# QoE-Aware Virtual Machine Placement for Cloud Games

Hua-Jun Hong[1], De-Yu Chen[2], Chun-Ying Huang[3], Kuan-Ta Chen[2], and Cheng-Hsin Hsu[1]
[1]Department of Computer Science, National Tsing Hua University
[2]Institute of Information Science, Academia Sinica
[3]Department of Computer Science, National Taiwan Ocean University

***Abstract*—We study an optimization problem to maximize the cloud gaming provider's total profit while achieving just-good-enough Quality-of-Experience (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, in which the profit is not a concern and overall gaming QoE needs to be maximized. We conduct extensive trace-driven simulations, which show that the proposed heuristic algorithms: (i) achieve close-to-optimal solutions, (ii) scale to large cloud gaming services with 3000+ servers and 1000+ gamers, and (iii) outperform the state-of-the-art placement heuristic, e.g., by up to 3.5 times in terms of net profits. We also present a prototype system and testbed using off-the-shelf virtualization software.**

## I. Introduction

Cloud gaming providers, such as Gaikai, Ubitus, and OnLive, offer on-demand gaming services to many gamers, who play games via thin clients running on their desktops, laptops, smartphones, and TV set-top boxes. Cloud gaming services are very attractive to gamers and developers [26], and a market study projects that the cloud gaming market is going to grow to 8 billion US dollars by 2017 [11]. In fact, some cloud gaming startups, such as Gaikai, were recently acquired by leading game developers, such as SONY [8], which shows that the tremendous market potential of cloud gaming has been appreciated by the game industry.

Offering cloud gaming services in a commercially-viable way is, however, very challenging as demonstrated by OnLive's financial difficulty [25]. 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. This diversity renders the dilemma of finding the best tradeoff between *profit* and *QoE* even harder.

*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 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.

This optimization problem is similar to the virtual network embedding problem [1], and is also NP-Complete. However, existing solutions for virtual embedding problem [1, 5, 6, 23, 30] concentrate on computational/storage intensive applications, without taking the *real-time* nature 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, 19, 27]. 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 [15] on two VM implementations to derive the game-dependent parameters for QoE and performance models (Sec. III).
- We formulate and propose two algorithms for the provider-centric VM placement problem (Sec. IV).
- 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 profit is not a concern and the overall gaming QoE needs to be maximized. We also propose two algorithms to solve the gamer-centric problem (Sec. V).
- Our extensive simulations indicate that our efficient algorithms: (i) result in close-to-optimal performance, as small as 1% and 14% gaps, (ii) scale to large cloud gaming services with 3000+ servers and 1000+ gamers, and (iii) outperform a state-of-the-art algorithm, e.g., up to 3.5 times of net profit increase (Sec. VI).

## II. Related Work

Marzolla et al. [22] 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. [12] 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] develop a migration algorithm that considers the historical migration for saving energy. Speitkamp and Bichler [28] 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. [31] create a performance

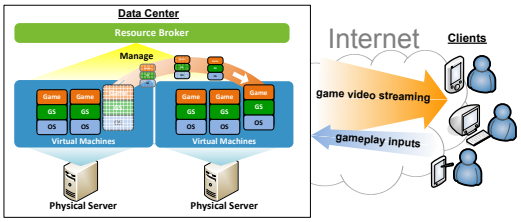



Fig. 1. The architecture of the considered cloud gaming service platform, where GS denotes cloud gaming server.

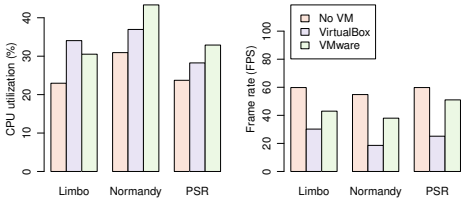



Fig. 2. Virtualization overhead depends on game and VM implementations.

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 [24] 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 QoE levels into consideration.

The benefits of game server consolidation have been studied for certain game genres. For example, Lee and Chen [18] address the server consolidation problem for Massively Multiplayer Online Role-Playing Game (MMORPG). In contrast, we consider cloud gaming that streams high-quality real-time videos to gamers, while MMORPG servers only send low bitrate status updates. Moreover, we explicitly optimize gaming QoE in this paper.

