

國立清華大學電機資訊學院資訊工程研究所

碩士論文

Department of Computer Science

College of Electrical Engineering and Computer Science

National Tsing Hua University

Master Thesis

虛擬機器配置方法優化雲端遊戲體驗

Placing Virtual Machines to Optimize Cloud Gaming Experience



洪華駿

Hua-Jun Hong

指導教授：徐正炘 博士

Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 103 年 06 月

June, 2014

國立清華大學
資訊工程研究所

碩士論文

虛擬機器配置方法優化雲端遊戲體驗



洪華駿
撰

103
06

Acknowledgments

I would like to express my gratitude to all the persons who help me to finish my thesis in this year. First, I would like to thank my parents who always support and help me a lot so that I can complete my works perfectly. Second, I would like to appreciate Prof. Kuan-Ta Chen, Prof. Chun-Ying Huang, Prof. Che-Rung Lee, and De-Yu Chen from Academia Sinica, National Taiwan Ocean University, and National Tsing Hua University. It is appreciated to have your suggestions and contributions in our joint work. The cooperation with you is a nice experience during this year. Finally, I would like to appreciate my adviser: Prof. Cheng-Hsin Hsu. Without your advising, I could not have the wonderful opportunity to attend this work and finish my thesis.



致謝

我要感謝所有在我就讀碩士班期間幫助過我的人。首先，我要謝謝我的父母一直支持著我 讓我能夠完美的完成一切的工作。接著，我要感謝我的合作者們陳寬達教授，黃俊穎教授，李哲榮教授以及 陳德禹學長，他們分別來自中央研究院，國立台灣海洋大學以及國立清華大學。謝謝你們在這個合作計劃中的 建議以及所有的貢獻，在我的碩士生涯，這是一份極好的合作經驗。最後，我要感謝我的指導 教授:徐正忻教授，如果沒有教授的指導，我沒有辦法擁有這麼好的機會參與此計劃並且完成我的碩士論文。



中文摘要

如何在雲端遊戲廠商以及玩家的遊戲品質體驗之間找到一個平衡點非常複雜，進而導致最佳化 雲端遊戲體驗變成了一件不容易的工作。我們解決了這項挑戰並且研究一個最佳化問題來 最大化雲端遊戲廠商的利益並且同時讓玩家有足夠的遊戲品質體驗。我們測量並且得到了遊戲品質體驗以及效能的數學模型，接著我們將問題轉換為數學式並得出最佳解，但最佳 解需要指數倍的運算時間，所以我們發展一個有效率啟發式演算法。我們也為了封閉式的 雲端遊戲環境提出了另一項方程式以及演算法，在這個情境下利益將不被考量，我們會最大化玩家的遊戲品質體驗。我們利用現存的虛擬化技術實作出了一個系統雛形以及小型實驗環境來驗證 我們演算法的效率以及實用性，我們的實驗指引了雲端遊戲廠商如何創造他們自己的盈利環境。接著我們延伸我們的實驗到擁有實際數據的模擬器上，此實驗說明了: (i) 此演算法接近最佳解， (ii) 能夠有兩萬個主機以及四萬個玩家， (iii) 比現行的虛擬機器配置演算法效能高出許多，例如: 擁有3.5倍的利益。在解決了虛擬機器配置問題之後，我們對最新的顯示卡進行測量，進而回答一 問題: 是否現行的顯示卡已經足夠供給雲端遊戲了呢?。與以前的研究不同的地方，我們得到了 許多違背過去常識的結果。其一，最新的顯示卡虛擬化技術可能會使分享式顯示卡的效能比專用的 顯示卡虛擬化還好，其二，越多的工作轉換不一定會導致幀數下降。總的來說，我們得知了最新的 顯示卡虛擬化技術已經足以分享給多個需要大量顯示卡效能的雲端遊戲玩家。最後，我們發現使用 最新顯示卡的主機的瓶頸可能會轉為處理器，而必須將影像的編碼從處理器移植到專用的編解碼晶片 上來得到較好的遊戲品質體驗。

Abstract

Optimizing cloud gaming experience is no easy task due to the complex tradeoff between gamer Quality of Experience (QoE) and provider net profit. We tackle the challenge and study an optimization problem to maximize the cloud gaming provider's total profit while achieving just-good-enough QoE. Moreover, we conduct experiments using a modern GPU and a cloud gaming platform to answer the following question: *Are modern GPUs ready for cloud gaming?* For the optimization problem, We conduct measurement studies to derive the QoE and performance models. We formulate and optimally solve the problem. The optimization problem has exponential running time, and we develop an efficient heuristic algorithm. We also present an alternative formulation and algorithms for closed cloud gaming services with dedicated infrastructures, where the profit is not a concern and overall gaming QoE needs to be maximized. We present a prototype system and testbed using off-the-shelf virtualization software, to demonstrate the practicality and efficiency of our algorithms. Our experience on realizing the testbed sheds some lights on how cloud gaming providers may build up their own profitable services. Moreover, we conduct extensive trace-driven simulations to evaluate our proposed algorithms. The simulation results show that the proposed heuristic algorithms: (i) produce close-to-optimal solutions, (ii) scale to large cloud gaming services with 20000 servers and 40000 gamers, and (iii) outperform the state-of-the-art placement heuristic, e.g., by up to 3.5 times in terms of net profits. For the measurement study of modern GPU, the observations are different from earlier studies, our measurement results reveal several findings that are counter to common beliefs. First, with the latest GPU virtualization technique, shared GPUs may run faster than dedicated GPUs. Second, more context switches not necessarily lead to lower FPS (frame-per-second). In summary, we conclude that modern GPUs are powerful enough and can be shared by multiple GPU-intensive cloud games. Last, we present some suggestions for future cloud gaming platforms, e.g., the latest GPU servers may be CPU-bounded, which require the platforms to offload the video encoding from CPUs to dedicated codec chips for good gaming experience.

Contents

Acknowledgments	i
致謝	ii
中文摘要	iii
Abstract	iv
1 Introduction	1
2 Related Work	4
2.1 General Cloud Applications	4
2.2 Cloud Games	5
2.3 GPU Virtualization on Cloud Gaming Systems	5
3 Measurement Studies	7
4 VM Placement Problem and Solution	9
4.1 System Overview	9
4.2 Notations and Models	10
4.3 Problem Formulation	11
4.4 Proposed Algorithm	12
5 Alternative Formulation and Algorithms for Closed Systems	14
6 System Implementation and Testbed	16
6.1 Prototype Implementation	16
6.2 Testbed and Practical Concerns	17
6.3 Experiment–Performance Gains of the Migrationless Algorithms	19
7 Trace-Driven Simulations	22
7.1 Setup	22
7.2 Results	23
8 GPU Consolidation for Cloud Games	26
8.1 Methodology	26
8.1.1 Workload Generators	26
8.1.2 Experiment Setup	27
8.1.3 Performance Metrics	27
8.1.4 Measurement Utilities	27

8.2 Measurement Results	28
9 Conclusion and Future Work	32
Bibliography	34



List of Figures

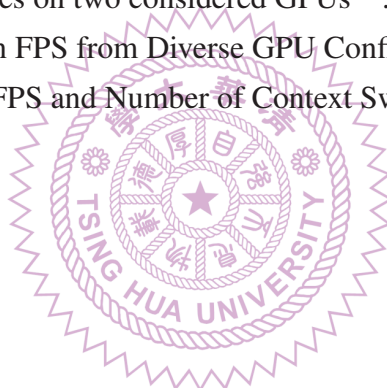
1.1	The architecture of cloud gaming services, where GS denotes cloud gaming server.	2
2.1	Virtualization overhead depends on game and VM implementations. . . .	6
2.2	Measurement results for CPU utilization, GPU utilization, frame rate, and processing delay. Sample results from Limbo.	6
4.1	The pseudocode of the QDH algorithm.	11
5.1	The pseudocode of the QDH' algorithm.	15
5.2	The implemented prototype system.	15
5.3	The cloud gaming testbed in our lab.	15
6.1	Live migration.	17
6.2	Comparisons between QDH_L/QDH'_L and QDH/QDH' : (a) net profits and (b) quality.	18
6.3	Comparisons between QDH_L/QDH'_L and QDH/QDH' : (a) net profits and (b) quality.	18
6.4	Comparisons between QDH_L/QDH'_L and OPT_L/OPT'_L : (a) net profits and (b) quality.	19
6.5	Provider-centric simulation results with synthetic traces: (a) net profits and (b) used servers.	21
6.6	Gamer-centric simulation results with synthetic traces.	21
7.1	Fairness in QoE levels on different game genres.	22
7.2	Simulation results with WoW traces: (a) net profits, (b) used servers, and (c) QoE levels.	23
7.3	Running time of VM placement algorithms.	24
7.4	Impacts of number of gamers on: (a) net profits and (b) QoE levels. . . .	24
8.1	Our testbed with a cloud game server (right) and four game clients (left), connected by a Fast Ethernet switch (middle).	28

8.2	Comparing the pass-through and vGPU: (a) resulting FPS, (b) 2D benchmark scores, and (c) 3D benchmark scores.	28
8.3	GPU consolidation overhead: (a) resulting FPS with various numbers of VMs, (b) fully loaded time ratio from vGPU ₈ , and (c) CPU _{vm} from vGPU ₈	29
8.4	End-to-end performance of a cloud game platform: (a) resulting FPS, (b) CPU _{vm} utilization with pass-through GPU, and (c) CPU _{vm} utilization with vGPU ₂	29
1	Levels of virtualization.	39
2	Two types of virtualization: (a) multiple guests with a single operating system and (b) multiple guests with individual operating systems.	40



List of Tables

3.1	R-square Values of Different Games/VM	8
4.1	Symbols Used Throughout This Paper	13
7.1	Running Time in Seconds	25
8.1	Specifications of Two GPUs.	27
8.2	Achieved frame rates on two considered GPUs	29
8.3	Sanctuary Scores in FPS from Diverse GPU Configurations	30
8.4	Relation Between FPS and Number of Context Switches	30





Chapter 1

Introduction

To offer on-demand gaming services to many gamers using heterogeneous client computers, including game consoles, desktops, laptops, smartphones, and set-top boxes, increasingly more service providers push computer games to powerful cloud servers and stream the game scenes to a simple application running on client computers [37]. Such on-demand game services are referred to as *cloud gaming* by various companies, such as Gaikai, Ubitus, and OnLive. Market research predicts that the cloud gaming market is going to grow to 8 billion USD by 2017 [12], and some leading game development companies [9] have seriously considered this new opportunity. Therefore, we expect to see many more cloud gaming services soon.

Offering cloud gaming services in a commercially-viable way is, however, very challenging as demonstrated by OnLive's financial difficulty [34]. The main challenge for cloud gaming providers is to find the best tradeoff between two contradicting objectives: *reducing the hardware investment* and *increasing the gaming Quality-of-Experience (QoE)*. Satisfactory gaming QoE demands for high-end hardware, which may incur huge financial burden; meanwhile, using low-end hardware leads to less pleasing gaming QoE, which may drive gamers away from the cloud gaming services. Moreover, different game genres impose diverse hardware requirements, which may result in insufficient or wasted hardware resources if server resources are not well planned. For example, the servers configured for cutting-edge 3D first person shooter games may be an overkill for 2D casual games. The server diversity renders the dilemma of finding the best tradeoff between *profit* and *QoE* even harder.

