Placing Virtual Machines to Optimize Cloud Gaming Experience

Hua-Jun Hong, De-Yu Chen, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu

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. 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. Last, 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.

Index Terms—Cloud gaming, remote rendering, live video streaming, real-time encoding, performance evaluation, performance optimization

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 [2]. 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 [3], and some leading game development companies [4] 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 [5]. 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.

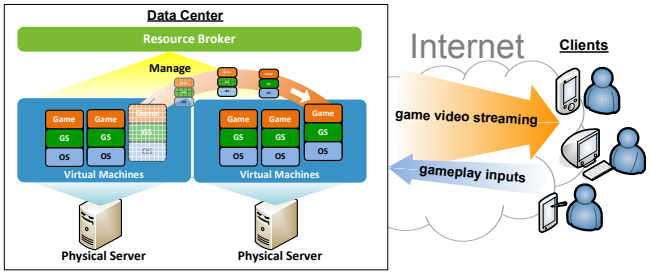


Fig. 1. The architecture of cloud gaming services, where GS denotes cloud gaming server.

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 gaming QoE in a cost-effective way, as illustrated

A preliminary version of this manuscript [1] appeared in the Proceedings of the 12th Annual Workshop on Network and Systems Support for Games (NetGames 2013) as a 2-page short paper.

- *H. Hong and C. Hsu are with Department of Computer Science, National Tsing Hua University, Hsin Chu, Taiwan.*
- *C. Huang is with Department of Computer Science and Engineering, National Taiwan Ocean University, Kee Lung, Taiwan.*
- *D. Chen and K. Chen are with Institute of Information Science, Academia Sinica, Taipei, Taiwan.*

in Fig. 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 [6], and thus is NP-Complete. The existing solutions for network embedded problems [6], [7], [8], [9], [10], 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 [11], [12], [13]. Hence, the existing virtual network embedding algorithms do *not* work for cloud gaming providers. To the best of our knowledge, this paper is the first attempt to tackle the VM placement problem to maximize the cloud gaming QoE.

In particular, this paper makes the following contributions:

- We conduct extensive measurement studies using an open-source cloud gaming platform, GamingAnywhere [14] on two VM implementations to derive the game-dependent parameters for QoE and performance models (Sec. 3).
- We formulate and propose two algorithms for the provider-centric VM placement problem (Sec. 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 (Sec. 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 (Sec. 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 (Sec. 7).

2 Related Work

2.1 General Cloud Applications

Optimizing general cloud applications has been studied in cloud environments. For example, Zaman et al. [15] 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. [16] 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 [15], [16], 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. [17] 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. [18] 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. [19] 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 [20] 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. [21] 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 [22] 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 [23] 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. [24] and Wu et al. [25] are complementary to our work, as they concentrate on minimizing the queueing delay of a cloud gaming system, while we

focus on the user experience during the game sessions. For example, Duong et al. [24] develop resource provisioning and waiting queue scheduling algorithms to admit selective incoming gamers for the best profit under user-specified maximal waiting times. Wu et al. [25] 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) [14] 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. [1].

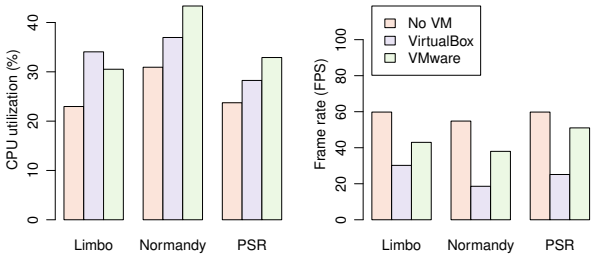


Fig. 2. Virtualization overhead depends on game and VM implementations.

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 [14] 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. [26].

TABLE 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

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 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. 3. This figure reveals that the CPU utilization, GPU utilization, frame rate, and processing delay can be modeled as sigmoid functions of the number of VMs on a physical server, which are also plotted in Fig. 3 as the curves. We notice that several basic functions, such as $ax+b$, ax^2+bx+c, a/x, $a-a/x$, $y=\log(x)$, and $y=\sqrt{x}$ may also be used as the regression models [27]. 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 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 1 shows that sigmoid functions model the VM measurement results very well. The precise fitted sigmoid models are detailed in Sec. 4.2, and the empirically derived parameters are used in Secs. 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.

