2016 IEEE 9th International Conference on Cloud Computing

Inter-player Delay Optimization in Multiplayer Cloud Gaming

Yuchi Chen
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
yuchi_chen@sfu.ca

Jiangchuan Liu
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
Email: jcliu@sfu.ca

Yong Cui
Department of Computer Science and Technology
Tsinghua University
Beijing, China
Email: cuiyong@tsinghua.edu.cn

***Abstract*—Novel cloud computing technology makes multiplayer cloud gaming a reality, where players play games that do not run on local devices, but on servers in the cloud. Nevertheless, the necessary communication between player's local device and cloud server increases the response delay of the gaming session. Besides the degrade of responsiveness, the inter-player delay, which is the difference of response delays perceived by players who are interacting with each other, can significantly affect the fairness of the multiplayer game. In this paper, we first introduce the Inter-player Delay Optimization (IDO) problem that aims at minimizing this inter-player delay, while preserving good-enough absolute response delay experienced by players. We further propose an efficient heuristic algorithm to solve the IDO problem. The evaluation result of a comprehensive simulation using a large-scale real-world data trace shows that IDO can reduce up to about 30% of maximum inter-player delay among interacting players comparing with the other existing solution.**

***Keywords*-cloud gaming, inter-player, delay, VM placement.**

I. Introduction

As a novel application of cloud computing technology, cloud gaming has become an attractive online gaming service. An increasing number of service providers have been involved in this new generation of gaming paradigm, such as SONY PlayStation Now, NVIDIA GeForce NOW, Ubitus GameNow, etc. Unlike traditional online games, the cloud-based game does not run on player's local device, but on the remote server in the cloud Data Centers (DCs). The remote server operates the game logic and streams the game scene back to player's client through the Internet, so that the client can continuously show the game scene to the player, just like the game is running by the client. Cloud gaming enables players to play arbitrary high-quality games with less powerful devices. With recent technology evolutions and service promotions carried out by major players in the industry, cloud gaming is expected to undergo a series of dramatic upgrades in the near future [1].

Nevertheless, providing cloud gaming service with high Quality of Experience (QoE) is a challenging task. One essential problem with cloud gaming is that the network connection between the player's client and the cloud server seriously affects the responsiveness of the game session, which is assessed by the *absolute response delay*, i.e. the time duration between the player issuing a command and the corresponding game frame being displayed to him. As is presented in some measurement studies [2], the absolute response delay of a cloud gaming system can be as high as over 550ms. This means that the player's QoE can be hardly guaranteed, especially in a real-time online Role Playing Game (RPG) that usually requires this response delay to be lower than 500ms [3, 4].

Another important issue is that cloud gaming system can exaggerate the *inter-player delay*, i.e. the difference of absolute response delay among players. The high inter-player delay can lead to frustrating unfairness among interacting players. As is revealed by Zander et al [5], in a real-time First Person Shooter (FPS) game players with lower absolute response delay have a clear advantage over bots with larger delay. More precisely, as the inter-player delay increases from 100ms to 400ms, the performance (i.e. the kill rate in the FPS game playing) of players decreased by up to 40%. Apparently, high inter-player delay issue can be even more harmful to QoE than the high absolute response delay in multiplayer gaming [6].

Generally, the absolute response delay perceived by cloud gaming players is determined by three kinds of temporal costs [7]: (1) *network delay*, the time required for the cloud server to receive player's command and the video frames are streamed back to the player's client, (2) *playout delay*, the time required by the client to decode and play the received video stream, and (3) *processing delay*, the time needed by the cloud server to generate the video frames according to player's input. Although network delay occupies most part of the absolute response delay, some measurement studies show that the processing delay can also take over 30% of the absolute response delay[7]. The processing delay of some cloud gaming services can be even as much as over 200ms[8], which can significantly reduce the responsiveness of cloud game session. As such both network and processing delay should be taken into account carefully when resolving the high inter-player delay issue.

In this paper, we identify and address this important but not yet well-solved problem: the *Inter-player Delay Optimization* problem (IDO) in the multiplayer cloud gaming scenario. We first introduce and formulate the IDO problem,