GamingAnywhere (GA) [15] is an open cloud gaming system, which provides a platform for experimenting different optimization techniques for cloud gaming. We use GA to derive the performance and QoS models for different games on different VMs, and to develop VM placement algorithms.

## III. 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 [15] on VMware workstation 9 and VirtualBox 4.2.6. The GA client runs on another machine without VMs. The two machines running GA server and client are Windows 7, 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 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 [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 gives some sample results, which reveals that: (i) VMs lead to nontrivial overhead, (ii) different VMs result in different 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. This figure, along with the R-square values, given in our technical report [14], demonstrate that our models closely follow the measurement results. The precise fitted sigmoid models are detailed in Sec. IV-B, and the empirically derived parameters are used in Secs. VI and VII. 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 service providers are likely to build data centers with one or very few types of machines, which can be profiled offline beforehand.

## IV. VM Placement Problem and Solution

### A. System Overview

Fig. 1 illustrates the system architecture of a 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 data centers at diverse locations. The gamers run game clients on desktops, laptops, mobile devices, and set-top boxes to access the cloud gaming platform 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. The games may have diverse resource requirements, including CPU, GPU, and memory [7], 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 [21]. Last, we note that the broker can be a virtual service running on a server or a server farm for higher scalability.

### B. Notations and Models

We study the VM placement problem, in which the VM placement decisions affect network delay, processing delay, and

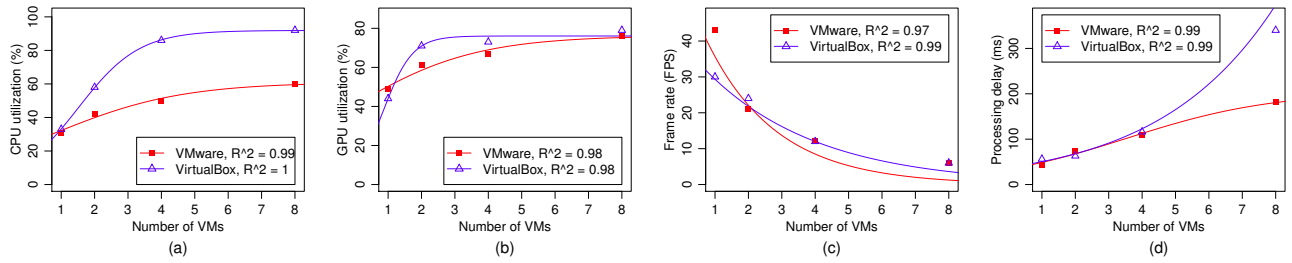



Fig. 3. Measurement results for CPU utilization, GPU utilization, frame rate, and processing delay. Sample results from Limbo.

operational cost. We write the network delay between server $s$ ($1 \le s \le S$) and gamer $p$ ($1 \le p \le P$) 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 [13]. Let $p$ be the game played by a specific gamer, we use $f_p(v)$ and $d_p(v)$ to denote the frame rate and processing delay when serving $p$ with a server running $v$ VMs. 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, 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 a sigmoid function $u(v) = \frac{\delta_1}{1+e^{-\delta_2 v+\delta_3}}$ and $z(v) = \frac{\zeta_1}{1+e^{-\zeta_2 v+\zeta_3}}$, 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 $w_s(v) = c_s(u(v)+z(v))$ be the operational cost of imposing CPU and GPU utilization $u(v)$ and $z(v)$ on $s$, where $c_s$ is a cost term consisting of various components, such as electricity, maintenance, and depreciation. Moreover, we allocate $G_v$ GB memory to each VM, whereas each physical server is equipped with $G_p$ GB memory. Last, we consider GA servers to stream at $B$ kbps. We let $W$ be the number of data centers, and use $\mathbf{S}_w$ ($1 \le w \le W$) to denote the set of servers in data center $w$. We let $B_w$ be the uplink bandwidth of data center $w$ ($1 \le w \le W$). Our bandwidth model is general, as the mapping between servers and data centers is flexible: 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 [19, 27] 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 [19, 27] 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 $g$ with frame rate $f_p$ and response time $\tilde{d}_{s,p}$. Inspired by the linear QoE model in [19], 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 [19]. Last, we use $Q_p$ to denote the maximal tolerable QoE degradation of gamer $p$.