4 VM Placement Problem and Solution

We study the provider-centric problem in this section.

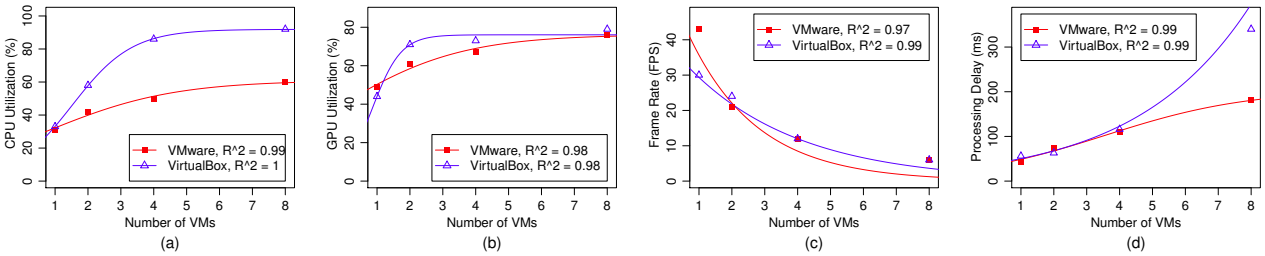


Fig. 3. Measurement results for CPU utilization, GPU utilization, frame rate, and processing delay. Sample results from Limbo.

4.1 System Overview

Fig. 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 Sec. 5. The games may have diverse resource requirements, including CPU, GPU, and memory [28], 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 2 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 [30]. 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. 3 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,

TABLE 2
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

1: **for** each gamer $p = 1, 2, \ldots, P$ **do**
2: **sort** servers on network latency to p in asc. order
3: **for** each server $s = 1, 2, \ldots, S$ **do**
4: **if** serving p on s satisfies Eqs. (2)–(8) **then**
5: **let** $x_{s,p} = 1$
6: **break**
7: **return** x

Fig. 4. The pseudocode of the QDH algorithm.

we use $u_s(v)$ and $z_s(v)$ to model the CPU and GPU utilizations of server s running v VMs. Fig. 3 shows that $u_s(v)$ and $z_s(v)$ can also be written as sigmoid functions $u_s(v) = \frac{\delta_1}{1+e^{-\delta_2 v+\delta_3}}$ and $z_s(v) = \frac{\zeta_1}{1+e^{-\zeta_2 v+\zeta_3}}$, where δ_1–δ_3 and ζ_1–ζ_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 [11], [12] 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. [11], [12] 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 [11], 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. [11]. Last, we use Q_p to denote the maximal tolerable QoE degradation of gamer p.

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:

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

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

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

$$v_s = \textstyle\sum_{p=1}^{P} x_{s,p}, \ \forall s; \tag{4}$$

$$1 = \textstyle\sum_{s=1}^{S} x_{s,p}, \ \forall p; \tag{5}$$

$$Q_p \geq \gamma_{p,1} f_p + \gamma_{p,2} \tilde{d}_p, \ \forall p; \tag{6}$$

$$B_w \geq B \textstyle\sum_{s \in \mathbf{S}_w} \sum_{p=1}^{P} x_{s,p}, \ \forall w; \tag{7}$$

$$G_s \geq G \textstyle\sum_{p=1}^{P} x_{s,p}, \ \forall s; \tag{8}$$

$$x_{s,p} \in \{0, 1\}, \ \forall 1 \leq s \leq S, 1 \leq p \leq P. \tag{9}$$

The objective function in Eq. (1) maximizes the provider's net profit, i.e., the difference between the collected fee and cost. Eqs. (2) and (3) derive the frame rate and response time as intermediate variables. In Eq. (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. (5) ensures that each gamer is served by a single server. Eq. (6) makes sure that the gaming QoE degradation is lower than the user-specified maximal tolerant level. Eqs. (7) and (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. (1)–(9) can be optimally solved using optimization solvers, such as CPLEX [31]. 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 exceeded. Fig. 4 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. (2)–(9). It is clear that the QDH algorithm runs in polynomial time.