2159-6190/16 $31.00 © 2016 IEEE
DOI 10.1109/CLOUD.2016.96

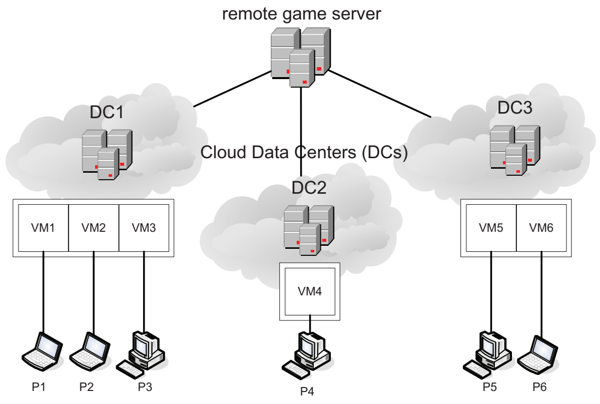


Figure 1. Typical multiplayer cloud gaming framework

which aims at minimizing the inter-player delay among interacting players, while preserving good-enough absolute response delay experienced by players. We further propose an efficient algorithm to solve the IDO problem by two major steps. Based on the proposed algorithm, we conduct a comprehensive evaluation based on the realworld trace dataset called World of Warcraft Avatar History Dataset [9], which is useful for simulating the interact patterns in realworld multiplayer online games. We compare the performance of our algorithm with the other existing solution, showing that our algorithm can overperform the existing mechanism under various scenarios with various scales of players and DCs.

II. Problem Statement

The typical framework of cloud-based online gaming system discussed in this paper is illustrated in Figure 1. Different from traditional online gaming system, the game client runs not on the player's local device, but on the Virtual Machines (VMs) that are hosted by cloud servers in the cloud DCs. On one hand, these VMs appear as the players' devices from the perspective of remote game server. On the other hand, the VMs play the role of cloud game server that runs the game logic. The client running on player's device obtains the player's command, transfer it to the VM, receive the video frame sent back by the VM and play it out for the player.

Comparing with traditional online gaming, it is more difficult to preserve good-enough responsiveness in the cloud gaming system, due to the reason that the network delay and processing delay are essentially required for maintaining the game session. Although keeping the response delay below an acceptable threshold is a priority in the online gaming system, reducing the inter-player delay can be an even more important task [10]. Many successful real-time multiplayer games use specific mechanisms to get rid of high inter-player delay issue [11, 12]. However such methods may fail in the cloud gaming scenario, since the remote game server has no idea of the additional network and processing delay in the cloud gaming system. This means that the responsibility of eliminating the impact of such kinds of delay is left to the cloud gaming service providers.

Generally, the major task in improving the player's QoE in multiplayer gaming is resolving the high inter-player delay issue [6]. The challenge in this Inter-player Delay Optimization (IDO) problem is roughly three-fold,

- Optimizing the inter-player delay among interacting players: this is the main but no easy task since this is a tradeoff between the optimization of both network delay and processing delay among all players;
- Preserving the good-enough absolute response delay: this is a fundamental requirement of maintaining acceptable QoE for players;
- Limitation of available resources of cloud data centers: usually a cloud gaming service provider uses a series of cloud data centers to serve a large scale of players. The resources of physic machines, such as the CPU-time, memory and the storage space, are virtualized as a whole. The optimization of VM placement should obey the limitation of available resources of each DC.

In the following sections, we describe the IDO problem model as well as our approach to get rid of these challenges.

III. IDO: Inter-player Delay Optimization Problem

Since both the network and processing delay are determined by the placement of VMs assigned to players, the IDO problem is essentially a VM placement optimization problem. Intuitively it is similar to the bin packing problem, which is known as an NP-hard problem [13], but we should make several assumptions to convert it into the IDO problem. First we assume a scenario where players are using the common cloud gaming service, but may be served by different cloud data centers. Under this assumption, the network Round-Trip Time (RTT) between each cloud data center and the remote game server can vary in a limited range. The second assumption is that the knowledge of interaction pattern among players can be learned based on the locations of their avatars in the game [14]. We mainly focus on the case where avatars of two players in the same region group up as an interaction pair, and there may be multiple such pairs within this region.

Table 1 lists all the symbols used in the model. We assume that there are N players, indexed by i and j, playing through the cloud gaming service. The cloud gaming service provider operates M Data Centers (DCs), indexed by m. Each of players uses the client provided by the service provider to access one of the DCs, and is then served by a VM assigned by the DC. Following the parameters described above, the IDO problem is formulated as follows,

$$\min\left\{\max_{i,j\in N} g_{ij}\right\} \tag{1}$$

s.t.

$$g_{ij} \geq |D_i - D_j|\, z_{ij}, \qquad \forall i,j \in N; \tag{2}$$

$$r_{im} = n_{im} + p(v_m) + d_m, \quad \forall i \in N, \forall m \in M; \tag{3}$$

$$v_m = \sum_{i=1}^{P} x_{im}, \qquad \forall m \in M; \tag{4}$$

$$D_i = \sum_{m=1}^{M} r_{im} x_{im}, \qquad \forall i \in N; \tag{5}$$

$$p(v) = \frac{\alpha}{1+e^{-\beta v}}, \qquad \forall v \in \mathbb{N}; \tag{6}$$

