Table of Contents

[Introduction 3](#_Toc405876842)

[Literature Survey 7](#_Toc405876843)

[System Analysis 9](#_Toc405876844)

[3.1. Latency Measurement of a Large Scale Cloud Gaming System 9](#_Toc405876845)

[*A. Cloud Union’s Platform 9*](#_Toc405876846)

[*B. Measurement of Queuing Delay 11*](#_Toc405876847)

[*C. Measurement of Response Delay 13*](#_Toc405876848)

[*D. Implications 15*](#_Toc405876849)

[Problem Definition 16](#_Toc405876850)

[A. System Architecture 16](#_Toc405876851)

[B. Problem Formulation 17](#_Toc405876852)

[C. Design of Online Control Algorithm 19](#_Toc405876853)

[Experimental Evaluation 25](#_Toc405876854)

[A. Experiment Settings 25](#_Toc405876855)

[B. Comparison of Gaming Latency 27](#_Toc405876856)

[C. Comparison of Server Provisioning Cost 28](#_Toc405876857)

[D. Stability of Virtual Queues 30](#_Toc405876858)

[CONCLUSION 32](#_Toc405876859)

[Bibliography 33](#_Toc405876860)

Abstract

As a new paradigm, cloud gaming allows users to play high-end video games instantly without downloading or installing the original game software. In this paper, authors first conduct a series of well-designed active and passive measurements on a large-scale cloud gaming platform and identify the significant diversity in the queuing delay and response delay among users. authors note that the latency problem largely results from users specified request routing and inelastic server provisioning.

To address latency problem of the cloud gaming platform, they further propose an online control algorithm called iCloudAccess to perform intelligent request dispatching and server provisioning. Our main objective is to cut down the provisioning cost of cloud gaming service providers while still ensuring the user quality-of-experience requirements.

Authors formulate the problem as a constrained stochastic optimization problem and apply the Lyapunov optimization theory to derive the online control algorithm with provable upper bounds. They also conduct extensive trace-driven simulations to evaluate the effectiveness of our algorithm, and our results show that our proposed algorithm achieves significant gain

**Chapter 1**

# Introduction

Since the first video game was launched on the market around 45 years ago, we have witnessed a series of significant revolutions in the video game industry. In recent years, the emergence of cloud gaming has provided a promising approach to make gaming (especially high-end 3-D video games) more affordable and accessible to game players. The basic idea of cloud gaming is to render video games in the cloud- and stream-encoded game scenes to players via the broadband networks. Users can interact with the game application by sending the control signals (e.g., key strokes and mouse clicks) to the cloud server. Users are relieved of downloading or installing the original game software. With such cloud assisted gaming mode, users can easily play high-end 3-D video games on any device, such as PC, set-top box (STB), iPad, and smart phone, whenever and wherever possible

The potential of cloud gaming has already attracted a great amount of attention from many industrial practitioners, ventures, and researchers. It is predicted that the size of global video game market revenue will grow to U.S. $78 billion in 2017, among which cloud gaming market is expected to expand the most [1]. However, it is very challenging to build a cloud gaming platform that can provide users with high quality of experience (QoE).

Existing cloud gaming systems generally rely on a set of geographically distributed data centers to serve users in different regions. A user request will be directed to a data center according to certain policies (e.g., proximity). Due to the specialized hardware requirements [e.g., Graphics Processing Unit (GPU)], normally a dedicated cloud server, which can be either a physical machine or a virtual machine, will be allocated to a player exclusively upon receiving the request. The cloud server is responsible for game rendering and streaming encoded game scenes to the client.

When the cloud platform cannot provision enough servers to meet user demand timely, user requests have to be queued for a period. As online game players are pretty impatient [2], if queuing delay is too long, it will result in the loss of user accesses. In addition, short response delay is also highly desired to ensure the interactivity of cloud gaming applications. Here, response delay refers to the time difference between receiving a User input at the client side and the game scenes updated on the user’s screen [3]. Especially, the increase of response delay is intolerable for real-time video games [e.g., first-person shooter (FPS) games]. To better understand the problems and challenges therein, as the first step, we need to take a close look at the real cloud gaming systems. Starting from the first-hand observations, we can identify the underlying causes and address the problems in a right way for cloud gaming systems.

In this paper, we focus on understanding and mitigating the latency problem of cloud gaming services from the perspective of cloud gaming service providers (CGSPs). To this purpose, we first conduct a measurement study on a large-scale cloud gaming platform in China. Our measurement results show that both queuing delay and response delay exhibit significant temporal and spatial diversity, and such diversity are largely caused by user-specified request dispatching and inelastic cloud resource provisioning. Especially, we are the first to study the queuing phenomenon in a real-world cloud gaming system. To improve the QoE of game players in different regions, we further propose an online control algorithm called iCloudAccess, which reduces queuing delay and response delay by smart request dispatching and server provisioning among data centers. Our proposed iCloudAccess can minimize the time-average server provisioning cost so as to be cost- effective for CGSPs. Finally, we validate the effectiveness of our proposed solution via extensive simulations. In summary, our main contributions in this paper can be listed as follows.

1. We conducted an in-depth measurement study of CloudUnion, which is the first cloud gaming system in China and also proprietary. We collected a large amount of Wire shark traces and analyzed the communications between the client and servers in the cloud. With both passive and active measurements, we are able to unveil the architecture and internal mechanisms of CloudUnion
2. We developed a customized crawler to query the status of user requests and obtained the queuing information of each data center. We observed that players have to wait in the queue for a rather long period due to improper request routing to data centers in hot regions. We also measured the response delay when accessing different data centers at different time points. The response delay shows similar temporal and spatial variation. Such diversity is mostly due to user-specified request dispatching and inelastic server provisioning of the cloud platform.
3. We proposed an online control algorithm called iCloudAccess to reduce the gaming latency by intelligent request dispatching and server provisioning. Our objective is to minimize the time-average provisioning cost of CGSPs while still ensuring user QoE. We formulated the problem into a stochastic constrained optimization problem and applied the Lyapunov optimization theory to design the online algorithm. Our theoretical analysis shows that our algorithm can approach the optimality with an explicitly provable performance upper bound.
4. We performed extensive trace-driven simulations to verify the effectiveness of our proposed iCloudAccess algorithm in the practical settings. Our simulation results show that, compared with other alternatives, iCloudAccess can save more than 30% of provisioning cost and reduce the queuing delay and response delay significantly

2.2. Cloud gaming: concept and technologies

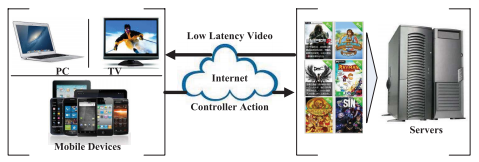


Fig 1: Concept of cloud gaming.

Cloud gaming is also called gaming on demand or gaming as a service. Essentially, cloud gaming has much in common with a video-on-demand service, but is more interactive. The player’s computer receives streaming video (and audio) and sends the keyboard, mouse, and controller input actions to the cloud gaming server over the broadband networks. Fig. 1 briefly shows the concept of cloud gaming.