1: **for** each gamer $p = 1, 2, \ldots, P$ **do**
2: **sort** servers on quality degradation $q_p(\cdot)$ in asc. order
3: **for** each server $s = 1, 2, \ldots, S$ **do**
4: **if** serving p on s satisfies Eqs. (2)–(5), (7)–(8) **then**
5: **let** $x_{s,p} = 1$
6: **break**
7: **return** $\mathbf{x}$

Fig. 5. The pseudocode of the QDH′ algorithm.

5 Alternative Formulation and Algorithms for Closed Systems

The provider-centric problem presented in Sec. 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. (1)–(9), and we first replace the objective function in Eq. (1) with:

$$\min \left[\sum_{p=1}^{P} \gamma_{p,1} f_p + \sum_{p=1}^{P} \gamma_{p,2} \tilde{d}_p \right], \tag{10}$$

which minimizes the total QoE degradation. In particular, the QoE degradation is reduced when f_p increases 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. (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 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. (2)–(5), (7)–(8). QDH′ runs in polynomial time.

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. 6. 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

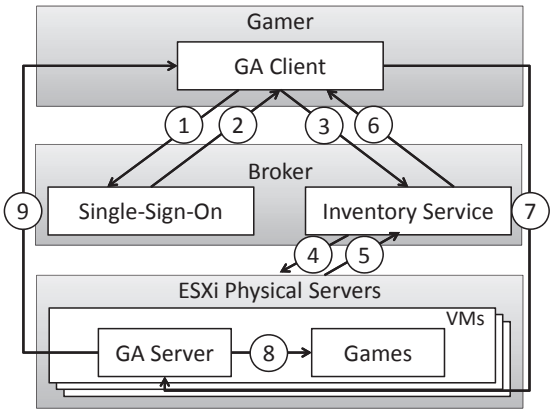


Fig. 6. The implemented prototype system.

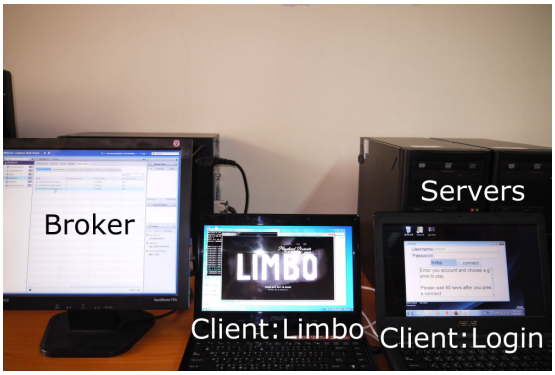


Fig. 7. The cloud gaming testbed in our lab.

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. 6 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. 7. 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

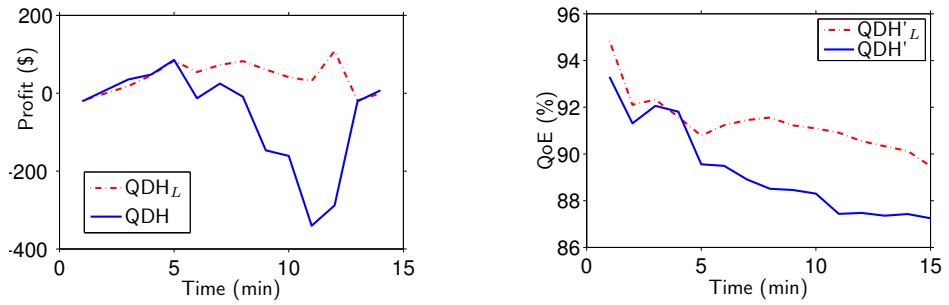


Fig. 9. Comparisons between QDH_L/QDH'_L and QDH/QDH': (a) net profits and (b) quality.

cards, and several i5 client computers. The broker, physical servers and client computers are connected via Gigabit Ethernet. We enable the CPU hardware support for virtualization 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. 8 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 [32]. 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.