$$\tilde{D} \geq D_i, \qquad \forall i \in N; \tag{7}$$

$$C_m \geq \sum_{i=1}^{P} c_i x_{im}, \qquad \forall m \in M; \tag{8}$$

$$x_{im} \in \{0,1\}, \qquad \forall i \in N, \forall m \in M; \tag{9}$$

$$1 \geq \sum_{m=1}^{M} x_{im}, \qquad \forall i \in N. \tag{10}$$

The IDO problem described by the objective (1) is to minimize the maximal inter-player delay among all pairs of interacting players. The constraint (2) defines this inter-player delay, i.e. the difference of absolute response delay between two interacting players. The equation (3) defines the calculation of the absolute response delay of player *i* when he is served by center *m*, that is, the response delay is the combination of the network delay between the player's client and the VM, the processing delay of the VM, and the network delay between the VM and the remote game server. In this equation the playout delay is omitted. The reason is that playout delay is too short to significantly affect the absolute response delay, as it usually accounts for only about 10% of processing delay [7]. The equation (4) counts the number of VMs running inside the center *m*. The equation (5) indicates that the final response delay of player *i* depends on the DC his client is connected to. The equation (6) determines the function of calculating the process delay of a center that is running *v* VMs. We uses the sigmoid function revealed by Hong et al [15] to describe the relationship between the number of VMs and the average processing delay of VMs in the DC. The model parameter α and β can be derived from the actual measurements within different scenarios. The constraint (7) defines the maximal acceptable absolute response delay, i.e. the absolute response delay perceived by all players cannot be higher than the threshold $\tilde{D}$. The constraint (8) indicates that the resources used by VMs should not exceed the capability of the DC. The constraint (9) and constraint (10) restricts that each player should be served by at most one DC.

Table I. Symbols and Definitions

Symbols	Definitions
N	Number of players
M	Number of cloud data centers (DCs)
D_i	Response delay experienced by player i
$\tilde{D}$	Maximum tolerable response delay
r_{im}	Response delay when player i is served by DC m
n_{im}	Network delay between player i's client and DC m
d_m	Network delay between DC m and the remote game server
$p(v)$	Processing delay of the DC that is serving v VMs
c_i	Amount of resources required by player i's VM
C_m	Amount of available resources in DC m
z_{ij}	Interact state between player i and player j
g_{ij}	Difference of response delay between player i and player j

It is easy to see that IDO is not a straightforward reduction of the bin packing problem, in the sense that the value (i.e. the inter-player delay) of each object (i.e. players' VMs) not only depends on the bin (i.e. the DC) that it is finally placed in, but also depends on the other objects that are co-placed with it. For instance, consider the case where the placement of a certain player's VM is different in two solutions. In this case, not only the inter-player delay perceived by this player and his partner is different, but also the inter-player delay of every other pair of interacting players. The IDO problem can be resolved by a Dynamic Programming (DP) algorithm, but basically such a DP method faces a great risk to regress to an exhaustive search over solution spaces, which can take huge exponential running time. As such the feasibility of DP is fairly limited due to the real-time requirements, especially when the amount of players increases to a large scale. Therefore we focus on developing an efficient heuristic method to achieve a near-optimal solution to this problem.

IV. Inter-player Delay Optimization algorithm

The IDO problem can be regarded as the combination of two sub-problems: (1) make a proper VM placement plan for every player's VM among all available DCs, in order to minimize the absolute response delay perceived by every player, and (2) make a proper VM placement that can optimize the inter-player delay among all interacting players. According to the equation (3), the absolute response delay of a player is determined by the connecting DC and the VM assigned to him. Therefore, the first sub-problem can be regarded as a reduction of Set-Cover problem[16, 17], where the VMs are customer points and the cloud data center are feasible facilities, and an optimal VM placement is the optimal cover of customer points by feasible centers that minimizes the response delay experienced by each player. The second sub-problem is more tricky since the solution to the first sub-problem may not guarantee the inter-player delay to be minimal. As such, further adjustment of VMs may be needed in order to reduce the inter-player delay between interacting players, which is based on both the interacting state of players and the available sources of DCs. Hereby a near-optimal solution can be achieved following this heuristic local search process.

We propose an efficient heuristic algorithm to tackle this

IDO problem, which consists of two phases to solve the two sub-problems respectively. The first phase is to find out the most suitable placement of all players' VMs, and the second phase is to minimize the inter-player delay between interacting players by adjusting the placement solution produced in the first phase.

A. VM Placement Phase

In this phase IDO algorithm adopts the basic idea of the classic heuristic algorithm for solving the set cover problem [16]. At the first step, IDO generates the *circular convex set* per DC, a subset of all possible covering patterns of VMs by each DC. More precisely, for the DC m, the circular convex set $P_m = \{\{i_1\}, \{i_1, i_2\}, \cdots, \{i_1, i_2, \cdots, i_N\}\}$, where all N VMs $i_1, i_2, \cdots, i_N$ are fully included. Each subset within this circular convex set represents the covering of VMs by the DC m. This circular convex set can be generated in $O(N)$ running time, and is sufficient for composing the near-optimal covering solution [17].