### C. Problem Formulation

We let $x_{s,p} \in \{0,1\}$ ($1 \le p \le P, 1 \le s \le S$) 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:

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

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

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

$$v_s = \sum_{p=1}^{P} x_{s,p},\ \forall s; \quad (4)$$

$$1 = \sum_{s=1}^{S} x_{s,p},\ \forall p; \quad (5)$$

$$Q_p \ge \gamma_{p,1} f_p + \gamma_{p,2}\tilde{d}_p,\ \forall p; \quad (6)$$

$$B_w \ge B\sum_{s\in\mathbf{S}_w}\sum_{p=1}^{P} x_{s,p},\ \forall w; \quad (7)$$

$$G_p \ge G_v \sum_{p=1}^{P} x_{s,p},\ \forall s; \quad (8)$$

$$x_{s,p} \in \{0,1\},\ \forall 1 \le s \le S, 1 \le p \le P; \quad (9)$$

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.

### D. Proposed Algorithm

The provider-centric formulation in Eqs. (1)–(9) can be optimally solved using optimization solvers, such as CPLEX [9]. We refer to the solver-based algorithm as OPT. The OPT algorithm gives optimal solutions at the expense of exponential time 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. The pseudocode of QDH is given in [14] due to the space limitations. 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.

## V. Alternative Formulation and Algorithms for Closed Systems

The provider-centric problem presented in Sec. IV 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 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$'$. Its pseudocode is also given in [14]. 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 individual servers. 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.

## VI. Trace-Driven Simulations

### A. Setup

We have built a simulator for the VM placement problem using a mixture of C/C++, Java, CPLEX, and Matlab. We have implemented the QDH/QDH$'$ and OPT/OPT$'$ algorithms in our simulator. For comparisons, we have also implemented a VM placement algorithm that places each VM on a random gamer 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 [10] to obtain the IP addresses of OnLive data centers in Virginia, California, and Texas. For gamers, we develop a BitTorrent crawler using libtorrent [20] 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 [16]. We ran our crawler on August 13, 2013 with 4494 torrents downloaded from IsoHunt [17], which gave us 22395 IP addresses and 5875 US IP addresses. Next, we measure the network latencies among gamer/server IP pairs using King [13], 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 [29] in our simulations. Each gamer plays a game randomly chosen from Limbo, PSR, and Normandy. We also vary the number of servers $S \in \{192, 384, 768, 1536, 3072\}$. 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 $c_s = 1$. We conduct all the simulations on an Intel i7 3.4 GHz PC. We consider the following performance metrics:

- *Net profit.* The total provider profit in every minute.
- *Quality of Experience.* The gaming QoE normalized in the range of $[0\%, 100\%]$.
- *Running time.* The time of executing each algorithm.
- *Number of used servers.* The number of servers that serve at least one gamer.

### B. Sample Results

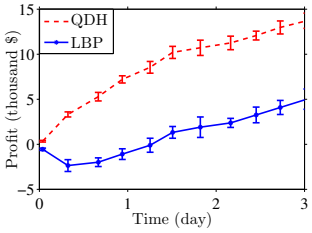
Due to the space limitations, more simulation results are given in [14].

**Necessity of QDH/QDH$'$.** The OPT/OPT$'$ algorithms can only solve small problems with less than 6 servers and 10 gamers. Moreover, the proposed QDH/QDH$'$ algorithms result in close-to-optimal performance, up to 86% and 99% in the provider- and gamer-centric scenarios, respectively. Therefore, we no longer consider OPT/OPT$'$ in the rest of this paper.

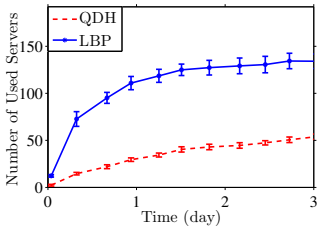
**Performance of QDH/QDH$'$.** We plot the provider-centric results in Fig. 4(a), which shows that QDH significantly outperforms LBP: up to 3 times difference. This can be explained by Fig. 4(b), which shows that QDH turns on fewer servers to achieve higher net profits. We plot the gamer-centric results in Fig. 5, which reveals that QDH$'$ constantly outperforms LBP: up to 5% QoE gap. Moreover, the confidence intervals show that QDH$'$ leads to more consistent QoE levels among individual gamers, achieving better *fairness*. Fig. 6 plots the aggregate QoE of three games, which shows that both QDH$'$ and LBP are relatively fair to different game genres, while QDH 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(a), which shows that while QDH outperforms LBP in the first half of the simulation (by up to 3.5 times), LBP performs better in the second half. This can be explained by Figs. 7(b) and 7(c), which reveal that QDH turns on more servers at meet the just-good-enough QoE level. More specifically, Fig. 7(c) shows that LBP fails to deliver good QoE levels: as low as 0% is observed. This figure also shows that QDH$'$ always achieves 80+% QoE levels.