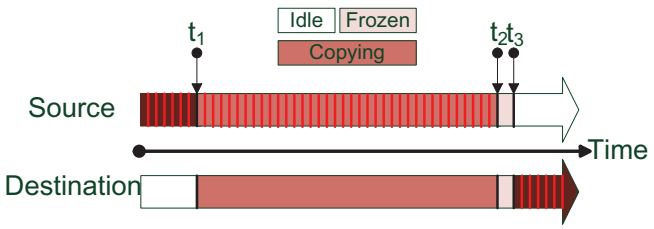
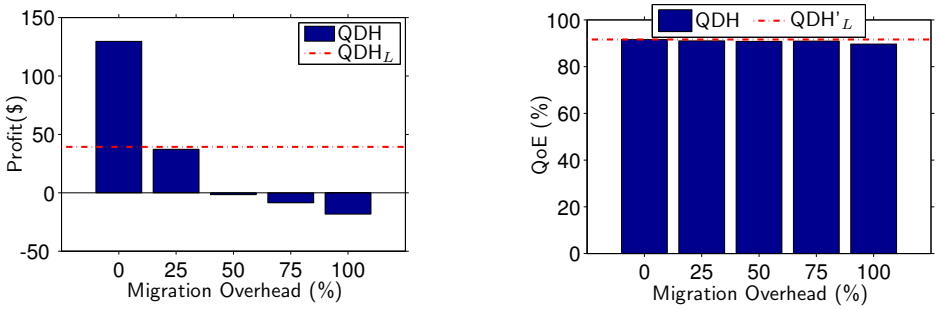


Fig. 8. Live migration.

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 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. 8. 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 and 5, we never migrate the ongoing game sessions. Intuitively, QDH_L/QDH'_L run faster, yet achieve better performance as the high migration time is avoided. Moreover, QDH'_L is an optimal migrationless algorithm. We will show this in evaluation sections (Secs. 6.3 and 7).

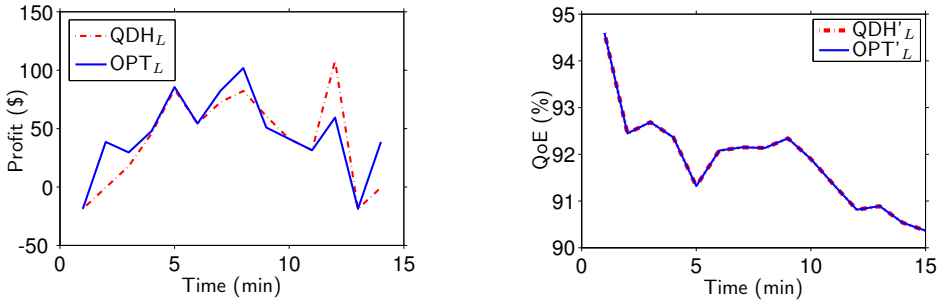
6.3 Experiment–Performance Gains of the Migrationless Algorithms

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 [33]. Last, we set $D = 90\%$, $T = 15$ minutes and consider the two performance metrics:



(a) (b)

Fig. 10. Comparisons between QDH_L/QDH'_L and QDH/QDH′: (a) net profits and (b) quality.



(a) (b)

Fig. 11. Comparisons between QDH_L/QDH'_L and OPT_L/OPT'_L: (a) net profits and (b) quality.

- *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. 9(a) and 9(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. 10. This figure reveal 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 paper.

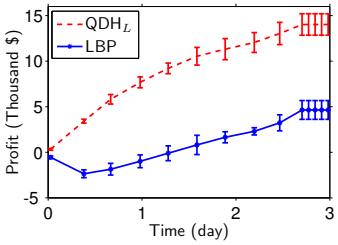
Last, we compare the QDH_L/QDH'_L algorithms against the migrationless optimal solution that exhaustively checks all servers for each new gamer. We refer to the migrationless optimal solutions as OPT_L/OPT'_L. Fig. 11 reports the average performance over time. Fig. 11(a) shows that QDH_L and OPT_L result 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. 11(b) reveals that QDH'_L leads to exactly the same (optimal) performance in QoE, compared to OPT'_L. Fig. 11 shows the merits of QDH_L and QDH'_L.

7 Trace-Driven Simulations

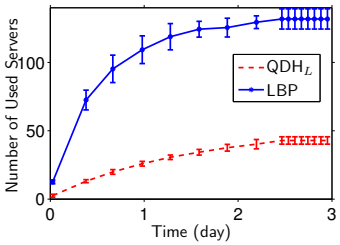
In this section, we consider large-scale evaluations using detailed simulations.

7.1 Setup

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)



(b)

Fig. 12. Provider-centric simulation results with synthetic traces: (a) net profits and (b) used servers.