Unlike traditional PC games, cloud gaming offers many novel features: firstly, with cloud gaming, players are relieved of expensive hardware investment and constant upgrades. A thin client (e.g., STB, laptop, and mobile device) with a broadband Internet connection is enough to play any video games; secondly, cloud gaming allows games to be platform independent and players do not need to worry about the compatibility issues when playing games. It is possible to play games on any operating system (e.g., Mac, Linux, and Android) or device (e.g., PC, mobile phone, and smart TV); thirdly, cloud gaming allows users to start playing games instantly, without the need to download and install the game images; and finally, cloud gaming makes copyright protection much easier, as games can only run on remote servers. For game publishers, cloud gaming is an attractive form for digital rights management.

However, cloud gaming services pose stringent requirements on bandwidth and latency. To stream high-quality video game comparable with that rendered by a high-end gaming PC, a player needs a broadband Internet connection. The cloud gaming platform should also be able to provide sufficiently short latency for real-time gaming and react to user inputs timely.

General-purpose thin clients, such as virtual network computing [4], cannot satisfy the stringent requirements of cloud gaming on response time and frame rate [5]. Existing cloud gaming systems employ highly optimized H.264/AVC codecs (e.g., ×264) [6] to perform real-time video encoding/decoding on captured game frames. To reduce the processing latency, multithreading is widely used to better leverage the computation power of multicore CPUs and GPUs. The network communications between the client and the gaming server can be based on different real-time communication protocols, such as RTP [7] and RTSP [8].

Chapter 2

# Literature Survey

Cloud gaming utilizes advanced streaming and cloud computing technology to enable users to play on any device, anytime, anywhere. However, it is very challenging to build a highly scalable, low-latency cloud gaming platform.

In the aspect of system design, there have been quite a few commercial cloud gaming systems being deployed, such as OnLive [9], GaiKai [10], CloudUnion [11], Ubitus [22], and BigFish Games [23]. GamingAnywhere [3] is the first open- source cloud gaming system, which is extensible, portable, and fully configurable. Choy et al. [24] noted that fully centralized infrastructure is unable to meet the requirements of latency-sensitive applications, such as cloud gaming. They further proposed a new hybrid cloud gaming platform called EdgeCloud [25] to augment cloud data centers with resources from end hosts and reduce the user latency. Despite that server provisioning has been studied by many researchers in the field of cloud computing [26], [27], however, different from other cloud services (e.g., cloud storage and cloud video), cloud gaming is inherently latency sensitive. For cloud-based video- on-demand (VoD) applications, researchers have designed different kinds of optimal resource provisioning strategies [28], [29] to scale up and down resources provisioned in multiple geo-distributed data centers according to the demand dynamics. Although cloud gaming bears similarity with VoD in some extent, its stringent requirements on bandwidth and latency make resource provisioning much more challenging than that for VoD. Marzolla et al. [30] considered the dynamic resource provisioning problem in massively multiplayer online games (MMOGs). But MMOG is also different from cloud gaming. In MMOG services, a single server can serve many users simultaneously.

To deepen the understanding of cloud gaming systems, researchers have also conducted a few measurement studies. Chen et al. [12] measured the response latency of two cloud gaming platforms, namely, OnLive and StreamMyGame. In their measurements, they invented a novel delay measurement method by using the hooking mechanism in Windows to inject the instrumentation code. Claypool et al. [31] conducted a detailed

Measurement study on traffic characteristics of OnLive and showed that OnLive rapidly sends large packets downstream, but less frequently sends much smaller packets upstream, which is significantly different from traditional game clients. Manzano et al. [32] measured and analyzed the distribution of packet size and packets inter arrival time of two cloud gaming systems, OnLive and GaiKai. Jarschel et al. [33] focused on the evaluation of user QoE in cloud gaming systems and studied the variation of QoE scores (e.g., mean opinion score) under various network conditions.

Our work differs from the previous work in the following aspects: first, we identified that queuing delay and response delay exhibit both temporal and spatial diversity, and noted that such diversity is mostly due to inefficient request dis- patching and cloud resource provisioning; second, instead of considering only single data center, we focused on the scenario with multiple regions and multiple data centers and considered the problems of request dispatching and server provisioning jointly; and finally, we took both provisioning cost and user QoE into account during problem formulation and designed an online control algorithm with explicitly provable performance bounds. Our algorithm can minimize the provisioning cost of operators without sacrificing user QoE.

**Chapter 3**

# System Analysis

To better understand the latency components of cloud gaming services, we first conduct extensive measurements on the cloud gaming platform of CloudUnion [11], which was the first to launch cloud gaming services in China and its subscribers have exceeded 3 00 000 as of July 2012.

## 3.1. **Latency Measurement of a Large Scale Cloud Gaming System**

### Cloud Union’s Platform

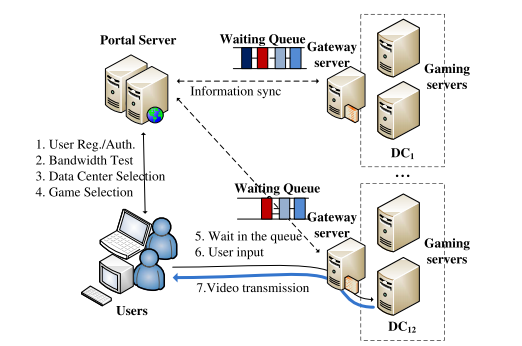


Fig 2 Architecture of Cloud Union’s cloud gaming platform

CloudUnion now offers more than 200 games via its cloud platform. Users can start playing games by either downloading the Cloud Union’s client or directly using the web browser (e.g., Internet Explorer, Firefox, and Chrome). Cloud Union’s platform supports game streaming in a variety of resolutions, ranging from 320 × 240 to 1024 × 768. The minimum bandwidth requirement for the client is 2 Mb/s, but 6+Mb/s bandwidth is recommended for high-quality game streaming. The measurement of CloudUnion is challenging because the Cloud Union’s protocol is proprietary. To understand the underlying protocol, we had to collect a large amount of Wireshark traces from multiple gaming sessions and analyze the communications between the

Client and servers in the cloud platform. From the protocol analysis, we find that the Cloud Union’s infrastructure can be as shown in Fig. 2.

To improve the QoE of users in different regions, CloudUnion deploys its data centers in 12 geographically distributed locations. A portal server is responsible for user registration, authentication, and bandwidth test. After a user logs into the system, it should first manually choose a data center from a list and then select a preferred game. A user normally chooses a data center in a nearby region. The request will be routed to a gateway server of the selected data center. Upon receiving a user request, the cloud gaming platform will launch a dedicated server1 to run the game specified in the request and stream the gaming video to the user client. When the capacity of a data center cannot meet the demand, user requests routed to that data center will be held in a waiting queue.

In the whole gaming session, there are two major latency components for cloud game players.

* 1. **Queuing delay**: the difference between the time a user request enters the waiting queue of a data center and the time the data center starts to serve the request. The queuing phenomenon occurs when a data center does not have available resources (e.g., CPU, memory, and storage) to launch new gaming servers. In such a case, users have to wait for a while before being served. Normally, queuing delay ranges from a few seconds to several hours. If queuing delay is too long, players will have to choose another data center or abort playing the game.
  2. **Response delay**: the difference between the time the client sends a player’s command to the server and the time the generated video frame is decoded and presented on the screen. According to [12], response delay can be further divided into network delay (at the network side), processing delay (at the server side), and playout delay (at the client side). Processing delay and playout delay are determined by video/audio codecs and the hardware configuration (e.g., CPU and memory) of the cloud server/client. Network delay is determined by the geographical distance between the user and the data center and the network bandwidth condition. Response delay has significant impacts on the interactivity of cloud gaming.

