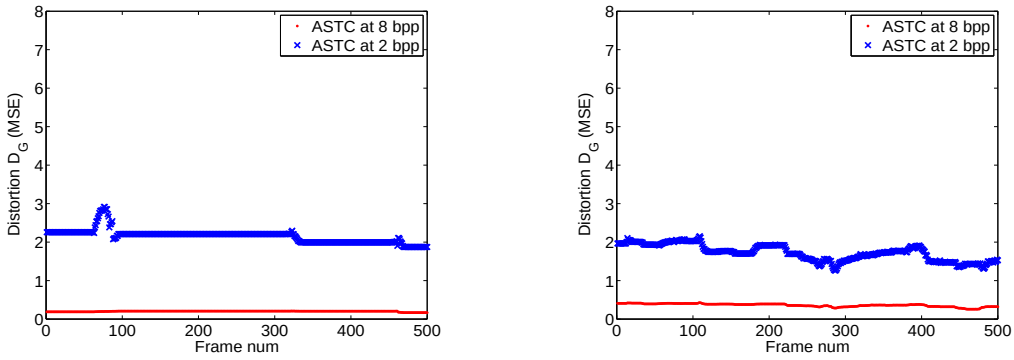


(a) (b)

Figure 6.8: Experimental results for verifying the relationship between the distortion D_V and the bit rate R_V^t: (a) for Orc warrior sequence, and (b) for Human warrior sequence.

and Equation (6.7). We capture two sequences from the online game WoW [131]: one is the Orc warrior sequence and the other is the Human warrior sequence. Figure 6.8 shows the relationship and fitting results of the average distortion D_V and the bit rate R_V^t. It can be seen from Figure 6.8 that the distortion D_V would decrease as the increase of R_V^t. Furthermore, we use the rate-distortion model in Equation (6.6) to fit the relationship between D_V and R_V^t and draw the fitted function curve in Figure 6.8(a) and Figure 6.8(b), where all the true (R_V^t, D_V) points fall into the 95% confidence bounds of the fitting function. The narrow confidence bounds indicate that the rate-distortion model in Equation (6.6) is reliable.

Similarly, we conduct experiments to verify Equation (6.7). We assume that the client has collected all the required geometry meshes and textures compressed by ASTC. During the test, the frame rendered at cloud server uses the original textures, while the frame rendered at client uses the reconstructed textures. Figure 6.9 shows the distortion of each rendered frame given an ASTC bit rate. From Figure 6.9, we can see that the distortion with ASTC at 8 bpp is lower than that at 2 bpp. Figure 6.9(a) and Figure 6.9(b) also indicate that the frames rendered with same bit rate textures have a relatively flat distortion, which justifies Equation (6.7).



(a) (b)

Figure 6.9: Distortion of each frame in ASTC: (a) for Orc warrior sequence, and (b) for Human warrior sequence.

6.3.4 Response Delay Analysis

Besides distortion and rate, response delay is another important user-perceived QoS metric in cloud gaming. The response delay is defined as the duration from the time when the user gives an input on the client device to the time when the resulting game frame is displayed to the user. The response delay in cloud gaming consists of multiple components. Similar to [82], we divide the response delay into the following four components.

- **Network delay** d_{net}^t: d_{net}^t represents the network round-trip-time (RTT), which includes the delays for transmitting user input to server and sending game frame back to client.
- **Rendering delay** d_{ren}^t: d_{ren}^t represents the delay for the game software at cloud server to process user input and render the next game frame.
- **Encoding delay** d_{enc}^t: d_{enc}^t represents the time for the cloud server to capture the game frame and encode it into bit stream for the client.
- **Decoding delay** d_{dec}^t: d_{dec}^t represents the delay for the client to decode and display the game frame.

Thus, the total response delay d_{tot} can be formulated as the sum of the four components, which is given by

$$d_{tot}^t = d_{net}^t + d_{ren}^t + d_{enc}^t + d_{dec}^t. \tag{6.8}$$

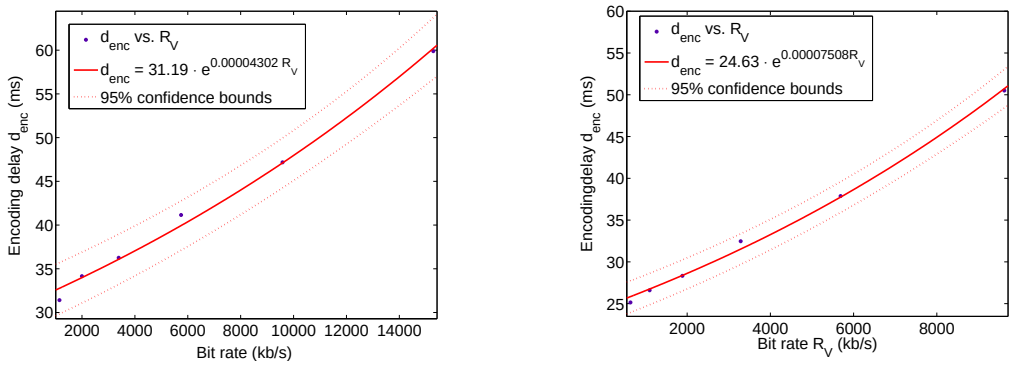
Among the four components, d_{net}^t can be measured by the Internet Control Message Protocol (ICMP) Ping command from the client to the server, and d_{ren}^t is game dependent, which can be acquired from the rendering frame rate at cloud server. If the cloud gaming system is proprietary and closed, d_{ren}^t can also be approximated from the PC version of the game. Similar measurement techniques have been used in study [82].

Measuring the encoding delay d_{enc}^t is not straightforward, because d_{enc}^t is determined by the encoding complexity, while the encoding complexity relates with the distortion and the bit rate. In H.264/AVC encoding, the motion estimation is the most time consuming part, which takes more than 90% encoding time [136]. Similar to [136, 137], we approximate the encoding time by the motion estimation time (MET) in our work. The major operation in motion estimation is the sum of absolute difference (SAD). Thus, the MET can be acquired as the total number of CPU clock cycles consumed by all the SAD operations divided by the number of clock cycles per second [136]. Therefore, the encoding delay d_{enc}^t can be given by [136]

$$d_{enc}^t \approx MET = \frac{M(2\lambda+1)^2\eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, \quad \text{if } t < t_\theta, \tag{6.9}$$

where M is the number of macroblocks in a frame, λ is the motion estimation search range, η is the number of reference frames, $(2\lambda+1)^2\eta$ is the total number of SAD in a 3D search cube for each macroblock, $\alpha(R_V^t)$ is the ratios of the actual number of SAD operations to the theoretical total number of SAD operations, $M(2\lambda+1)^2\eta \cdot \alpha(R_V^t)$ represents the actual number of SAD operations in a frame, c_0 is the number of CPU cycles required for one SAD operation, and f_{CLK} is the clock frequency of the CPU. As presented in [136], the function $\alpha(\cdot)$ in Equation (6.9) can be fitted as an exponential function. Equation (6.9) is the motion estimation time for full search. In practical encoder, the fast search methods, like diamond search and hexagonal search, are usually used. To accommodate the fast search methods, the motion estimation time in Equation (6.9) can be revised by multiplying a coefficient ϑ, where $0 < \vartheta \leq 1$. We utilize the exhaustive full search to achieve the minimum predictive residuals.

In order to justify Equation (6.9), we conduct experiments and draw the actual encoding delay d_{enc}^t and the bit rate R_V^t in Figure 6.10. The model in Equation (6.9) is used as the function to fit the relationship of d_{enc}^t and R_V^t. Figure 6.10 indicates that the relationship can be well fitted by the function in Equation (6.9), and all actual (R_V^t,



(a) (b)

Figure 6.10: Experimental results for verifying the relationship between the encoding delay d_{enc}^t and the bit rate R_V^t: (a) for Orc warrior sequence, and (b) for Human warrior sequence.

d_{enc}) points are within the 95% confidence bounds.

Furthermore, when $t \geq t_\theta$, the locally rendered graphics frame is displayed. In such a case, the cloud server will skip encoding the frame. Therefore, the encoding delay is approximated as 0. Based on the above analysis, we can formulate the encoding delay at cloud server as

$$d_{enc}^t \approx \begin{cases} \frac{M(2\lambda+1)^2 \eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, & \text{if } t < t_\theta, \\ 0, & \text{if } t \geq t_\theta, \end{cases} \tag{6.10}$$

As the last component, the decoding delay d_{dec}^t cannot be directly measured by the cloud server. When cloud server transmits the currently encoded frame to client, the decoding of the frame has not occurred yet. In the previous cloud gaming study [82], the delay at client is derived by measuring the total response delay and then subtracting the delays for network, rendering, and encoding. Therefore, the measured decoding delay is not for the frame which is currently encoded, but for the previous frame. Inspired by study [82], we use the most recent decoding delay to approximate the current decoding delay at client. Specifically, the actual decoding delay at time $t-1$ is measured by the client and sent back to the cloud server as the decoding delay d_{dec}^t.

With the four delay components d_{net}^t, d_{ren}^t, d_{enc}^t, and d_{dec}^t, the cloud server can calculate the total response delay d_{tot}^t by Equation (6.8). During game playing, the cloud server can dynamically adjust encoding parameters to satisfy the total response delay constraint.

For example, if the current network delay d_{net}^t increases, the cloud server can choose a larger quantization step size to reduce the encoding delay d_{enc}^t and accordingly reduce the total response delay d_{tot}^t.

6.3.5 Rate Allocation Algorithm

Based on the above analysis, the optimal rate allocation problem in Equation (6.1) can be formulated as

$$
\begin{aligned}
\underset{\{R_V^t, R_G^t\}}{\text{Minimize}} \quad & \mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + \mathcal{D}_{cur}^{t} + \mathcal{D}_{fut}^{t+} \\
\text{subject to} \quad & \\
& \mathcal{D}_{cur}^{t} = \begin{cases} D_V, & \text{if } t < t_\theta, \\ D_G, & \text{if } t \geq t_\theta, \end{cases} \\
& \mathcal{D}_{fut}^{t+} = \begin{cases} (\tilde{t} - t) D_V \cdot f, & \text{if } \tilde{t} < t_\theta, \\ \big((t_\theta - t) D_V \ + \ (\tilde{t} - t_\theta) D_G\big) \cdot f, & \text{if } \tilde{t} \geq t_\theta, \end{cases} \\
& D_V = \tfrac{\rho\mu}{R_V^t - \nu} + \gamma, \\
& D_G = \sigma_{ASTC}, \\
& \mathcal{R}_{tot}^t \leq R_{max}, \\
& \mathcal{R}_{tot}^t = R_V^t + R_G^t, \\
& d_{tot}^t \leq d_{max}, \\
& d_{tot}^t = d_{net}^t + d_{ren}^t + d_{enc}^t + d_{dec}^t, \\
& d_{enc}^t \approx \begin{cases} \frac{M(2\lambda+1)^2 \eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, & \text{if } t < t_\theta, \\ 0, & \text{if } t \geq t_\theta. \end{cases}
\end{aligned}
\tag{6.11}
$$

In Equation (6.11), the target of the rate allocation is to minimize the total distortion $\mathcal{D}_{tot}$ for the given bandwidth R_{max} and the given response delay d_{max}, by optimizing source rates for video stream R_V^t and graphics stream R_G^t. A higher R_V^t is required to provide a video frame with a lower distortion. But the increase of R_V^t will reduce R_G^t and accordingly extends the time for video streaming. Moreover, a higher R_V^t will cause a longer encoding delay. On the other hand, if a higher bit rate is allocated to R_G^t, the distortion D_V will increase, since a lower rate is left for R_V^t. Therefore, it is infeasible to simultaneously minimize the total distortion, bit rate, and response delay. The optimal rate allocation problem in Equation (6.11) is to find the Pareto optimal tradeoff [138]

among the optimization criteria.

According to the relationship of current time t, time t_θ, and total playing time $\tilde{t}$, the optimization problem Equation (6.11) can be studied in the following cases.

Case 1: Given $t_\theta \leq t \leq \tilde{t}$, i.e. $\Omega_\theta \subseteq \Omega^t$, the client has received all the required game data. Therefore, the locally rendered frame is displayed at client. We can get the following theorem.

Theorem 6.1. *Given $t_\theta \leq t \leq \tilde{t}$, the rate allocation to video stream is $R_V^t = 0$.*

Proof. Given $t_\theta \leq t \leq \tilde{t}$, i.e. $\Omega_\theta \subseteq \Omega^t$, Equation (6.11) is converted to

$$\begin{aligned} \underset{\{R_V^t, R_G^t\}}{\text{Minimize}} \quad & \mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + \mathcal{D}_{cur}^{t} + \mathcal{D}_{fut}^{t+} \\ \text{subject to} \quad & \\ & \mathcal{D}_{cur}^{t} = D_G, \\ & \mathcal{D}_{fut}^{t+} = (\tilde{t} - t) D_G \cdot f, \\ & D_V = \frac{\rho\mu}{R_V^t - \nu} + \gamma, \\ & D_G = \sigma_{ASTC}, \\ & R_V^t + R_G^t \leq R_{max}, \\ & d_{net}^t + d_{ren}^t + d_{enc}^t + d_{dec}^t \leq d_{max}, \\ & d_{enc}^t \approx 0. \end{aligned} \tag{6.12}$$

The objective can be represented as $\mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + D_G + (\tilde{t} - t) D_G \cdot f$, where the total distortion is only determined by D_G and has nothing with D_V. Since R_V^t has no effect on the total distortion, R_V^t will be reduced to 0. Theorem 6.1 is proved. □

Intuitively, if the client can independently render the graphics frame, there is no need for the cloud server to stream the encoded video frames after t_θ. Accordingly, R_V^t will be reduced to 0.

Case 2: Given $t \leq \tilde{t} \leq t_\theta$, the expected game playing time $\tilde{t}$ is no longer than t_θ. Thus, the client cannot receive all the required game data before the client leaves the scene. We then have the following theorem.

Theorem 6.2. *Given $t \leq \tilde{t} \leq t_\theta$, the optimal solution $({R_V^t}^*,\ {R_G^t}^*)$ to Equation (6.11) is ${R_V^t}^* = \max\{R_V^t | R_V^t \leq R_{max},\ d_{enc}^t(R_V^t) \leq d_{max} - d_{net}^t - d_{ren}^t - d_{dec}^t\}$, ${R_G^t}^* = R_{max} - {R_V^t}^*$.*

Proof. Given $t \leq \tilde{t} \leq t_\theta$, Equation (6.11) is converted to

$$\begin{aligned} \underset{\{R_V^t, R_G^t\}}{\text{Minimize}} \quad & \mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + \mathcal{D}_{cur}^{t} + \mathcal{D}_{fut}^{t+} \\ \text{subject to} \quad & \\ & \mathcal{D}_{cur}^{t} = D_V, \\ & \mathcal{D}_{fut}^{t+} = (\tilde{t} - t) D_V \cdot f, \\ & D_V = \frac{\rho\mu}{R_V^t - \nu} + \gamma, \\ & D_G = \sigma_{ASTC}, \\ & R_V^t + R_G^t \leq R_{max}, \\ & d_{net}^t + d_{ren}^t + d_{enc}^t + d_{dec}^t \leq d_{max}, \\ & d_{enc}^t \approx \frac{M(2\lambda+1)^2 \eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, \end{aligned} \tag{6.13}$$

where $\mathcal{D}_{tot}$ is determined by D_V and has no relationship with D_G. Since D_V is a monotonically decreasing function of R_V^t, the optimal allocation ${R_V^t}^*$ is the largest R_V^t which simultaneously meets $R_V^t \leq R_{max}$ and $d_{net}^t + d_{ren}^t + d_{enc}^t(R_V^t) + d_{dec}^t \leq d_{max}$, i.e. ${R_V^t}^* = \max\{R_V^t | R_V^t \leq R_{max},\ d_{enc}^t(R_V^t) \leq d_{max} - d_{net}^t - d_{ren}^t - d_{dec}^t\}$. After allocating rate to R_V^t, the remaining rate will be allocated to R_G^t, and ${R_G^t}^* = R_{max} - {R_V^t}^*$. Therefore, Theorem 6.2 is proved. □

The proof of Theorem 6.2 is given in Appendix **??**. The intuition of Theorem 6.2 is that, if the client cannot render a high-quality frame from the accumulated graphics data at the client by the end of current session, the available source rate should be allocated to R_V^t as much as possible. In such a situation, the video stream will be always used.

