Play Request Dispatching for Efficient Virtual Machine Usage in Cloud Gaming

Yusen Li, Xueyan Tang, *Senior Member, IEEE,* and Wentong Cai, *Member, IEEE*

***Abstract*—Cloud gaming is becoming increasingly popular. The basic idea of cloud gaming is to run games on cloud servers and let players interact with games through thin clients. As the player population grows, the cloud gaming service provider needs to maintain a large number of cloud servers for running the game instances requested by the players. A primary concern of the cloud gaming service provider is the total running cost of the cloud servers. In this paper, we study the problem of how to dispatch the play requests to the cloud servers in a cloud gaming system. We show that the dispatching strategy of play requests may heavily affect the total service cost of the cloud gaming system. The play request dispatching problem can be considered as a variant of the dynamic bin packing problem. However, we show that the classical bin packing algorithms such as First Fit and Best Fit are not efficient in terms of resource usage in cloud gaming due to the diurnal workload pattern of online games. To address this issue, we propose an efficient request dispatching algorithm that assigns play requests according to the predicted ending times of game sessions. We also assess several classes of prediction algorithms and select a neural-network based algorithm to predict the ending times of game sessions. We conduct extensive evaluations of the proposed algorithms using real traces from different types of online games. The experimental results show that the proposed dispatching algorithm with neural-network based prediction can reduce the resource waste of the cloud servers and thus decrease the total service cost compared to the First Fit and Best Fit algorithms. The reduction in the resource waste is particularly significant for match-based games such as DotA and World of Tank.**

***Index Terms*—cloud gaming, service cost, play request dispatching, bin packing.**

I. Introduction

Recently, cloud gaming has attracted a great deal of interests from both academia and industry [1], [2]. In a cloud gaming system (see Figure 1), the games run on cloud servers and the players interact with games through thin clients. Specifically, the cloud servers run the game instances, render the 3D graphics, encode them into 2D videos, and stream them to the thin clients. The thin clients then decode and display the video streams. In this way, players can easily play games without installing them locally. Moreover, players can play GPU intensive games on their computers, tablets or even mobile phones without dedicated hardware equipped. Many companies have started to offer cloud gaming services, such as OnLive [1] and GaiKai [3]. The cloud gaming market has been forecasted to reach 8 billion US dollars in 2017 [4].

Yusen Li, Xueyan Tang and Wentong Cai are with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue 50, 639798, Singapore. E-mail: {S080007@e.ntu.edu.sg, ASXYTANG@ntu.edu.sg, ASWTCAI@ntu.edu.sg}

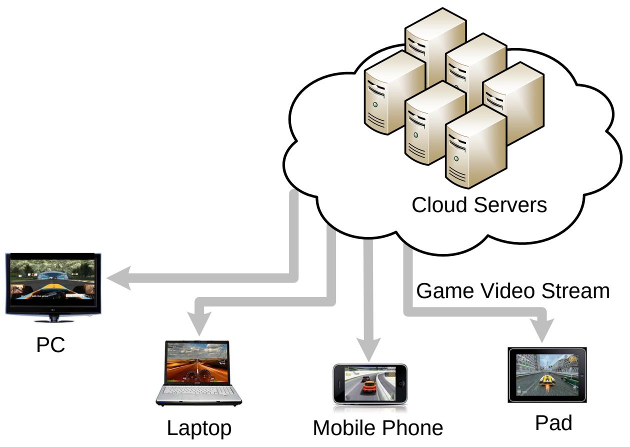


Fig. 1. Cloud Gaming

Running a game instance requires a certain amount of computational resources. Therefore, the number of game instances that can run concurrently on a cloud server is dependent on the server capacity and the resource requirement of games. In order to serve a large player population, the cloud gaming service provider needs to prepare sufficient number of cloud servers for running the game instances. However, the high workload variability of online gaming makes server provisioning a challenging issue. As shown by empirical data (see Figure 3 in Section III), the number of active players varies greatly over the day. Thus, setting up too many servers would lead to resource waste during slow gaming times whereas not maintaining enough servers would result in servers becoming overloaded during peak gaming times. The on-demand resource provisioning service in public clouds provides a promising solution to the above problem. For example, Amazon EC2 provides g2.2xlarge virtual machine instances intended for graphics-intensive GPU computing applications such as cloud gaming. The users can rent the resources (i.e., virtual machines) on an as-needed basis in response to workload variation and pay for only the resources that they actually use. This approach frees the users from the complexities of purchasing, engineering and maintaining hardware infrastructures. The famous cloud gaming company GaiKai [3] has used two public clouds [5].

Consider a cloud gaming system that rents virtual machines to serve as cloud servers. A main concern of the service provider is how to minimize the total renting cost of the virtual machines used. In general, the total renting cost is proportional to the total running hours of all the virtual machines used.