### Measurement of Queuing Delay

Our trace analysis shows that the portal server keeps track of the status of the waiting queue of each data centre. By querying the portal server with the Cloud Union’s protocol, we can obtain queuing information of each data centre, including the number of user requests in the waiting queue, the position of a user request in the queue, and the estimated waiting time of a user request. To automate the querying job, we developed a customized crawler to query the portal server every 30 s. The crawler was continuously running for 40 days (from Mar. 24, 2013 to May 2, 2013) and logged all the queuing information of data centres. In the following, we will describe our main findings

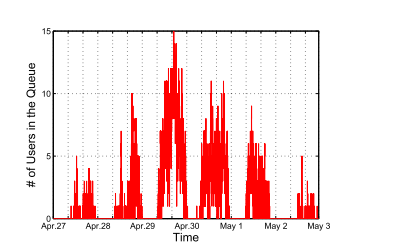


Fig 3 Number of users in the waiting queue of a data center during the period from Apr. 27, 2013 to May 2, 2013.

We first examine the queuing status of a data center in a hot region. Fig. 3 plots the number of users in the waiting queue of that data center over one week (from Apr 27, 2013 to May 2, 2013). We can observe that the queuing phenomenon occurred every day and became more serious during the period of Apr 29–May 1, 2013, which are the holidays in China. The results show that the current server provisioning strategy used by CloudUnion is not elastic as expected, and cannot provision enough number of gaming servers in a timely manner.

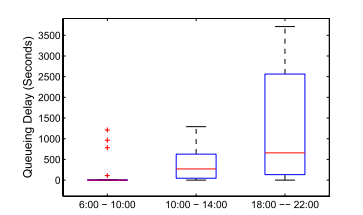


Fig 4 Queuing delay experienced by a user at different time slots of a day.

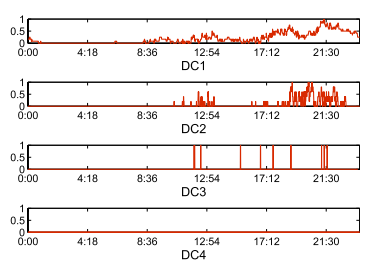


Fig 5 Frequency of queuing phenomena in different data centers.

To study queuing delay in detail, we write a shell script to simulate the login behavior of a user and monitor its waiting time in the queue. By performing multiple logins from the same location to a data center at different time slots of a day, we can calculate the median and variance of the queuing delay. Fig. 4 shows the distribution of queuing delay at different time slots of a day. It is observed that there is almost no queuing delay for most requests during the morning time (6:00–10:00), while queuing delay becomes very serious in the night time (18:00–22:00), with over 50% requests being queued for more than 500 s

Fig. 5 shows the frequency of queuing phenomena of four data centers in different regions over a period of 24 h (on Apr 26, 2013). To ease our comparison, we normalize the number of users in the waiting queue by the maxi- mum number of users observed in the waiting queue in the 24-h period. From the figure, we can clearly observe that there exists a significant diversity in terms of queuing frequency among different data centers. For DC1, which is located in a hot region, a new user request has to wait in a queue instead of being served immediately for 2/3 of a day. On the contrary, there is no queuing phenomenon for DC4. For data centers with queuing phenomena (e.g., DC1, DC2, and DC3), it is found that queuing occurs more frequently during the period of 8 AM–12 AM and 6 PM–12 PM and the peak occurs at around 9 PM, which is normally the leisure time.

Our measurement results note that queuing delay exhibits both temporal and spatial diversity. Such diversity in queuing delay is largely caused by the sub optimality of user-specified data center selection and the inefficiency of cloud resource provisioning. Actually, such kind of queuing problem widely exists in any online service system when the pace of resource provisioning cannot keep up with the increase of request arrivals. For example, when flash crowds occur, it is easy to observe the queuing phenomenon for online service systems. Even with cloud computing, it still takes time (e.g., ranging from a few seconds to tens of minutes) to allocate and provision a new Virtual Machine (VM) instance (see [13] – [15]), depending on the VM size. The queuing problem can be alleviated by faster VM provisioning, but cannot disappear because of the unpredictability of user demand. Therefore, we believe that cloud gaming systems outside of China (e.g., OnLive and GaiKai) will also have the same queuing problem.

### Measurement of Response Delay

Queuing delay determines how long a user should wait before running a game, while response delay determines how interactive a cloud game is during a game session. The direct measurement of response delay is very difficult due to the proprietary nature of the CloudUnion system. Instead, we adopted a method similar to [12] to measure the response delay. The basic idea is to calculate the time difference between the time a hot key is pressed and the time the updated screen is shown at the user side. The pressing of a hot key can be simulated by utilizing the hooking mechanism in Windows, and the screen update can be detected by examining the color changes of a specific set of pixels.

From the same location (i.e., Guangzhou), we initiated multiple game sessions simultaneously and measured the corresponding response delay to different data centers. Fig. 6 shows the difference of response delay when selecting different data centers. The response delay exhibits significant spatial diversity, with DC1 being the lowest and DC3 being the highest

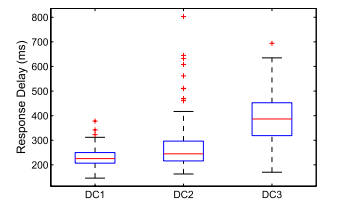


Fig 6 Response delay when selecting different data centers

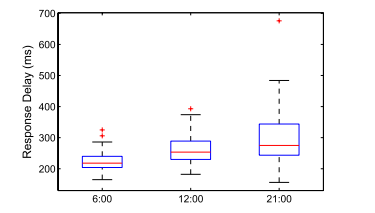


Fig 7 Temporal diversity of response delay over a day.

For the same data center, we also initiated a series of game sessions at different time slots of a day and measured the dynamics of response delay. Fig. 7 clearly indicates that response delay also exhibits temporal diversity. The average response delay is the lowest in the morning and increases to the peak in the evening. The above diversity in response delay is determined by multiple factors. Among three components of response delay, processing delay and playout delay are pretty stable under a certain hardware/software configuration. The more volatile delay component is network delay, which is determined by the selection of data centers.

### Implications

In the current cloud gaming system, for users with only partial information, it is difficult if not impossible to make the best choice among data centers. A random selection of a data center may result in either long queuing delay or intolerable response delay. However, request dispatching cannot completely solve the latency problem by itself, especially when the total capacity of provisioned cloud servers cannot meet the surge of user demand. Therefore, request dispatching should work together with server provisioning. From the perspective of CGSPs, they need to consider the problems of request dispatching and server provisioning jointly to reduce the access latency for game players. Meanwhile, the CGSPs should take the provisioning cost into account and optimize user QoE as they can under their budget constraint to be commercially successful. Note that, in addition to what has been reported in this paper, we have also measured and analysed many other aspects of the CloudUnion system, including user behaviour, traffic pattern, and user-perceived quality. The complete measurement results are reported in our technical report [16]. In terms of this paper, our main focus is to address the request dispatching and resource provisioning problems in the existing cloud gaming system. For this purpose, we only use a small fraction of our measurement results on queuing delay and response delay as a motivating example to show the problem.