Case 3: Given $t < t_\theta < \tilde{t}$, the client will display the decoded video frame from t to t_θ and then display the rendered frame from t_θ to $\tilde{t}$. The distortion D_V is determined by R_V^t, and the length of the video stream is determined by R_G^t. Thus, it requires an optimal rate allocation between R_V^t and R_G^t to minimize the total distortion. By analyzing Equation (6.11), we can get the following Lemma.

Lemma 6.1. *Given $t < t_\theta < \tilde{t}$, Equation (6.11) is a convex optimization problem.*

Proof. Given $t < t_\theta < \tilde{t}$, the optimization problem in Equation (6.11) can be converted

to

$$\begin{aligned}
&\underset{\{R_V^t, R_G^t\}}{\text{Minimize}} \quad \mathcal{D}_{tot} = \mathcal{D}_{pas}^{t-} + \mathcal{D}_{cur}^{t} + \mathcal{D}_{fut}^{t+} \\
&\text{subject to} \\
&\qquad \mathcal{D}_{cur}^{t} = D_V, \\
&\qquad \mathcal{D}_{fut}^{t+} = \big((t_\theta - t)D_V \ + \ (\tilde{t} - t_\theta)D_G\big) \cdot f, \\
&\qquad D_V = \tfrac{\rho\mu}{R_V^t - \nu} + \gamma, \\
&\qquad D_G = \sigma_{ASTC}, \\
&\qquad R_V^t + R_G^t \leq R_{max}, \\
&\qquad d_{net}^t + d_{ren}^t + d_{enc}^t + d_{dec}^t \leq d_{max}, \\
&\qquad d_{enc}^t \approx \tfrac{M(2\lambda+1)^2\eta\cdot\alpha(R_V^t)\cdot c_0}{f_{CLK}},
\end{aligned} \tag{6.14}$$

where $\alpha(R_V^t)$ is an exponential function of R_V^t. In Equation (6.14), the bandwidth constraint and the response delay constraint are both convex, since the bandwidth constraint is a linear real-value function and the response delay constraint is the sum of convex functions. To prove the convexity of the objective function, we derive the Hessian matrix of $\mathcal{D}_{tot}$ as following.

$$\nabla^2\mathcal{D}_{tot}(R_V^t, R_G^t) = \begin{bmatrix} \frac{2(t_\theta - t)\rho\mu}{(R_V^t - \nu)^3} & \frac{B_{\{\Omega_\theta \setminus \Omega^t\}}\rho\mu}{{R_G^t}^2(R_V^t - \nu)^2} \\ \frac{B_{\{\Omega_\theta \setminus \Omega^t\}}\rho\mu}{{R_G^t}^2(R_V^t - \nu)^2} & \frac{2B_{\{\Omega_\theta \setminus \Omega^t\}}(D_V + D_G)}{{R_G^T}^2} \end{bmatrix} \tag{6.15}$$

We can find that the Hessian matrix $\nabla^2\mathcal{D}_{tot}(R_V^t, R_G^t)$ is positive semidefinite, and thus the objective function in Equation (6.14) is convex [138]. Based on the above analysis, given $t < t_\theta < \tilde{t}$, Equation (6.11) is a convex optimization problem. The Lemma 6.1 is proved. □

Since the optimization problem in Equation (6.11) is a convex optimization, we can use Lagrange multiplier method [138] to solve it. The Lagrange function of Equation (6.11) is given by

$$L(R_V^t, R_G^t, \lambda, \kappa) = \mathcal{D}_{tot}(R_V^t, R_G^t) + \lambda \cdot \left[\mathcal{R}_{tot}^t(R_V^t, R_G^t) - R_{max}\right] + \kappa \cdot \left[d_{tot}^t(R_V^t, R_G^t) - d_{max}\right], \tag{6.16}$$

where $\lambda \geq 0$ and $\kappa \geq 0$ are Lagrange multipliers associated with the inequality constraints. According to KKT conditions, the optimal primal solution ${R_V^t}^*$ and ${R_G^t}^*$ and dual solution λ^* and κ^* exist such that the following conditions can be satisfied simulta-

neously.

$$\begin{cases} \left.\dfrac{\partial L(R_V^t, R_G^t, \lambda, \kappa)}{\partial R_V^t}\right|_{R_V^{t\,*}, R_G^{t\,*}, \lambda^*, \kappa^*} = 0, & \text{(6.17a)} \\ \left.\dfrac{\partial L(R_V^t, R_G^t, \lambda, \kappa)}{\partial R_G^t}\right|_{R_V^{t\,*}, R_G^{t\,*}, \lambda^*, \kappa^*} = 0, & \text{(6.17b)} \\ \lambda^* \cdot \left[\mathcal{R}_{tot}^t(R_V^{t\,*}, R_G^{t\,*}) - R_{max}\right] = 0, & \text{(6.17c)} \\ \kappa^* \cdot \left[d_{tot}^t(R_V^{t\,*}, R_G^{t\,*}) - d_{max}\right] = 0. & \text{(6.17d)} \end{cases}$$

To solve the above equations, we use the Newton's method [138] to iteratively find the optimal solutions to the optimization problem in Equation (6.11). Suppose $\boldsymbol{\delta^k} = (R_V^{t\,k+1} - R_V^{t\,k}, R_G^{t\,k+1} - R_G^{t\,k})^T$, which is the vector of update directions of primal variables in Newton's method. Given $(R_V^{t\,k}, R_G^{t\,k}, \lambda^k, \kappa^k)$, $(\boldsymbol{\delta^k}, \lambda^{k+1}, \kappa^{k+1})$ can be solved from the following linear system [138].

$$\begin{pmatrix} \nabla^2 L(R_V^{t\,k}, R_G^{t\,k}, \lambda^k, \kappa^k) & \nabla \mathcal{R}_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) & \nabla d_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) \\ \nabla \mathcal{R}_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) & 0 & 0 \\ \nabla d_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta^k} \\ \lambda^{k+1} - \lambda^k \\ \kappa^{k+1} - \lambda^k \end{pmatrix} = \begin{pmatrix} -\nabla \mathcal{D}_{tot}(R_V^{t\,k}, R_G^{t\,k}) \\ R_{max} - \mathcal{R}_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) \\ d_{max} - d_{tot}^t(R_V^{t\,k}, R_G^{t\,k}) \end{pmatrix} \tag{6.18}$$

With the solved $\boldsymbol{\delta^k}$, we can have $(R_V^{t\,k+1}, R_G^{t\,k+1})^T = (R_V^{t\,k}, R_G^{t\,k})^T + \boldsymbol{\delta^k}$. By repeating the above process, the optimal rate allocation $R_V^{t\,*}$ and $R_G^{t\,*}$ to Equation (6.11) will be achieved until exceeding the maximum number of iterations or $\boldsymbol{\delta^k} \leq \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is the vector of tolerable errors.

Based on the above discussions, we develop an efficient rate allocation algorithm, as illustrated in Algorithm 6.1. In Algorithm 6.1, the cloud server collects the required model parameters, determines the relationships of t, t_θ, and $\tilde{t}$, and then finds the corresponding optimal rate allocation $R_V^{t\,*}$ and $R_G^{t\,*}$. The rate allocation algorithm is conducted in every Δt, which is set as one second in our work.

Algorithm 6.1: Rate Allocation Algorithm for Hybrid Streaming Framework

Initialize model parameters $\rho,\ \mu,\ \nu,\ \gamma,\ \sigma_{ASTC}$, and the game data set Ω_θ for the requested game.
Estimate the expected game playing time $\tilde{t}$ according to the user's historical statistics, and set $t = 0$.
Set $t_\theta = \infty$, if client device has no graphics rendering capacity,
while $t\ <\ \tilde{t}$ **do**
  Check the game data set Ω^t in the sync buffer at cloud side.
  if $\Omega_\theta \subseteq \Omega^t$ **then**
    ${R_V^t}^* = 0$, ${R_G^t}^* = R_{max}$,
  else
    Compute $\mathcal{D}_{tot}({R_V^t}^*|_{\tilde{t}\leq t_\theta}, {R_G^t}^*|_{\tilde{t}\leq t_\theta})$, where
    ${R_V^t}^*|_{\tilde{t}\leq t_\theta} = \max\{R_V^t | d_{enc}^t(R_V^t) \leq d_{max} - d_{net}^t - d_{ren}^t - d_{dec}^t,\ \ R_V^t \leq R_{max}\}$,
    ${R_G^t}^*|_{\tilde{t}\leq t_\theta} = R_{max} - {R_V^t}^*$.
    Compute $\mathcal{D}_{tot}({R_V^t}^*|_{\tilde{t}>t_\theta}, {R_G^t}^*|_{\tilde{t}>t_\theta})$, where ${R_V^t}^*|_{\tilde{t}>t_\theta}$ and ${R_G^t}^*|_{\tilde{t}>t_\theta}$ can be acquired by repeatedly solve Equation (6.18) until exceeding the maximum number of iterations or $\boldsymbol{\delta^k} \leq \boldsymbol{\epsilon}$.
    if $\mathcal{D}_{tot}({R_V^t}^*|_{\tilde{t}\leq t_\theta}, {R_G^t}^*|_{\tilde{t}\leq t_\theta}) \leq \mathcal{D}_{tot}({R_V^t}^*|_{\tilde{t}>t_\theta}, {R_G^t}^*|_{\tilde{t}>t_\theta})$ **then**
      ${R_V^t}^* = {R_V^t}^*|_{\tilde{t}\leq t_\theta}$, and ${R_G^t}^* = {R_G^t}^*|_{\tilde{t}\leq t_\theta}$.
    else
      ${R_V^t}^* = {R_V^t}^*|_{\tilde{t}>t_\theta}$, and ${R_G^t}^* = {R_G^t}^*|_{\tilde{t}>t_\theta}$.
    end if
  end if
  $t\ =\ t + \Delta t$
end while

6.4 Performance Evaluation

In this section, we conduct experiments to evaluate the proposed hybrid streaming framework and the rate allocation algorithm.

6.4.1 Experiment Setup

In our experiments, a server with Intel i7 CPU 3.07GHz, 12G RAM, and NVIDIA GeForce GTX 660 is used as the cloud server, and a laptop with Intel i5 2.5GHz, 4G RAM, and Intel HD Graphics 4000 is used as the client device. We tested two games: the multi-player online role-playing game WoW released by Blizzard Entertainment [131] and the first person shooter game Angry Bots released by Unity Technology [139]. Since the game

WoW is a proprietary game, the game source is closed to the public. Thus, we exported the game models and textures of WoW and rendered 3D game scenes with these assets. Two WoW game scenes are used in the test: the Orc warrior scene and the Human warrior scene. The game Angry Bots is an open-source game. The game assets and scripts are available to the public. Besides the game software, we use WoW Model Viewer to export the WoW game assets, use OpenSceneGraph to render the 3D game scene, use OpenMesh to generate progressive mesh representation of game models, use Live555 to set up the RTSP server for video streaming, use x264 [140] as video codec, and use ffplay [141] as the media player at client. The screen resolution of all games is set to 1280×720 (720P). For fair comparisons, we need to make sure the game screens are compared with identical game environment, i.e. the avatars in the game are played with the same movement and the identical actions. To achieve this, we record the avatar's animation path. By replaying the animation event, the cloud server can control the avatars to repeatedly move in certain game scenes and perform the exactly same actions. Each game test lasts for five minutes. The game screenshots are shown in Figure 6.11. During the evaluation, the captured game frame at cloud server is taken as the benchmark. We adopt PSNR as the visual quality metric by calculating the distortion between the displayed game frame at client and the benchmark at server. To simulate the network conditions, we use dummynet [142] to control the network bandwidth. Specifically, the maximal bit rate R_{max} varies from 1 Mbps to 6 Mbps. When R_{max} is under 2 Mbps (inclusive), the low rate textures are transmitted to client to save bandwidth. Otherwise, the high rate textures are used.

6.4.2 Comparison with Video Streaming Approach

We first compare the proposed hybrid streaming framework with the video streaming approach. The comparison of PSNR performance under different bandwidths is shown in Figure 6.12. We can see that the proposed hybrid streaming framework can achieve a significantly higher average PSNR than the video streaming under the same bandwidth constraint. The PSNR of the hybrid streaming increases by 8.35 dB, 6.54 dB, and 8.61 dB in average than the video streaming in Orc warrior, Human warrior, and Angry Bots scenes, respectively. In the hybrid streaming framework, cloud server gradually transmits game data to client. After receiving the required game data, the client can

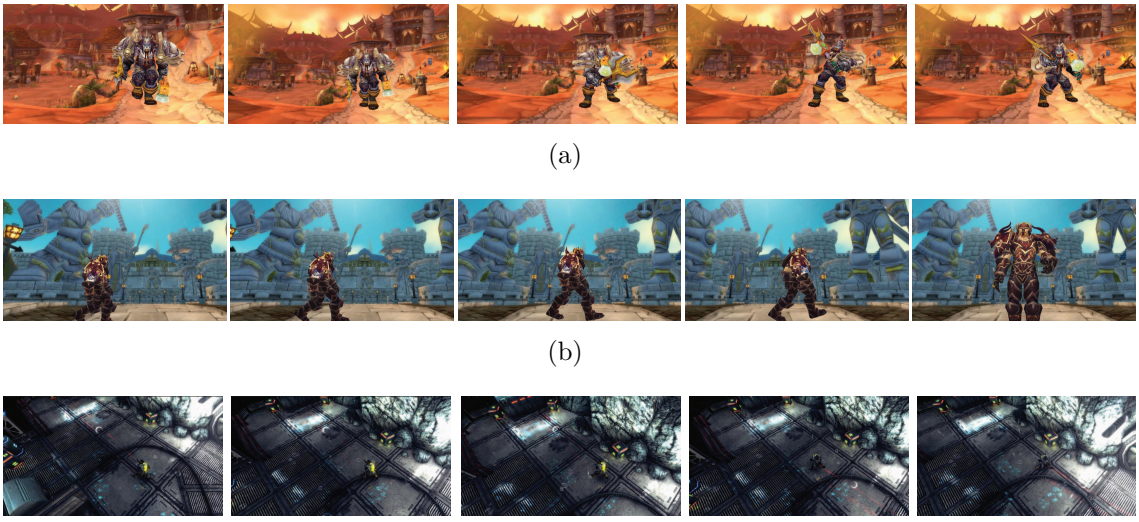


Figure 6.11: Game screenshots: (a) WoW Orc warrior scene, (b) WoW human warrior scene, and (c) Angry Bots scene.

locally render a game frame, which has a better quality than the decoded video frame. Thus, the average PSNR is improved. As shown in Figure 6.12, the PSNR curve of the hybrid streaming becomes flat when R_{max} is relatively large, which corresponds to the distortion of compressed textures. To give a clear illustration, Figure 6.13 and Figure 6.14 compare the temporal PSNR performance when R_{max} is 2 Mbps and 6 Mbps, respectively. In Figure 6.13 and Figure 6.14, the initial PSNR of hybrid streaming is lower than the video streaming, because part of bit rates are allocated for delivering the game data and the quality of the video frame is accordingly affected. After a short period, the client has received all the game data and rendered a high quality frame. Thus, the PSNR of hybrid streaming increases. As shown in Figure 6.13(c), the hybrid streaming has an initial PSNR of 29.62 dB, which is 5.86 dB lower than video streaming, but after $t_\theta = 120$ seconds, the hybrid streaming reaches 44.25 dB, which is 7.52 dB higher than video streaming. Overall, the average PSNR in hybrid streaming is 37.91 dB, which increases by 3.65 dB compared to video streaming in Figure 6.13(c). Comparing Figure 6.13 and Figure 6.14, the cloud server spends less time on delivering game data when R_{max} is 6 Mbps, and as a result, t_θ in Figure 6.14 can be reached earlier than that in Figure 6.13.