**Scalability.** We plot the running time in Fig. 8, which shows the QDH/QDH$'$ algorithms terminate in real time: $< 250$ ms. We then increase the number of servers $S$. We found that it takes QDH/QDH$'$ at most 7.15 s to solve a VM placement problem with more than three thousand servers and a thousand




(a)




(b)

Fig. 4. Provider-centric results with synthetic traces: (a) net profits and (b) used servers.

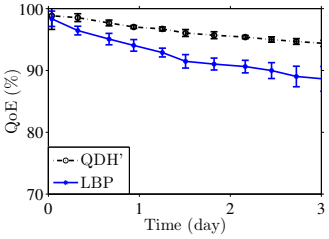



Fig. 5. Gamer-centric simulation results with synthetic traces.

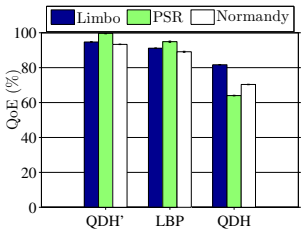
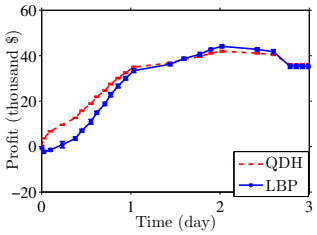
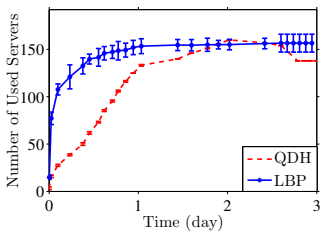


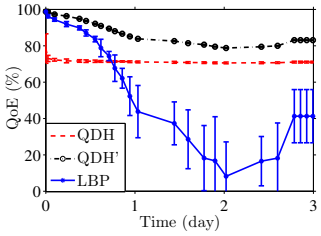

Fig. 6. Fairness in QoE levels on different game genres.




(a)




(b)




(c)

Fig. 7. Simulation results with WoW traces: (a) net profits, (b) used servers, and (c) QoE levels.

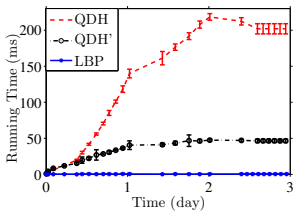
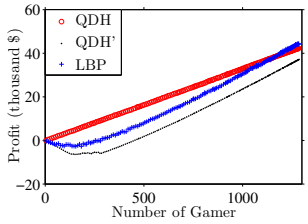
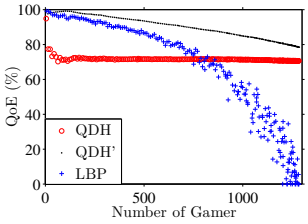



Fig. 8. Running time of the algorithms.




(a)




(b)

Fig. 9. Impacts of number of gamers on: (a) net profits and (b) QoE levels.

of concurrent 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. 9(a) and 9(b) to study the relation between the performance and number of gamers. We made two observations on these figures. First, more gamers lead to higher profits and lower QoE levels, and QDH/QDH′ successfully achieve their design objectives. Second, LBP only works with few gamers, as illustrated in Fig. 9(b).

## VII. System Implementation and Testbed

### A. Prototype Implementation

We have implemented a complete cloud gaming system consisting of a broker, physical servers, and GA servers/clients, as illustrated in Fig. 10. We adopt VMWare ESXi 5.1 as the virtualization software on physical servers. 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. We integrate the GA client and server with VMware ESXi and vCenter. In particular, the GA client provides interface for gamers to send their accounts and passwords to the broker. Upon being authenticated, 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. The broker then instructs the chosen physical server to launch a VM and send the VM's IP address to the GA client. Last, the GA client connects to the GA server, and instructs the GA server to run the user-specified game. This starts a new GA game session.