After generating the circular convex sets of all centers, IDO checks if the covering solution represented by a set is feasible, according to the total resource cost of covered VMs and the available source of the DC. If such a solution is valid, IDO computes the *relative covering cost* of each set. This relative covering cost is defined as the sum of the absolute response delay of all players whose VMs are covered by the DC of each set, divided by the number of covered VMs. During each iteration IDO conducts the calculation of relative covering cost, and choose the circular convex set with the minimal cost. When a circular convex set is chosen, IDO removes the VMs it covers from all the other remaining sets, meaning that each VM can only be located in a certain DC. Hereby at least one player will be properly served in each iteration, which makes at most N iterations in total.

B. Adjustment Phase

After the VM placement phase, a near-optimal placement of each player's VM is derived. This VM placement solution however does not guarantee that the inter-player delay between interacting players is minimized, hence further adjustment of this placement solution is necessary. We use a local search strategy to achieve the goal of minimizing the maximal inter-player delay while preserving low-enough absolute response delay.

For each pair of interacting players, IDO tries to place the VM of the player with higher absolute response delay to another feasible DC, and check if the inter-player delay between two players is reduced. IDO also checks if the absolute response delay of the moved VM is still under the threshold $\hat{D}$. In this method IDO tries to find the modification with the minimal inter-player delay between two players in each iteration. Since the interacting groups are disjoint, it takes $O(N)$ running time to go through all pairs of interacting players. During each iteration it takes $O(M)$ running time to check the available DC for each pair of interacting players. Hereby IDO takes $O(MN)$ running times to adjust the VM placement to a near-optimal solution.

Algorithm 1 Inter-player Delay Optimization Algorithm

```
Input: $\{z_{ij}\}$: interact state, $\vec{C_m}$: resources per center $m$,
    $\vec{C_i}$: cost per player $i$, $\{n_{im}\}$: network delay between
    player $i$ and center $m$, $\vec{d_m}$: network delay from center
    $m$ to server
Output: $\{x_{im}\}$
Derive all circular convex sets $P_m$ for each DC $m \in M$,
    set $COVER = \emptyset$
repeat
Remove the circular convex set $P_m$ that does not satisfy
    the resource constraint of DC $m$, based on the inequation (8)
Calculate the relative covering cost $\sum_{i \in P_m} D_i$ of each
    circular convex set $P_m$ for $m \in M$ based on equation
    (3) and (5)
    Find $m_0$ such that
    $\frac{\sum_{i \in P_{m_0}} D_i}{|P_{m_0}|} = \mathbf{max}_{i \in N, m \in M}(\frac{\sum_{i \in P_m} D_i}{|P_m|})$
    $COVER \leftarrow COVER \cup P_{m_0}$
    for $m \leftarrow 1$ to $M$
      $P_m \leftarrow P_m - P_{m_0}$
    end for
until $P_m = \emptyset$ for all $m \in M$
for $i \leftarrow 1$ to $N$
  if $z_{ij} = 1$, find $m_1$ such that $|D_i - D_j|$ is minimized,
    where $i \in P_{m_1}$ or $j \in P_{m_1}$,
  $x_{im_1} \leftarrow 1$ if $i \in P_{m_1}$, $x_{jm_1} \leftarrow 1$ if $j \in P_{m_1}$
end for
return $\{x_{im}\}$
```

The pseudo code of IDO is shown in Algorithm 1. IDO tends to arrange less DCs to support all players, thus VMs are more likely to consolidate in certain DCs. This follows the nature of multiplayer online gaming systems, which adopts server consolidation to reduce the cost of maintaining game contents and sessions [14]. Although some measurement studies show that higher consolidation degree of workloads may not degrade the performance of a single server [18], consolidations at a large scale impact the processing performance of servers in DC, in the sense that consolidating VMs can result in higher processing delay. However, the consolidation degree is properly under control in IDO process since it takes the processing delay into account (as a part of absolute response delay). Hereby IDO is capable of reduce the number of necessary DCs while preserve a tolerable processing delay. In the following section we show the good performance of IDO algorithm with the comprehensive evaluation using the real world data trace, and compare IDO with the other existing mechanism from several perspectives.

V. Performance Evaluations

A. Experiment Setup

We use the well-developed avatar history dataset of World of Warcraft (WoW) as the input of the experiment [9]. This dataset records the avatars' game play times and a number of attributes, such as their races, classes, current level, and locations within the game world, during the period between Jan. 2006 and Jan. 2009. The dataset consists of records of more than 90,000 avatars, which are useful for deriving the interaction pattern between players.