We also investigate the impact of response delay on the visual quality of cloud gaming.

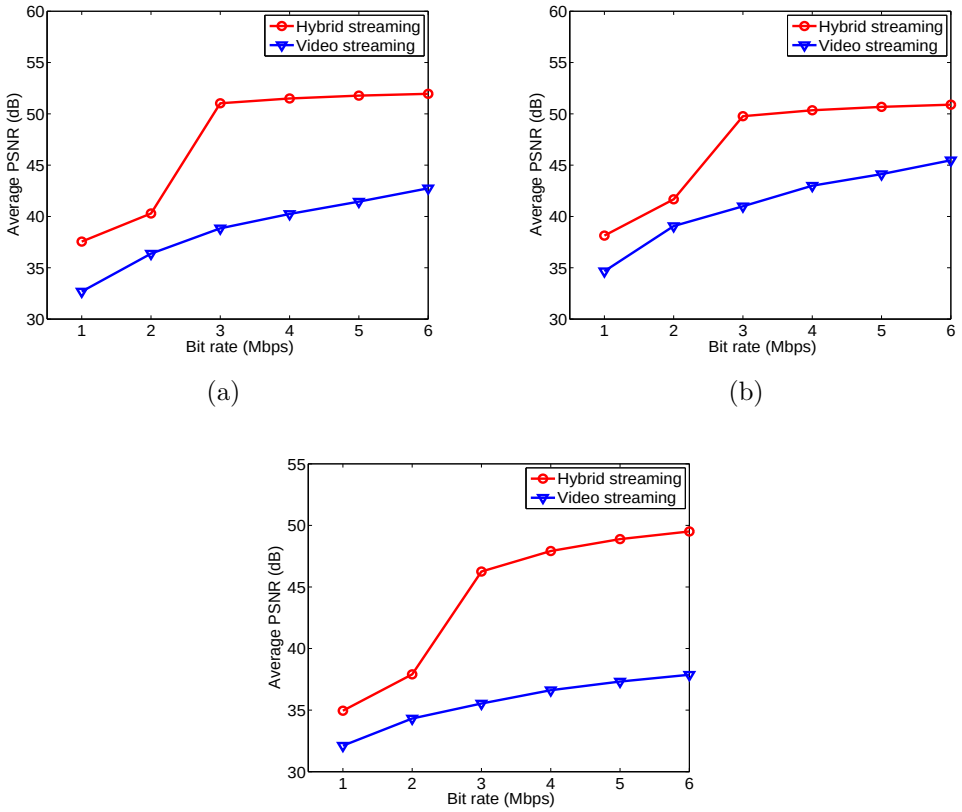


Figure 6.12: Comparison of the PSNR performance under different bandwidths: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

In the experiments, the network delay is set as 40 ms, the maximal bit rate R_{max} is 6 Mbps, and the response delay d_{max} is set as $\{120,\ 130,\ 140,\ 150,\ 200,\ 300\}$ ms. Figure 6.15 shows the comparison of PSNR for different response delays. From Figure 6.15, it can be seen that the average PSNR increases with d_{max} but becomes quite flat for a larger d_{max}. This is because that a low response delay d_{max} leads to a low encoding delay d_{enc}, which constrains the coding bit rate. When increasing d_{max}, the limit of coding rate is increased, leading to a higher visual quality. But when d_{max} increases to a certain value, the bandwidth R_{max} becomes the new bottleneck to limit coding rate, and hence the average PSNR tends to be flat. As shown in Figure 6.15, d_{max} has a less impact on the hybrid streaming framework. We analyzed the reason. In hybrid streaming, when R_V^t is constrained by d_{max}, the extra bit rates will be allocated to R_G^t, leading to a faster

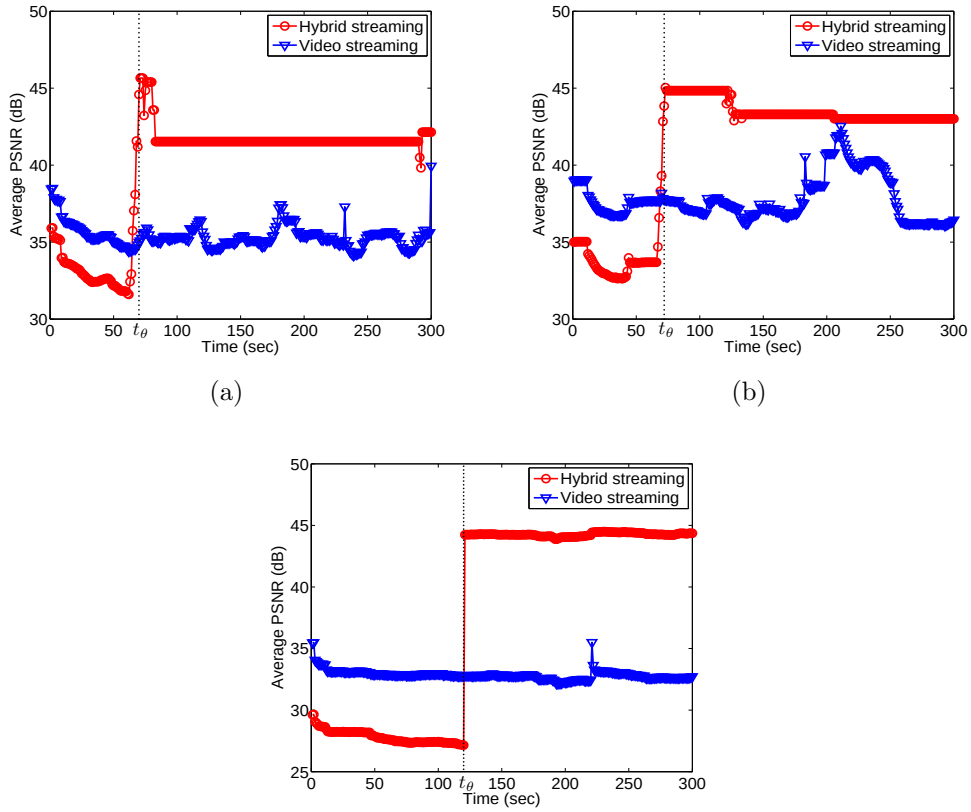


Figure 6.13: Comparison of the temporal PSNR performance when $R_{max} = 2$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

delivery of game data. Once receiving all the game assets, the client can locally render the graphics frame. Furthermore, given $d_{max} = 150$ ms and $R_{max} = 6$ Mbps, we compare the PSNR under different network delays in Figure 6.16. It can be seen from Figure 6.16 that the average PSNR decreases with the increase of network delay, and the reduction of PSNR in the hybrid steaming is lower than the video streaming. The reason is that a high PSNR of locally rendered frame is not affected by the network delay. As shown in Figure 6.16(a), when the network delay $d_{net} = 80$ ms, the hybrid streaming can achieve the average PSNR of 49.41 dB, which represents a 19.66 dB improvement over the video streaming.

Next, we compare the overall data size in the hybrid streaming and the video streaming. Figure 6.17 presents the overall data size with the maximal bit rate R_{max} fixed at

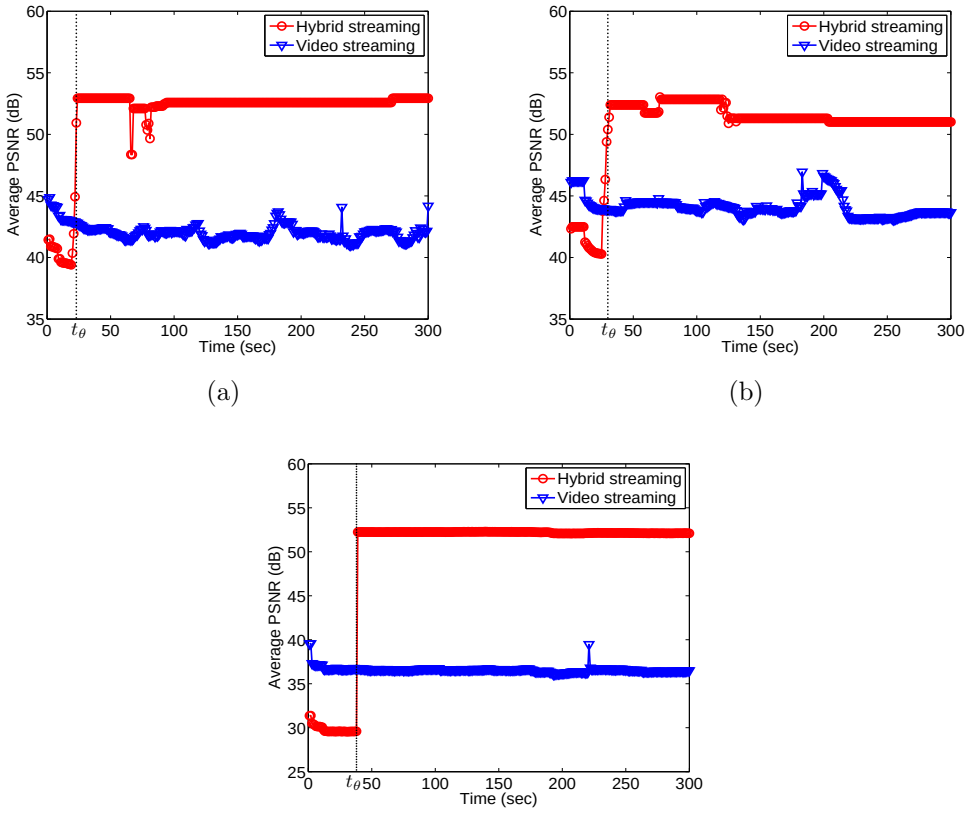
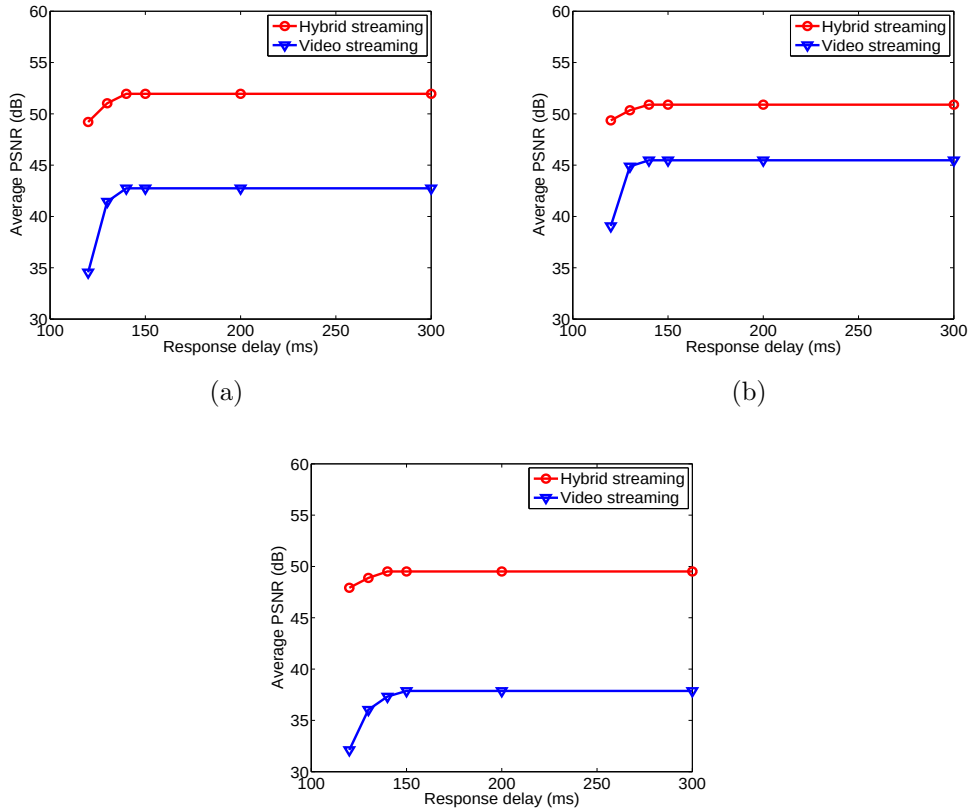


Figure 6.14: Comparison of the temporal PSNR performance when $R_{max} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

2 Mbps and 6 Mbps. It can be seen that the hybrid streaming can effectively save the overall transmission data size. Compared to the video streaming, the hybrid streaming in Figure 6.17(c) reduces the overall transmission data size by 60.55% with $R_{max} = 2$ Mbps and by 85.08% with $R_{max} = 6$ Mbps. Moreover, we observe that the overall transmission in hybrid streaming at 6 Mbps is almost same as that at 2 Mbps. This is because that a higher R_{max} can allocate a higher rate for graphics stream R_G^t and accordingly reduce the length of video streaming period, leading to a lower overall data size. To give a clear illustration, Figure 6.18 shows the accumulated data size in each second when R_{max} is 2 Mbps. In Figure 6.18, the hybrid streaming has the similar transmission data size as the video streaming at the beginning period. After t_θ, the required game data has been received by client and the locally rendered frame is displayed as game frame. Since then,



(a)

(b)

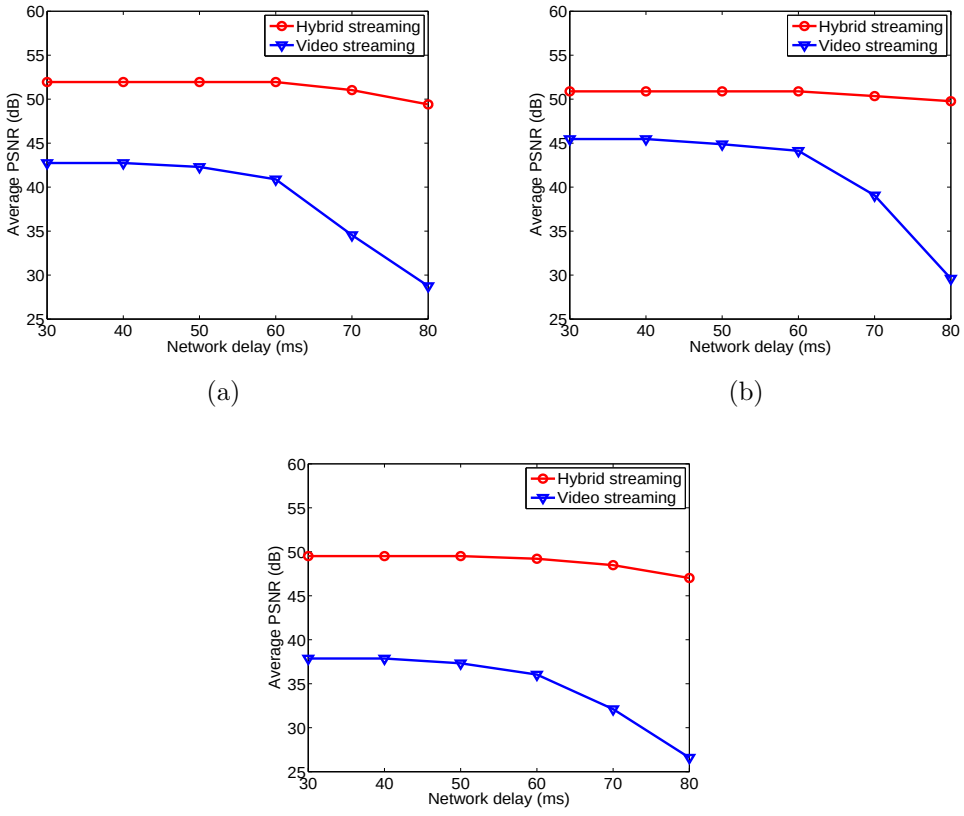
(c)

Figure 6.15: Comparison of the PSNR performance for different response delays when $R_{max} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

there is no more transmission between the server and the client in hybrid streaming. As a result, the hybrid streaming consumes less overall bit rates compared to video streaming.