Since cloud gaming services push games to cloud servers, *server consolidation* enables dynamic resource allocation among game servers serving multiple gamers for better overall performance and lower operational cost. In this paper, we study the problem of efficiently consolidating multiple cloud gaming servers on a physical machine using modern virtual machines (VMs), such as VMware and VirtualBox, in order to provide high

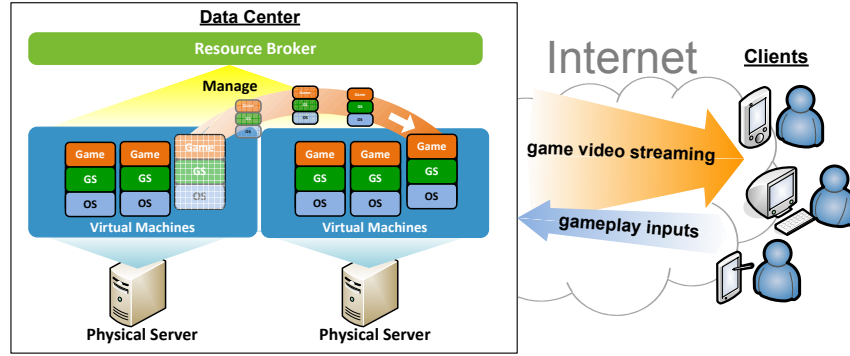


Figure 1.1: The architecture of cloud gaming services, where GS denotes cloud gaming server.

gaming QoE in a cost-effective way, as illustrated in Fig. 1.1. We consider the VM placement problem to maximize the total profit while providing the just-good-enough QoE to gamers. This problem is referred to as *provider-centric* problem throughout this paper.

The considered problem is a variation of the virtual network embedding problem [1], and thus is NP-Complete. The existing solutions for network embedded problems [1, 5, 6, 31, 43], however, are designed for computational/storage intensive applications, without taking the *real-time* requirements of cloud gaming (and other highly interactive applications) into consideration. In particular, unlike computational/storage intensive applications that demand for high CPU/disk throughput, cloud games demand for high QoE, in terms of, e.g., responsiveness, precision, and fairness [4, 25, 39]. Hence, the existing virtual network embedding algorithms do *not* work for cloud gaming providers.

To reduce the operation costs by consolidation policy, however, results in degraded user-perceived quality, and may drive gamers away from cloud gaming services. While virtualizing CPUs, network interfaces, and storages is rather mature, virtualizing GPUs is still considered experimental. In fact, several papers warn the potentially poor performance in terms of low frame rate, high response time, and low video quality when GPUs are shared among multiple VMs [13, 38]. For example, Shea and Liu [38] show that the frame rate of Doom 3 is lower than 40 FPS (frame-per-second) even if Xen and KVM are configured with one-to-one GPU pass-through, which renders sharing the GPU among more than two VMs virtually impossible. Nonetheless, in the past couple of years, the GPU virtualization technology has been dramatically improved, which *may* have solved the performance issue of VM consolidation for computer games. To the best of our knowledge, this paper is the first attempt to tackle the VM placement problem to maximize the cloud gaming QoE and conduct a measurement study to quantify the performance of GPU virtualization technology for cloud gaming systems.

In particular, this paper makes the following contributions:

- We conduct extensive measurement studies using an open-source cloud gaming platform, GamingAnywhere [20] on two VM implementations to derive the game-dependent parameters for QoE and performance models (Ch. 3).
- We formulate and propose two algorithms for the provider-centric VM placement problem (Ch. 4).
- We extend the provider-centric VM placement problem into a *gamer-centric* problem for closed cloud gaming services, e.g., in hotels, Internet cafes, and amusement parks, where the overall gaming QoE needs to be maximized using already-deployed infrastructures. We also propose two algorithms to solve the gamer-centric problem (Ch. 5).
- We present a prototype system built by off-the-shelf components, and quantify the implication of live *migration*, which refers to moving a running VM from one physical server to another. We augment our algorithms to accommodate to high migration overhead, resulting in efficient and practical algorithms (Ch. 6).
- Our extensive trace-driven simulations indicate that: (i) our efficient algorithms result in close-to-optimal performance, as small as 0% and 10% gaps, (ii) the efficient algorithms scale to large cloud gaming services with twenty thousands of servers and more than forty thousands gamers, and (iii) the efficient algorithms outperform a state-of-the-art algorithm by large, e.g., up to 3.5 times of net profit increase (Ch. 7).
- we conduct real experiments using modern GPUs and a real cloud gaming platform [19] to answer the following question: *Are modern GPUs ready for cloud gaming?* Our experiment results reveal several insights that have never been reported in the literature. For example, we observe that: (i) virtualized GPUs may *outperform* pass-through GPUs, and (ii) cloud servers with modern GPUs may become *CPU-bounded*. These observations are quite different from common beliefs. Our experiments not only answer the above question, but also shed some lights on better designs of future cloud gaming platforms for high user-perceived quality (Ch. 8).

Chapter 2

Related Work

2.1 General Cloud Applications

Optimizing general cloud applications has been studied in cloud environments. For example, Zaman et al. [44] propose an auction-based mechanism for dynamic provision and allocation of VMs to maximize the provider's profit and improves the total utilization of cloud resources. Lin et al. [28] formulate the data replication problem in the clouds as a mathematical optimization problem and propose several algorithms for the I/O intensive applications. In our work, we formulate the VM placement problem of cloud gaming systems and propose optimization algorithms to solve the problem. Different from these two studies [28, 44], we optimize the real-time cloud games with an objective of maximizing the provider's profit by QoE-aware algorithms while optimizing the gaming quality at the same time.

VM migration techniques have been investigated for non real-time applications. Marzolla et al. [30] utilize the live migration technology to move the VMs away from the the lightly loaded physical servers and thus the empty servers can be switched to low-power mode. Ferreto et al. [16] create a dynamic server consolidation algorithm with migration control and avoid unnecessary migrations to reduce the number of powerd on servers and migration cost. Chen et al. [3] find that virtual machines do not usually use all their resources, and they create an algorithm which also considers the migration cost according to the records of migration history for saving energy. Speitkamp and Bichler [40] present a heuristic solution which approximates the optimal solution by not only considering the cost but also determining whether the problem size can be optimally solved. Nathuji et al. [46] create a performance interference model and classify the applications into different resource bounds using historical data. The applications are then consolidated on physical servers for better Quality of Service (QoS). Zhu and Tung [33] also consider the interference and implement a system to determine the placement of VMs to avoid the

interference and meet the desired QoS values. None of the aforementioned studies take cloud gaming QoE levels into consideration.

2.2 Cloud Games

The benefits of game server consolidation have been studied for certain game genres. For example, Lee and Chen [24] address the server consolidation problem for Massively Multiplayer Online Role-Playing Game (MMORPG). In particular, they propose a zone-based algorithm to leverage spatial locality of gamers in order to reduce the hardware requirements at the servers. Their work is different from ours for two reasons. First, we consider cloud gaming that streams high-quality real-time videos to gamers, while MMORPG servers only send low bitrate status updates. Second, we explicitly optimize gaming QoE in this paper, while they only attempt to save energy at the data centers without taking QoE into consideration.

Duong et al. [15] and Wu et al. [42] are complementary to our work, as they concentrate on minimizing the queuing delay of a cloud gaming system, while we focus on the user experience during the game sessions. For example, Duong et al. [15] develop resource provisioning and waiting queue scheduling algorithm to admit selective incoming gamers for the best profit under user-specified maximal waiting times. Wu et al. [42] also propose an online control algorithm to quickly serve users in the waiting queue. Compared to their work, we optimize the gaming QoE after a user is admitted in the system; such QoE maximization is arguably more important, as gamers typically can only tolerate a few minutes of waiting time, but each game session may last for hours.

Most of the cloud gaming systems, including Gaikai, Ubitus, and OnLive are proprietary and closed, and thus measuring cloud gaming performance and QoS on them is hard, if not impossible. We employ GamingAnywhere (GA) [20] for our experiments, which is an open cloud gaming system. In particular, we use GA to derive the performance and QoS models for different games on different VMs, and to develop VM placement algorithms. Last, our initial investigations on the QoE-aware virtual machine placement problems were reported in Hong et al. [18].

2.3 GPU Virtualization on Cloud Gaming Systems

GPU architecture has been constantly changed, which renders virtualizing GPU for concurrent access from multiple VMs rather challenging. Dowty and Sugerman [13] discuss several GPU virtualization techniques, which can be roughly classified into software-based and pass-through. The software-based GPU virtualization is compatible with more

GPU hardware, while the pass-through approach achieves better performance. The software-based virtualization is more flexible and has been adopted by VMWare [13], and is used in prototyping optimization algorithms for GPU scheduling [45]. The pass-through approach can further be classified into: (i) one-to-one fixed pass-through and (ii) one-to-many mediated pass-through [13].

Until very recently, commercial products did not support mediated pass-through and refer to fixed one-to-one pass-through as pass-through. In fact, the performance of one-to-one fixed pass-through GPUs for cloud gaming has not been studied until late 2013 [38]. More precisely, Shea and Liu [38] conduct extensive experiments to quantify the performance gap between native hardware (without virtualization) and pass-through GPU (with virtualization). They find that the native hardware significantly outperforms pass-through GPU, and conclude that sharing a GPU among multiple cloud gaming VMs will lead to unacceptable low frame rate. Their measurement results also reveal that the low frame rate may be partially attributed to the excessive context switches. Our current paper extends Shea and Liu [38] in the sense that: (i) we consider modern GPUs that support more advanced pass-through approach, and (ii) we quantify the performance of GPUs shared by multiple VMs. To our best knowledge, the performance of these modern GPUs has not been measured in the literature.

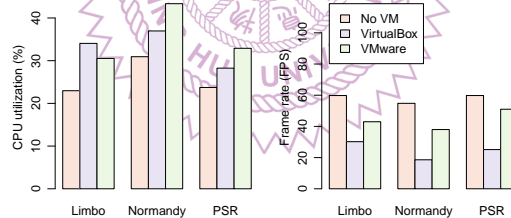


Figure 2.1: Virtualization overhead depends on game and VM implementations.

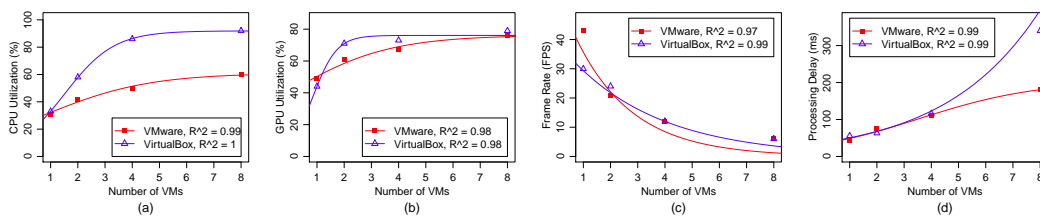


Figure 2.2: Measurement results for CPU utilization, GPU utilization, frame rate, and processing delay. Sample results from Limbo.

Chapter 3

Measurement Studies

We conduct measurement studies to model the implications of consolidating multiple cloud gaming servers on a physical machine. We set up the GA server [20] on VMware workstation 9 and VirtualBox 4.2.6. The GA client runs on another machine without VMs. The two Windows 7 machines running GA server and client are connected via a wired network, and they are equipped with Intel i7 3.4 GHz CPU and 24 GB memory, and Intel i5 2.8 GHz CPU and 4 GB memory, respectively. We install a NVidia Quadro 6000 GPU on the GA server. We choose three games in different genres: Limbo, Sudden Strike: Normandy (Normandy), and Police Supercars Racing (PSR), and measure various performance metrics over 5-min game sessions with different configurations. We consider four metrics relevant to the VM placement problem: (i) *CPU utilization*: the average CPU load measured on the physical server, (ii) *GPU utilization*: the average GPU load measured on the physical server, (iii) *frame rate*: the average number of frames streamed per second, and (iv) *processing delay*: the average time for the GA server to receive, render, capture, encode, and transmit a frame, which is measured by the techniques proposed in Chen et al. [2].