**Chapter 4**

# Problem Definition

In this section, we design an online control algorithm called iCloudAccess to speed up the accesses of cloud game players. ICloudAccess provides a cost-effective approach to stream video games with low latency by smart request dispatching among data centers and dynamic cloud resource provisioning.

**4.1:** ICloudAccess: Reducing Latency via Online Request Dispatching and Server Provisioning

### System Architecture

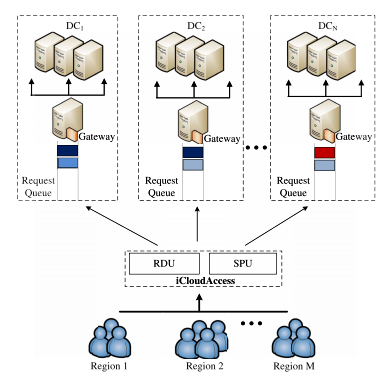
****

Fig 8: System architecture.

Fig. 8 describes the role of *iCloudAccess* in the cloud gaming platform. *ICloudAccess* contains two major components as follows

1. Request dispatching unit (RDU) that is responsible for dispatching requests intelligently.
2. Server provisioning unit (SPU) that is responsible for adjusting the number of game servers provisioned at each data center according to user demand.

The operations of RDU and SPU are performed at different timescales. For every incoming user request, RDU needs to dispatch the request timely and the dispatching operation needs to be completed within a few seconds; however, the provisioning of cloud servers cannot be conducted in a real-time manner. Normally, an SPU adjusts the provisioning of cloud servers for each data center periodically, in the order of hours.

By intelligently dispatching a request to a data center with a short waiting queue, queuing delay of a user request can be significantly reduced. However, the selection of a data center cannot violate the constraint on response delay. When the provisioned servers are not enough to support the demand, the SPU will start to provision more cloud servers in the corresponding data center. Next, we will study how to design online algorithm to optimize request dispatching and server provisioning.

### Problem Formulation

In our system model, we assume that the cloud gaming platform contains N geographically distributed data centers to provide gaming services to users spreading over M regions. A user request will be routed to one data center for service (as shown in Fig. 8).

The time is divided into a series of slots. Let λi (t) be the total number of requests generated by users in the ith region at a time slot t and λij (t ) be the number of user requests that are generated from the ith region and routed to the jth data center for service at a time slot t . Then we have



****

For game players, their QoE is sensitive to both queuing delay and response delay. Queuing delay is determined by the load status of the assigned data center, while response delay is largely determined by network delay between the client and the remote gaming server, as processing delay and playout delay are pretty stable. Therefore, we define the QoE function of a user k as a function of queuing delay Dq(k) and network delay Dr*(k)* , namely

G(k) = ɸ(Dq(k) , Dr(k))………(3)

Where ɸ (•) is a generic function that can be defined according to user sensitivity to different types of delay. For example, ɸ (•) can be simply defined as a network delay when the player does not care about the waiting time in the queue. Assume that G (k) is an increasing and convex function, and upper-bounded by Gmax , i.e., G(k) ≤ Gmax , ∀k.

To meet the dynamic user demand, the cloud gaming service provider has to scale up or down the provisioned servers at each data center. Assume that the server provisioning actions occur every m time slots, which are in an hourly timescale. For the jth data center, suppose nj (t ) be the total number of provisioned servers at the time slot *t* and cj(t ) be the cost to provision a server at the time slot *t* . The total server provisioning cost of the jth data center at the time slot *t* is given by c­j­(t )•nj(t). Considering all the data centers, the long-term time-average server provisioning cost of a cloud gaming platform can be defined as



From the perspective of CGSPs, the main objective is to provide users with a good QoE while reducing its provisioning cost at the same time. To this purpose, we formulate the problem into the constrained stochastic optimization problem.



s.t 





In the above problem, the objective is the long-term time average server provisioning cost that should be minimized as much as possible. Constraint (8) is used to guarantee that the time-average QoE value of newly arrived users should

Be less than a threshold €. Intuitively, when response delay is below a certain threshold, which is enough to ensure good user experience, a further reduction of response delay brings marginal benefit to users. In this case, the service operator

Cares more about the minimization of server provisioning cost, which directly impacts their revenue in running the service. Uj (t) in constraint (8) denotes the set of newly arrived user requests that enter the waiting queue of the jth data center.

Solving the above optimization problem, we can obtain the optimal request dispatching strategy λ (t) = (λij,∀i, j) and the optimal server provisioning strategy n(t) = (nj(t),∀j).

Different games have different latency and QoE requirements. To incorporate the difference between game genres into the optimization problem, we can redefine constraint (8) in the above problem for each game genre separately. For

different game genres, the threshold in the constraint can be set as different values. By transforming the constraints via introducing a set of virtual queues, we can obtain the request dispatching decision for different game genres in one operation round of our online algorithm.

### Design of Online Control Algorithm

To solve the constrained stochastic optimization in problem P1, we exploit the Lyapunov optimization theory to design online control strategies. A major benefit of Lyapunov optimization is that it does not require future information about user demand. Taking actions to greedily minimize the drift plus penalty in each time slot, it can provide performance with explicit bounds.

In the framework of Lyapunov optimization, the original stochastic optimization problem can be transformed to an optimization problem of minimizing the Lyapunov drift plus penalty. Using Lyapunov optimization, the time-average constraints in problemP1can be transformed into a set of queue stability constraints [17].

A set of virtual queues H={H1,H2,...,HN} are introduced. Hj denotes the virtual QoE queue of the jth data center and Hj(t) denotes the queue backlogs in the jth data center at the time slot *t*. The evolution of the queue Hj can be described as



It can be proven that when the virtual queue Hj is stable, then the QoE constraint (8) can also be satisfied (see Lemma 4.1).

***Lemma 4.1***: If the virtual queue Hj is stable, then the QoE constraint (8) can be satisfied. That is



*Proof*: See Appendix A in our technical report [21] for the proof details.

Let Qj be the workload queue of the jth data center and Qj(t)be the amount of workload queued in the jth data center at the time slot *0*. Suppose the average duration of a gaming session be μ. Then, the update of the workload queue Qj can

be described by



The provisioning operation performs every m time slot; thus, we have nj(t)=nj(t +τ),1≤τ≤m−1. Define the Lyapunov function as  and them-step Lyapunov drift an .According to the Lyapunov optimization framework, we can then obtain the drift plus penalty by adding the provisioning cost into the drift, namely



where Q(t) ={Q1(t),Q2(t),...,QN(t)}, H(t) ={H1(t),H2(t),...,HN(t)},and Vis a tenable parameter. The original problemP1can then be transformed to the following optimization problem P2



s.t. 



To solve problemP2, a key step is to derive the upper bound of the drift plus penalty, which we will derive in *Lemma 4.2*.

***Lemma 4.2***: For any feasible  and the drift plus penalty is upper bounded by, namely



Where



Proof: Please see Appendix B in our technical report [21] for the proof details.