6.4.3 Comparison with Graphics Streaming Approach

In this subsection, we compare the proposed hybrid streaming framework with the graphics streaming approach. In graphics streaming, the game assets are transmitted to client. After receiving all the necessary assets, the client can start rendering the game frame. For fair comparison, we use the same textures and game models in the hybrid streaming and graphics streaming. Figure 6.19 compares the PSNR performance under different bandwidths. As shown in Figure 6.19, the hybrid streaming can achieve a higher PSNR



(a) (b)

(c)

Figure 6.16: Comparison of the PSNR performance under different network delays when d_{max} = 150 ms and R_{max} = 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

than the graphics streaming under the same bandwidth constraint. The PSNR of hybrid streaming increases by 3.31 dB, 3.68 dB, and 4.40 dB in average compared to the graphics streaming for the Orc warrior, Human warrior, and Angry Bots scenes, respectively. The reason is that graphics streaming takes a buffering period to collect all the required game assets. In this period, the client cannot render a high quality frame and has a very low PSNR. As a result, the overall quality is degraded. Compared to the graphics streaming, the proposed framework applies the streamed video frame in the initial period, and accordingly has a higher initial PSNR. To view the PSNR performance in more detail, Figure 6.20 and Figure 6.21 illustrate the temporal PSNR with the fixed R_{max} at 2 Mbps and 6 Mbps, respectively. From Figure 6.20 and Figure 6.21, we can

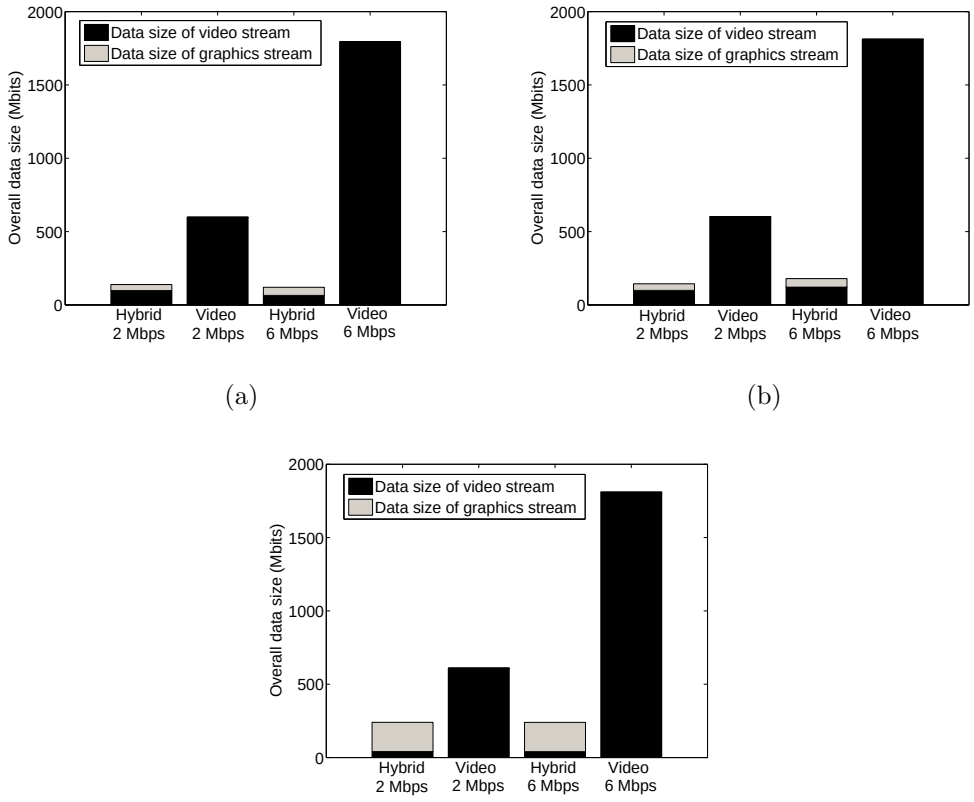


Figure 6.17: Comparison of the overall data size when R_{max} is fixed at 2 Mbps and 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

see that graphics streaming has a relatively low PSNR at the beginning, due to the lack of game data. In contrast, the proposed hybrid streaming achieves a higher PSNR in this period. As a cost, hybrid streaming takes a longer period to transmit all the game data to client. We also compare the overall data size in the hybrid streaming and the graphics streaming. Figure 6.22 shows the overall transmission data size with R_{max} fixed at 2 Mbps and 6 Mbps. Compared to the graphics streaming in Figure 6.22, the hybrid streaming consumes a higher transmission data size, due to the video streaming in the beginning period.

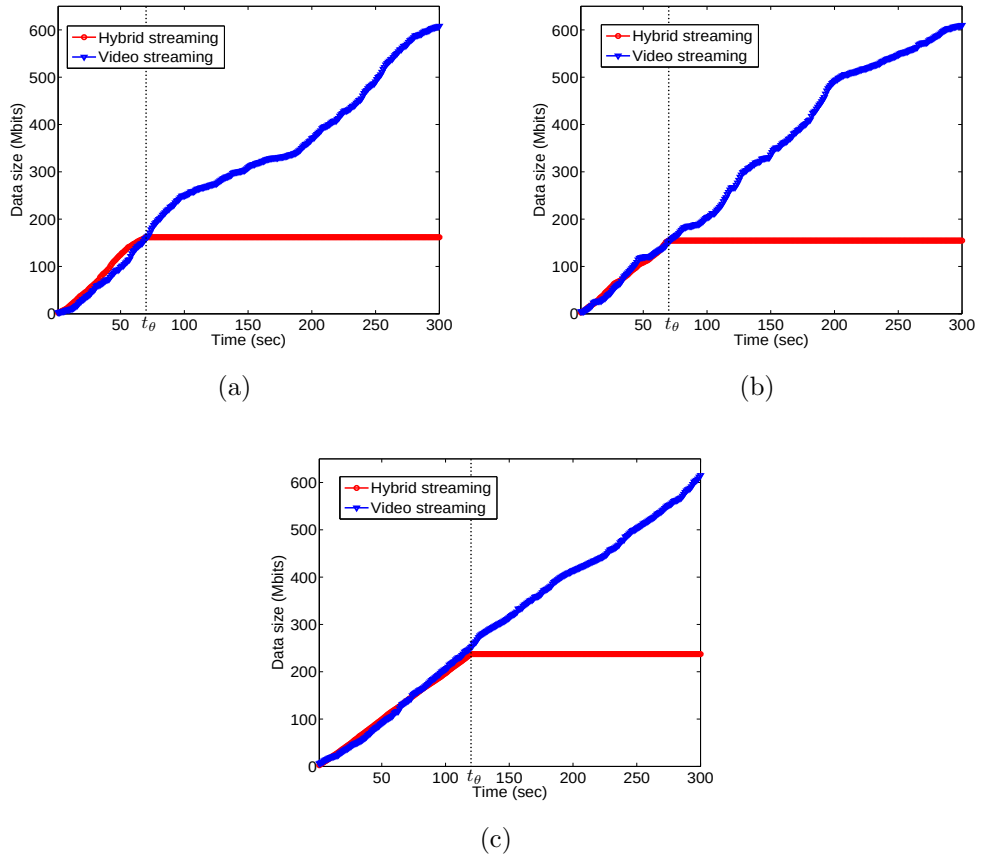


Figure 6.18: Comparison of the accumulated data size in each second when R_{max} is fixed at 2 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

6.4.4 Evaluation of Rate Allocation Algorithm

The proposed rate allocation algorithm is evaluated in this subsection. In the experiments, we select different rate allocation pairs of (R_V^t, R_G^t) and compare the P-SNR performance. The selected rate allocation pairs include: 1)the optimal allocation, which is acquired by solving the optimization problem in Equation (6.11); 2) $(R_V^t, R_G^t) = (0.7R_{max}, 0.3R_{max})$ represents rate allocation with a high rate for video stream; 3) $(R_V^t, R_G^t) = (0.5R_{max}, 0.5R_{max})$ represents even rate allocation; and 4) $(R_V^t, R_G^t) = (0.3R_{max}, 0.7R_{max})$ represents rate allocation with a high rate for graphics stream. Special cases, like $(R_V^t, R_G^t) = (R_{max}, 0)$ and $(R_V^t, R_G^t) = (0, R_{max})$, which represent the video streaming and graphics streaming respectively, have been compared

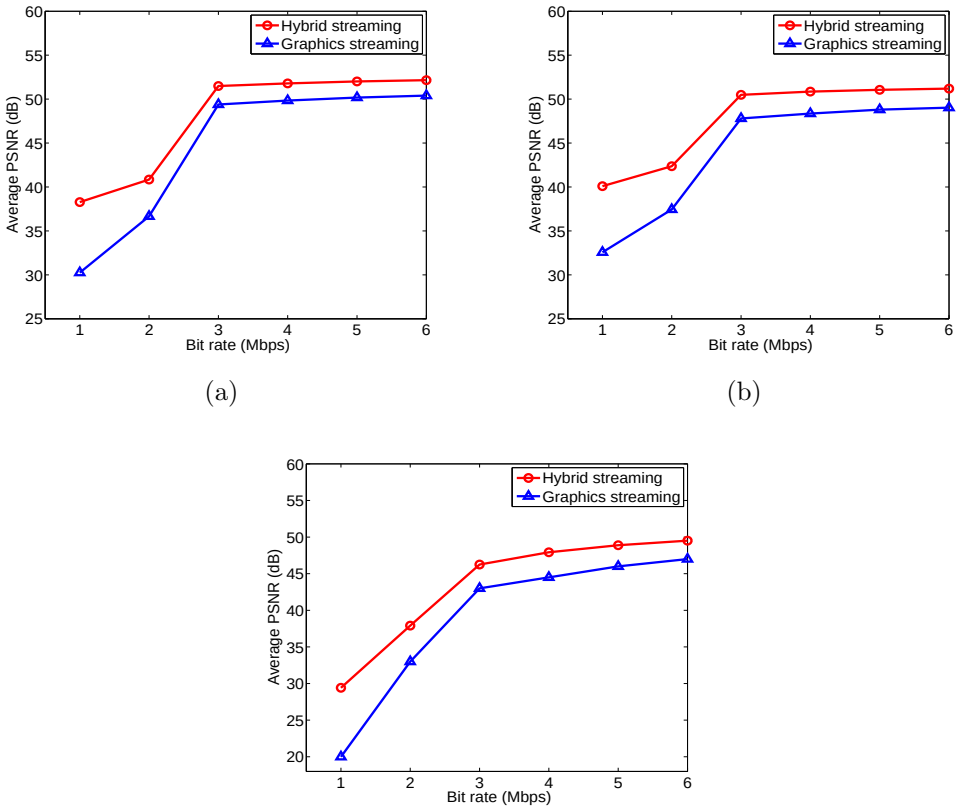


Figure 6.19: Comparison of the PSNR performance under different bandwidths: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

before and hence are not included in this evaluation. Figure 6.23 compares the average PSNR under different bandwidths when the response delay d_{max} is fixed at 150 ms. It can be seen from Figure 6.23 that for a given maximal response delay d_{max}, the proposed rate allocation algorithm can achieve a higher average PSNR compare to the other alternative rate allocation pairs. From Figure 6.23, we can also find that the rate allocation pair $(0.7R_{max}, 0.3R_{max})$ achieves a high PSNR when R_{max} is lower than 2 Mbps, but a low PSNR when R_{max} is 6 Mpbs. The reason is that the PSNR in video encoding is not linearly increased with R_V^t. Therefore, when R_V^t is relatively high, the increment of R_V^t cannot get an equivalent return on the PSNR performance, but will limit the allocation of R_G^t, leading to a slow improvement on the overall quality. To give a more detailed illustration, Figure 6.24 illustrates the temporal PSNR in the first 100 seconds when

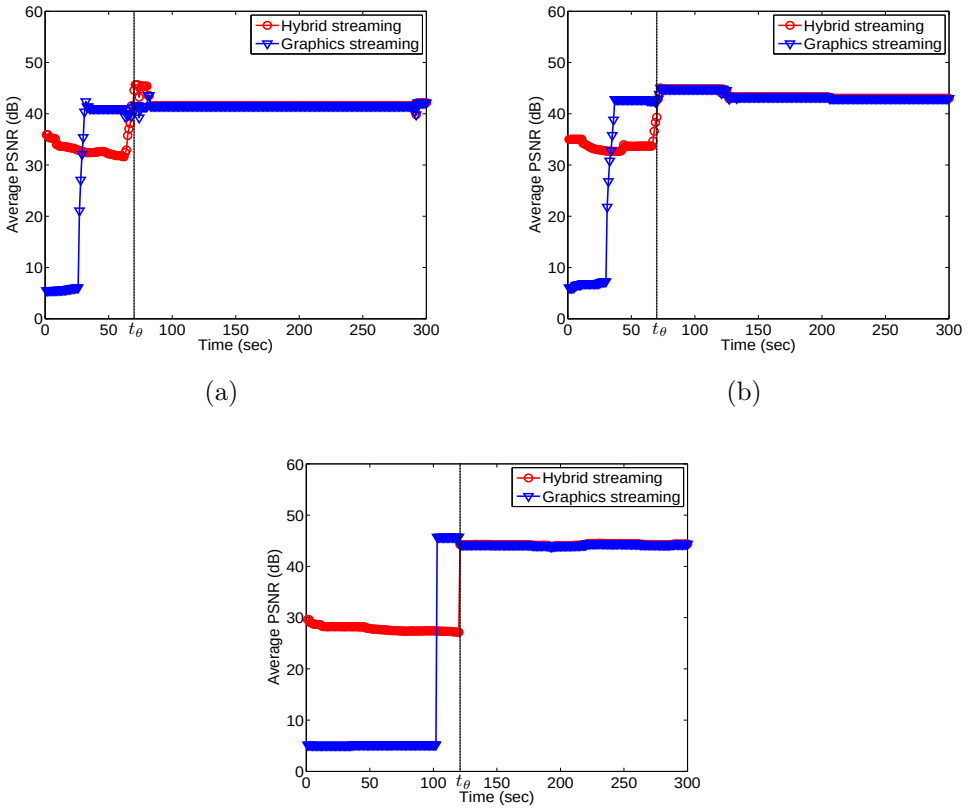


Figure 6.20: Comparison of the temporal PSNR performance when $R_{max} = 2$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

$R_{max} = 6$ Mbps. As shown in Figure 6.24, the rate allocation pair $(0.7R_{max}, 0.3R_{max})$ reaches the best quality at the beginning, but starts locally rendering later than other allocations. For the Angry Bots scene, the game assets have not been completely received in the first 100 seconds with the allocation pairs $(0.7R_{max}, 0.3R_{max})$. Compared to the other rate allocation pairs, the proposed rate allocation algorithm can achieve an optimal balance between the quality of video streaming and the length of video streaming.