We first compare the performance of GA running on the host OS and that running on a single VM with all available resources allocated to it. Fig. 2.1 gives some sample results, which reveals that: (i) VMs lead to nontrivial overhead, (ii) different VMs result in different amount of overhead, and (iii) different games incur different workloads that may have distinct performance implications on different VMs. Hence, more extensive measurements are required to derive the prediction model of GA performance in each game/VM pair.

Next, we vary the number of VMs on the server, while equally dividing the 8 CPU cores among all VMs. In particular, we conduct the measurements with 1, 2, 4, and 8 VMs. We plot the sample results from Limbo in Fig. 2.2. This figure reveals that the CPU utilization, GPU utilization, frame rate, and processing delay can be modeled as

Table 3.1: R-square Values of Different Games/VM

Game	VM	CPU	GPU	FPS	DELAY
Limbo	VMware	0.9910	0.9837	0.9767	0.9955
	VirtualBox	1.0000	0.9877	0.9933	0.9996
Normandy	VMware	0.9999	1.0000	0.9865	0.9995
	VirtualBox	0.9991	0.9986	0.9764	0.9995
PSR	VMware	0.5758	0.9961	0.9917	0.9974
	VirtualBox	0.9898	0.9360	0.9969	0.9943

sigmoid functions of the number of VMs on a physical server, which are also plotted in Fig. 2.2 as the curves. We notice that several basic functions, such as $ax + b$, $ax^2 + bx + c$, a/x , and $a - a/x$ may also be used as the regression models [23]. After trying these basic functions, we find that the sigmoid functions fit our measurements much better. Therefore, we employ the sigmoid functions in this paper, and report their *R-square* values in Table 3.1. The R-square values indicate how close the sigmoid functions follow the real measurements: the deviation is smaller when the R-square value approaches 1. Hence, Table 3.1 shows that sigmoid functions model the VM measurement results very well.

The precise fitted sigmoid models are detailed in Ch. 4.2, and the empirically derived parameters are used in Chs. 6 and 7. We acknowledge that the model parameters depend not only on the pairs of game/VM but also on game server specifications and operating systems. This however is not a serious concern, as cloud gaming providers are likely to build data centers with one or very few types of machines, which can be profiled offline beforehand. In extreme cases where the physical servers are more heterogeneous, our measurement approach may adopt online regression for incremental adaptations.

Chapter 4

VM Placement Problem and Solution

We study the provider-centric problem in this section.

4.1 System Overview

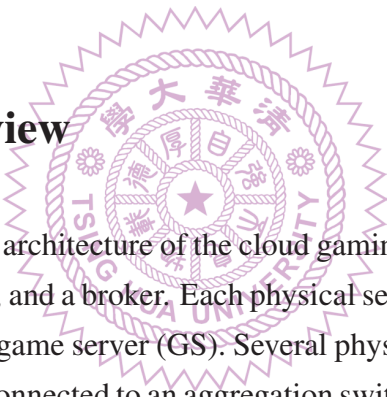


Fig. 1.1 illustrates the system architecture of the cloud gaming platform, which consists of S physical servers, P gamers, and a broker. Each physical server hosts several VMs, while every VM runs a game and a game server (GS). Several physical servers are mounted on a rack, and multiple racks are connected to an aggregation switch. The aggregation switches are then connected to the Internet via a core switch. Physical servers are distributed in several data centers at diverse locations. The gamers run game clients on desktops, laptops, mobile devices, and set-top boxes to access cloud games via the Internet.

The broker is the core of our proposal. The broker consists of a resource monitor and implements the VM placement algorithm. It is responsible to: (i) monitor the server workload and network conditions, and (ii) place the VMs of individual gamers on physical servers to achieve the tradeoff between QoE and cost that is most suitable to the cloud gaming service. In particular, for public cloud gaming services, the provider's profit is more important, while for closed cloud gaming services, the gaming QoE is more critical. We study the former case in this section, and will consider the later case in Ch. 5. The games may have diverse resource requirements, including CPU, GPU, and memory [8], while the paths between gamers and their associated servers have heterogeneous network resources, such as latency and bandwidth. Moreover, gamers can tolerate different QoE levels for different game genres [29]. Last, we note that the broker can be a virtual service running on a server or a server farm for higher scalability.

4.2 Notations and Models

Table 4.1 gives the symbols used in this paper. We study the VM placement problem, in which the VM placement decisions affect network delay, processing delay, and operational cost. We write the network delay between server s ($1 \leq s \leq S$) and gamer p ($1 \leq p \leq P$) as $e_{s,p}$ which is essentially the round-trip time between them. The $e_{s,p}$ values may be measured by various network diagnostic tools, such as Ping and King [17]. We use $f_p(v)$ and $d_p(v)$ to denote the frame rate and processing delay when serving gamer p with a server running v VMs, which depend on the game played by p . Fig. 2.2 reveals that sigmoid functions can model $f_p(v)$ and $d_p(v)$ well, and we write them as $f_p(v) = \frac{\alpha_{p,1}}{1+e^{-\alpha_{p,2}v+\alpha_{p,3}}}$ and $d_p(v) = \frac{\beta_{p,1}}{1+e^{-\beta_{p,2}v+\beta_{p,3}}}$, where $\alpha_{p,1}-\alpha_{p,3}$ and $\beta_{p,1}-\beta_{p,3}$ are model parameters derived from regression. Furthermore, we use $u_s(v)$ and $z_s(v)$ to model the CPU and GPU utilizations of server s running v VMs. Fig. 2.2 shows that $u_s(v)$ and $z_s(v)$ can also be written as sigmoid functions $u_s(v) = \frac{\delta_1}{1+e^{-\delta_2v+\delta_3}}$ and $z_s(v) = \frac{\zeta_1}{1+e^{-\zeta_2v+\zeta_3}}$, where $\delta_1-\delta_3$ and $\zeta_1-\zeta_3$ are the model parameters. We denote g_p as the hourly fee paid by gamer p . We let $w_s(v) = c_s(u_s(v) + z_s(v))$ be the operational cost of imposing CPU and GPU utilization $u_s(v)$ and $z_s(v)$ on s , where c_s is a cost term consisting of various components, such as electricity, maintenance, and depreciation. Moreover, we allocate G GB memory to each VM, whereas physical server s is equipped with G_s GB memory. Last, we consider GA servers to stream at B kbps. We let W be the number of data centers, and use \mathbf{S}_w ($1 \leq w \leq W$) to denote the set of servers in data center w . We let B_w be the uplink bandwidth of data center w ($1 \leq w \leq W$). Our bandwidth model is general, as the mapping between servers and data centers is flexible. For example, if the last-mile links are the bottleneck, we may create a *virtual* data center for each server, such that $|\mathbf{S}_w| = 1$, $\forall w$.

We next model the QoE of cloud gaming. Recent studies [25, 39] suggest that the response time of user inputs directly affects QoE levels. The response time $\tilde{d}_{s,p}(v)$ is the sum of processing delay, network delay, and playout delay. The playout delay is the time duration of receiving, decoding, and displaying a frame at the client. Since playout delay is not affected by VM placements, we do not include it in our model for brevity, and write $\tilde{d}_{s,p}(v) = d_p(v) + e_{s,p}$. We generalize the QoE models in Lee et al. [25, 39] to be a function of both response time and frame rate. More specifically, we let $q_p(f_p, \tilde{d}_{s,p})$ be the gaming QoE degradation observed by gamer p with frame rate f_p and response time $\tilde{d}_{s,p}$. Inspired by the linear QoE model in [25], we write $q_p(f_p, \tilde{d}_{s,p}) = \gamma_{p,1}f_p + \gamma_{p,2}\tilde{d}_{s,p}$, where $\gamma_{p,1}$ and $\gamma_{p,2}$ are model parameters that can be derived by the methodology presented in Lee et al. [25]. Last, we use Q_p to denote the maximal tolerable QoE degradation of gamer p .

```

1: for each gamer  $p = 1, 2, \dots, P$  do
2:   sort servers on network latency to  $p$  in asc. order
3:   for each server  $s = 1, 2, \dots, S$  do
4:     if serving  $p$  on  $s$  satisfies Eqs. (4.2)–(4.8) then
5:       let  $x_{s,p} = 1$ 
6:       break
7: return  $\mathbf{x}$ 

```

Figure 4.1: The pseudocode of the QDH algorithm.

4.3 Problem Formulation

We let $x_{s,p} \in \{0, 1\}$ ($1 \leq p \leq P, 1 \leq s \leq S$) be the decision variables, where $x_{s,p} = 1$ if and only if gamer p is served by a VM on server s . With the notations defined above, we formulate the provider-centric problem as:

$$\max \left[\sum_{p=1}^P \sum_{s=1}^S x_{s,p} g_p - \sum_{s=1}^S c_s \left(\frac{\delta_1}{1 + e^{-\delta_2 v_s + \delta_3}} + \frac{\zeta_1}{1 + e^{-\zeta_2 v_s + \zeta_3}} \right) \right] \quad (4.1)$$

$$\text{s.t. } f_p = \alpha_{p,1} / \left(1 + e^{-\alpha_{p,2} \sum_{s=1}^S (x_{s,p} v_s) + \alpha_{p,3}} \right), \quad \forall p; \quad (4.2)$$

$$\tilde{d}_p = \frac{\beta_{p,1}}{1 + e^{-\beta_{p,2} \sum_{s=1}^S (x_{s,p} v_s) + \beta_{p,3}}} + \sum_{s=1}^S e_{s,p} x_{s,p}, \quad \forall p; \quad (4.3)$$

$$v_s = \sum_{p=1}^P x_{s,p}, \quad \forall s; \quad (4.4)$$

$$1 = \sum_{s=1}^S x_{s,p}, \quad \forall p; \quad (4.5)$$

$$Q_p \geq \gamma_{p,1} f_p + \gamma_{p,2} \tilde{d}_p, \quad \forall p; \quad (4.6)$$

$$B_w \geq B \sum_{s \in \mathbf{S}_w} \sum_{p=1}^P x_{s,p}, \quad \forall w; \quad (4.7)$$

$$G_s \geq G \sum_{p=1}^P x_{s,p}, \quad \forall s; \quad (4.8)$$

$$x_{s,p} \in \{0, 1\}, \quad \forall 1 \leq s \leq S, 1 \leq p \leq P. \quad (4.9)$$

The objective function in Eq. (4.1) maximizes the provider's net profit, i.e., the difference between the collected fee and cost. Eqs. (4.2) and (4.3) derive the frame rate and response time as intermediate variables. In Eq. (4.4), we define another intermediate variable v_s to keep track of VMs on each server s , and we evenly allocate the cores among all VMs on a server. Eq. (4.5) ensures that each gamer is served by a single server. Eq. (4.6) makes sure that the gaming QoE degradation is lower than the user-specified maximal tolerant level. Eqs. (4.7) and (4.8) impose bandwidth and memory constraints on each data center and sever, respectively. In summary, the formulation maximizes the provider's profit while serving each gamer with a (user-specified) just-good-enough QoE level.

4.4 Proposed Algorithm

The provider-centric formulation in Eqs. (4.1)–(4.9) can be optimally solved using optimization solvers, such as CPLEX [10]. We refer to the solver-based algorithm as OPT. The OPT algorithm gives optimal solutions at the expense of exponential computation complexity. Therefore, we use OPT for benchmarking and propose an efficient heuristic algorithm, called Quality-Driven Heuristic (QDH), below.