To minimize the upper bound1and find the optimal server provisioning decision ,normally we need to have the future information of queue backlogs (i.e., Q and H). In our algorithm design, the requirement is relaxed by approximating

the future queue state (Q(t+τ),H(t+τ )), τ=1,...,m−1, with(Q(t),H(t)) for t =m,2m,.... Such approximation leads to a relaxed upper bound of the drift plus penalty as proven in Lemma 4.3.

***Lemma 4.3***: For any feasible  and , the drift plus penalty is upper bounded by, namely



Where is the relaxed upper bound and satisfies

Proof: Please see Appendix C in our technical report [21] for the proof details.

The optimization problem can be then solved by finding a strategy to minimize the relaxed upper bound. It can be easily verified thatis a convex function of  and. Thus, we can solve the minimization ofefficiently by exploiting standard convex optimization tools (e.g., CVX) and obtain the online decisions of RDU and SPU (i.e.  and). At the end of each time slot, all queues are updated accordingly

***Algorithm 1:* iCloudAccess**: **Online Control Algorithm for RDU and SPU**

⎬

**Input:**

The values of N, M, m, μ, €, V;

Prices of on-demand cloud servers;

Number of incoming user requests λij(t);

Network delay between regions and data centers, dij (t);

**Output:**

RDU and SPU decision 

1. Initialization step: Let *t* =0, and set Qj(0)=0, Hj(0)= 0, for j =1,2,...,N.
2. while the cloud gaming service is running do
3. if (*t* mod m) == 0 then
4. Monitor the queue backlog Q(t), H(t) and the real-time information of cj(t)for each data center j.
5. Determine the SPU decision  by solving 
6. End if
7. Update information of network delay between a region i and a data center j, and the amount of user requests from a region i (i.e., λi(t))for i =1,2,...,M, j = 1,2,...,N.
8. Determine the RDU decision  by solving 
9. Update Q and H according to (9) and (11), respectively
10. End While

The details of our online algorithm for requesting dispatching and server provisioning are given in Algorithm 1. The algorithm generates the decision λ(t) on request dispatching every time slot and the decision  on server rovisioning

Every *m* time slot. In our algorithm, the time complexity lies in solving the optimization problem in each round. Efficient linear programming tools can be applied to resolve the timeslotted linear programming problem. Compared with the VM provisioning delay, the computational time of our online algorithm incurs little time overhead. Our online algorithm can approach the optimal solution

Of the original optimization problem within infinitely small distance. The distance to the optimality is determined by the tuning parameter V. Theorem 1 explicitly proves the performance bound on the server provisioning cost of our

algorithm.

***Theorem 1***:The performance bound of the time-average server provisioning cost induced by our online algorithm (by solving problemP2) can be given by



Where  denotes the in fimum of the average server provisioning cost over all stable policies, and   is the upper bound of the function





Proof: Please see Appendix D in our technical report [21] for the proof details.

Although our algorithm design uses the estimates of queue backlogs, which may differ from the real value, it has been proven in [18] that the estimate bias will only cause a loosening of the upper bound. By choosing a large value of V,

We can obtain the same results as that when using the accurate queue backlog information

**Chapter 5**

# Experimental Evaluation

In this section, we develop a discrete-event simulator and conduct a set of experiments to evaluate the effectiveness of our proposed online algorithm.

## Experiment Settings

In our simulation, we consider a scenario in which a CGSP has ten distributed data centers to serve users spreading over 15 regions. User requests are dispatched without delay and the CGSP adjusts the number of cloud servers provisioned at each data center every 30 min. To make our simulation more realistic, we use the workload model obtained from the real online game traces [19] to simulate user arrival and departure 2.The mean gaming session is around 30 min. Response delay consists of three components: network delay, processing delay, and playout delay. On the basis of our measurements of CloudUnion, we assume that network delay between a data center and a region is randomly distributed in the range (100, 500) ms. We further assume that the sum of the left two delay components (namely, processing delay and playout delay) follows a uniform distribution in (50, 100) ms, which approximates the measurement results of response delay in Fig. 6. Note that the parameter settings vary with different hardware and software configurations.

The provisioning cost of a cloud server is set based on the Amazon EC2’s pricing model. We use the real pricing traces for Linux/UNIX VM instances obtained from the Amazon’s website [20]. In the traces, the price of VM instances takes values from a finite set within the range [0.05, 0.07] (in units of $ per hour) and changes dynamically. In the experiment, we assume that the prices at each data center are independent and identically distributed (i.i.d.).

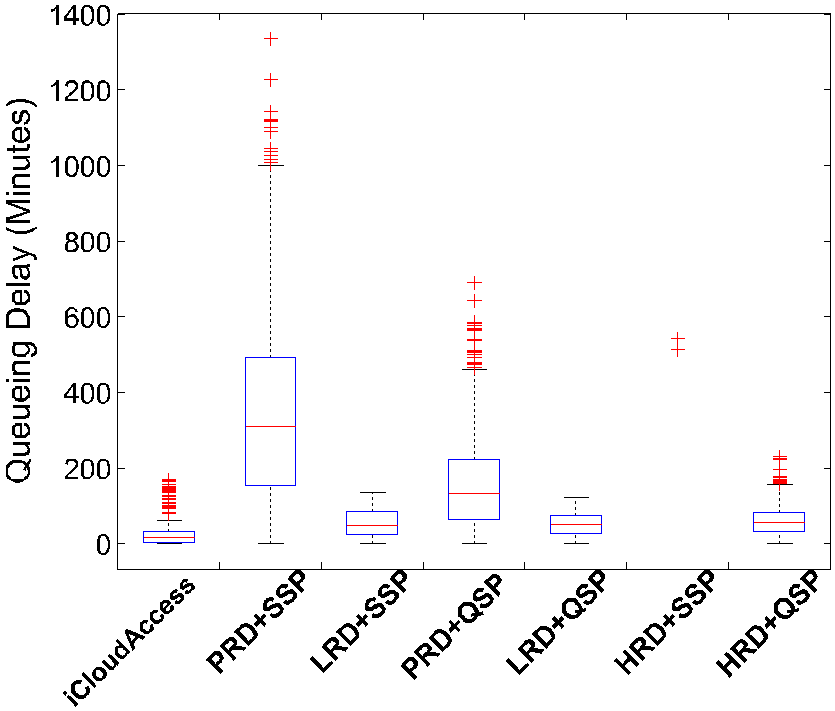
We compare our proposed iCloudAccess algorithm with four other alternatives, each of which is a combination of a request dispatching strategy and a server provisioning strategy for data centers. In our simulation, we consider the following three typical request-dispatching strategies.

1. Proximity-aware request dispatching (PRD), in which requests are always dispatched to a data center with the lowest network delay (for example, a data center in a nearby region). It can maximize the reduction of the response delay for a user.
2. Load-aware request dispatching (LRD), in which requests are always routed to a data center with the lowest workload level.
3. Hybrid-weight request dispatching (HRD), in which we consider a weighted sum of the normalized proximity factor and the normalized load factor as the metric. Requests are routed to a data center with the smallest weighted sum. HRD can be thought of as a hybrid of PRD and LRD. By default, the weight parameter is set as 1 to achieve a balance between the proximity factor and the load factor.

Two server provisioning strategies are used for comparison in our experiments.