We choose different parts of datasets to establish different scenarios, and we use disjoint sets (where 1000 records are included per set) in each scenario. Based on the intuitive observation that the players co-located in the same region tend to cooperate or fight with each other, players are grouped up mainly based on the location of their avatars. We assume that the interactions among players are pairwise, in order to simulate the most usual interactive behaviors in WoW, such as bargaining and dueling. Furthermore, we also use this dataset to simulate the distribution of DCs of the cloud gaming service provider. We take every region as a game world served by a single DC. As such we establish 100 DCs in the simulations, and the number of DCs varies in different experiments. Two kinds of network delay, i.e. the network delay between the players' clients and DCs, and the network delay between DCs and the game server, are intuitively set to reflect the fact that they can vary dramatically throughout regions and time periods [19]. Basically the network delay between players and DCs ranges from 5 to 200ms, while the network delay between DCs and the remote game server varies from 10 to 50ms.

We use the above setup to evaluate our heuristic algorithm from three perspectives: (1) its capability of minimizing the inter-player delay among interacting players, (2) its capability of preserving a low-enough absolute response delay experienced by every player, and (3) its ability of efficiently arranging DCs to support all players, while preserving low-enough processing delay. From these perspectives, we compare the performances of our IDO algorithm with the Quality-Driven Heuristic algorithm (QDH), a VM placement algorithm that arranges the VMs according to the network delay between players and servers [15].

B. Capability of Minimizing the Inter-player Delay

The major objective of IDO problem is to minimize the inter-player delay among interacting players. In the following experiments, we change the scale of workloads to assess the ability of IDO algorithm to reduce the inter-player delay. More precisely, we conduct experiments in three kinds of scenarios where there are: (1) 100 players and 50 DCs, (2) 500 players and 50 DCs and (3) 1000 Players and 50 DCs. Within each case we randomly choose player records as input. We adopt similar setups in the other following experiments.

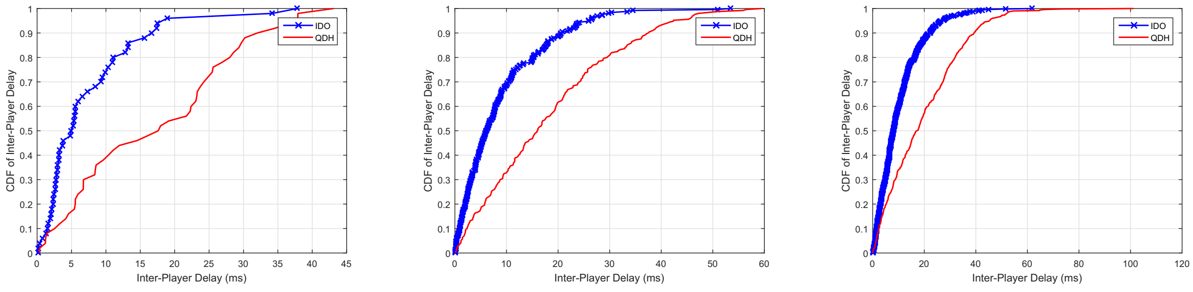
Figure 2 depicts the Cumulative Distribution Function (CDF) of the inter-player delays of interacting players provided by IDO algorithm and QDH within these cases. In all of these cases IDO algorithm significantly outperforms QDH, i.e. IDO algorithm provides the result where inter-player delay among players is low, comparing with the QDH's result. Figure 3 shows that IDO algorithm reduces the maximal inter-player delay by about 30% comparing with QDH. Even when there is a large amount of workload (i.e. 1000 players with 50 DCs), IDO algorithm can still keep the maximal inter-player delay under 70ms, so that the interacting players can hardly perceive the delay gap between them. In the case where there are not so sufficient DCs (i.e. only 10 DCs are used to support 500 players), IDO algorithm performs similarly with QDH, i.e. the relative difference of maximum inter-player delays provided by IDO and QDH is only around 10%. Although IDO algorithm tends to consolidate VMs on a set of DCs, When DCs are not sufficient IDO has no choice but to arrange more DCs to fulfill all requirements. The reason why QDH performs poorer when DCs are sufficient is that it tends to evenly assign VMs to DCs, and it does not take the difference of response delays among players into account.

In further experiments, as the number of player increases, the inter-player delay of interacting players in IDO's result does not vary explicitly. This means that IDO algorithm is able to reach near-optimal solution despite the variation in number of players and DCs. Hereby, comparing with QDH, IDO algorithm is capable of reducing the variance of inter-player delay under different scenarios.

C. Capability of Preserving Low-Enough Absolute Response Delay

Beside of minimizing the inter-player delay, IDO algorithm can also preserve the good-enough responsiveness of the game session, i.e. maintaining the absolute response delay experienced by players below a tolerable threshold. In the following experiments, we vary both the numbers of players and DCs in order to assess IDO's ability to guarantee acceptable absolute response delay under different circumstances. Beside of minimizing the inter-player delay, IDO algorithm can also preserve the good-enough responsiveness of the game session, i.e. maintaining the absolute response delay experienced by players below a tolerable threshold. In the following experiments, we vary both the numbers of players and DCs in order to assess IDO's ability to guarantee acceptable absolute response delay under different circumstances.