The QDH algorithm is built upon an intuition: it is desirable to consolidate more VMs on a server as long as the user-specified maximal tolerate QoE degradation is not exceed. Fig. 4.1 illustrates the pseudocode of the QDH algorithm. For each gamer, the algorithm first sorts all servers on the network latency to that gamer. It then iterates through the servers in the ascending order and creates a VM for the gamer on the first server that can support this gamer without violating constraints in Eqs. (4.2)–(4.9). It is clear that the QDH algorithm runs in polynomial time.



Table 4.1: Symbols Used Throughout This Paper

Sym.	Description
S	Number of physical servers
P	Number of gamers
s	Index of a physical server
p	Index of a gamer
$e_{s,p}$	Round-trip time between physical server s and gamer p
v	Number of VMs running on physical server s
$f_p(v)$	Frame rate when serving gamer p with a server running v VMs
$d_p(v)$	Processing delay when serving gamer p with a server running v VMs
α	Frame rate model parameter
β	Processing delay model parameter
$u_s(v)$	CPU utilizations of server s running v VMs
$z_s(v)$	GPU utilizations of server s running v VMs
δ	CPU utilization model parameter
ζ	GPU utilization model parameter
g_p	Hourly fee paid by gamer p
$w_s(v)$	Operational cost of CPU and GPU
c_s	Cost term consisting of various components
G	Memory size of each VM
G_s	Memory size of physical server s
B	Streaming bit rate of GA server
W	Number of data centers
w	Index of a data center
S_w	Set of servers in data center w
$\tilde{d}_{s,p}(v)$	Sum of processing delay, network delay, and playout delay
q_p	Game QoE degradation
γ	QoE degradation model parameter
Q_p	Max tolerable QoE degradation of gamer p
$x_{s,p}$	Decision variable of the problem formulation
t_1	Start time of migration
t_2	Start time of synchronization before end of migration
t_3	End time of migration
D	Probability of each gamer joins (leaves) a gamer session
ω	Normalized migration overhead

Chapter 5

Alternative Formulation and Algorithms for Closed Systems

The provider-centric problem presented in Ch. 4 is suitable to public cloud gaming services. For closed cloud gaming services, e.g., in hotels, Internet cafes, and amusement parks, maximizing the overall QoE is more important as the network bandwidth is dedicated to cloud gaming. Therefore, we present the gamer-centric formulation and algorithms in this section. We start from the provider-centric formulation in Eqs. (4.1)–(4.9), and we first replace the objective function in Eq. (4.1) with:

$$\min \left[\sum_{p=1}^P \gamma_{p,1} f_p + \sum_{p=1}^P \gamma_{p,2} \tilde{d}_p \right], \quad (5.1)$$

which minimizes the total QoE degradation. In particular, the QoE degradation is reduced when f_p increase or d_p decreases as the empirically derived $\gamma_{p,1}$ is negative and $\gamma_{p,2}$ is positive. Next, we remove the constraints in Eq. (4.6) as the new objective function has taken the QoE into consideration. This yields the gamer-centric problem formulation. We develop a solver-based algorithm for the gamer-centric formulation, which is referred to as OPT'.

We also propose an alternative QDH for the gamer-centric problem, which is called QDH'. Fig. 5.1 illustrates the heuristic algorithm. For each gamer, the algorithm first computes its quality degradation levels on individual servers. It sorts the servers on the quality degradation if serving that gamer using each server. Then, the algorithm iterates through the servers and creates a VM for the gamer on the first server that can support the gamer without violating any constraints in Eqs. (4.2)–(4.5), (4.7)–(4.8). QDH' runs in polynomial time.

```

1: for each gamer  $p = 1, 2, \dots, P$  do
2:   sort servers on quality degradation  $q_p(\cdot)$  in asc. order
3:   for each server  $s = 1, 2, \dots, S$  do
4:     if serving  $p$  on  $s$  satisfies Eqs. (4.2)–(4.5), (4.7)–(4.8) then
5:       let  $x_{s,p} = 1$ 
6:       break
7: return  $\mathbf{x}$ 

```

Figure 5.1: The pseudocode of the QDH' algorithm.

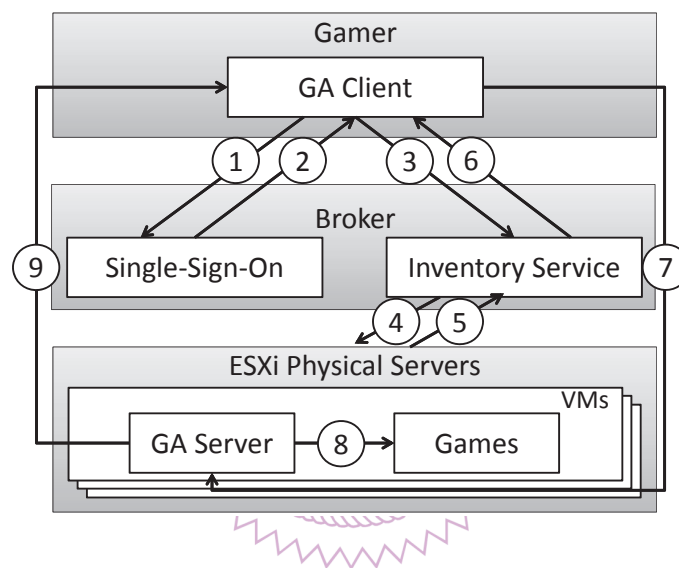


Figure 5.2: The implemented prototype system.



Figure 5.3: The cloud gaming testbed in our lab.

Chapter 6

System Implementation and Testbed

We conduct small-scale evaluations using a real testbed in the section.

6.1 Prototype Implementation

We have implemented a complete cloud gaming system consisting of a broker, physical servers, and GA servers/clients, as illustrated in Fig. 5.2. We adopt VMWare ESXi 5.1 as the virtualization software on physical servers. ESXi allows us to create VMs on physical servers, and each VM hosts a GA server and a game chosen by the corresponding gamer. We employ VMware vCenter 5.1 as the platform for our broker, which is comprised of Single-Sign-On for user authentication and Inventory Service for managing/monitoring the VMs on ESXi servers. The Inventory Service comes with different APIs, and we use its Java API to interface with the vCenter on the broker so as to control ESXi servers on all physical servers.

Fig. 5.2 shows the flow of our system. We integrate the GA client and server with VMware ESXi and vCenter. In particular, the GA client provides an interface for gamers to send their accounts and passwords to the broker (①). Upon being authenticated (②), the GA client sends the user-specified game to the broker, and the broker determines where to create a new VM for that game based on the status of all physical servers and networks (③). The broker then instructs the chosen physical server to launch a VM (④) and sends the VM's IP address to the GA client (⑤, ⑥). Last, the GA client connects to the GA server (⑦), instructs the GA server to run the user-specified game (⑧), and sends the stream of game to GA Client (⑨). This starts a new GA game session.

6.2 Testbed and Practical Concerns

We set up a testbed using the prototype system in our lab, which is shown in Fig. 8.1. The testbed contains an i7 3.2 GHz broker with the management web page, several i5 3.5 GHz physical servers with NVidia Quadro 6000 cards, and several i5 client computers. The broker, physical servers and client computers are connected via Gigabit Ethernet. We enable the CPU hardware support and conduct the following experiments.

We measure the overhead of launching a VM running Windows 7 and compare it against that of natively booting up Windows 7 on the same machine. We found out that both experiments take ~ 50 s, showing little additional overhead due to virtualization. We next measure the overhead of live migrations. Fig. 6.1 illustrates a sample migration of a gamer from the source VM to the destination VM, where the areas with virtual patterns indicate the VM currently used by the gamer. More generally, there are two types of migrations: (i) live migration and (ii) stop-and-copy [7]. With live migration, the memory and disk pages of the source VM are first copied over to the destination VM. Meanwhile, the gamer still connects to the source VM and may produce *dirty* memory and disk pages. The system iteratively copies those dirty pages to the destination VM until either there is no more dirty pages, or the number of new dirty pages is more than the number of copied dirty pages. Next, the *synchronization* starts: (i) the system freezes both VMs, (ii) the system copied over the remaining pages, and (iii) the gamer is then served by the destination VM. We let t_1 and t_3 be the start and end times of the migration procedure, and t_2 be the time synchronization starts. Using the notations, live migration copies pages between t_1 and t_2 without stopping gamers from using the VMs, and freezes VMs between t_2 and t_3 . Our testbed supports live migration and we conduct diverse experiments to quantify the migration overhead.

We discover that the live-migration time t_1 to t_3 of 20, 30, and 40 GB VM images are about 6, 9, and 11 minutes in our testbed. In addition, the frozen time t_2 to t_3 are always less than 3 seconds. These three various VM image sizes are roughly mapped to the three considered games: Limbo, PSR, and Normandy. Given that the migration time are

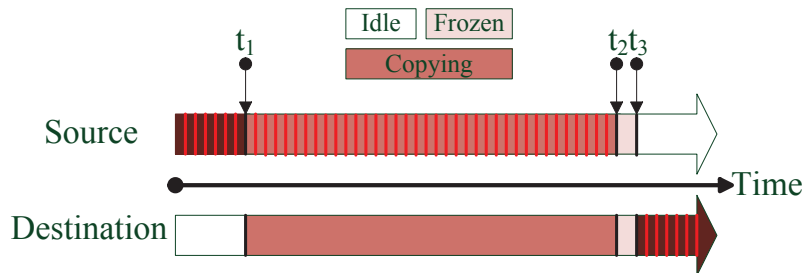


Figure 6.1: Live migration.

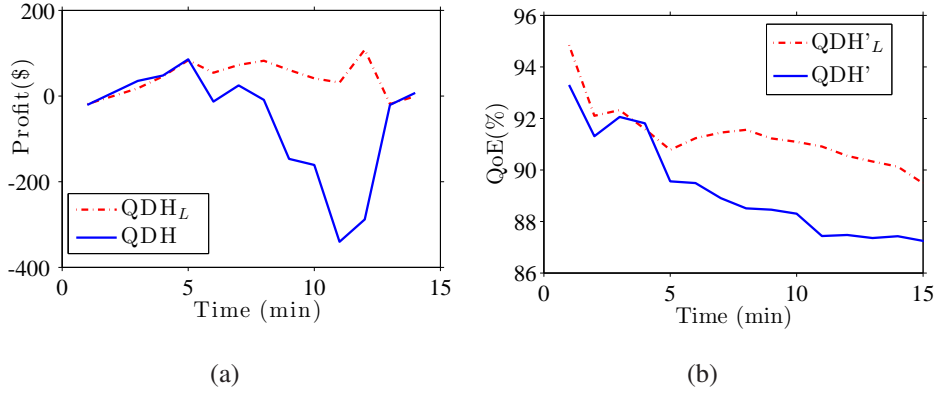


Figure 6.2: Comparisons between QDH_L/QDH'_L and QDH/QDH': (a) net profits and (b) quality.

non-trivial, recomputing the VM placement problems for all gamers (including those with ongoing sessions) may lead to unacceptable QoE degradation even with live migration. The major cause of the QoE degradation is the duplicated resource reservations: when migrating a gamer, both the source and destination VMs consume resources as shown in Fig. 6.1. Hence, we propose an *migrationless* version of the proposed QDH/QDH' algorithms, which do not migrate the running VMs to avoid the degradation caused by migration time. We denote the new algorithms as QDH_L/QDH'_L, which only intelligently place the VMs of incoming gamers that have not started the game sessions. That is, by getting rid of the outermost loops in Figs. 4.1 and 5.1, we never migrate the ongoing game sessions. Intuitively, QDH_L/QDH'_L run faster, yet achieve better performance as the high migration time are avoided. Moreover, QDH'_L is an optimal migrationless algorithm. We will show this in evaluation sections (Chs. 6.3 and 7).