### B. Testbed and Practical Concerns.

We set up a testbed using the prototype system in our lab, which is shown in Fig. 11. The testbed contains an i7 3.2 GHz broker with the management web page, two i5 3.5 GHz physical servers, and two i5 client computers. The broker, physical servers and client computers are connected via Gigabit Ethernet. We measure the migration overhead, which is the delay of moving an ongoing session from a physical server to another one. We found that the migration delays of 10, 20, and 40 GB VM images are about 3, 6, and 11 minutes.

**Adaptive QDH/QDH′ algorithms.** To cope with long migration delays, we propose adaptive variations of QDH/QDH′ algorithms, which only launch the VMs for new gamers, and never migrate VMs running ongoing game sessions. We refer to the new adaptive algorithms as $\text{QDH}_\text{A}$ and $\text{QDH}'_\text{A}$, and we quantify their performance via simulating a large cloud gaming system with 3072 servers using the WoW traces. We

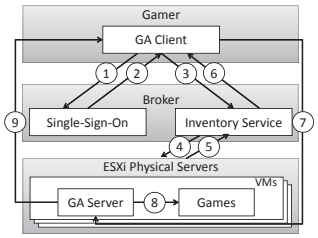



Fig. 10. The prototype system.


Fig. 11. The cloud gaming testbed.

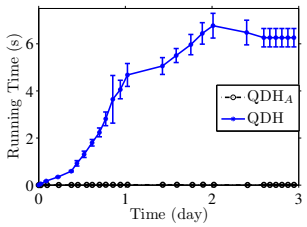



Fig. 12. Time reduction of $\text{QDH}_\text{A}$.

TABLE I
INITIAL RESULTS FROM A GPU ACCELERATED TESTBED

| Mean Performance | QDH | QDH′ | LBP |
|---|---|---|---|
| Frame Rate (fps) | 26 | 33 | 19 |
| Net Profit ($) | 250 | 180 | 103 |
| QoE (%) | 80 | 85 | 77 |

found that the performance ratio of $\text{QDH}_\text{A}$/QDH is $> 89\%$ and $\text{QDH}'_\text{A}$/QDH′ is $> 99\%$, while the running time is reduced by $\sim$500 times as illustrated in Fig. 12. Our evaluations show that $\text{QDH}_\text{A}$ and $\text{QDH}'_\text{A}$ perform very well with smaller time complexity, and may support larger cloud gaming systems.

## VIII. 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 [15], 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 extensive trace-driven simulations, we demonstrate that: (i) the efficient algorithms achieve up to 99% (gamer-centric) and 86% (provider-centric) performance compared to the optimal algorithms, while the optimal algorithms do not scale to a large number of game servers and gamers, (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 $< 7.15$ s on a PC for a system with 3000+ servers and 1000+ gamers.

This work can be extended in several directions. For example, we may develop more comprehensive system models, which consider other resource types and heterogeneous servers, and support online parameter adaptation. Another potential extension is to leverage the modern virtualization software the comes with some GPU supports. Our preliminary experiments with 10 gamers and 8 physical servers equipped with NVidia Quadro 6000 cards confirm the merits of QDH/QDH′, which are summarized in Table I. More rigorous experiments on the GPU accelerated testbed are among our future tasks.

## References


[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] 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.

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

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

[10] DigSites. http://digsitesvalue.net/s/onlive.com.

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

[12] 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.

[13] 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.

[14] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu. QoE-aware virtual machine placement for cloud games. Technical report, September 2013. http://nmsl.cs.nthu.edu.tw/dropbox/netgames13TR.pdf.

[15] 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.

[16] ip2c. http://firestats.cc/wiki/ip2c.

[17] IsoHunt. http://isohunt.com/.

[18] 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.

[19] 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.

[20] libtorrent. http://www.rasterbar.com/products/libtorrent/.

[21] C. Mark and C. Kajal. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, November 2006.

[22] 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.

[23] 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.

[24] 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.

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

[26] P. Ross. Cloud computing's killer app: Gaming. *IEEE Spectrum*, 46(3):14, March 2009.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.