Figure 4 illustrates the CDF of the absolute response delays of interacting players provided by the IDO algorithm and QDH. As can be seem from the results, IDO algorithm still generally outperforms QDH when the workload is light (i.e. 100 players with 50 DCs), in the sense that more players perceive lower absolute response delay. However,

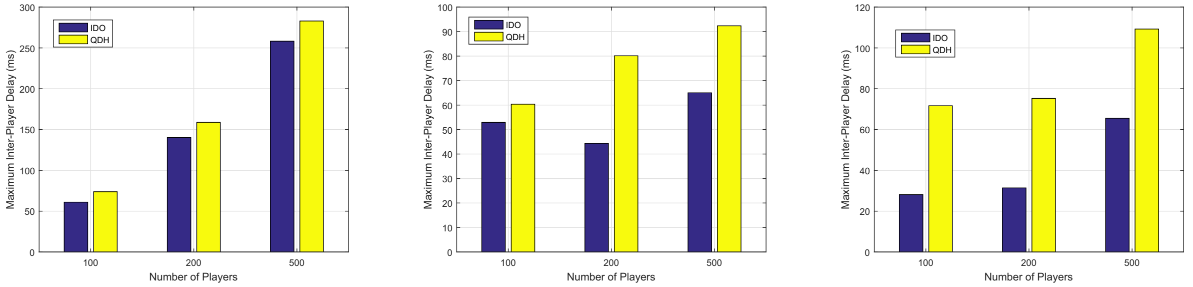


(a) 100 Players and 50 DCs

(b) 500 Players and 50 DCs

(c) 1000 Players and 50 DCs

Figure 2. CDF of inter-player delays Under Different Workloads



(a) 10 DCs

(b) 50 DCs

(c) 100 DCs

Figure 3. Maximum inter-player delays Under Different Workloads

when the number of players increases, the minimal absolute response delay provided by IDO algorithm significantly increases, as can be seem from Figure 4(b) and Figure 4(c). In these two cases QDH can reach a better lower-bound and similar higher-bound with the result of IDO algorithm in the worst case. The reason is that IDO tends to make the absolute response delay perceived by two interacting players as similar as possible, this means that a player who can have better performance has to regress to his partner in order to preserve the fairness between them. This result is in consistence with the purpose of IDO algorithm. Furthermore, even in case where workload is heavy (e.g. 1000 players with 50 DCs), IDO still keeps the maximal absolute response delay below 230ms. Given that the threshold of tolerable absolute response delay in a massive multiplayer game is 500ms, this result is relatively acceptable.

D. Efficientiveness of DC Usage and Preserving Processing Delay

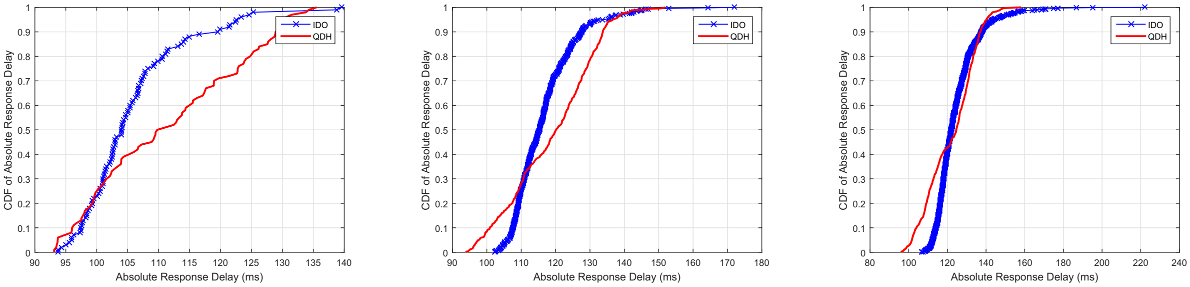
IDO algorithm tends to consolidate VMs over a set of DCs, i.e. it is capable of reducing the number of necessary DC to support all players. However, this strategy may lead to a risk of increasing the processing delay, since the processing delay is positively relative to the number of VMs [15]. In the following experiments we vary the number of players to quantify the ability of IDO algorithm to preserve the processing delay. We also change the number of available DCs, intending to verify that if IDO can efficiently make use of DCs.

Figure 5 depicts the processing delay under scenarios with different workloads, i.e. the different scale of players' VMs. Comparing with QDH, IDO algorithm does not preserve the change of processing delay within a small range. However the evaluation results still show that IDO algorithm can maintain the processing delay under a tolerable threshold. Even in a scenario at scale, i.e. where 1000 players are interacting through 50 DCs, the IDO algorithm can keep the maximal processing delay below 110ms, which is relatively acceptable in the case where the workload is heavy.