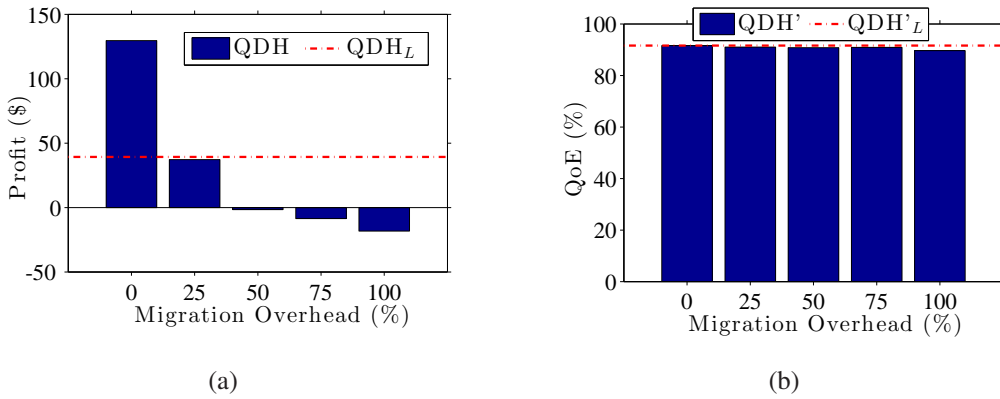


Figure 6.3: Comparisons between QDH_L/QDH'_L and QDH/QDH': (a) net profits and (b) quality.

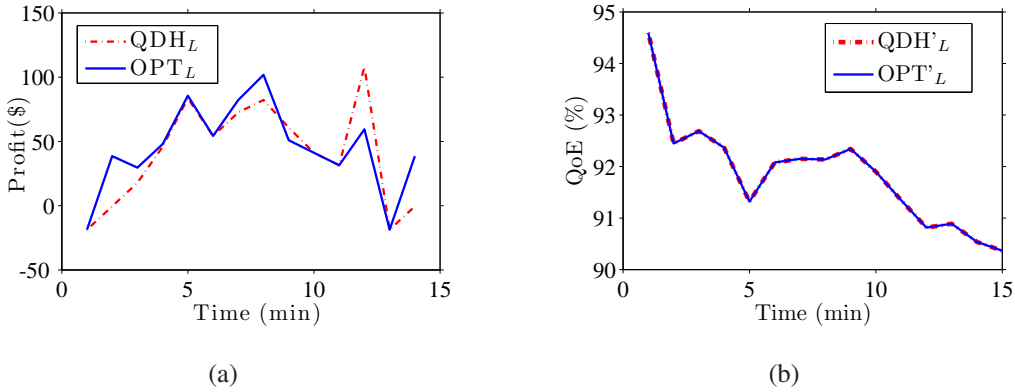


Figure 6.4: Comparisons between QDH_L/QDH'_L and OPT_L/OPT'_L: (a) net profits and (b) quality.

6.3 Experiment–Performance Gains of the Migrationless Algorithms

Setup. To quantify the QDH_L/QDH'_L algorithms, we employ a testbed with 9 physical servers, 15 gamers, and 3 games—Limbo, PSR, and Normandy. In every minute, each gamer joins (leaves) a game session with a probability of $D\%$ ($1 - D\%$), where D is a system parameter. Each simulation lasts for T minutes. We assume that each physical server can serve up to two VMs and each VM launches a randomly selected game. In each simulation, we measure the fps and processing delay, and use them in the quality model. Also, we measure the CPU and GPU utilizations, and use them in the profit model. We inject realistic network latency (see Sec. 7.1) using dummynet [14]. Last, we set $D = 90\%$, $T = 15$ minutes and consider the two performance metrics:

- *Net profit.* The total provider profit in every minute.
- *Quality of Experience.* The gaming QoE normalized in the range of $[0\%, 100\%]$.

Results. We make three observations on the performance of the QDH_L/QDH'_L algorithms. First, QDH_L/QDH'_L outperform QDH/QDH'. In particular, Figs. 6.2(a) and 6.2(b) show that the gains between QDH_L/QDH'_L and QDH/QDH' are up to 396 dollars and 4% QoE. A closer look reveals that the performance gains are due to high migration overhead.

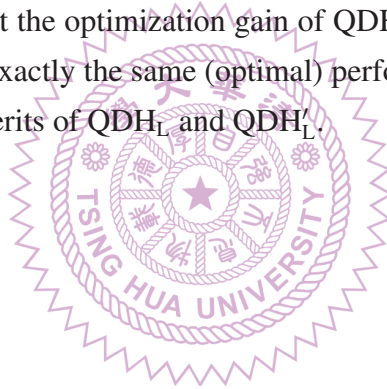
Due to the increasingly higher computing power, the migration overhead will be gradually reduced and the performance gains of QDH_L/QDH'_L may be diminishing. To better understand the trend, we let ω be the normalized migration overhead, where $0 \leq \omega \leq 1$. For example, setting $\omega = 1/3$ means the migration overhead becomes 1/3 of the current one. We vary different ω values and plot the average results in Fig. 6.3. This figure reveals that the migrationless algorithms QDH_L/QDH'_L still outperform the ordinary algorithms

QDH/QDH' even when $\omega = 25\%$ and $\omega = 5\%$. However, such a steep technology advance is less likely to become a reality in the short term. Hence, we no longer consider the QDH/QDH' algorithms in the rest of this thesis.

Last, we compare the QDH_L/QDH'_L algorithms against the migrationless optimal solution that exhaustively checks all servers for each new gamer. We refer to the migrationless optimal solutions as OPT_L/OPT'_L. Fig. 6.4 reports the average performance over time. Fig. 6.4(a) shows that QDH_L and OPT_L result in similar net profit.

More specifically, the OPT_L algorithm outperforms the QDH_L algorithm in the first half of the experiment, but the QDH_L occasionally performs better in the second half. A closer look indicates that since both algorithms are migrationless, once game sessions start, they will be executed until the gamers leave. Therefore, even though OPT_L selects the best VM placements for the *incoming* gamers, it cannot foresee the future (e.g., when will the gamers leave), and thus its profit may be lower than that of the QDH_L algorithm. Nonetheless, the overall profit of QDH_L is still 10% lower than the optimum.

A closer look depicts that the optimization gain of QDH_L is merely 10%. Fig. 6.4(b) reveals that QDH'_L leads to exactly the same (optimal) performance in QoE, compared to OPT'_L. Fig. 6.4 shows the merits of QDH_L and QDH'_L.



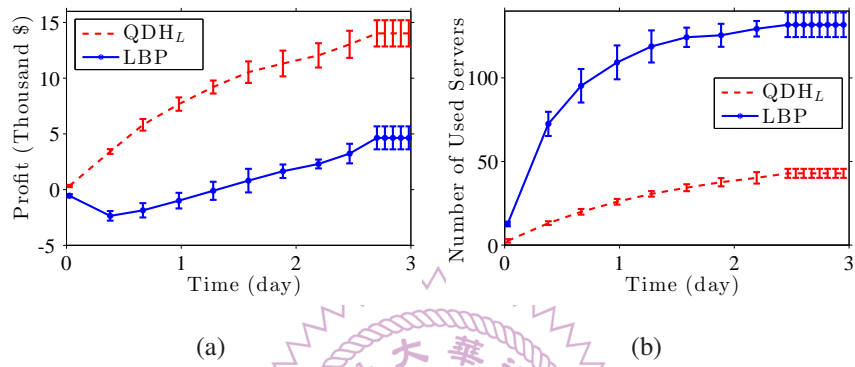


Figure 6.5: Provider-centric simulation results with synthetic traces: (a) net profits and (b) used servers.

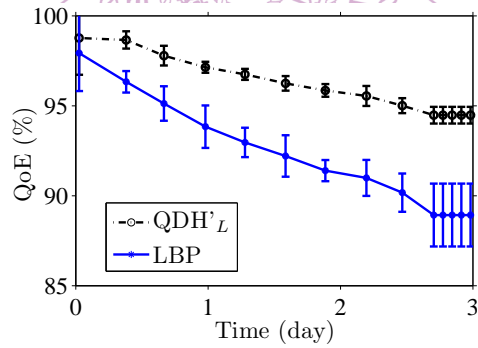


Figure 6.6: Gamer-centric simulation results with synthetic traces.

Chapter 7

Trace-Driven Simulations

In this section, we consider large-scale evaluations using detailed simulations.

7.1 Setup

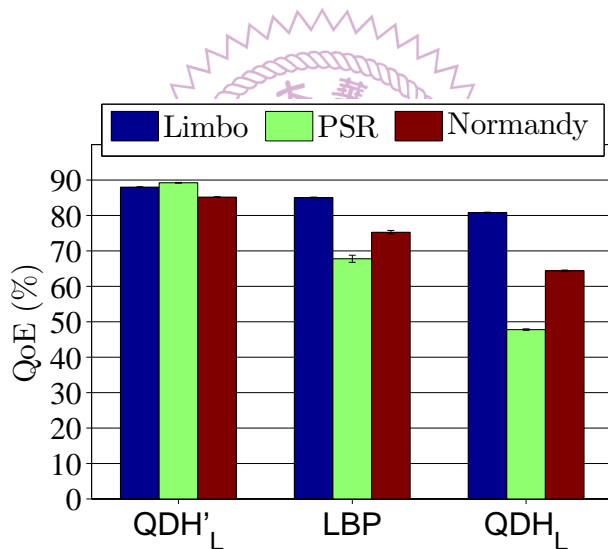


Figure 7.1: Fairness in QoE levels on different game genres.

We have built a simulator for the VM placement problem using a mixture of C/C++, Java, and Matlab. We have implemented the QDH_L/QDH'_L algorithms in our simulator. For comparisons, we have also implemented a VM placement algorithm that places each VM on a random game server that is not fully loaded and in the data center geographically closest to the gamer. This baseline algorithm is referred to as Location Based Placement (LBP) algorithm. We collect gamer and server IP addresses and the latency between each gamer/server IP pair in order to drive our simulator. For servers, we use DigSites-Value [11] to obtain the IP addresses of OnLive data centers in Virginia, California, and Texas. For gamers, we develop a BitTorrent crawler using libtorrent [27] to collect peer IP addresses and then use them as gamer IP addresses. Since OnLive only hosts game

servers in the US, we filter out non-US gamer IP addresses using ip2c [21]. We ran our crawler on August 13, 2013 with 4494 torrents downloaded from IsoHunt [22], which gave us 22395 IP addresses and 5875 US IP addresses. Next, we measure the network latencies among gamer/server IP pairs using King [17], since we have no control over neither end systems. We drop the IP addresses without complete latency results to all servers, which leads to 412 gamer IP addresses.