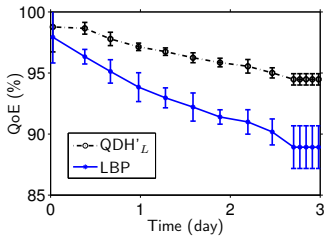


Fig. 13. Gamer-centric simulation results with synthetic traces.

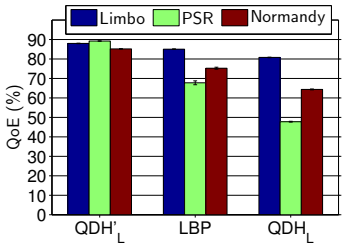


Fig. 14. Fairness in QoE levels on different game genres.

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 DigSitesValue [34] to obtain the IP addresses of OnLive data centers in Virginia, California, and Texas. For gamers, we develop a BitTorrent crawler using libtorrent [35] 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 [36]. We ran our crawler on August 13, 2013 with 4494 torrents downloaded from IsoHunt [37], which gave us 22395 IP addresses and 5875 US IP addresses. Next, we measure the network latencies among gamer/server IP pairs using King [30], 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.

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 [38] 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:

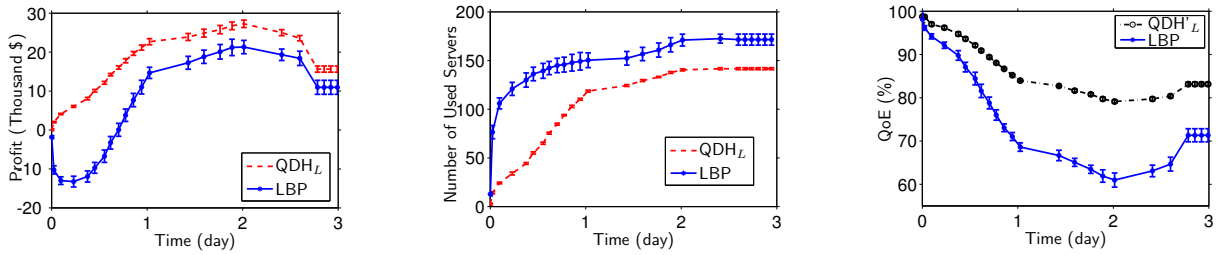
- *Net profit.*
- *Quality of Experience.*
- *Running time.* The time of executing each algorithm.
- *Number of used servers.* The number of servers that serve at least one gamer.

7.2 Results

Performance of QDH$_L$/QDH$'_L$. We plot the provider-centric results in Fig. 12(a), which shows that QDH$_L$ significantly outperforms LBP: up to 3.5 times difference. This can be explained by Fig. 12(b), which shows that QDH$_L$ turns on fewer servers to achieve higher net profits. We plot the gamer-centric results in Fig. 13, which reveals that QDH$'_L$ constantly outperforms LBP: up to 5% QoE gap. Moreover, the confidence intervals show that QDH$'_L$ leads to more consistent QoE levels among individual gamers, achieving better *fairness*. Fig. 14 plots the aggregate QoE of three games, which shows that both QDH$'_L$ and LBP are relatively fair to different game genres, while QDH$_L$ 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. 15(a), which shows that QDH$_L$ 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. 15(b) and 17(b), which reveals that while QDH$_L$ shutdown more servers and it always allows all gamers meets 60+% QoE which is the just-good-enough QoE level. Fig. 15(c) shows that QDH$'_L$ outperforms LBP up to 130%.

Scalability. We plot the running time in Fig. 16, which shows the QDH$_L$/QDH$'_L$ 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



(a) (b) (c)

Fig. 15. Simulation results with WoW traces: (a) net profits, (b) used servers, and (c) QoE levels.