1. Stable server provisioning (SSP), in which a fixed num- ber of cloud servers are provisioned at each data center. The server provisioning does not change over time. It is a simple yet widely adopted server provisioning strategy for CGSPs. In our experiment, we assume that the time- average number of user requests for each data center is known beforehand, and the number of provisioned cloud servers equals the time-average number of user requests. Such assumption is ideal and only made for the comparison purpose.
2. Queuing-aware server provisioning (QSP), in which the CGSP makes provisioning decision based on the observations on the backlogs of the waiting queue. At every decision time point, if there are k users in the waiting queue, the CGSP will provision k/2 additional cloud servers in the next period; otherwise, if there are no waiting users, the CGSP only continues to provision the number of cloud servers in use in the next period.

Therefore, we have seven methods in comparison, namely, 1) PRD + SSP; 2) LRD + SSP; 3) PRD + QSP; 4) LRD + QSP; 5) HRD + SSP; 6) HRD + QSP; and 7) our proposed iCloudAccess algorithm



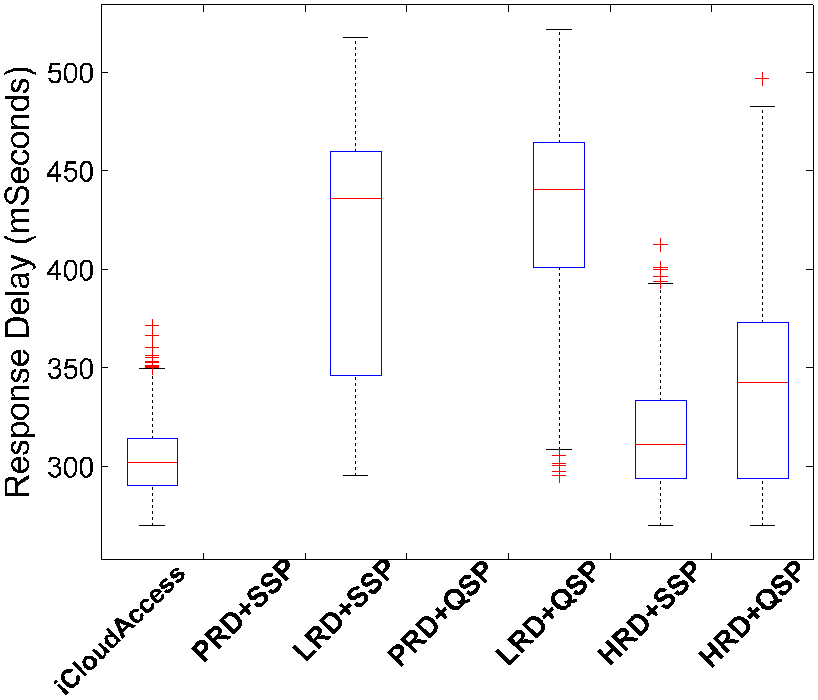


Fig 9 Comparison of gaming latency. (a) Queuing delay. (b) Response delay.

## Comparison of Gaming Latency

Fig. 9 shows the comparison of gaming latency among different methods. From the results in Fig. 9(a), we can clearly observe that our proposed iCloudAccess can reduce the mean and variance of queuing delay significantly. Among five methods, PRD + SSP perform the worst due to the serious queuing problem caused by PRD in the hot regions. Our iCloudAccess method achieves the lowest average queuing delay, and the variance of queuing delay is also small. Most of the outliers occur in the early stage of the experiment. For the two methods with QSP, the queuing delay can also be reduced greatly, as the provisioning decision takes the queue state into account.

Fig. 9(b) plots the response delay incurred by different methods. Two methods (PRD + SSP and PRD + QSP) with PRD achieve the lowest response delay as they always dispatch requests to the data center with the lowest network delay. However, PRD may cause serious queuing delay in the hot regions if the paces of server provisioning cannot keep up with the increase of user demand [one example is PRD + SSP in Fig. 9(a)]. The response delay of LRD + SSP and LRD + QSP is high because the selection of data centers is ignorant of proximity (or latency). Our proposed iCloudAccess approaches the performance of PRD + SSP and PRD +

QSP with very small distance (normally with an increase of 10–20 Ms). However, we will show in Section V-C that the provisioning cost of our method is the lowest among seven.

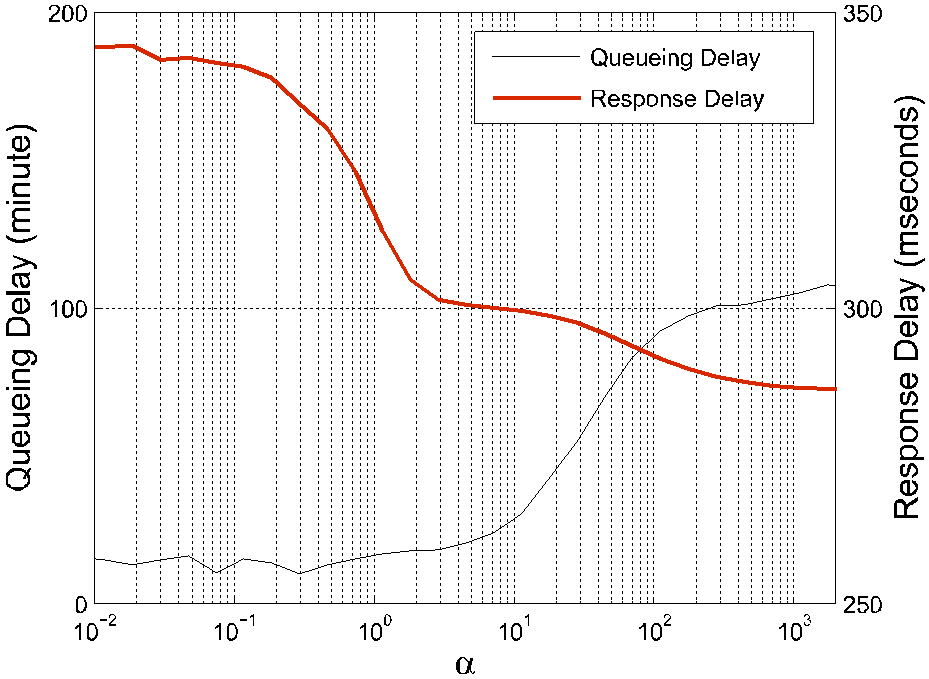


fig 10 Impacts of the tunable parameter α on queuing delay and response delay.

Methods, which is of more importance for the cloud gaming server providers.

To examine the impacts of different values of *α* on the QoE function, we conducted additional experiments with various values of *α*. From the results in Fig. 10, we can observe that, with the increase of *α*, the response delay decreases significantly, while the queuing delay increases in the meanwhile. The parameter *α* determines user sensitivity to different kinds of delay. We also compare our proposed strategy with other approaches under various values of *α*, and the results show that our proposed solution still outperforms other alternatives. PRD and LRD can be thought of as two special cases of HRD by varying the weight. More experimental results about HRD can be found in our technical report [21].