6.5 Chapter Summary

In this chapter, we propose a novel hybrid streaming framework for cloud gaming to provide users with a high quality gaming experience under the constraints of bandwidth

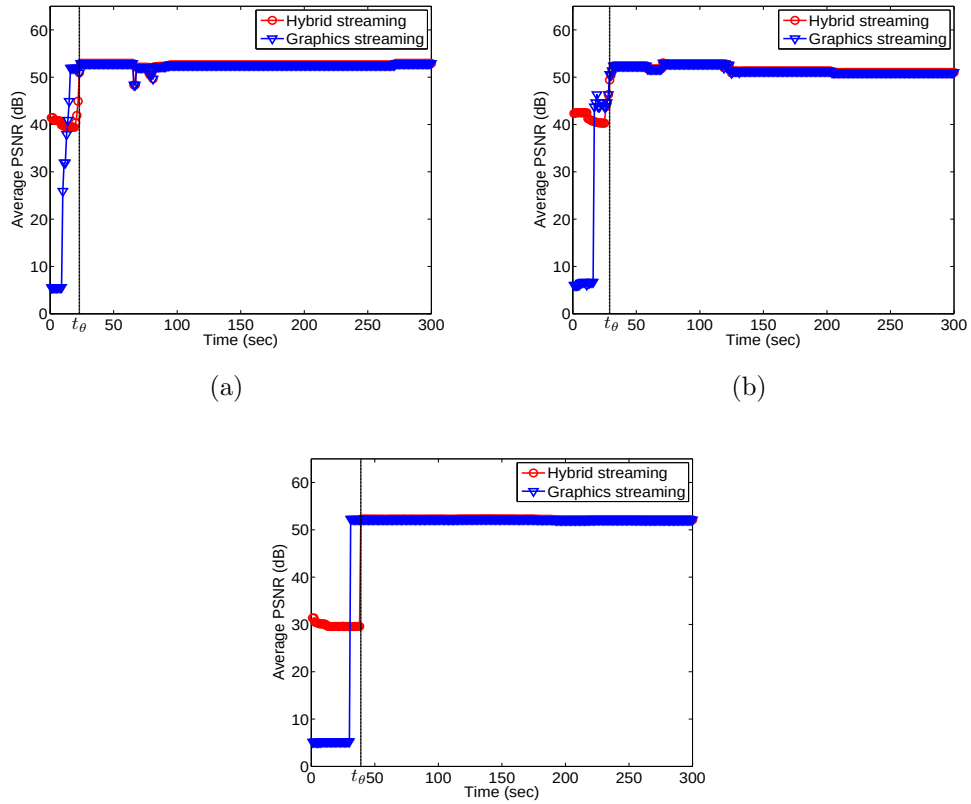


Figure 6.21: Comparison of the temporal PSNR performance when $R_{max} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

and response delay. The proposed framework jointly applies video streaming and graphics streaming. Based on the proposed framework, we study the rate allocation problem as a d-R-D optimization, in which we optimize the allocated rates between the video stream and the graphics stream to minimize the total distortion under the bandwidth and response delay constraints. We perform in-depth analysis on the relationship among the video distortion, the source rate, and the response delay in cloud gaming, and propose a practical rate allocation algorithm. Experimental results demonstrate that the proposed hybrid streaming framework can optimally allocate source rates between video stream and graphics stream to achieve the minimal distortion under the bandwidth and response delay constraints.

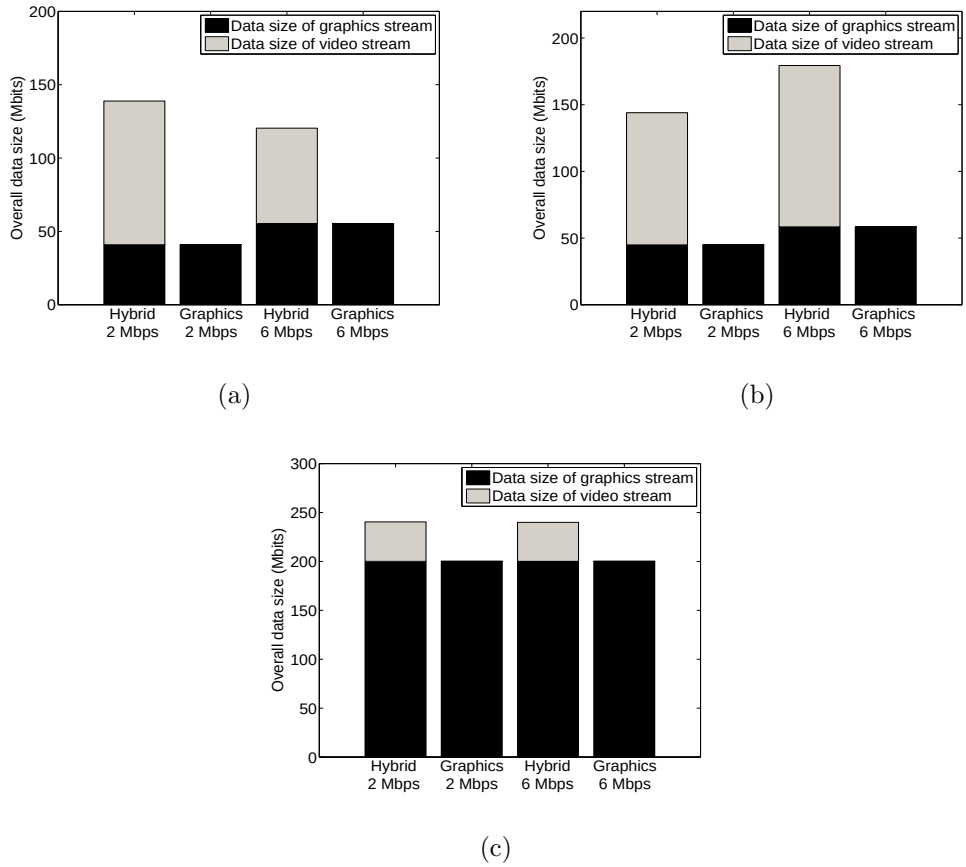


Figure 6.22: Comparison of the overall data size when R_{max} is fixed at 2 Mbps and 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

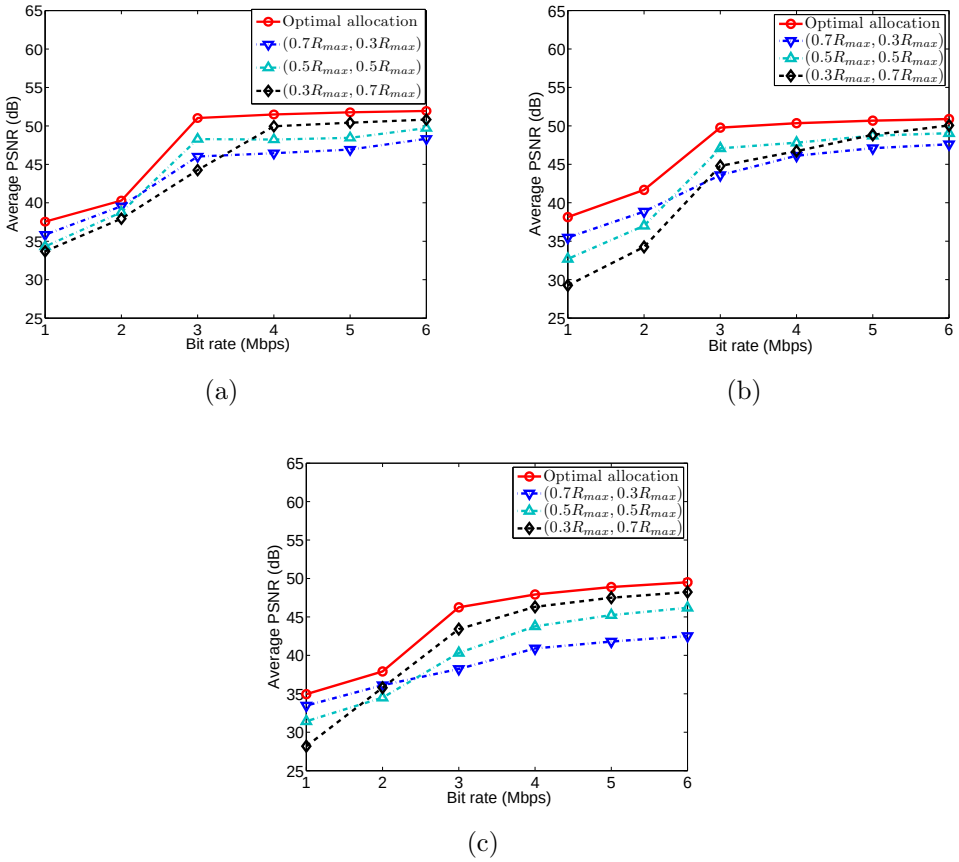


Figure 6.23: Comparison of the average PSNR under different bandwidths with d_{max} fixed at 150 ms: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

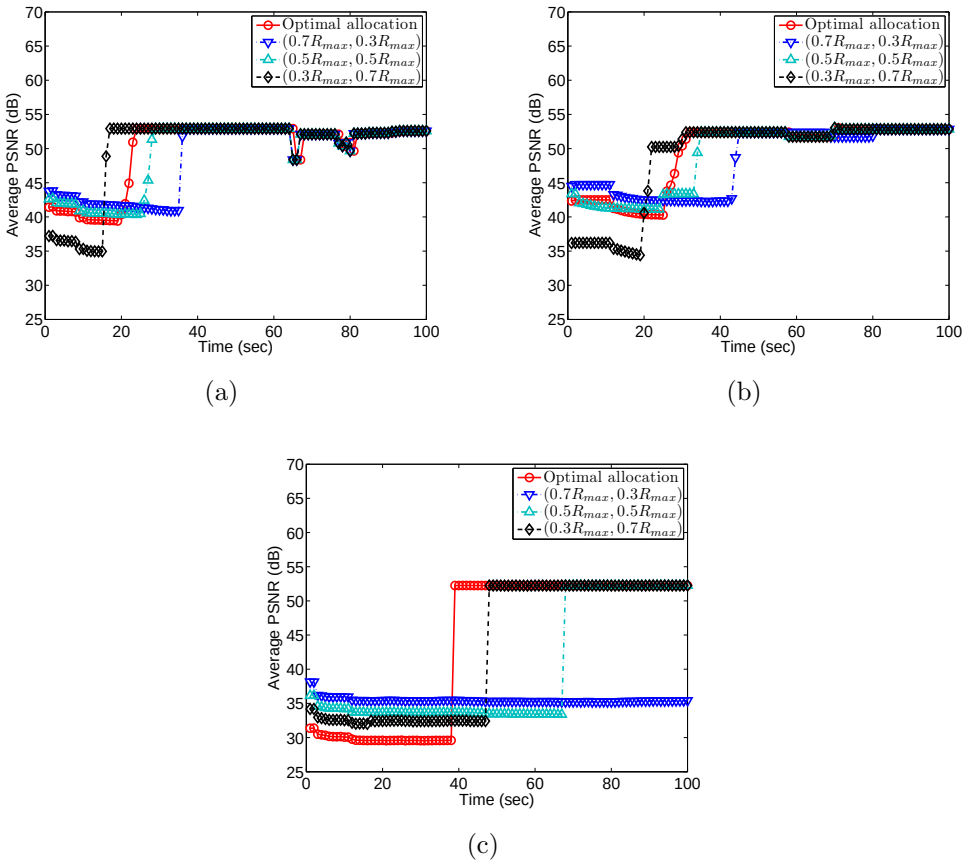


Figure 6.24: Comparison of the temporal PSNR in the first 100 seconds with R_{max} fixed at 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we study resource allocation for cloud-based multimedia services to guarantee the quality of service (QoS). The major contribution is a coherent cloud resource management framework, involving the cloud service modeling, dynamic resource configurations, user level and task level workload scheduling, and quality of experience (QoE) optimization for cloud gaming.

In **Chapter 3**, we studied the modeling for cloud based multimedia services. We introduced a queueing model to characterize the service process in multimedia cloud. The proposed queueing model was composed by three concatenated queueing systems, which were the schedule queue, the computation queue, and the transmission queue. We analyzed the equilibrium demands and derived the relationships between the response time and the allocated cloud resources. Based on the proposed queueing model, we investigated resource optimization problems for multimedia cloud computing in three different scenarios: single-service scenario, multi-service scenario, and priority-service scenario. In each scenario, we formulated and solved the response time minimization problem and the resource cost minimization problem, respectively. The proposed resource allocation schemes can optimally allocate cloud resources for different service scenarios to achieve the minimal response time under a certain budget or provide the satisfactory services at the minimal resource cost

In **Chapter 4**, we investigated the dynamic resource configuration. We first proposed

a two-time-scale resource configuration (TRC) scheme. The proposed scheme allocated virtual machines (VMs) in a mid-long time scale to reduce the cost, and dynamically re-allocated VMs in a fine-grained time scale to cope with the varying workload. Based on the TRC scheme, we studied resource optimization problems in single-site cloud scenario and multi-site cloud scenario, respectively. In each scenario, we formulated the resource cost minimization problem, which optimally allocated VMs to achieve the minimal resource cost, and the response time minimization problem, which dynamically adjusted VMs according to varying demands to minimize the average response time. Since the formulated problems were integer nonlinear programming, we proposed heuristics to efficiently allocate resources. Evaluation results showed that the proposed dynamic VMs configuration schemes not only allocated VMs to attain a low resource cost, but also dynamically reconfigured VMs to reduce the average response time.

In **Chapter 5**, we presented the study on workload scheduling to best utilize the allocated resources. Specifically, we investigated the user level and task level workload scheduling, respectively. At the user level, we applied queuing theory to model the request scheduling in cloud services. Based on the model, we optimized the workload assignment to minimize the required response time or minimize the total resource cost. We proposed lightweight heuristics to efficiently solve the formulated problems. In addition, we studied the workload scheduling at the task level. We introduced a directed acyclic graph to model the precedence constraints among tasks. Based on the model, we formulated and solved the execution time minimization problem for sequential structure, parallel structure, and mixed structure services. By conducting simulations, we demonstrated that the proposed workload scheduling scheme can make the best use of allocated resources to achieve a low response time.

In **Chapter 6**, we proposed a hybrid streaming framework for cloud gaming. The proposed framework jointly applied video streaming and graphics streaming. In the framework, the cloud server not only transmitted the encoded video frame, but also progressively transmitted graphics data. The received graphics data was used to render a graphics frame, which provided an additional reference candidate for video encoding. As the accumulation of graphics data at client, the frame rendered from the graphics buffer generated a lower residual to the captured frame than the previous video frame, which reduced the encoding bit rate. Based on the proposed framework, we formulated the rate allocation problem as a delay-rate-distortion (d-R-D) optimization, in which

we optimized the allocated rates between the video stream and the graphics stream to minimize the total distortion under the bandwidth and response delay constraints. We performed thorough analysis on the rate-distortion relationship and the response delay in cloud gaming, and proposed a practical rate allocation algorithm. Extensive simulations demonstrated that the proposed hybrid streaming framework can optimally allocate source rates to achieve a low distortion under the bandwidth and response delay constraints.