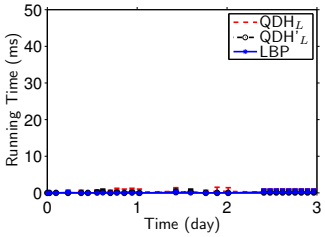
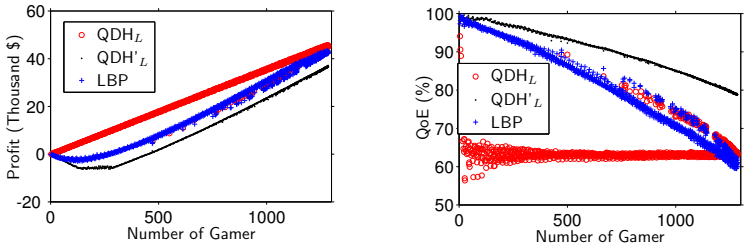


Fig. 16. Running time of VM placement algorithms.



(a) (b)

Fig. 17. Impacts of number of gamers on: (a) net profits and (b) QoE levels.

TABLE 3
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

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 3 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. 17(a) and 17(b) to study the relation between the performance and number of gamers. These figures show that more gamers lead to higher profits and lower QoE levels, and QDH_L/QDH'_L successfully achieve their design objectives.

8 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 [14], 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. Via testbed and extensive trace-driven simulations, 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, and (iii) the efficient algorithms terminate in < 2.5 s on a commodity PC even for large services with 20000 servers and 40000 gamers.

This work can be extended in several directions. For example, 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.

References

[1] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. 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.

[2] P. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectrum*, vol. 46, no. 3, p. 14, March 2009.

[3] "Distribution and monetization strategies to increase revenues from cloud gaming," http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf.

[4] "Cloud gaming adoption is accelerating ... and fast!" http://www.nttcom.tv/2012/07/09/cloud-gaming-adoption-is-acceleratingand-fast/.

[5] "OnLive launches new company to avoid bankruptcy," http://techland.time.com/2012/08/20/onlive.

[6] 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*, vol. 15, no. 2, pp. 909 – 928, 2012, accepted to appear.

[7] 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*, vol. 38, no. 2, pp. 17–29, April 2008.

[8] 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*, vol. 41, no. 2, pp. 38–47, April 2011.

[9] 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*, San Diego, CA, March 2010, pp. 1–9.

[10] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. of IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, April 2009, pp. 783–791.

[11] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. 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)*, October 2012, pp. 117–124.

[12] 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)*, Barcelona, Spain, July 2011, pp. 1–6.

[13] 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)*, Ottawa, Canada, October 2011, pp. 2:1–2:6.

[14] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Transactions on Multimedia Computing Communications and Applications*, pp. 1–25, Jan 2014.

[15] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 129–141, July-December 2013.

[16] J. Lin, C. Chen, and J. Chang, "QoS-aware data replication for data-intensive applications in cloud computing systems," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 101–115, July-December 2013.

[17] 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)*, Lucca, Italy, June 2011, pp. 1–6.

[18] T. Ferreto, M. Netto, R. Calheiros, and C. Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, October 2011.

[19] 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)*, Dublin, Ireland, May 2011, pp. 594–601.

[20] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, December 2010.

[21] 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)*, Honolulu, HI, June 2012, pp. 170–179.

[22] 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)*, Paris, France, April 2010, pp. 237–250.

[23] 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)*, Miami, FL, February 2010, pp. 435 – 442.

[24] 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.

[25] 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.

[26] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proc. of ACM International Conference on Multimedia (MM'11)*, Scottsdale, AZ, November 2011, pp. 1269–1272.

[27] M. Kutner, C. Nachtsheim, J. Neter, and W. Li, *Applied linear statistical models*, 5th ed. McGraw-Hill Higher Education, 2004.

[28] M. Claypool, "Motion and scene complexity for streaming video games," in *Proc. of the International Conference on Foundations of Digital Games (FDG'09)*, Port Canaveral, FL, April 2009, pp. 34–41.

[29] C. Mark and C. Kajal, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, November 2006.

[30] 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)*, Boston, MA, November 2002, pp. 5–18.

[31] "IBM ILOG CPLEX optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[32] 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*, vol. 2, pp. 273–286, 2005.

[33] "DummyNet," http://info.iet.unipi.it/~luigi/dummynet/.

[34] "DigSites," http://digsitesvalue.net/s/onlive.com.

[35] "libtorrent," http://www.rasterbar.com/products/libtorrent/.

[36] "ip2c," http://firestats.cc/wiki/ip2c.

[37] "IsoHunt," http://isohunt.com/.

[38] "War of Warcraft avatar history dataset," http://mmnet.iis.sinica.edu.tw/dl/wowah/.