## Comparison of Server Provisioning Cost

Fig. 11 shows the total server provisioning cost incurred by different methods during the simulation period. Our simulation lasts for 3000 min. PRD + SSP and LRD + SSP incur the highest provisioning cost, and our proposed iCloudAccess has the lowest provisioning cost. Fig. 11 shows the total server provisioning cost incurred by different methods during the simulation period. Our simulation lasts for 3000 min. PRD + SSP and LRD + SSP incur the highest provisioning cost, and our proposed

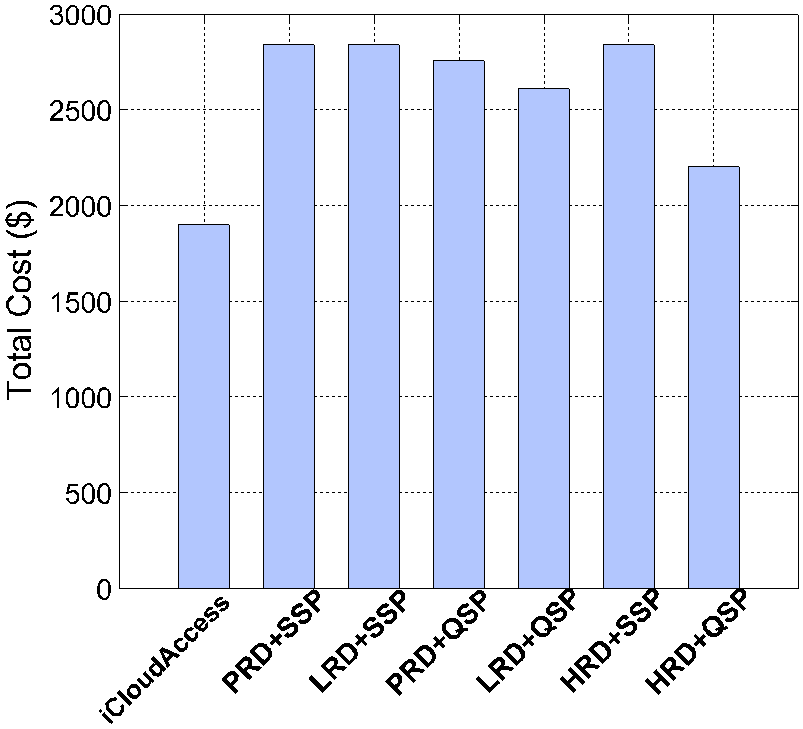


fig 11 Comparison of server provisioning cost

Fig. 11 shows the total server provisioning cost incurred by different methods during the simulation period. Our simulation lasts for 3000 min. PRD + SSP and LRD + SSP incur the highest provisioning cost, and our proposed iCloudAccess has the lowest provisioning cost. More accurately, our method can reduce 33% of the provisioning cost compared with PRD + SSP, LRD + SSP, and HRD + SSP, 31% of the provisioning cost compared with PRD + QSP, 27% of the provisioning cost compared with LRD + QSP, and 16% of the provisioning cost compared with HRD + QSP.

As stated in Theorem 1, the cost saving of our iCloudAccess method depends on the parameter setting of V, which determines the distance to the optimal value. The parameter *V* also determines the trade-off between queue size and penalty cost in the Lyapunov drift-penalty optimization framework. The size of queue *H* affects the user QoE, while the penalty cost indicates the server provisioning cost. When a small value of *V* is chosen, our algorithm will prefer to improve the user QoE, but it is at the cost of a higher server provisioning cost. On the contrary, when the value of *V* is large, the algorithm prefers to reduce the server provisioning cost, but at the cost of a lower level of user QoE. The choice of *V* depends on the budget and the desired user QoE level to be achieved by the service provider. In our experiment, the value of *V* is set as 100. With a larger value of *V,* we can further approach the infimum of provisioning cost.

Fig. 12 shows the impact of the parameter *V* on the server provisioning cost. When the value of *V* increases from 100 to 108, the total provisioning cost can be further reduced from over 1800 U.S. to nearly 950 U.S., resulting in a reduction of over 60% provisioning cost compared with the best one of the other four methods. However, when the value of *V* is big enough (e.g., *>*108), there is marginal benefit on further reducing the provisioning cost by increasing the value of *V.* CGSPs can set an appropriate value of *V* according to their server provisioning budget and the preferred user QoE.

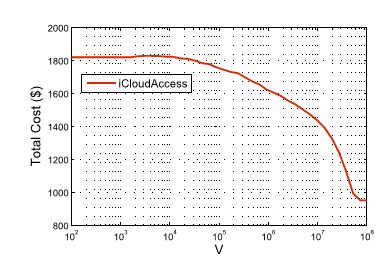


fig 12 Impact of the parameter V on the provisioning cost.

## Stability of Virtual Queues

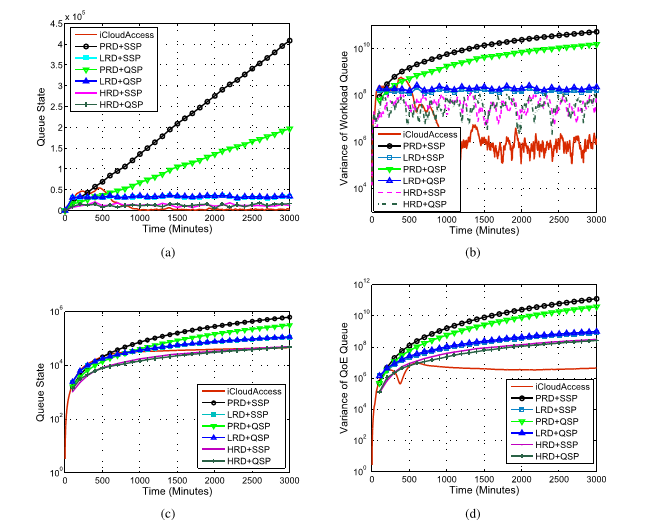


Fig 13 Stability of virtual queues. (a) Evolution of the workload queue. (b) Variance of the workload queue. (c) Evolution of the QoE queue. (d) Variance

The stability of virtual queues is important for long-running control algorithms. To analyse the stability of virtual queues, we plot the evolution of queue state and its corresponding variance in Fig. 13. The state of the workload queue Q (t) is measured by the total amount of workload held in the workload queues [i.e., Q j (j = 1, N )] of all data centers namely, 

Fig. 13(a) shows the evolution of the state of workload queue. It is observed that iCloudAccess can stabilize the workload queue to a rather low level after 1000 min. Fig. 13(b) further shows the variance of the workload queues of different data centers, namely, the variance of Q j (t )( j = 1, ... , N ).The variance under iCloudAccess is much lower than other four methods and tends to be stable after around 900 min. It implies that our proposed iCloudAccess method can well balance the workload distribution among data centers. We also track the state of the QoE queue H (t ) in Fig. 13(c). We measure the state of the QoE queue by the total queue backlogs in H j ( j = 1, ... , N ) at the time slot *t* ,

i.e.,  The total queue backlogs of iCloudAccess tends to be stable after 600 min. Recall that we have proven in Lemma 4.1 that if the virtual queue H j is stable, then the QoE constraint (8) can be satisfied. Our results clearly show that our proposed iCloudAccess can ensure the QoE constraint by stabilizing the virtual queue H j . Fig. 13(d) shows the variance of H j (t )( j = 1, ... , N ). It is found that the variance of queue backlogs under a iCloudAccess is much smaller than that under the other four strategies after 600 min. It means that our proposed iCloudAccess method can balance the queue backlogs among different data centers. Therefore, users assigned to different data centers can have similar user experience