7.2 Future Work

Despite the advancements and research efforts have been made on cloud-based multimedia services, there are still many open problems in multimedia cloud computing. The following directions are worth for future exploration.

7.2.1 Big Data on Multimedia Cloud

Living in the information era, we are surrounded by an enormous amount of digital content. According to IDC digital universe study [143], the total amount of generated digital content in 2013 is 4.4 trillion gigabytes, and this number will be doubled in every two years. In such a situation, big data is emerged as the huge data sets, whose size and complexity are beyond the processing capability of personal computers. Cloud computing provides possible solutions to store and process big data. However, big data processing introduces challenges into cloud. Big data analysis requires distributed processing on cloud. However, due to the huge data size, the internal data transmission will introduce a large delay. Service providers, therefore, demand an efficient scheduling scheme which can support distributed processing with the minimal data transmission delay. The proposed directed acyclic graph (DAG) model in Chapter 5 can be used to model this problem, in which the distributed task is represented by the vertex and the delay of data transmission between two tasks is represented by the edge. An optimal scheduling problem can be formulated based on the model.

7.2.2 Collaborative Media Computing and Rendering

With the hardware upgrades for client devices in recent years, it becomes possible to collaboratively utilize resources on cloud server and client devices to compute, process, and render media contents. Cloud servers are powerful in computation, but cloud processing introduces a large transmission delay. By applying collaborative media computing and rendering, the local processing on client devices can effectively eliminate the transmission delay and accordingly improve the user experience. In Chapter 6, we have studied the collaboration of using graphics processing capability in cloud servers and client devices to improve the cloud gaming experience. As the next step, we will extend our work to other media applications and explore the collaborative media computing and rendering.

7.2.3 Joint Resource Allocation of Cloud and Network

The current research work on resource allocation focuses on the resources in cloud data center, while ignoring the varying network conditions, which may affect the quality of user experience. For example, the cloud-gaming users sometimes cannot receive high-quality frames rendered at the cloud server, due to the unpredicted network performance and the limited bandwidth at client side. When allocating resources to cloud gaming servers, we should take the network performance into account. Therefore, a joint resource allocation of cloud computing and network conditions is necessary. By jointly optimizing the resources in the cloud, at the clients, and in the networks, we can achieve the highest end-to-end quality of experience for users.

7.2.4 Optimization of Internal Traffic Management in Cloud

In current cloud data center, the bandwidth resource is shared by all virtual servers. Due to the varying workload, the bandwidth demand at each server is dynamically changing, leading to an unpredicted internal transmission performance. Similar to the traffic management in physical world, cloud computing can control the internal traffic routing of user's work flow, and the physical positions of the start and destination servers. By optimizing the positions of servers and the traffic path, we can achieve the best network performance and reduce the risk of internal transmission congestion for cloud computing.

7.2.5 Mobile multimedia applications

Driven by the rapid advances of smart phones and mobile devices, mobile users not only expect a broadband Internet connection, but also desire a universal multimedia experience. However, due to the limited power and processing capacity, it is still a challenge to conduct the computation-intensive multimedia processing on mobile devices. Fortunately, the emerging cloud computing provides a solution. In this thesis, the proposed resource allocation methods are generic resource management schemes on cloud side, which are independent of client devices and transmission routes. In mobile applications, the wireless transmission and the content adaption have to be specially considered. In the next step, we will extend our work to the specific mobile multimedia applications by investigating how the proposed resource allocation methods can improve the performance of mobile multimedia applications.

Appendix A

Proofs

A.1 Derivation of Solution to the Optimization Problem (3.4)

According to Lagrange multiplier method [100], the optimal solution to the optimization problem (3.4) must simultaneously satisfy all conditions in Equation (3.5). We first analyze the conditions in Equation (3.5). Since $\lambda^{(t)} - S^{(t)} < 0$ and $\mu_1 \left(\lambda^{(t)} - S^{(t)}\right) = 0$, we can get that the Lagrange multiplier $\mu_1 = 0$. Similarly, we can get $\mu_{2i} = 0$ ($\forall i = 1, 2, \ldots, N$) from $p_i\lambda^{(t)}F - C_i^{(t)} < 0$ and $\mu_{2i}\left(p_i\lambda^{(t)}F - C_i^{(t)}\right) = 0$, and get $\mu_3 = 0$ from $\lambda^{(t)}D - B^{(t)} < 0$ and $\mu_3\left(\lambda^{(t)}D - B^{(t)}\right) = 0$. Since $\mu_1 = 0$, $\mu_{2i} = 0$ ($\forall i = 1, 2, \ldots, N$), and $\mu_3 = 0$, the Lagrange function is simplified as

$$L\left(S^{(t)}, C_1^{(t)}, C_2^{(t)}, \ldots, C_N^{(t)}, B^{(t)}, \mu_4\right) = T_{sing}^{tot(t)} + \mu_4\left((\alpha S^{(t)} + \beta\sum_{i=1}^{N} C_i^{(t)} + \gamma B^{(t)})\bar{t} - \mathcal{C}_{max}\right),$$

Next, we calculate the partial derivation of Lagrange function with respect to each optimization variable as follows.

$$\begin{array}{rcl}
\frac{\partial}{\partial S^{(t)}} L\left(S^{(t)}, C_1^{(t)}, C_2^{(t)}, \ldots, C_N^{(t)}, B^{(t)}, \mu_4\right) & = & -\frac{1}{(S^{(t)} - \lambda^{(t)})^2} + \mu_4\alpha\bar{t} = 0, \\
\frac{\partial}{\partial C_i^{(t)}} L\left(S^{(t)}, C_1^{(t)}, C_2^{(t)}, \ldots, C_N^{(t)}, B^{(t)}, \mu_4\right) & = & -\frac{p_i F}{(C_i^{(t)} - p_i\lambda^{(t)}F)^2} + \mu_4\beta\bar{t} = 0, \ \forall i = 1, 2, \ldots, N \\
\frac{\partial}{\partial B^{(t)}} L\left(S^{(t)}, C_1^{(t)}, C_2^{(t)}, \ldots, C_N^{(t)}, B^{(t)}, \mu_4\right) & = & -\frac{D}{(B^{(t)} - \lambda^{(t)}D)^2} + \mu_4\gamma\bar{t} = 0.
\end{array} \tag{A.1}$$

Considering $S^{(t)} > \lambda^{(t)}$, $\mu_4 \geq 0$ and $-\frac{1}{(S^{(t)}-\lambda^{(t)})^2} + \mu_4 \alpha \bar{t} = 0$, we can derive that $\mu_4 > 0$ and $S^{(t)} = \sqrt{\frac{1}{\mu_4 \alpha \bar{t}}} + \lambda^{(t)}$. Similarly, we can derive that $C_i^{(t)} = \sqrt{\frac{p_i F}{\mu_4 \beta \bar{t}}} + p_i \lambda^{(t)} F$ ($\forall i = 1, 2, \ldots, N$), and $B^{(t)} = \sqrt{\frac{D}{\mu_4 \gamma \bar{t}}} + \lambda^{(t)} D$. Due to $\mu_4 \left(\left(\alpha S^{(t)} + \beta \sum_{i=1}^N C_i^{(t)} + \gamma B^{(t)} \right) \bar{t} - \mathcal{C}_{max} \right) = 0$ and $\mu_4 > 0$, we can derive that $\left(\alpha S^{(t)} + \beta \sum_{i=1}^N C_i^{(t)} + \gamma B^{(t)} \right) \bar{t} - \mathcal{C}_{max} = 0$. By representing $S^{(t)}$, $C_i^{(t)}$ ($\forall i = 1, 2, \ldots, N$), $B^{(t)}$ in terms of μ_4, we can have the following derivation.

$$
\begin{aligned}
& \left(\alpha S^{(t)} + \beta \sum_{i=1}^N C_i^{(t)} + \gamma B^{(t)} \right) \bar{t} - \mathcal{C}_{max} \\
= \; & \alpha \bar{t} \left(\sqrt{\tfrac{1}{\mu_4 \alpha \bar{t}}} + \lambda^{(t)} \right) + \beta \bar{t} \sum_{i=1}^N \left(\sqrt{\tfrac{p_i F}{\mu_4 \beta \bar{t}}} + p_i \lambda^{(t)} F \right) + \gamma \bar{t} \left(\sqrt{\tfrac{D}{\mu_4 \gamma \bar{t}}} + \lambda^{(t)} D \right) - \mathcal{C}_{max} \\
= \; & \alpha \bar{t} \sqrt{\tfrac{1}{\mu_4 \alpha \bar{t}}} + \alpha \bar{t} \lambda^{(t)} + \beta \bar{t} \sum_{i=1}^N \sqrt{\tfrac{p_i F}{\mu_4 \beta \bar{t}}} + \beta \bar{t} \sum_{i=1}^N p_i \lambda^{(t)} F + \gamma \bar{t} \sqrt{\tfrac{D}{\mu_4 \gamma \bar{t}}} + \gamma \bar{t} \lambda^{(t)} D - \mathcal{C}_{max} \\
= \; & \tfrac{\sqrt{\alpha \bar{t}}}{\sqrt{\mu_4}} + \sqrt{\beta \bar{t}} \sum_{i=1}^N \tfrac{\sqrt{p_i F}}{\sqrt{\mu_4}} + \tfrac{\sqrt{\gamma \bar{t}} \sqrt{D}}{\sqrt{\mu_4}} + (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t} - \mathcal{C}_{max} \\
= \; & \tfrac{1}{\sqrt{\mu_4}} \left(\sqrt{\alpha \bar{t}} + \sqrt{\beta \bar{t}} \sum_{i=1}^N \sqrt{p_i F} + \sqrt{\gamma \bar{t}} \sqrt{D} \right) + (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t} - \mathcal{C}_{max} \\
= \; & 0.
\end{aligned}
\tag{A.2}
$$

From Equation (A.2), we can derive that $\frac{1}{\sqrt{\mu_4}} = \frac{\mathcal{C}_{max} - (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t}}{\sqrt{\alpha \bar{t}} + \sqrt{\beta \bar{t}} \sum_{i=1}^N \sqrt{p_i F} + \sqrt{\gamma \bar{t}} \sqrt{D}}$. With the representation of $\frac{1}{\sqrt{\mu_4}}$, we can achieve the solution to the optimization problem (3.4) as follows

$$
\begin{aligned}
S^{(t)*} &= \sqrt{\tfrac{1}{\mu_4 \alpha \bar{t}}} + \lambda^{(t)} \\
&= \tfrac{\mathcal{C}_{max} - (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t}}{\left(\sqrt{\alpha} + \sqrt{\beta F} \sum_{i=1}^N \sqrt{p_i} + \sqrt{\gamma D} \right) \sqrt{\alpha t}} + \lambda^{(t)}, \\
C_i^{(t)*} &= \sqrt{\tfrac{p_i F}{\mu_4 \beta \bar{t}}} + p_i \lambda^{(t)} F \\
&= \tfrac{\left(\mathcal{C}_{max} - (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t} \right) \sqrt{p_i F}}{\left(\sqrt{\alpha} + \sqrt{\beta F} \sum_{i=1}^N \sqrt{p_i} + \sqrt{\gamma D} \right) \sqrt{\beta t}} + p_i \lambda^{(t)} F, \quad \forall i = 1, 2, \ldots, N, \\
B^{(t)*} &= \sqrt{\tfrac{D}{\mu_4 \gamma \bar{t}}} + \lambda^{(t)} D \\
&= \tfrac{\left(\mathcal{C}_{max} - (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t} \right) \sqrt{D}}{\left(\sqrt{\alpha} + \sqrt{\beta F} \sum_{i=1}^N \sqrt{p_i} + \sqrt{\gamma D} \right) \sqrt{\gamma \bar{t}}} + \lambda^{(t)} D, \\
\mu_1 &= 0, \\
\mu_{2i} &= 0 \quad \forall i = 1, 2, \ldots, N, \\
\mu_3 &= 0, \\
\mu_4 &= \left(\tfrac{\sqrt{\alpha \bar{t}} + \sqrt{\beta \bar{t}} \sum_{i=1}^N \sqrt{p_i F} + \sqrt{\gamma \bar{t}} \sqrt{D}}{\mathcal{C}_{max} - (\alpha + \beta F + \gamma D) \lambda^{(t)} \bar{t}} \right)^2.
\end{aligned}
\tag{A.3}
$$

Therefore, the optimal solution to the optimization problem in Equation (3.4) can be achieved.

A.2 Proof of Theorem 3.1

Proof. Substituting the optimal solution (3.6) to the objective function in Equation (3.4), we can get

$$T_{sing}^{tot(t)} = \frac{\left(\sqrt{\alpha} + \sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i} + \sqrt{\gamma D}\right)^2}{\frac{\mathcal{C}_{max}}{t} - (\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)})} = \frac{\left(\xi + \sqrt{\beta F}y\right)^2}{\delta}, \tag{A.4}$$

where $\xi = \sqrt{\alpha} + \sqrt{\gamma D} > 0$, $y = \sum_{i=1}^{N}\sqrt{p_i} > 0$, and $\delta = \frac{\mathcal{C}_{max}}{t} - (\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)})$. We can find that the schedule probabilities only affect the term y. In the following proof, we will first prove that $T_{sing}^{tot(t)}$ monotonically increases with y. Based on this conclusion, we will find the schedule probabilities to achieve the minimal y and accordingly achieve the minimal $T_{sing}^{tot(t)}$. Similarly, we will obtain the schedule probabilities for the maximal $T_{sing}^{tot(t)}$.

As the first step, we will prove that $T_{sing}^{tot(t)}$ is a monotonically increasing function of y. Given $0 < y_1 < y_2$, we can get that $(\xi + \sqrt{\beta F}y_1)^2 < (\xi + \sqrt{\beta F}y_2)^2$, due to $\xi > 0$ and $\sqrt{\beta F} > 0$. Moreover, according to constraints in Equation (3.4), we can get $\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)} = \alpha\lambda^{(t)} + \beta F\sum_{i=1}^{N} p_i\lambda^{(t)} + \gamma D\lambda^{(t)} < \alpha S + \beta\sum_{i=1}^{N} C_i + \gamma B \leq \frac{\mathcal{C}_{max}}{t}$. Thus, $\delta = \frac{\mathcal{C}_{max}}{t} - (\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)})$ in Equation (A.4) is positive. We therefore derive that $\frac{(\xi+\sqrt{\beta F}y_1)^2}{\delta} < \frac{(\xi+\sqrt{\beta F}y_2)^2}{\delta}$ for any given $0 < y_1 < y_2$. So, $T_{sing}^{tot(t)}$ in Equation (A.4) is a monotonically increasing function of y. Thus, the minimal $T_{sing}^{tot(t)}$ will be achieved at the minimal y, while the maximal $T_{sing}^{tot(t)}$ will be achieved at the maximal y.