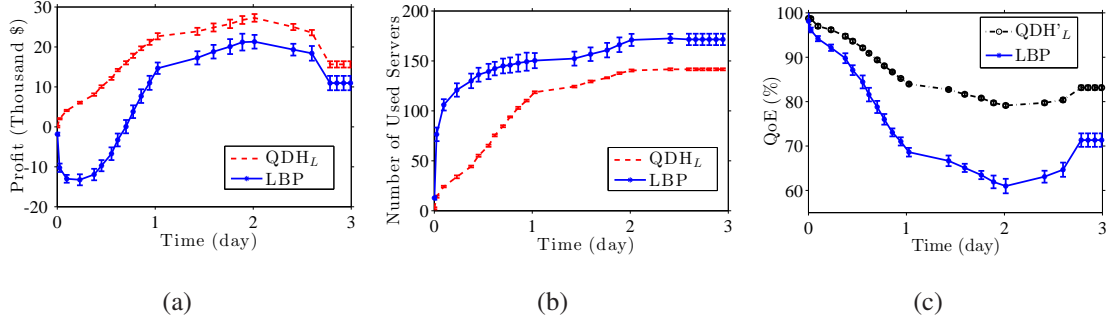


Figure 7.2: Simulation results with WoW traces: (a) net profits, (b) used servers, and (c) QoE levels.

We conduct a three-day simulation for each scenario using different algorithms. The gamers arrive at the broker following a Poisson process with a mean time interval of 4 minutes and each gamer plays for a duration uniformly chosen from $\{300, 600, 1200, 2400, 4800\}$ minutes. In addition to the synthetic gamer arrival traces, we also employ real World of Warcraft (WoW) traces [41] in our simulations. Each gamer plays a game randomly chosen from Limbo, PSR, and Normandy. We also vary the number of servers $S \in \{192, 384, 768, 1536, 3072\}$ and the migration overheads of 6, 9, and 11 minutes for Limbo, PSR, and Normandy respectively. During each simulation, we run the scheduling algorithm once every minute and we report the mean performance results among all gamers, and 95% confidence intervals whenever applicable. If not otherwise specified, we set $S = 192$, $\gamma_{p,1} = -0.1$, $\gamma_{p,2} = 0.1$, $g_p = 1$, and $c_s = 2$. We conduct all the simulations on an Intel i7 3.4 GHz PC. We consider the following performance metrics:

- *Net profit.*
- *Quality of Experience.*
- *Running time.* The time of executing each algorithm.
- *Number of used servers.* The number of servers that serve at least one gamer.

7.2 Results

Performance of QDH_L/QDH'_L. We plot the provider-centric results in Fig. 6.5(a), which

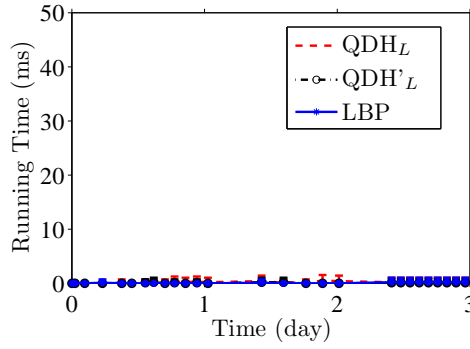


Figure 7.3: Running time of VM placement algorithms.

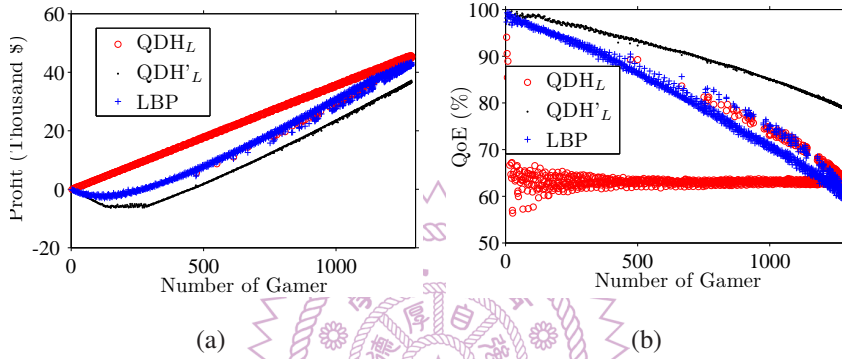


Figure 7.4: Impacts of number of gamers on: (a) net profits and (b) QoE levels.

shows that $QDHL_L$ significantly outperforms LBP: up to 3.5 times difference. This can be explained by Fig. 6.5(b), which shows that $QDHL_L$ turns on fewer servers to achieve higher net profits. We plot the gamer-centric results in Fig. 6.6, which reveals that $QDHL'_L$ constantly outperforms LBP: up to 5% QoE gap. Moreover, the confidence intervals show that $QDHL'_L$ leads to more consistent QoE levels among individual gamers, achieving better *fairness*. Fig. 7.1 plots the aggregate QoE of three games, which shows that both $QDHL'_L$ and LBP are relatively fair to different game genres, while $QDHL_L$ maximizes the net profits by devoting more resources to less complicated games.

Performance results from WoW traces. In the following, we report results from the WoW traces. We plot the provider-centric results in Fig. 7.2(a), which shows that $QDHL_L$ always outperforms LBP with the difference between the two algorithms up to 20+ thousand dollars. And in the first quarter of the simulation, LBP runs into a big deficit problem. This can be explained by Figs. 7.2(b) and 7.4(b), which reveals that while $QDHL_L$ shutdown more servers and it always allow all gamers meets 60+% QoE which is the just-good-enough QoE level. Fig. 7.2(c) shows that $QDHL'_L$ outperform LBP up to 130%.

Scalability. We plot the running time in Fig. 7.3, which shows the $QDHL_L/QDHL'_L$

Table 7.1: Running Time in Seconds

# of Servers	QDH_L		QDH'_L	
	Mean	Max	Mean	Max
5000	0.215	0.853	0.02	0.05
10000	0.379	0.967	0.05	0.07
15000	0.557	1.9	0.07	0.12
20000	0.819	2.52	0.12	0.23

algorithms terminate in real time: < 1.5 ms. We then increase the number of servers S , and report the average running time with two assumptions that all gamers which we get from WoW trace will not leave the game session and we repeat the WoW trace 30 times to scale up the number of total gamers in our system. Table 7.1 shows that it takes QDH_L/QDH'_L at most 2.5s to solve a VM placement problem with more than 20000 servers and 40000 gamers. This is relatively short compared to the initialization time of modern computer games.

Number of gamers. The number of gamers in WoW traces is varying in time, and we present two scatter plots in Figs. 7.4(a) and 7.4(b) to study the relation between the performance and number of gamers. This figures show that more gamers lead to higher profits and lower QoE levels, and QDH_L/QDH'_L successfully achieve their design objectives.

Chapter 8

GPU Consolidation for Cloud Games

We have formulated the VM placement problem and proposed algorithms to maximize cloud providers' profit while achieving just-good-enough QoE for gamers, or maximize the QoE for gamers without consider the profit. After that, in this chapter, we conduct experiments using a modern GPU and a cloud gaming platform to answer the following question: *Are modern GPUs ready for cloud gaming?*

8.1 Methodology

In this section, we present the methodology to measure the performance of modern GPUs.

8.1.1 Workload Generators

We generate GPU workload using two kinds of applications within Windows 7 running as the guest OS. The details are given below.

- **Game.** Three games are chosen from all game genres: Fear2, which is a first-person shooter game, LEGO Batman, which is an action game, and Limbo, which is a scroll-based puzzle game.
- **Benchmark.** We use Sanctuary for overall GPU benchmark and Cadalyst for detailed (2D versus 3D) GPU benchmark.

We use `tinytask` to record the mouse and keyboard inputs of each game play for 3 minutes. We then replay the same user inputs in multiple experiments with different configurations to ensure fair comparisons. The game graphics detail levels are kept as default. The Sanctuary benchmark gives an FPS number as the overall score. The Cadalyst benchmark gives four 2D scores on ortho lines, radial lines, texts/blocks, and erase/zoom, and we denote them as $2D_1$, $2D_2$, $2D_3$, and $2D_4$ in the figures. The Cadalyst benchmark also

gives four 3D scores on rotate wireframe, rotate hidden, rotate conceptual, and rotate realistic, which are referred to as $3D_1$, $3D_2$, $3D_3$, and $3D_4$ in the figures. We set the resolution to 1920x1200.

Table 8.1: Specifications of Two GPUs.

GPU	Year	Core	Memory	No. Inst.	vSGA	vGPU
Quadro 6000	2010	448	6 GB	1	Yes	No
K2	2013	3072	8 GB	2	Yes	Yes

8.1.2 Experiment Setup

We setup a XenServer 6.2 with a Xeon 2.1 GHz 16-core CPU and 64 GB memory for the cloud gaming server. We conduct experiments with two GPU cards: Nvidia Quadro 6000 (released in 2010) and Nvidia K2 (released in 2013). Their specifications are given in Table 8.1. The K2 GPU has two physical GPU instances, and each instance can be configured to be in one of the following modes: (i) pass-through, (ii) vGPU with up to 8 VMs, (iii) vGPU with up to 4 VMs, and (iv) vGPU with up to 2 VMs. vGPU is the latest Nvidia GPU virtualization technique realizing mediated pass-through [13]. We refer to these modes as: PassThrough, vGPU₂, vGPU₄, and vGPU₈, respectively. Quadro 6000 GPU does not support vGPU and only works with vSGA, which is software-based GPU sharing. By default, the XenServer allocates 1 CPU core and 2GB memory to Dom0, which is responsible for managing VMs. The remaining CPU cores and memory are equally divided among the VMs running Windows 7.

8.1.3 Performance Metrics

In addition to the scores given by Sanctuary and Cadalyst benchmark, we also consider the following performance metrics.

- **FPS.** The number of rendered frames per second.
- **Context Switch.** The number of context switches in Dom0.
- **CPU Utilization.** The CPU load of Dom0 (CPU_{dom0}) and each VM (CPU_{vm}).
- **GPU Utilization.** The load of GPUs.

8.1.4 Measurement Utilities

XenServer is based on CentOS Linux, and Linux utilities may be compiled on XenServer. For example, we tried to compile PAPI and Perf. However, they do not work on XenServer

even after being compiled, because the XenServer kernel has been patched. We end up with the following utilities.

- **Fraps.** To measure the FPS of the foreground window.
- **Sar.** To measure the number of context switches.
- **Xentop.** To measure the CPU utilization of Dom0 and VMs.
- **Nvidia-smi.** To measure the GPU utilization under vGPU.
- **GPU-Z.** To measure the GPU utilization of pass-through GPUs.



Figure 8.1: Our testbed with a cloud game server (right) and four game clients (left), connected by a Fast Ethernet switch (middle).

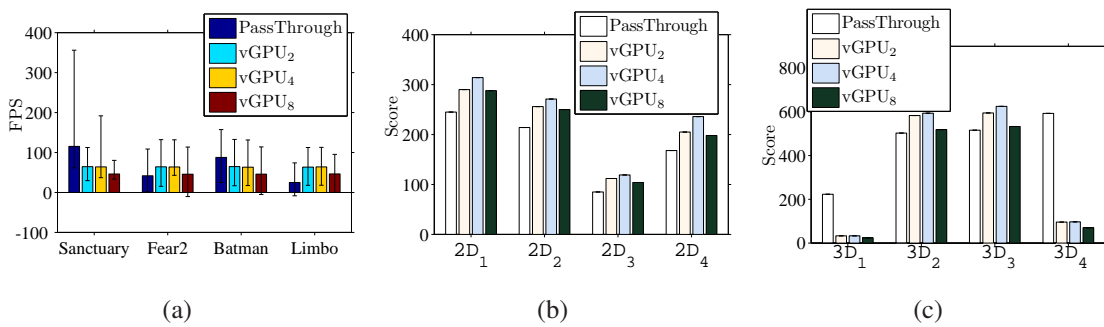


Figure 8.2: Comparing the pass-through and vGPU: (a) resulting FPS, (b) 2D benchmark scores, and (c) 3D benchmark scores.