Given a workload, we shall show that the total running hours of all the virtual machines used is highly dependent on how the play requests of the players are assigned to the virtual machines. In our previous work [6], [7], we have modeled the problem of dispatching play requests to cloud servers for minimizing the total renting cost as a variant of the dynamic bin packing (DBP) problem, which is named the MinTotal DBP problem. We theoretically analyzed the worst-case performance of the classical bin packing algorithms such as First Fit and Best Fit for the MinTotal DBP problem. First Fit attempts to put a new item into the first bin (i.e., the earliest opened bin) that can accommodate the item, while Best Fit attempts to assign a new item to the bin with the smallest residual capacity that can accommodate it. First Fit and Best Fit are easy to implement in that they make decisions based on the current system state only. However, in this paper, we show that First Fit and Best Fit bin packing algorithms may lead to high resource waste of the virtual machines in cloud gaming due to the diurnal workload pattern of online games.

This paper considerably extends a preliminary conference version [8]. The contributions of this paper are as follows. We propose an efficient play request dispatching algorithm which assigns play requests according to the predicted ending times of game sessions. The ending times of game sessions may be predicted according to user habits, historical user behaviors, or based on the expected game session length etc. With the predicted ending times, our request dispatching algorithm aims to "pack" the game sessions that have similar ending times to the same virtual machine, so that the virtual machine can keep a high level of resource utilization during its running hours. We also assess several classes of prediction algorithms. Based on the results, we choose a neural-network based approach for predicting the ending times of game sessions. We evaluate the proposed dispatching algorithm using real traces from different types of online games. The experimental results show that the dispatching algorithm with neural-network based prediction can reduce the resource waste of the virtual machines, thereby decreasing the total running hours compared to First Fit and Best Fit. The reduction in the resource waste is particularly significant for match-based games.

The rest of this paper is structured as follows. Section II summarizes the related work. Section III introduces the play request dispatching problem in cloud gaming and evaluates the performance of the classical bin packing algorithms. Our proposed play request dispatching algorithm is presented in Section IV. In Section V, we evaluate several classes of prediction algorithms for predicting the ending times of game sessions. In Section VI, we evaluate the proposed dispatching algorithm with neural-network based prediction using real game traces. Finally, conclusions are made and future work is discussed in the Section VII.

II. Related Works

Cloud gaming systems have been implemented for both commercial use and research studies [1], [2], [9]. GamingAnyWhere [2] is the first open source cloud gaming system, which provides a complete cloud gaming testbed for researchers and developers. OnLive [1] is the first company offering a commercial cloud gaming platform. As mentioned in the introduction, the basic idea of cloud gaming is to render games on cloud servers and stream videos to thin clients. A large amount of research work has been conducted towards measuring and optimizing the video-based cloud gaming solution. The measurement works have mainly concentrated on measuring the latency and network traffic of the existing commercial cloud gaming platforms [10], [11], [12], [13]. The optimization works have mainly focused on video encoding techniques and graphic rendering techniques for bit rate reduction [14], [15], [16], [17], [18].

In the area of resource provisioning, extensive research has been conducted on assigning players among multiple game servers in large-scale online games, with objectives of enhancing the interactivity and consistency [19], [20], [21], [22]. However, not much work has been done on the resource management issues in cloud gaming systems. Wu *et al.* [23] studied the request dispatching and server provisioning issues in a multi-region multi-datacenter cloud gaming system. The objective of their work is to reduce queueing delay and response delay. Hong *et al.* [24] considered how to consolidate the game instances on the physical servers to strike a balance between the Quality of Experience (QoE) perceived by players and the net profit of the service provider. They found that both the QoE and the provider's profit are highly dependent on how the virtual machines are placed on the physical servers. Some heuristic algorithms were proposed to maximize the provider's profit while guaranteeing the QoE. However, it was assumed that the operational cost of the cloud gaming service provider is proportional to the CPU and GPU utilizations of its servers. This is valid when the service provider maintains its own server infrastructure. But it does not really match the pay-as-you-go billing method of the on-demand resources offered by public clouds, which charges the customers based on the total running hours of all the virtual machines used. If the service provider rents server resources from public clouds, the servers used would be charged according to their running hours regardless of their utilizations. Thus, to save the total renting cost, it is important to reduce the total running hours of the servers.