To achieve the minimal $T_{sing}^{tot(t)}$ in Equation (A.4), we need to find the minimal $y = \sum_{i=1}^{N}\sqrt{p_i}$. We have the inequation $y^2 = (\sum_{i=1}^{N}\sqrt{p_i})^2 = \sum_{i=1}^{N} p_i + \sum_{i=1}^{N}\sum_{i'=i+1}^{N}(2\sqrt{p_i}\sqrt{p_{i'}})$ $\geq \sum_{i=1}^{N} p_i = 1$, and the equality holds if and only if $p_i p_{i'} = 0$ $(i \neq i')$. Therefore, solving the equations $\sum_{i=1}^{N} p_i = 1$ and $p_i p_{i'} = 0$ $(i \neq i')$, we can get that y reaches the minimal value 1 when probability p_i is 1 and all other probabilities $p_{i'}$ $(i' \neq i)$ are 0. Therefore, when schedule probability p_i $(\forall i = 1, \ldots, N)$ is 1 and all other probabilities $p_{i'}$ $(i' \neq i, i' = 1, \ldots, N)$ are 0, we can achieve the minimal response time, which is given

by

$$T_{sing}^{tot(t)(min)} = \frac{\left(\sqrt{\alpha} + \sqrt{\beta F} + \sqrt{\gamma D}\right)^2}{\frac{\mathcal{C}_{max}}{t} - \left(\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)}\right)}. \tag{A.5}$$

The first statement in Theorem 3.1 is proved.

Similarly, we will obtain the maximal y, leading to the maximal $T_{sing}^{tot(t)}$. From inequation $(\sqrt{p_i} - \sqrt{p_{i'}})^2 \geq 0$, we can derive that $p_i + p_{i'} \geq 2\sqrt{p_i}\sqrt{p_{i'}}$, $(\forall i, i' = 1, \ldots, N)$. Thus, we can derive the following relationship

$$\begin{array}{lllll}
y^2 & = (\sum_{i=1}^{N} \sqrt{p_i})^2 & & & \\
 & = \sum_{i=1}^{N} p_i + \sum_{i=1}^{N} \sum_{i'=i+1}^{N} (2\sqrt{p_i}\sqrt{p_{i'}}) & & & \\
 & \leq \sum_{i=1}^{N} p_i + \sum_{i=1}^{N} \sum_{i'=i+1}^{N} (p_i + p_{i'}) & & & \\
= & p_1+ & p_2+ & p_3 + \cdots + & p_{N-1} + p_N \\
+(N-1) & p_1+ & p_2+ & p_3 + \cdots + & p_{N-1} + p_N \\
+ & (N-2) & p_2+ & p_3 + \cdots + & p_{N-1} + p_N \\
+ \cdots & & & & \\
+ & & & & p_{N-1} + p_N \\
= N \sum_{i=1}^{N} p_i & & & & \\
= N, & & & &
\end{array}$$

and the equality holds if and only if $p_i = p_{i'}$ $(i \neq i', \forall i, i' = 1, \ldots, N)$. Since $\sum_{i=1}^{N} p_i = 1$, we can get that when $p_1 = p_2 = \ldots = p_N = \frac{1}{N}$, the maximal $y = \sqrt{N}$ is achieved. Thus, when the schedule probabilities are all equal, i.e. $p_1 = p_2 = \cdots = p_N = \frac{1}{N}$, the maximal response time is achieved. The maximal response time is given by

$$T_{sing}^{tot(t)(max)} = \frac{\left(\sqrt{\alpha} + \sqrt{N}\sqrt{\beta F} + \sqrt{\gamma D}\right)^2}{\frac{\mathcal{C}_{max}}{t} - \left(\alpha\lambda^{(t)} + \beta F\lambda^{(t)} + \gamma D\lambda^{(t)}\right)}. \tag{A.6}$$

The second statement in Theorem 3.1 is proved. □

A.3 Proof of Theorem 3.2

Proof. Substituting the optimal analytical solution in (3.8) to the objective function in Equation (3.7), we can get the total resource cost as

$$\mathcal{C}_{sing}^{tot(t)} = \frac{(\sqrt{\alpha} + \sqrt{\beta F}\sum_{i=1}^{N}\sqrt{p_i} + \sqrt{\gamma D})^2\bar{t}}{\tau} + (\alpha + \beta F + \gamma D)\lambda^{(t)}\bar{t} = \frac{(\xi + \sqrt{\beta F}y)^2\bar{t}}{\tau} + \varpi, \tag{A.7}$$

where $\xi = \sqrt{\alpha} + \sqrt{\gamma D} > 0$, $y = \sum_{i=1}^{N}\sqrt{p_i} > 0$, and $\varpi = (\alpha + \beta F + \gamma D)\lambda^{(t)}\bar{t} > 0$.

Similar to the proof of Theorem 1, we first prove that the resource cost $\mathcal{C}_{sing}^{tot(t)}$ in Equation (A.7) is a monotonically increasing function of y. Given $0 < y_1 < y_2$, we can derive that $\frac{(\xi+\sqrt{\beta F}y_1)^2\bar{t}}{\tau} + \varpi < \frac{(\xi+\sqrt{\beta F}y_2)^2\bar{t}}{\tau} + \varpi$, since $\xi > 0$, $\bar{t} > 0$, $\tau > 0$ and $\varpi > 0$. So, $\mathcal{C}_{sing}^{tot(t)}$ in Equation (A.7) is a monotonically increasing function of y. Thus, the minimal resource cost will be achieved at the minimal y, while the maximal resource cost achieved at the maximal y.

To achieve the minimal resource cost, we need to find the minimal y. From the proof in Theorem 1, we can get that $y = \sum_{i=1}^{N}\sqrt{p_i}$ reaches the minimal value 1 when probability p_i ($\forall i = 1, \ldots, N$) is 1 and all other probabilities $p_{i'}$ ($i' \neq i, i' = 1, \ldots, N$) are 0. The minimal total resource cost is given by

$$\mathcal{C}_{sing}^{tot(t)(min)} = \frac{(\sqrt{\alpha} + \sqrt{\beta F} + \sqrt{\gamma D})^2\bar{t}}{\tau} + (\alpha + \beta F + \gamma D)\lambda^{(t)}\bar{t}. \tag{A.8}$$

Therefore, the first statement in Theorem 3.2 is proved.

To get the maximal resource cost, we need to find the maximal y. According to the proof in Theorem 1, $y = \sum_{i=1}^{N}\sqrt{p_i}$ reaches the maximal value $\sqrt{N}$ when schedule probability $p_1 = p_2 = \ldots = p_N = \frac{1}{N}$. The maximal total resource cost is given by

$$\mathcal{C}_{sing}^{tot(t)(max)} = \frac{(\sqrt{\alpha} + \sqrt{N}\sqrt{\beta F} + \sqrt{\gamma D})^2\bar{t}}{\tau} + (\alpha + \beta F + \gamma D)\lambda^{(t)}\bar{t}. \tag{A.9}$$

Thus, the second statement in Theorem 3.2 is proved. □

A.4 Proof of Theorem 3.3

Proof. When the number of services is 1 in the multi-service scenario (suppose that the only provided service is class-i service), the optimization problem in Equation (3.12) will be changed to:

$$\begin{aligned} \underset{\{S^{(t)},C_i^{(t)},B^{(t)}\}}{\text{Minimize}} \quad & T_{mult}^{tot(t)} = \frac{1/S^{(t)}}{1-\lambda_i^{(t)}/S^{(t)}} + \frac{F_i/C_i^{(t)}}{1-\lambda_i^{(t)}F_i/C_i^{(t)}} + \frac{D_i/B^{(t)}}{1-\lambda_i^{(t)}D_i/B^{(t)}} \\ \text{subject to} \quad & \\ & \lambda_i^{(t)} < S^{(t)}, \\ & \lambda_i^{(t)}F_i < C_i^{(t)}, \\ & \lambda_i^{(t)}D_i < B^{(t)}, \\ & (\alpha S^{(t)} + \beta C_i^{(t)} + \gamma B^{(t)})\bar{t} \leq \mathcal{C}_{max}. \end{aligned} \tag{A.10}$$

We can find that the optimization problem in Equation (3.4) is identical to the optimization problem in Equation (A.10), when the schedule probabilities in Equation (3.4) are $p_i = 1$ and $p_{i'} = 0$ ($i' \neq i$). According to Theorem 3.1, the minimal response time in Equation (3.4) will be achieved under such schedule probability settings. Therefore, Theorem 3.3 is proved. □

A.5 Proof of Theorem 3.4

Proof. When there is only one class of service provided (suppose that it is class-i service), the resource cost minimization problem in Equation (3.13) is changed to the following:

$$\begin{aligned} \underset{\{S^{(t)},C_i^{(t)},B^{(t)}\}}{\text{Minimize}} \quad & \mathcal{C}_{mult}^{tot(t)} = (\alpha S^{(t)} + \beta C_i^{(t)} + \gamma B^{(t)})\bar{t} \\ \text{subject to} \quad & \\ & \lambda_i^{(t)} < S^{(t)}, \\ & \lambda_i^{(t)}F_i < C_i^{(t)}, \\ & \lambda_i^{(t)}D_i < B^{(t)}, \\ & \frac{1/S^{(t)}}{1-\lambda_i^{(t)}/S^{(t)}} + \frac{F_i/C_i^{(t)}}{1-\lambda_i^{(t)}F_i/C_i^{(t)}} + \frac{D_i/B^{(t)}}{1-\lambda_i^{(t)}D_i/B^{(t)}} \leq \tau_i. \end{aligned} \tag{A.11}$$

We can find that the optimization problem in Equation (3.7) is the same as the optimization problem in Equation (A.11) when the schedule probabilities in Equation (3.7) are $p_i = 1$, $p_{i'} = 0$ $(i' \neq i)$. According to Theorem 3.2, the minimal resource cost in Equation (3.7) is achieved under such schedule probability settings. Therefore, Theorem 3.4 is proved. □

A.6 Proof of Theorem 3.5

Proof. To prove Theorem 3.5, we first prove that the optimal solution for optimization problem in Equation (3.12) is also a feasible solution for optimization problem in Equation (3.17). Suppose that the optimal solution for Equation (3.12) is $\{S^{(t)}, C_1^{(t)}, \ldots, C_M^{(t)}, B^{(t)}\}$, which satisfies all constraints in Equation (3.12). With the same services, the corresponding resource allocation $\{S^{(t)}, C^{(t)} = \sum_{i=1}^{M} C_i^{(t)}, B^{(t)}\}$ is also a feasible solution for optimization problem in Equation (3.17), since it can meet all constraints in Equation (3.17). Therefore, with the resource allocation $\{S^{(t)}, C_1^{(t)}, \ldots, C_M^{(t)}, B^{(t)}\}$, the response time for processing class-1 service in multi-service scenario is given by $T_{mult}^{tot(1)(t)} = T_{mult}^{sche(1)(t)} + T_{mult}^{comp(1)(t)} + T_{mult}^{tran(1)(t)}$, while the corresponding response time in priority-service scenario is given by $T_{prio}^{tot(1)(t)} = T_{prio}^{sche(1)(t)} + T_{prio}^{comp(1)(t)} + T_{prio}^{tran(1)(t)}$.

Next, we will compare the response time of schedule, computation, and transmission in $T_{mult}^{tot(1)}$ and those in $T_{prio}^{tot(1)}$, respectively. In schedule phase, since $\lambda^{(t)} = \sum_{i=1}^{M} \lambda_i^{(t)} \geq \lambda_1^{(t)}$, $(T_{mult}^{sche(1)(t)} = \frac{1}{S^{(t)} - \lambda^{(t)}}) \geq (T_{prio}^{sche(1)(t)} = \frac{1}{S^{(t)} - \lambda_1^{(t)}})$; in computation phase, since $C_1^{(t)} \leq C^{(t)}$, $(T_{mult}^{comp(1)(t)} = \frac{F_1}{C_1^{(t)} - \lambda_1^{(t)} F_1}) \geq (T_{prio}^{comp(1)(t)} = \frac{F_1}{C^{(t)} - \lambda_1^{(t)} F_1})$; in transmission phase, according to the conclusion in [96] that the highest priority class in priority queue always has the smaller response time than its counterpart in $M/H_M/1$ queue, we can get that $T_{mult}^{tran(1)} \geq T_{prio}^{tran(1)}$. So, based on the above comparisons, we can get that $T_{mult}^{tot(1)} \geq T_{prio}^{tot(1)}$. Therefore, Theorem 3.5 is proved. □

References

[1] "Worldwide and Regional Public IT Cloud Services 2014 - 2018 Forecast," https://www.idc.com/getdoc.jsp?containerId=251730, [Online: accessed May 20, 2015].

[2] "Facebook data center," http://www.datacenterknowledge.com/the-facebook-data-center-faq/, [Online: accessed May 20, 2015].

[3] "Gmail," http://www.google.ca/about/datacenters/, [Online: accessed May 20, 2015].

[4] "Microsoft Office Online," https://office.com/start/default.aspx, [Online: accessed May 20, 2015].

[5] "OnLive," https://www.onlive.com/, [Online: accessed April 10, 2015].

[6] "US Federal Government Cloud," http://aws.amazon.com/federal/, [Online: accessed April 10, 2015].

[7] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[8] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, 2010.

[9] "Netflix case study," http://aws.amazon.com/solutions/case-studies/netflix/, [Online: accessed May 20, 2015].

[10] D. Minarolli and B. Freisleben, "Utility-based resource allocation for virtual machines in cloud computing," in *Proc. of IEEE ISCC*, 2011, pp. 410–417.

[11] P. Calyam, R. Patali, A. Berryman, A. M. Lai, and R. Ramnath, "Utility-directed resource allocation in virtual desktop clouds," *Computer networks*, vol. 55, no. 18, pp. 4112–4130, 2011.

[12] D. Minarolli and B. Freisleben, "Virtual machine resource allocation in cloud computing via multi-agent fuzzy control," in *Proc. of IEEE CGC*, 2013, pp. 188–194.

[13] A. Rai, R. Bhagwan, and S. Guha, "Generalized resource allocation for the cloud," in *Proc. of ACM Symposium on Cloud Computing*, 2012, p. 15.

[14] A. Nathani, S. Chaudhary, and G. Somani, "Policy based resource allocation in IaaS cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 94–103, 2012.

[15] "Amazon EC2," http://aws.amazon.com/ec2/, [Online: accessed May 20, 2015].

[16] "MSAzure," http://www.microsoft.com/windowsazure/, [Online: accessed May 20, 2015].

[17] "IBM Cloud Computing," http://www.ibm.com/cloud-computing/us/en/, [Online: accessed May 20, 2015].

[18] "AWS Elastic Beanstalk," https://aws.amazon.com/elasticbeanstalk/, [Online: accessed May 20, 2015].

[19] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," *Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.

[20] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on iaas cloud systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 5, pp. 666–677, 2012.

[21] A. Khiyaita, M. Zbakh, H. El Bakkali, and D. El Kettani, "Load balancing cloud computing: state of art," in *Proc. of IEEE Network Security and Systems*, 2012, pp. 106–109.

[22] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[23] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proc. of IEEE Power Aware Computing and Systems*, vol. 10, 2008.

[24] "GaiKai," http://www.gaikai.com/, [Online: accessed May 20, 2015].

[25] "CiiNOW," http://www.ciinow.com/, [Online: accessed May 20, 2015].

[26] M. Hemmati, A. Javadtalab, A. A. Nazari Shirehjini, S. Shirmohammadi, and T. Arici, "Game as video: bit rate reduction through adaptive object encoding," in *Proc. of ACM Network and Operating Systems Support for Digital Audio and Video*, 2013, pp. 7–12.

[27] H. Ahmadi, S. Khoshnood, M. R. Hashemi, and S. Shirmohammadi, "Efficient bitrate reduction using a game attention model in cloud gaming," in *Proc. of IEEE Haptic Audio Visual Environments and Games*, 2013, pp. 103–108.

[28] S. Garfinkel, *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. MIT Press, 1999.

[29] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.

[30] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[31] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Proc. of IEEE Information Technology: New Generations*, 2010, pp. 684–689.

[32] "GoGrid," http://www.gogrid.com/, [Online: accessed May 20, 2015].