[39] 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)*, Guangzhou, China, July 2010, pp. 332–336.

[40] S. Nanda and T. Chiueh, "A survey of virtualization technologies," Tech. Rep., February 2005, www.ecsl.cs.sunysb.edu/tr/TR179.pdf.

[41] R. Rose, "Survey of system virtualization techniques," Tech. Rep., March 2004, http://www.robertwrose.com/vita/rose-virtualization.pdf.

[42] 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*, vol. 36, no. SI, pp. 361–376, December 2002.

Hua-Jun Hong is a Masters student at the National Tsing Hua University, Taiwan. He received his B.S. in Computer Science from National Tsing Hua University, too. His research interests are in cloud computing, cloud gaming, and multimedia networking.

De-Yu Chen is a research assistant at the Institute of Information Science of Academia Sinica. He received his M.S. in Computer Science from National Taiwan University in 2009, and his B.B.A. in Business Administration from National Taiwan University in 2006. His research interests include cloud computing, distributed computing, and network traffic analysis.

Chun-Ying Huang (S'03–M'08) is an Associate Professor at the Department of Computer Science and Engineering, National Taiwan Ocean University. He received his B.S. in Computer Science from National Taiwan Ocean University in 2000 and M.S. in Computer Information Science from National Chiao Tung University in 2002. Dr. Huang received his Ph.D. in Electrical Engineering Department from National Taiwan University in 2007. His researches focus on computer network and network security issues, including traffic measurement and analysis, malicious behavior detection, and multimedia networking systems. Dr. Huang is a member of ACM, CCISA, IEEE, and IICM.

Kuan-Ta Chen (a.k.a. Sheng-Wei Chen) (S'04–M'06) is an Associate Research Fellow at the Institute of Information Science and the Research Center for Information Technology Innovation (joint appointment) of Academia Sinica. Dr. Chen received his Ph.D. in Electrical Engineering from National Taiwan University in 2006, and received his B.S. and M.S. in Computer Science from National Tsing-Hua University in 1998 and 2000, respectively. His research interests include quality of experience, multimedia systems, and social computing. He has been an Associate Editor of IEEE Transactions on Multimedia since 2011. He is a member of ACM, IEEE, IICM, and CCISA.

Cheng-Hsin Hsu (S'09–M'10) received the B.Sc. degree in mathematics and M.Sc. degree in computer science and information engineering from National Chung-Cheng University, Taiwan, in 1996 and 2000, respectively. He received the M.Eng. degree in electrical and computer engineering from the University of Maryland, College Park, in 2003 and the Ph.D. degree in computing science from Simon Fraser University, Burnaby, BC, Canada, in 2009. He is an assistant professor at National Tsing Hua University at Hsin Chu, Taiwan. His research interests are in the area of multimedia networking and distributed systems. He is a member of the IEEE and ACM.

APPENDIX

VIRTUALIZATION

Many different virtualization technologies, such as JVM, virtual storage, virtual machines, have been proposed in the literature [39], [40], [41], [42]. 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. 18, 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. 19. The first type (Fig. 19(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. 19(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.

Fig. 18. Levels of virtualization.

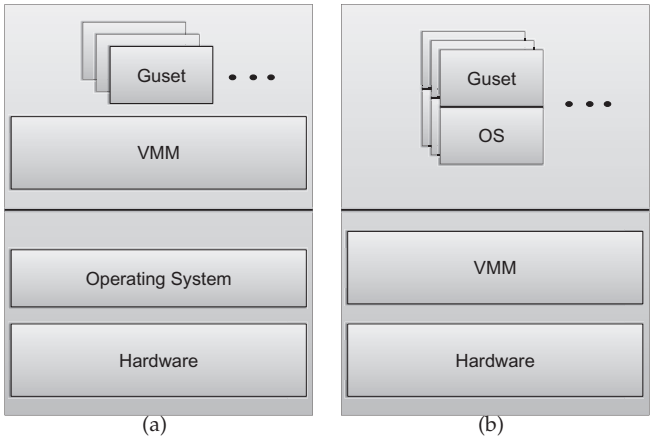


Fig. 19. Two types of virtualization: (a) multiple guests with a single operating system and (b) multiple guests with individual operating systems.