8.2 Measurement Results

Performance edge and scalability of vGPU. We set up a cloud gaming testbed [19] as illustrated in Fig. 8.1, and compare the performance of Quadro 6000 (software-based

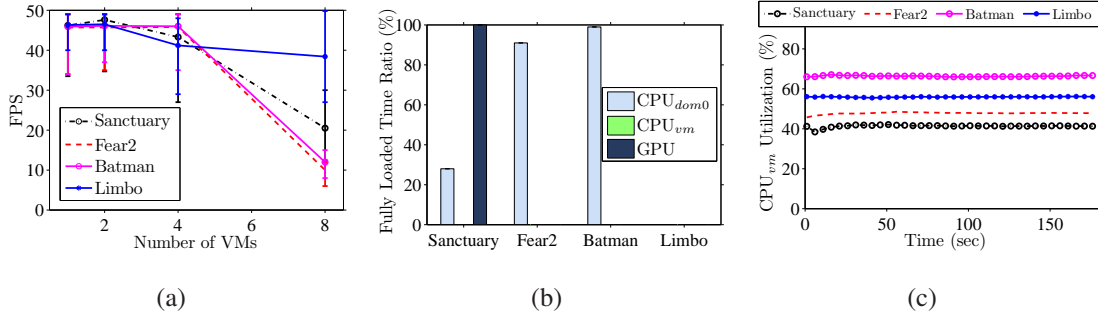


Figure 8.3: GPU consolidation overhead: (a) resulting FPS with various numbers of VMs, (b) fully loaded time ratio from vGPU₈, and (c) CPU_{vm} from vGPU₈.

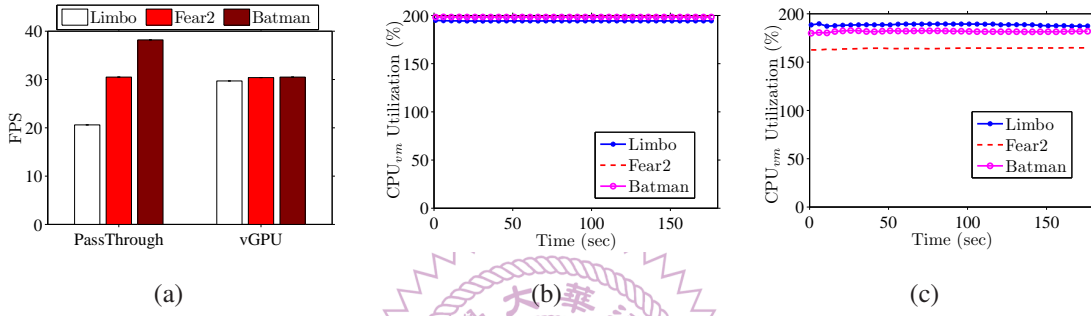


Figure 8.4: End-to-end performance of a cloud game platform: (a) resulting FPS, (b) CPU_{vm} utilization with pass-through GPU, and (c) CPU_{vm} utilization with vGPU₂.

vSGA) and K2 (mediated pass-through vGPU). We report the achieved average FPS from Limbo on the clients in Table 8.2. This table shows that K2 outperforms Quadro 6000 with up to 3.87 times of FPS increases. Furthermore, the FPS achieved by K2 does not drop too much even with 8 VMs, which shows its scalability. This experiment demonstrates the huge edge of vGPU (mediated pass-through) over vSGA (software-based virtualization). Hence, we no longer consider Quadro 6000 and vSGA in the rest of this thesis.

Independence of the two K2 GPU instances. We use Sanctuary to measure the FPS of pass-through, vGPUs, and mixed configurations. To isolate the performance measurement on GPUs, we conduct the next 4 measurements on the cloud game server. We report the average results in Table 8.3, which shows that the two GPU instances of K2 operate independently, as the FPS values are pretty stable. Without loss of generality, we only enable a GPU instance in the remaining experiments.

Table 8.2: Achieved frame rates on two considered GPUs

# of VMs	Quadro 6000	K2	Speed-up (times)
2 VMs	22.3	32.8	1.47
4 VMs	13.1	26.9	2.05
8 VMs	7.0	27.1	3.87

Table 8.3: Sanctuary Scores in FPS from Diverse GPU Configurations

GPU Configuration	PassThrough	vGPU
1 PassThrough on 1 Instance	150.4	x
2 PassThrough on 2 Instances	146.6	x
1 PassThrough + 4 vGPU ₄	142.7	45.9
4 VMs with vGPU ₄ on 1 Instance	x	42.7
8 VMs with vGPU ₄ on 2 Instances	x	42.8

Shared GPUs may outperform dedicated GPUs. We next compare the performance of pass-through, vGPU₂, vGPU₄, and vGPU₈ using Sanctuary, Fear2, Batman, and Limbo, where pass-through refers to one-to-one fixed pass-through. We plot the resulting FPS in Fig. 8.2(a). This figure reveals a surprising observation: vGPU results in higher FPS than pass-through when executing Limbo and Fear2. This is different from the common belief. We therefore take a step further by running Cadalyst and reporting the 2D and 3D scores in Figs. 8.2(b) and 8.2(c). These two figures show that vGPU₂ outperforms pass-through in all 2D operations by up to 15%. Moreover, vGPU₂ also leads to better scores than pass-through when executing some 3D operations (two out of four scores). Similar observations are also true for vGPU₄ and vGPU₈. Such observations explain the inferior FPS of pass-through GPUs: Limbo and Fear2 heavily rely on 2D operations; in contrast, the 3D-intensive Batman performs better on pass-through GPUs. The take-away of this experiment is that state-of-the-art vGPU virtualization technique has been well optimized and works better than pass-through GPUs for some game genres. In fact, we believe that *sharing a GPU among multiple VMs running games is now a reality*.

Table 8.4: Relation Between FPS and Number of Context Switches

Game	FPS			No. Context Switches		
	vGPU ₈	vGPU ₄	Ratio	vGPU ₈	vGPU ₄	Ratio
Fear2	45.8	64.9	0.7	9472	14149	0.67
Batman	43.3	41.6	1.04	4325	3991	1.08
Limbo	39.2	64.8	0.6	10700	13927	0.76

The performance of modern GPUs is no longer dominated by the overhead due to context switches. An earlier study [38] reports that more context switches incur higher overhead and reduce the rendered FPS. We try to reproduce this by running three games on vGPU₄ and vGPU₈, and measure the resulting FPS and number of context switches in Dom0. We report the results in Table 8.4, which shows that the FPS is proportional to the number of context switches. That is, more context switches indicate that VMs are busier rendering at higher FPS. This is quite different from the earlier study [38]. Another interesting observation is that vGPU never results in FPS > 68 throughout our experiments. This shows that an *FPS-aware* GPU scheduling algorithm, similar to Zhang

et al. [45], has been implemented in vGPU, making it suitable for sharing GPUs among VMs running cloud games.

Consolidation overhead and root cause analysis. Next, we quantify the consolidation overhead by configuring the K2 GPU into vGPU₈, and gradually adding more VMs (from 1 to 8). We then measure the FPS, CPU_{dom0}, CPU_{vm}, and GPU utilization under Sanctuary and different games. We plot the resulting FPS in Fig. 8.3(a). This figure shows that Limbo does not suffer from consolidation overhead, while all other games/benchmark do. We then compute the fraction of time each resource is fully loaded, and refer to it as *fully loaded time ratio* in %. We plot the sample ratio from vGPU₈ in Fig. 8.3(b), which shows that Sanctuary is bounded by GPU, while Fear2 and Batman are bounded by CPU_{dom0}. Moreover, as emphasized in Fig. 8.3(c), the CPU_{vm} never reaches 100% under vGPU₈. Our observations show that *allocating more CPU cores to Dom0 in XenServer will alleviate the high consolidation overhead for more complex games.*

Importance of hardware codecs. Last, we measure the end-to-end cloud gaming performance. We configure the server for pass-through and vGPU₂, assign 8 CPU cores to each VM, and measure the rendered FPS at clients and the CPU utilization at the server. We report the resulting FPS values in Fig. 8.4(a), which are between 20 and 42. The FPS results are less ideal to high-quality cloud gaming, and a closer look indicates that this is because of limitations of XenServer and Windows 7. In particular, because the cloud game server [19] relies on the CPUs for real-time video encoding, more CPU cores mean higher encoding speed. However, the free version of XenServer only supports exposing multiple virtual CPUs, rather than CPU cores, to each guest OS, and Windows 7 supports up to 2 CPUs. Hence, the guest Windows 7 only schedules the tasks to 2 CPUs while leaving the other 6 CPUs idle (see Figs. 8.4(b) and 8.4(c)), which renders lower FPS values. We note that K2 GPUs come with hardware H.264 codecs, and how to leverage these codecs for higher cloud gaming FPS is an interesting future task.

Chapter 9

Conclusion and Future Work

We studied the VM placement problems for maximizing: (i) the total net profit for service providers while maintaining just-good-enough gaming QoE, and (ii) the overall gaming QoE for gamers. The former problem is more suitable for public cloud gaming systems, while the later problem is more suitable for closed systems. We conducted extensive experiments using a real cloud gaming system [20], and two VMs to derive various system models. We formulated the two problems as optimization problems, and proposed optimal and efficient algorithms to solve them. After that, we designed and carried out a measurement study to understand whether the state-of-the-art GPU virtualization techniques are ready for cloud gaming. Different from earlier measurement studies [13, 38], we have found that the mediated pass-through GPU virtualization implemented in the latest GPUs enables efficient GPU sharing among multiple VMs.

Via testbed, extensive trace-driven simulations, and a measurement study of modern GPU, we demonstrate that: (i) the efficient algorithms achieve up to 90% (provider-centric) and 100% (gamer-centric) performance compared to the optimal algorithms, (ii) the efficient algorithms constantly outperform the state-of-the-art algorithm, e.g., up to 3.5 times in net profits, (iii) the efficient algorithms terminate in < 2.5 s on a commodity PC even for large services with 20000 servers and 40000 gamers, (iv) shared GPUs may outperform dedicated GPUs, and (v) they are rather scalable to the number of VMs. Hence, modern GPUs can be shared by VMs running GPU-intensive computer games.

This work can be extended in several directions: (i) we may develop more comprehensive system models, which may take other types of resources and heterogeneous server types into consideration, and support online parameter adaptation. (ii) we also observed that CPUs may become the bottleneck on cloud gaming servers, e.g., the cloud gaming platform [19] achieves less ideal FPS because the guest Windows 7 can only utilize up to 2 CPU cores, and leveraging the hardware H.264 codecs becomes crucial to build commercially-viable cloud gaming platforms. (iii) We may consider the individual QoE

of the gamers while we maximize the total QoE. (iv) About the higher performance of vGPU, we could analyze not only the 2D/3D operations but also the low-level instructions to know precise factors which result in the observation. (v) in the VM replacement algorithm, we may consider the gamer requirements to make the results more close our objective. (vi) The shared network storage may reduce the migration time. The most important future work is the six point. For our observation, if we can speed up 4 times of the total live migration time, the performance of QDH will be the same as the migrationless algorithm. We may leverage the shared network storage while we migrate the VMs in the same datacenter. With shared network storage, we can only migrate the memory instead the whole image of the VMs . If we allocate 4GB memory and 20GB disk to each VM, we can speed up 5 times with the memory-only migration skill. It will make our QDH algorithms become useful and outperform the migrationless algorithms.