**Chapter 6**

# CONCLUSION

The on-demand feature of cloud gaming enables users to play gaming without the hardware and software constraints. In this paper, we study the latency problem of a multiregion multidata center cloud gaming system. We first conduct a series of active and passive measurements on a large-scale cloud gaming service offering in China. From the measurement results, we observe that users suffer from high diversity in the queuing delay and response delay. To optimize the user experience and minimize the operational cost of CGSPs simultaneously, we formulate the problem into a constrained stochastic optimization problem and exploit the Lyapunov optimization framework to derive the online request dispatching and server provisioning strategies. Our proposed approach can significantly cut down the operational cost and reduce the latency at the same time. The work in this paper can provide useful guidelines for CGSPs to provision their services effectively. In the future, we plan to investigate the heterogeneity of user QoE requirements and study how to further optimize the cloud gaming infrastructure

# Bibliography

[1] (2012, Jul.). Distribution and Monetization Strategies to Increase Revenues from Cloud Gaming [Online]. Available: http://www.cgconfusa. com/report/documents/Content-5minCloudGamingReportHighlights.pdf

[2] C. Chambers, W.-C. Feng, S. Sahu, D. Saha, and D. Brandt, “Characterizing online games,” IEEE/ACM Trans. Netw., vol. 18, no. 3,

pp. 899–910, Jun. 2010.

[3] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, “GamingAnywhere: An open cloud gaming system,” in Proc. ACM Multimedia Syst. Conf., Feb. 2013, pp. 36–47.

[4] N. Tolia, D. Andersen, and M. Satyanarayanan, “Quantifying interactive user experience on thin clients,” IEEE Comput., vol. 39, no. 3, pp. 46–52, Mar. 2006.

[5] Y.-C. Chang, P.-H. Tseng, K.-T. Chen, and C.-L. Lei, “Understanding the performance of thin-client gaming,” in Proc. IEEE Int. Workshop Tech. Committee CQR, May 2011, pp. 1–6.

[6] (2013, Jul.). X264 Web Page [Online]. Available: http://www.videolan. org/developers/x264.html

[7] R. Frederick and V. Jacobson, “Rtp: A transport protocol for real-time applications,” IETF, London, U.K., Tech. Rep. RFC 3550, 2003.

[8] H. Schulzrinne, “Real time streaming protocol (RTSP),” IETF, London, U.K., Tech. Rep. RFC 2326, 1998.

[9] (2013). Onlive. [Online]. Available: http://www.onlive.com/ [10] (2013). Gaikai. [Online]. Available: http://www.gaikai.com/

[11] (2013). CloudUnion. [Online]. Available: http://www.yxyun.com/

[12] K. Chen, Y. Chang, P. Tseng, C. Huang, and C. Lei, “Measuring

the latency of cloud gaming systems,” in Proc. 19th ACM Int. Conf. Multimedia, 2011, pp. 1269–1272.

[13] H. LagarCavilla, J. Whitney, A. Scannell, P. Patchin, S. Rumble,

E. D. Lara, et al., “SnowFlock: Rapid virtual machine cloning for cloud computing,” in Proc. 4th ACM Eur. Conf. Comput. Syst., 2009, pp. 1–12.

[14] L. Shi, M. Banikazemi, and Q. Wang, “Iceberg: An image streamer for space and time efficient provisioning of virtual machines,” in Proc. ICPWorkshops, Sep. 2008, pp. 31–38.

[15] J. Zhu, Z. Jiang, and Z. Xiao, “Twinkle: A fast resource provisioning mechanism for internet services,” in Proc. IEEE INFOCOM, Apr. 2011, pp. 802–810.

[16] Z. Xue, D. Wu, and J. He, “A measurement study of a large-scale commercial cloud gaming system,” Dept. Comput. Sci., Sun Yat-sen Univ., Guangzhou, China, Tech. Rep. CS-2013-05, 2013.

[17] M. J. Neely, Stochastic Network Optimization with Application to

Communication and queuing Systems. San Rafael, CA, USA: Morgan

& Claypool, 2010.

[18] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, “Data centers power reduction: A two time scale approach for delay tolerant workloads,” in Proc. IEEE INFOCOM, Mar. 2012, pp. 1431–1439.

[19] M. Kwok and G. Yeung, “Characterization of user behavior in a multi-

Player online game,” in Proc. ACM SIGCHI Int. Conf. Adv. Comput. Entertainment Technol., 2005, pp. 69–74.

[20] (2013). Amazon EC2 Pricing [Online]. Available: http://aws.amazon. Com/ec2/pricing/

resource provisioning and adaptation in iaas clouds,” in Proc. IEEE 3rd

Int. Conf. Cloud Comput. Technol. Sci., Dec. 2011, pp. 312–319.

[28] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau, “Cloudmedia: When cloud on demand meets video on demand,” in Proc. 31st ICDCS, Jun. 2011, pp. 268–277.

[29] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, “Cost-effective partial migration of VoD services to content clouds,” in Proc. IEEE Int. Conf. Cloud Comput., Jul. 2011, pp. 203–210.

[30] M. Marzolla, S. Ferretti, and G. D’Angelo, “Dynamic resource provisioning for cloud-based gaming infrastructures,” Comput. Entertainment, vol. 10, no. 3, pp. 1–4, Oct. 2012

[21] D. Wu, Z. Xue, and J. He, “iCloudAccess: Cost-effective streaming of video games from the cloud with low latency,” Dept. Comput. Sci., Sun

Yat-Sen Univ., Guangzhou, China, Tech. Rep. CS-2013-07, 2013.

[22] (2013). Ubitus [Online]. Available: http://www.ubitus.net/en/index.html

[23] (2013). BigFish Games [Online]. Available:

http://www.bigfishgames.com/

[24] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “The brewing storm in

cloud gaming: A measurement study on cloud to end-user latency,” in Proc. IEEE 11th Annu. Workshop Netw. Syst. Support Games, Nov. 2012, pp. 1–6.

[25] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “Edgecloud: A new hybrid platform for on-demand gaming,” Univ. Wateloo, Waterloo, ON, Canada, Tech. Rep. CS-2012-19, 2012.

[26] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, “Resource provisioning for

cloud computing,” in Proc. Conf. Center Adv. Studies Collaborative Res.,

2009, pp. 101–111.

[27] T. N. B. Duong, X. Li, and R. S. M. Goh, “A framework for dynamic

[31] M. Claypool, D. Finkel, A. Grant, and M. Solano, “Thin to win? Net- work performance analysis of the OnLive thin client game system,” in Proc. IEEE 11th Annu. Workshop Netw. Syst. Support Games, Nov. 2012, pp. 1–6.

[32] M. Manzano, J. Hernandez, M. Uruena, and E. Calle, “An empirical study of cloud gaming,” in Proc. IEEE 11th Annu. Workshop Netw. Syst. Support Games, Nov. 2012, pp. 1–2.

[33] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, “An evaluation

of QoE in cloud gaming based on subjective tests,” in Proc. IEEE 5th

Int. Conf. IMIS, Jul. 2011, pp. 330–335.