The play request dispatching problem studied in this paper is related to the bin packing and interval scheduling problems. The classical bin packing problem aims to pack the items into the least number of bins. The problem and its variations have been studied extensively in both the offline and online versions [25], [26]. It is well known that even the offline version of the classical bin packing problem is NP-hard [27]. A variant of the classical bin packing problem is dynamic bin packing (DBP) [28]. This generalization assumes that items may arrive and depart at arbitrary times. The objective is to minimize the maximum number of bins concurrently used over time. The play request dispatching problem considered in this paper is a case of dynamic bin packing. However, we study the problem from a different perspective in that we aim to minimize the total renting cost. In our previous work [6], [7], we have formulated the play request dispatching problem for minimizing the total service cost of a cloud gaming system that rents servers from public clouds. We modeled the problem as a MinTotal DBP problem that aims to minimize the total cost of the bins used over time and theoretically analyzed the worst-case performance of the classical bin packing algorithms such as Best Fit and First Fit for the MinTotal DBP problem. However, no experimental evaluation was conducted using real gaming workloads. The interval scheduling problem [29] is also related to our play request dispatching problem. The classical interval scheduling problem considers a set of jobs, each associated with an interval in which the job should be processed. Each machine can process only a single job at any time. Given a fixed number of machines, the objective is to schedule a maximum feasible subset of jobs [30]. However, the online version of this problem has seldom been studied.

Our work is also related to the resource consolidation issues in cloud computing [31], [32], [33], [34], [35]. The works in [31], [33], [34] addressed the problems of assigning virtual machines to physical host machines in a cloud so that the resource utilization is maximized or the energy consumption is minimized. This problem is generally formulated as a bin packing problem and various approximation algorithms have been proposed. However, the MinTotal DBP model has never been considered in any of the existing works. Meng *et al.* [32] addressed the scalability issue of data center networks with network-aware virtual machine placement. It has been shown that a careful virtual machine placement can localize large traffic flows and thus reduce load at high-level switches in a data center network. They formulated the virtual machine placement problem as an optimization problem and proved its hardness. A two-tier algorithm was proposed to obtain an approximate solution. However, the problem formulated did not consider optimizing the total renting cost of the virtual machines. Xu *et al.* [35] formulated a multi-objective virtual machine placement problem. The problem aims to simultaneously optimize several objectives such as making efficient usage of multidimensional resources, avoiding hotspots, and reducing energy consumption. A modified genetic algorithm was proposed to deal with the huge solution space for large-scale data centers. However, only the offline scenario was considered in this paper and the online version of the problem was not studied.

III. Classical Bin Packing Algorithms for Play Request Dispatching

Suppose the cloud gaming service provider rents virtual machines as the cloud servers. When a play request is received, it should be assigned to an active virtual machine that has enough computational resources to run the game instance of this request. Several game instances may share the same virtual machine as long as the computational resources of the virtual machine are not saturated. If no active virtual machine is able to accommodate the play request, a new virtual machine should be started. In general, once a game instance starts, it will run on the same virtual machine during the entire game session. The migration of game instances from one virtual machine to another is not preferable due to large migration overheads and interruption to game play [24]. A virtual machine can be shut down and released for saving cost if no game instance is running on it.

In the online scenario, each play request must be dispatched immediately upon arrival and without any knowledge of the future play request arrivals. For a given set of play requests, the dispatching strategy directly affects the total running hours of the virtual machines used to serve all the play requests. Consider the example shown in Figure 2. Suppose for simplicity that all the play requests have the same computational resource requirements. Assume each virtual machine is able to host at most 2 game instances concurrently. Let each play request (or game session) be represented by a pair: (*starting time*, *ending time*). Assume there are three play requests to be served, which are represented by $r_1 = (0, 10)$, $r_2 = (0, 1)$, $r_3 = (0, 10)$. Since the three play requests arrive at the same time and two virtual machines are needed to host them. Suppose r_1 and r_2 are packed into the same virtual machine, and r_3 is assigned to the other virtual machine (see Figure 2(a)). In this case, the running hours of the virtual machine serving r_1 and r_2 would be $\max\{1, 10\} = 10$, and the running hours of the virtual machine serving r_3 would also be 10. Therefore, the total running hours of the two virtual machines is $10 + 10 = 20$. On the other hand, if r_1 and r_3 are packed into the same virtual machine, then r_2 would be assigned to the other virtual machine (see Figure 2(b)). In this case, the running hours of the virtual machine serving r_1 and r_3 would be 10, and the running hours of the virtual machine serving r_2 would be 1. Thus, the total running hours of the two virtual machines is $10 + 1 = 11$. It can be seen from the above example that the total running hours of the virtual machines is highly dependent on how the play requests are dispatched.