Bibliography

- [1] F. Bari, R. Boutaba, R. Esteves, M. Podlesny, G. Rabbani, Q. Zhang, F. Zhani, and L. Granville. Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, 15(2):909 – 928, 2012. Accepted to appear.
- [2] K. Chen, Y. Chang, P. Tseng, C. Huang, and C. Lei. Measuring the latency of cloud gaming systems. In *Proc. of ACM International Conference on Multimedia (MM'11)*, pages 1269–1272, Scottsdale, AZ, November 2011.
- [3] M. Chen, H. Zhang, Y. Su, X. Wang, G. Jiang, and K. Yoshihira. Effective VM sizing in virtualized data centers. In *Proc. of International Symposium on Integrated Network Management (IM'11)*, pages 594–601, Dublin, Ireland, May 2011.
- [4] P. Chen and M. Zark. Perceptual view inconsistency: An objective evaluation framework for online game quality of experience (QoE). In *Proc. of the Annual Workshop on Network and Systems Support for Games (NetGames'11)*, pages 2:1–2:6, Ottawa, Canada, October 2011.
- [5] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, April 2011.
- [6] N. Chowdhury, M. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. of IEEE INFOCOM 2009*, pages 783–791, Rio de Janeiro, Brazil, April 2009.
- [7] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machine. *Symposium on Networked Systems Design and Implementation*, 2:273–286, 2005.
- [8] M. Claypool. Motion and scene complexity for streaming video games. In *Proc. of the International Conference on Foundations of Digital Games (FDG'09)*, pages 34–41, Port Canaveral, FL, April 2009.

- [9] Cloud gaming adoption is accelerating ... and fast! <http://www.nttcom.tv/2012/07/09/cloud-gaming-adoption-is-acceleratingand-fast/>.
- [10] IBM ILOG CPLEX optimizer. <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>.
- [11] DigSites. <http://digsitesvalue.net/s/onlive.com>.
- [12] Distribution and monetization strategies to increase revenues from cloud gaming. <http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf>.
- [13] M. Dowty and J. Sugerman. GPU virtualization on VMware's hosted I/O architecture. *ACM SIGOPS Operating Systems Review*, 43(3):73–82, July 2009.
- [14] DummyNet. <http://info.i.et.unipi.it/~luigi/dummynet/>.
- [15] T. Duong, X. Li, R. Goh, X. Tang, and W. Cai. QoS-aware revenue-cost optimization for latency-sensitive services in IaaS clouds. In *Proc. of IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications (DS-RT'12)*, Dublin, Ireland, October 2012.
- [16] T. Ferreto, M. Netto, R. Calheiros, and C. Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, October 2011.
- [17] K. Gummadi, S. Saroiu, and S. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *Proc. of ACM SIGCOMM Internet Measurement Workshop (IMW'02)*, pages 5–18, Boston, MA, November 2002.
- [18] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu. QoS-aware virtual machine placement for cloud games. In *Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames'13)*, Denver, CO, December 2013.
- [19] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu. GamingAnywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10(1s):10:1–10:25, January 2014.
- [20] C. Huang, C. Hsu, Y. Chang, and K. Chen. Gaminganywhere: An open cloud gaming system. In *Proc. of the ACM Multimedia Systems Conference (MMSys'13)*, pages 36–47, Oslo, Norway, February 2013.

- [21] ip2c. <http://firestats.cc/wiki/ip2c>.
- [22] IsoHunt. <http://isohunt.com/>.
- [23] M. Kutner, C. Nachtsheim, J. Neter, and W. Li. *Applied linear statistical models*. McGraw-Hill Higher Education, 5th edition, 2004.
- [24] Y. Lee and K. Chen. Is server consolidation beneficial to MMORPG? a case study of World of Warcraft. In *Proc. of IEEE International Conference on Cloud Computing (CLOUD'10)*, pages 435 – 442, Miami, FL, February 2010.
- [25] Y. Lee, K. Chen, H. Su, and C. Lei. Are all games equally cloud-gaming-friendly? an electromyographic approach. In *Proc. of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE'05)*, pages 117–124, October 2012.
- [26] Y. Li, W. Li, and C. Jiang. A survey of virtual machine system: Current technology and future trends. In *Proc. of International Symposium on Electronic Commerce and Security (ISECS'10)*, pages 332–336, Guangzhou, China, July 2010.
- [27] libtorrent. <http://www.rasterbar.com/products/libtorrent/>.
- [28] J. Lin, C. Chen, and J. Chang. QoS-aware data replication for data-intensive applications in cloud computing systems. *IEEE Transactions on Cloud Computing*, 1(1):101–115, July-December 2013.
- [29] C. Mark and C. Kijal. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, November 2006.
- [30] M. Marzolla, O. Babaoglu, and F. Panzieri. Server consolidation in clouds through gossiping. In *Proc. of International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM'11)*, pages 1–6, Lucca, Italy, June 2011.
- [31] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. of IEEE INFOCOM 2010*, pages 1–9, San Diego, CA, March 2010.
- [32] S. Nanda and T. Chiueh. A survey of virtualization technologies. Technical report, February 2005. www.ecsl.cs.sunysb.edu/tr/TR179.pdf.
- [33] R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-clouds: Managing performance interference effects for QoS-aware clouds. In *Proc. of the European Conference on Computer systems (EuroSys'10)*, pages 237–250, Paris, France, April 2010.

- [34] OnLive launches new company to avoid bankruptcy. <http://techland.time.com/2012/08/20/onlive>.
- [35] S. Osman, D. Subhraveti, G. Su, and J. Nieh. The design and implementation of zap: a system for migrating computing environments. *ACM SIGOPS Operating Systems Review*, 36(SI):361–376, December 2002.
- [36] R. Rose. Survey of system virtualization techniques. Technical report, March 2004. <http://www.robertwrose.com/vita/rose-virtualization.pdf>.
- [37] P. Ross. Cloud computing’s killer app: Gaming. *IEEE Spectrum*, 46(3):14, March 2009.
- [38] R. Shea and J. Liu. On GPU pass-through performance for cloud gaming: Experiments and analysis. In *Proc. of ACM Annual Workshop on Network and Systems Support for Games (NetGames’13)*, Denver, CO, December 2013.
- [39] S. Shi, K. Nahrstedt, and R. Campbell. Distortion over latency: Novel metric for measuring interactive performance in remote rendering systems. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME’11)*, pages 1–6, Barcelona, Spain, July 2011.
- [40] B. Speitkamp and M. Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on Services Computing*, 3(4):266–278, December 2010.
- [41] War of Warcraft avatar history dataset. <http://mmnet.iis.sinica.edu.tw/dl/wowah/>.
- [42] D. Wu, Z. Xue, and J. He. iCloudAccess: Cost-effective streaming of video games from the cloud with low latency. *IEEE Transactions on Circuits and Systems for Video Technology*, January 2014. Accepted to appear.
- [43] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, April 2008.
- [44] S. Zaman and D. Grosu. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. *IEEE Transactions on Cloud Computing*, 1(2):129–141, July-December 2013.

- [45] C. Zhang, Z. Qi, J. Yao, M. Yu, and H. Guan. vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming. *IEEE Transactions on Parallel and Distributed Systems*, November 2013. Accepted to appear.
- [46] Q. Zhu and T. Tung. A performance interference model for managing consolidated workloads in QoS-aware clouds. In *IEEE International Conference on Cloud Computing (CLOUD'12)*, pages 170–179, Honolulu, HI, June 2012.

Virtualization

There are several different virtualization technologies, such as JVM, virtual storage, virtual machines, have been proposed in the literature [26, 32, 35, 36]. These technologies can be roughly classified into the following 6 kinds:

- *Application Virtualization (AV)*: We can utilize this technique to virtualize an environment designed for an application to make it work on different systems. Take JVM for example, with JVM, we can execute a Java program on different environments, such as Windows, Linux, and OS X. On the other hand, if we want to execute a C program on those systems, we need to recompile it and import a great number of libraries to make this program work. Another example is Wine, an application installed on Linux, which can make Windows executables work on Linux machines.
- *Resource Virtualization (RV)*: RV is a technique in which specific host resources, such as CPU, network, and memory will be virtualized for guests. Take Gluster for example, it is one of resource virtualization techniques called virtual storage (VS). Gluster is constructed by some bricks. Each of them can be assigned to storage devices of different types and from different manufacturers, and those bricks will be combined into a single volume that guests use without any knowledge of storage devices.
- *Operating System Level Virtualization (OSLV)*: OSLV enables multiple guests running on a single operating system kernel. This technique has the performance close to native machine, and dynamic resource management is feasible. On the other hand, it does not allow to run different kernels, so if a guest performs a system call and crashes, it may affect other guests. Also, it cannot create guest with Windows if the kernel system is UNIX-based. Examples: FreeBSD Jail, Solaris Zones, and OpenVZ.

- *Para-virtualization (PARA)*: Para-virtualization is not full software virtualization. Without binary translation, it modifies guests OS's to make directly communicating with hardware whenever feasible. Because it requires modified guest OS, it can only create virtual machines with open source operating system such as, Linux and FreeBSD.
- *Hardware Virtual Machine (HVM)*: We also call it virtualization with hardware assistance. Popular virtualization solutions, such as VMware, Xen, KVM, and VirtualBox, are all implemented with it. Without modified operating system and binary translation, this technique leverages the CPU (Intel-VT, AMD-V) virtualization supports and enable the guests to directly communicate with hardware.
- *Full Virtualization (Full)*: With software, this virtualization technique can completely virtualize the environment to support guests, so we do not need to worry about what operating system guest want to create. When guests send a kernel call, virtual machine monitor (VMM) will receive it and, as a communicating bus, sends the request to hardware with binary translation mechanism. But, because of the software virtualized environment and complex translation mechanism, significant performance degradation exists on guests.

In Fig. 1, we sort these virtualization techniques according to their level of virtualization, where techniques on the right have higher virtualization levels. The aforementioned six techniques can be further grouped into two types, as illustrated in Fig. 2. The first type (Fig. 2(a)) enables multiple guests running on the same OS kernel; therefore, the kernel calls from different guests may affect each other. The AV, RV, and OSLV belong to this type. The second type (Fig. 2(b)) isolates guests from each other by running each guest in its own OS.

Our cloud gaming testbed is built upon type-2 HVM solutions, such as VMware, which are more flexible at the expense of slightly higher virtualization overhead. To reduce the virtualization overhead, type-1 solutions may be adopted, which however may require more customizations in game software.



Figure 1: Levels of virtualization.

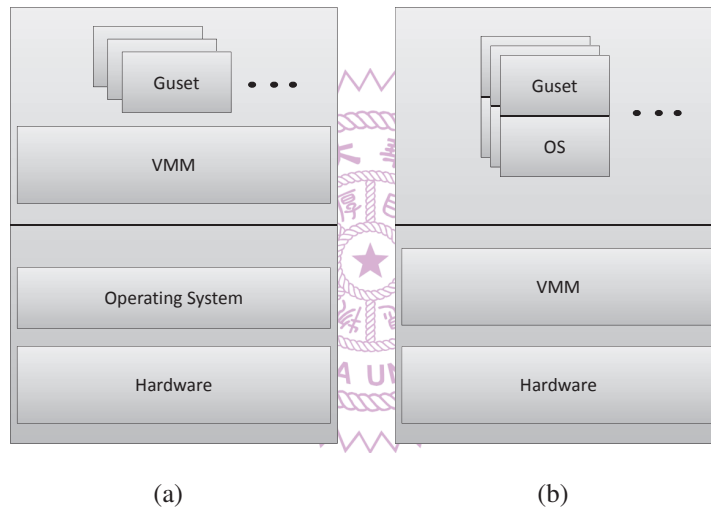


Figure 2: Two types of virtualization: (a) multiple guests with a single operating system and (b) multiple guests with individual operating systems.