[33] "Google App Engine," https://cloud.google.com/appengine/, [Online: accessed May 20, 2015].

[34] "Engine Yard," https://www.engineyard.com/, [Online: accessed May 20, 2015].

[35] "Google Docs," https://docs.google.com/, [Online: accessed May 20, 2015].

[36] "2014 celebrity photo leaks," http://en.wikipedia.org/wiki/2014_celebrity_photo_leaks, [Online: accessed May 20, 2015].

[37] "iCloud," https://www.icloud.com/, [Online: accessed May 20, 2015].

[38] "Amazon Data Loss," http://www.informationweek.com/cloud/infrastructure-as-a-service/9-worst-cloud-security-threats/d/d-id/1114085, [Online: accessed May 20, 2015].

[39] M. Menzel, R. Warschofsky, I. Thomas, C. Willems, and C. Meinel, "The service security lab: A model-driven platform to compose and explore service security in the cloud," in *Proc. of IEEE World Congress on Services*, 2010, pp. 115–122.

[40] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–5.

[41] D. Huang, X. Zhang, M. Kang, and J. Luo, "Mobicloud: Building secure cloud framework for mobile computing and communication," in *Proc. of IEEE International Symposium on Service Oriented System Engineering*, 2010, pp. 27–34.

[42] E. Bertino, F. Paci, R. Ferrini, and N. Shang, "Privacy-preserving digital identity management for cloud computing." *IEEE Data Engineering Bulletin*, vol. 32, no. 1, pp. 21–27, 2009.

[43] S. Guha, K. Tang, and P. Francis, "Noyb: Privacy in online social networks," in *Proc. of ACM Online Social Networks*, 2008, pp. 49–54.

[44] R. Brown *et al.*, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[45] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826–831.

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

[47] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[48] S. Ferretti, V. Ghini, F. Panzieri, and E. Turrini, "Seamless support of multimedia distributed applications through a cloud," in *Proc. of IEEE Cloud Computing*, 2010, pp. 548–549.

[49] W. Hui, C. Lin, and Y. Yang, "MediaCloud," *KSII Transactions on Internet and Information Systems*, vol. 6, no. 4, pp. 1153–1170, 2012.

[50] D. Miao, W. Zhu, C. Luo, and C. Chen, "Resource allocation for cloud-based free viewpoint video rendering for mobile phones," in *Proc. of ACM Multimedia*, 2011, pp. 1237–1240.

[51] S. Wang and S. Dey, "Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 870–883, 2013.

[52] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. Lau, "CloudMoV: Cloud-Based Mobile Social TV," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 821–832, 2013.

[53] N. Zhang, T. Mei, X.-S. Hua, L. Guan, and S. Li, "Interactive mobile visual search for social activities completion using query image contextual model," in *Proc. of IEEE Multimedia Signal Processing*, 2012, pp. 238–243.

[54] R. Nathuji and K. Schwan, "Virtualpower: coordinated power management in virtualized enterprise systems," in *ACM SIGOPS Operating Systems Review*, vol. 41, 2007, pp. 265–278.

[55] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," *Springer Journal of Middleware*, pp. 243–264, 2008.

[56] S. Chaisiri, B. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proc. of IEEE Asia-Pacific Services Computing Conference*, 2009, pp. 103–110.

[57] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 289–302, 2007.

[58] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-Tiered On-Demand resource scheduling for VM-Based data center," in *Proc. of IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 148–155.

[59] L. Staley, L. Cherkasova, and L. Cherkasova, "Building a performance model of streaming media applications in utility data center environment," in *Proc. of ACM/IEEE Conference on Cluster Computing and the Grid*, 2003, pp. 52–59.

[60] W. Lin and D. Qi, "Research on resource self-organizing model for cloud computing," in *Proc. of IEEE International Conference on Internet Technology and Applications*, 2010.

[61] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau, "Cloudmedia: When cloud on demand meets video on demand," in *Proc. of IEEE Distributed Computing Systems*, 2011, pp. 268–277.

[62] F. Wang, J. Liu, and M. Chen, "Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. of IEEE INFOCOM*, 2012, pp. 199–207.

[63] D. Niu, C. Feng, and B. Li, "Pricing cloud bandwidth reservations under demand uncertainty," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, 2012, pp. 151–162.

[64] H. Yu, D. Zheng, B. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, pp. 333–344, 2006.

[65] H. Wen, Z. Hai-ying, L. Chuang, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in *Proc. of IEEE International Conference on Electronic and Mechanical Engineering and Information Technology*, 2011, pp. 165–168.

[66] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. of IEEE INFOCOM*, 2013, pp. 190–194.

[67] R.-S. Chang, J.-S. Chang, and P.-S. Lin, "An ant algorithm for balanced job scheduling in grids," *Future Generation Computer Systems*, vol. 25, no. 1, pp. 20–27, 2009.

[68] J. Tai, J. Zhang, J. Li, W. Meleis, and N. Mi, "Ara: Adaptive resource allocation for cloud computing environments under bursty workloads," in *Proc. of IEEE Performance Computing and Communications Conference*, 2011, pp. 1–8.

[69] "PhotoSynth," https://photosynth.net/, [Online: accessed May 20, 2015].

[70] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, 2013.

[71] M. Silberstein, D. Geiger, A. Schuster, and M. Livny, "Scheduling mixed workloads in multi-grids: the grid execution hierarchy," in *Proc. of IEEE Symposium on High Performance Distributed Computing*, 2006, pp. 291–302.

[72] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "Gaminganywhere: an open cloud gaming system," in *Proc. of ACM Multimedia Systems Conference*, 2013, pp. 36–47.

[73] C.-Y. Huang, D.-Y. Chen, C.-H. Hsu, and K.-T. Chen, "Gaminganywhere: an open-source cloud gaming testbed," in *Proc. of ACM Multimedia*, 2013, pp. 827–830.

[74] W. Cai, M. Chen, and V. Leung, "Toward gaming as a service," *IEEE Internet Computing*, vol. 18, no. 3, pp. 12–18, 2014.

[75] M. Semsarzadeh, M. Hemmati, A. Javadtalab, A. Yassine, and S. Shirmohammadi, "A video encoding speed-up architecture for cloud gaming," in *Proc. of IEEE ICME Workshop*, 2014, pp. 1–6.

[76] W. Cai and V. C. Leung, "Multiplayer cloud gaming system with cooperative video sharing." in *Proc. of IEEE CloudCom*, 2012, pp. 640–645.

[77] "Kalydo," http://kalydo.com/technology/, [Online: accessed May 20, 2015].

[78] A. Jurgelionis, P. Fechteler, P. Eisert, F. Bellotti, H. David, J.-P. Laulajainen, R. Carmichael, V. Poulopoulos, A. Laikari, P. Perälä *et al.*, "Platform for distributed 3d gaming," *International Journal of Computer Games Technology*, vol. 2009, p. 1, 2009.

[79] P. Eisert and P. Fechteler, "Low delay streaming of computer graphics," in *Proc. of IEEE ICIP*, 2008, pp. 2704–2707.

[80] S. Wang and S. Dey, "Rendering adaptation to address communication and computation constraints in cloud mobile gaming," in *Proc. of IEEE Global Telecommunications Conference*, 2010, pp. 1–6.

[81] S. Shirmohammadi, "Adaptive streaming in mobile cloud gaming," *IEEE COMSOC MMTC E-Letter*, 2013.

[82] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, 2014.

[83] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. of ACM Multimedia*, 2011, pp. 1269–1272.

[84] "StreamMyGame," http://streammygame.com/smg/index.php, [Online: accessed May 20, 2015].

[85] M. T. Najaran and C. Krasic, "Scaling online games with adaptive interest management in the cloud," in *Proc. of IEEE Workshop on Network and Systems Support for Games*, 2010.

[86] B. Hariri and S. Shirmohammadi, "A statistical network traffic model for first-person shooter games," *Journal of Advances in Computer Networks*, vol. 2, no. 2, 2014.

[87] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, pp. 338–354, 1953.

[88] A. ParandehGheibi, M. Médard, A. Ozdaglar, and A. Eryilmaz, "Information theory vs. queueing theory for resource allocation in multiple access channels," in *Proc. of IEEE Symposium on Personal, Indoor and Mobile Radio Communication*, 2008, pp. 1–5.

[89] D. Wu, Z. Xue, and J. He, "icloudaccess: Costeffective streaming of video games from the cloud with low latency," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 2014.

[90] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[91] P. Suresh Varma, A. Satyanarayana, and R. Sundari, "Performance analysis of cloud computing using queuing models," in *Proc. of IEEE Cloud Computing Technologies, Applications and Management*, 2012, pp. 12–15.

[92] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Springer Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[93] M. Murphy, B. Kagey, M. Fenn, and S. Goasguen, "Dynamic provisioning of virtual organization clusters," in *Proc. of IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 364–371.

[94] M. Arlitt and C. Williamson, "Internet web servers: Workload characterization and performance implications," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631–645, 1997.

[95] S. M. Ross, *Stochastic processes*. John Wiley & Sons New York, 1996, vol. 2.

[96] D. Gross, *Fundamentals of queueing theory*. Wiley-India, 2008.

[97] B. Yang, F. Tan, Y. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," *Cloud Computing*, pp. 571–576, 2009.

[98] D. Ardagna, S. Casolari, and B. Panicucci, "Flexible distributed capacity allocation and load redirect algorithms for cloud systems," in *Proc. of IEEE International Conference on Cloud Computing*, 2011, pp. 163–170.

[99] J. Zheng and E. Regentova, "Qos-based dynamic channel allocation for gsm/gprs networks," *Network and Parallel Computing*, pp. 285–294, 2005.

[100] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[101] N. K. Jaiswal, *Priority queues*. Elsevier, 1968.

[102] L. Schrage and L. Miller, "The queue M/G/1 with the shortest remaining processing time discipline," *Operations Research*, pp. 670–684, 1966.

[103] A. Wolke and G. Meixner, "Twospot: A cloud platform for scaling out web applications dynamically," *Springer Journal of Towards a Service-Based Internet*, pp. 13–24, 2010.

[104] X. Nan, F. Guo, Y. He, and L. Guan, "vPresent: A cloud based 3D virtual presentation environment for interactive product customization," in *SPIE Optical Engineering + Applications*. International Society for Optics and Photonics, 2013.

[105] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley, 1998.

[106] G. Nemhauser and D. Bienstock, *Integer Programming and Combinatorial Optimization*. Springer, 2004, vol. 10.

[107] "Amazon EC2 Pricing," http://aws.amazon.com/ec2/#pricing, [Online: accessed May 20, 2015].

[108] C. Wu, B. Li, and S. Zhao, "Multi-channel live p2p streaming: Refocusing on servers," in *Proc. of IEEE INFOCOM.*, 2008.

[109] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems," in *Proc. of IEEE INFOCOM.*, 2011, pp. 421–425.

[110] J. L. Doob, *Stochastic processes*. New York Wiley, 1953, vol. 101.

[111] G. Box, G. Jenkins, and G. Reinsel, *Time series analysis: forecasting and control*. Wiley, 2011, vol. 734.

[112] X. Nan, Y. He, and L. Guan, "Optimization of workload scheduling for multimedia cloud computing," in *Proc. of IEEE ISCAS*, 2013, pp. 2872–2875.

[113] D. Xuan, W. Jia, W. Zhao, and H. Zhu, "A routing protocol for anycast messages," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 6, pp. 571–588, 2000.

[114] D. Cross and C. Harris, "Fundamentals of queuing theory," *John Wily and Sons*, 1998.

[115] J. Tang, W. P. Tay, and Y. Wen, "Dynamic request redirection and elastic service scaling in cloud-centric media networks," *IEEE Transactions on Multimedia*, 2014.

[116] J. Karlof, *Integer programming: theory and practice*. CRC Press, 2006.

[117] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press, 2001.

[118] M. Hendy and D. Penny, "Branch and bound algorithms to determine minimal evolutionary trees," *Elsevier Journal of Mathematical Biosciences*, vol. 59, no. 2, pp. 277–290, 1982.

[119] L. Kleinrock, *Queueing systems*. Wiley, 1975.

[120] M. Atkinson, J. Sack, N. Santoro, and T. Strothotte, "Min-max heaps and generalized priority queues," *ACM Journal of Communications*, vol. 29, no. 10, pp. 996–1000, 1986.

[121] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 2008, vol. 2.

[122] J. Lee and S. Leyffer, *Mixed integer nonlinear programming*. Springer, 2011, vol. 154.

[123] M. Bussieck, A. Pruessner *et al.*, "Mixed-integer nonlinear programming," *SIAG/OPT Newsletter: Views and News*, vol. 14, no. 1, pp. 19–22, 2003.

[124] Z. Zhao, Y. Zhao, Z. Gao, X. Nan, M. Mei, H. Zhang, H. Chen, X. Peng, Y. Chen, J. Guo *et al.*, "BUPT-MCPRL at TRECVID 2009," *TREC Video Retrieval Evaluation Online Proceedings, Gaithersburg, MD, USA*, 2009.

[125] J. Kelley, "The critical-path method: Resources planning and scheduling," *Journal of Industrial scheduling*, pp. 347–365, 1963.

[126] G. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. ACM spring joint computer conference*, 1967, pp. 483–485.

[127] P. Over, G. Awad, J. Fiscus, M. Michel, A. Smeaton, and W. Kraaij, *TRECVID 2009-goals, tasks, data, evaluation mechanisms and metrics*. National Institute for Standards and Technology (NIST), 2010.

[128] B. Blau, J. Erensen, T. H. Nguyen, and S. Verma, "Forecast: Video game ecosystem," Gartner Report, 2013.

[129] "Microsoft reportedly developing cloud gaming service," http://ca.ign.com/articles/2013/09/27/microsoft-reportedly-developing-cloud-gaming-service.

[130] H. Hoppe, "Progressive meshes," in *Proc. of ACM Computer Graphics and Interactive Techniques*, 1996, pp. 99–108.

[131] "World of warcraft," http://us.battle.net/wow/en/, [Online: accessed May 20, 2015].

[132] J. Nystad, A. Lassen, A. Pomianowski, S. Ellis, and T. Olson, "Adaptive scalable texture compression," in *Proc. of ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*. Eurographics Association, 2012, pp. 105–114.

[133] Y. Su and M.-T. Sun, "Fast multiple reference frame motion estimation for h. 264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 447–452, 2006.

[134] S. Ma, W. Gao, and Y. Lu, "Rate-distortion analysis for h. 264/avc video coding and its application to rate control," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 12, pp. 1533–1544, 2005.

[135] H. Wang and S. Kwong, "Rate-distortion optimization of rate control for h. 264 with adaptive initial quantization parameter determination," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 1, pp. 140–144, 2008.

[136] C. Li, D. Wu, and H. Xiong, "Delay - power-rate-distortion model for wireless video communication under delay and energy constraints," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 24, no. 7, pp. 1170–1183, 2014.

[137] Q. Chen, *Image and video processing for denoising, coding and content protection*. University of Florida, 2011.

[138] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[139] "Angrybots," http://u3d.as/content/unity-technologies/angry-bots/5CF, [Online: accessed May 20, 2015].

[140] "x264," http://www.videolan.org/developers/x264.html, [Online: accessed May 20, 2015].

[141] "ffplay," https://www.ffmpeg.org/ffplay.html, [Online: accessed May 20, 2015].

[142] "dummynet project," http://info.iet.unipi.it/~luigi/dummynet/, [Online: accessed May 20, 2015].

[143] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the Future*, vol. 2007, pp. 1–16, 2012.