Furthermore, Figure 6 shows that IDO algorithm can always use as less DCs as possible to support players, while QDH tends to use more DCs since it adopts a fairness strategy among DCs. This means that our algorithm can save the necessary resource cost of supporting a large number of players.

VI. Related Work

Optimizing VM placement. Specifically, there are majorly two studies closely related to our work on the topic of VM placement optimization. Wang et al [10] propose a Cloud-Based Distributed Interactive Applications (CDIA) framework and a bi-level heuristic algorithm that solving

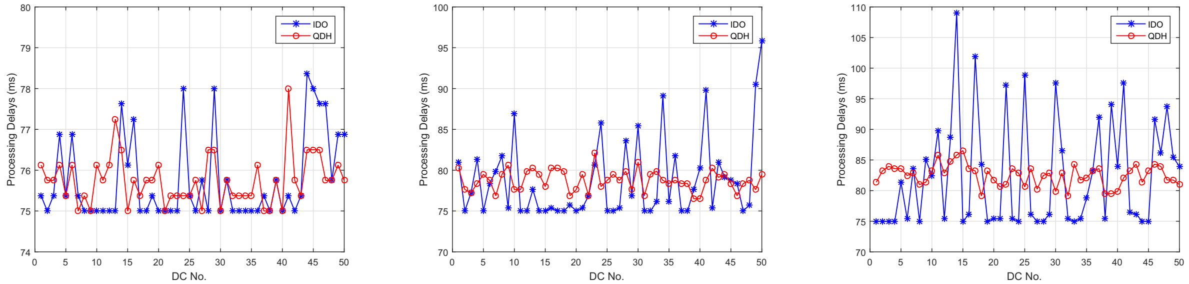


(a) 100 Players and 50 DCs

(b) 500 Players and 50 DCs

(c) 1000 Players and 50 DCs

Figure 4. CDF of Absolute Response Delay Under Different Workloads



(a) 100 Players and 50 DCs

(b) 500 Players and 50 DCs

(c) 1000 Players and 50 DCs

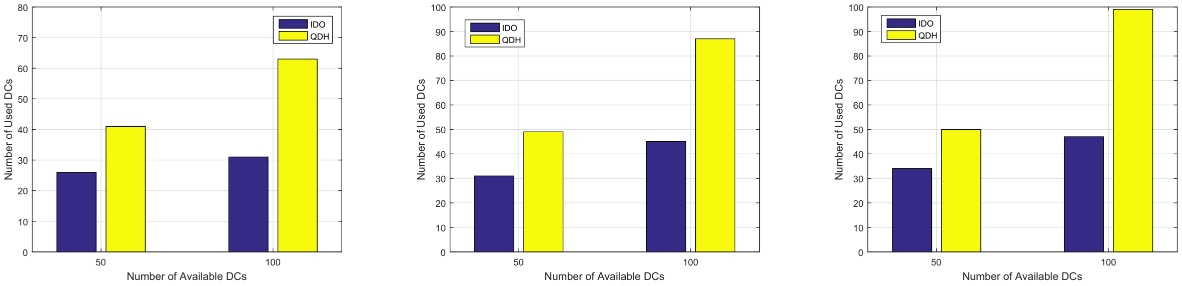
Figure 5. Processing Delay Under Different Workloads

the optimization problem of VM placement within this framework. They aim at reducing the interact latency between interacting VMs, and adopt a two-step method to partition this problem into load assignment and the resource assignment. Our works differs from this work in the sense that in the IDO problem scenario, the VMs of two players are not directly communicating with each other, but communicates through the remote game server, just as the hub-and-spoke communication model in traditional multiplayer online gaming. In addition, the solution proposed by Wang et al do not take the absolute response delay of every user into account, while our IDO algorithm will preserve low-enough absolute response delay for players. The other study is proposed by Hong et al [15], where they proposes a VM placement algorithm that can minimize the management and operation cost of the cloud gaming platform while preserve good-enough experience. However, although both our work and theirs are capable of reducing the necessary resources, they mainly focus on increasing the service provider's profit, while we aim at improving the players' gaming experience.

Previous studies have made a comparison on various VM placement algorithms from several perspectives [20]. Meng et al [21] propose the VM placement algorithm that optimizes the routing for inter-VM communication. Li et al [22] study the VM placement optimization problem over a set of physical machines (i.e. cloud servers), in order to reduce the number of necessary machines and the cost of network traffic among them. Calcavecchia et al [23] provide a model of VM placement that adopts statistical multiplexing and analysis of historical workload traces to assign VMs to certain physical machines. Tian et al [24] propose a VM placement and migration algorithm to reduce the cost of data centers by adaptively adjusting the assignment of data centers, virtual machines and video bitrate configurations for users.

Reducing processing delay. Chuah et al [25] propose a massive multiplayer online gaming platform based on cloud gaming system that uses the render ability of player's mobile device, so that both the network delay and processing delay are reduced. Choy et al [26] present the smart edge to reduce the initialization of VMs when players issue the gaming request, in order to reduce the processing delay on the server side. Cai et al [27] provide a multiplayer cloud gaming platform that takes advantage of peer-to-peer connection to share the jointing frames of the common game scenes, so that the processing delay at server side is partially hided by early frames delivered by other players.

As a complement of existing work, our work mainly focus on reducing the inter-player delay among interacting players, meanwhile maintaining the absolute response delay perceived by every player below the tolerable threshold.



(a) 100 Players

(b) 200 Players

(c) 500 Players

Figure 6. Number of Used DCs Under Different Workloads

VII. Conclusion

In this paper, we first identify and formulate the Inter-player Delay Optimization (IDO) problem that aims at minimizing the inter-player delay among interacting players, while preserving good-enough absolute response delay perceived by players. We further propose the heuristic IDO algorithm to efficiently reach the near-optimal solution to this problem. The comprehensive evaluation using a large-scale real-world data trace shows that our IDO algorithm can reduce up to about 30% maximal inter-player delay comparing with the other existing algorithm. Also the evaluation result shows that our IDO algorithm can help reduce the number of necessary DCs while supporting players at a large scale.

Acknowledgement

This publication was made possible by NPRP grant # [8-519-1-108] from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

Reference

[1] W. Cai et al. The future of cloud gaming [point of view]. *Proceedings of the IEEE*, 104(4):687–691, 2016.

[2] K. Chen et al. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2):480–495, 2014.

[3] R. Shea et al. Cloud gaming: architecture and performance. *IEEE Network*, 27(4):16–21, 2013.

[4] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[5] S. Zander et al. Achieving fairness in multiplayer network games through automated latency balancing. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 117–124. ACM, 2005.

[6] T. Henderson. Latency and user behaviour on a multiplayer game server. In *Networked Group Communication*, pages 1–13. Springer, 2001.

[7] K. Chen et al. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.

[8] Z. Wen et al. Qoe-driven performance analysis of cloud gaming services. In *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2014.

[9] Y. Lee et al. World of warcraft avatar history dataset. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 123–128. ACM, 2011.

[10] H. Wang et al. On design and performance of cloud-based distributed interactive applications. In *2014 IEEE 22nd International Conference on Network Protocols (ICNP)*, pages 37–46. IEEE, 2014.

[11] J. Färber. Network game traffic modelling. In *Proceedings of the 1st workshop on Network and system support for games*, pages 53–57. ACM, 2002.

[12] Yahn W Bernier. Latency compensating methods in client/server in-game protocol design and optimization. In *Game Developers Conference*, volume 98033, 2001.

[13] G. Michael and J. David. Computers and intractability: a guide to the theory of np-completeness. *WH Free. Co., San Fr*, 1979.

[14] Y. Lee and K. Chen. Is server consolidation beneficial to mmorpg? a case study of warcraft. In *2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 435–442. IEEE, 2010.

[15] H. Hong et al. Placing virtual machines to optimize cloud gaming experience. *IEEE Transactions on Cloud Computing*, 3(1):42–53, 2015.

[16] D. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on computing*, 11(3):555–556, 1982.

[17] D. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical programming*, 22(1):148–162, 1982.

[18] H. Hong et al. Gpu consolidation for cloud games: Are we there yet? In *Proceedings of the 13th Annual Workshop on Network and Systems Support for Games*, page 3. IEEE Press, 2014.

[19] J. Kim et al. Traffic characteristics of a massively multi-player online role playing game. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8. ACM, 2005.

[20] K. Mills et al. Comparing vm-placement algorithms for on-demand clouds. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 91–98. IEEE, 2011.

[21] X. Meng et al. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM*, pages 1–9. IEEE, 2010.

[22] X. Li et al. Let's stay together: Towards traffic aware virtual machine placement in data centers. In *INFOCOM*, pages 1842–1850. IEEE, 2014.

[23] M. Calcavecchia et al. Vm placement strategies for cloud scenarios. In *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 852–859. IEEE, 2012.

[24] H. Tian et al. On achieving cost-effective adaptive cloud gaming in geo-distributed data centers. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2064–2077, 2015.

[25] S. Chuah et al. Cloud gaming: a green solution to massive multiplayer online games. *Wireless Communications, IEEE*, 21(4):78–87, 2014.

[26] S. Choy et al. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems*, 20(5):503–519, 2014.

[27] W. Cai and V. Leung. Multiplayer cloud gaming system with cooperative video sharing. In *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 640–645. IEEE, 2012.