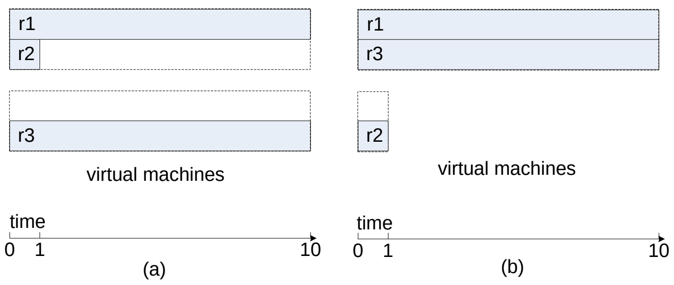


Fig. 2. Examples of request dispatching

A. Classical Bin Packing Algorithms

Since the play request dispatching is a variant of the bin packing problem, we first examine how the classical bin packing algorithms perform for this problem. We conducted some simulations to evaluate two most commonly used bin packing algorithms: First Fit and Best Fit. We consider the following implementations of First Fit and Best Fit for play request dispatching in cloud gaming:

- **First Fit Dispatching**: Each time a play request arrives, First Fit tries to assign it to the earliest started active virtual machine that has enough resources to run the game instance for the request. If no such virtual machine is found, a new virtual machine is started and the request is assigned to the new virtual machine.

- **Best Fit Dispatching**: Each time a play request arrives, Best Fit tries to assign it to the "best" active virtual machine, which is the one with the smallest amount of spare resources that can accommodate the game instance for the request. If no such virtual machine is found, a new virtual machine is started and the request is assigned to the new virtual machine.

We implemented an event-driven simulator to simulate the play request dispatching process by First Fit and Best Fit. In order to make the simulations more realistic, we use the real trace data from online games including the World of Warcraft Avatar History (WoWAH) dataset [36], the DotAlicious dataset, and the World of Tank (WoT) dataset [37].

B. WoWAH Dataset

World of Warcraft (WoW) is the most popular MMORPG which has over 12 million subscribers [38]. Hence, the data of WoW is often studied by researchers from various areas [39], [40], [41]. The WoWAH dataset records continual observations of the status of all players in a WoW realm in Taiwan over a 3-year period from January 2006 to January 2009. During the monitored period, 91065 players and 667032 game sessions associated with the players were observed. For each game session, the following information is recorded: player identifier, level, race, class, and zone.

C. DotAlicious Dataset

Defense of the Ancients (DotA) is an archetypal multiplayer online battle arena game. The players are divided into two teams with 5 players each. Each player controls an in-game avatar. In the match, each team tries to conquer the opponent team's main building. The DotAlicious dataset is collected from the DotA community. The DotA community has its own game servers and maintains lists of tournaments and results. It publishes information such as player rankings. The DotAlicious dataset contains 617069 matches that took place between April 2010 and February 2012. For each match, the start time, the duration and the community identifiers of the 10 participating players are recorded.

D. WoT Dataset

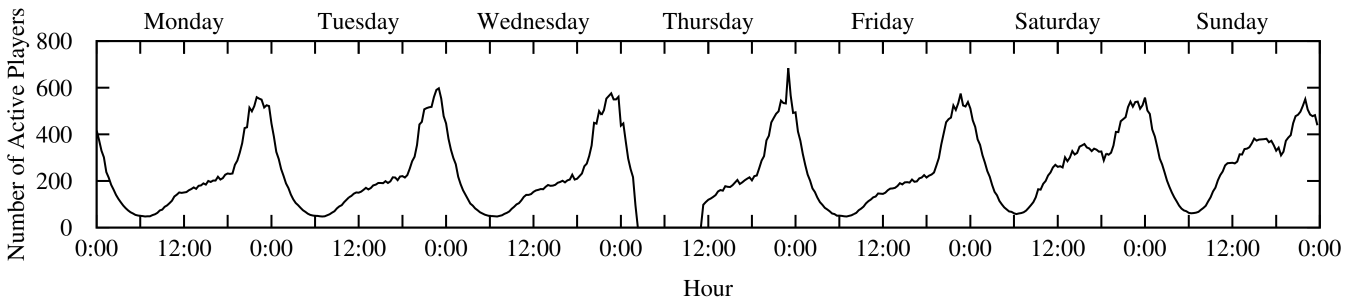
World of Tank (WoT) is one of the most popular Massively Multiplayer Online First Person Shooter games. The WoT dataset contains over 76000 matches which were played from August 2010 to July 2013. All the matches are team battles and there are 30 participating players in each match which are divided into two equal sized teams. For each match, the match duration and the map used are recorded.

Since DotA and WoT are matched-based games, we generate a separate game session for each participating player for each match in the DotAlicious dataset and the WoT dataset. This is because a separate instance needs to run in the cloud server for rendering graphics and video encoding for each player. We assume each player has a random waiting time between 0 to 60 seconds for matchmaking with other players before the match starts. Therefore, the start time of the game session for each player is set as: $session\_start\_time = match\_start\_time - \delta$, where δ is randomly generated from a uniform distribution [0, 60] seconds. The ending time of the game session for each player is given by the ending time of the match in which the player participated.

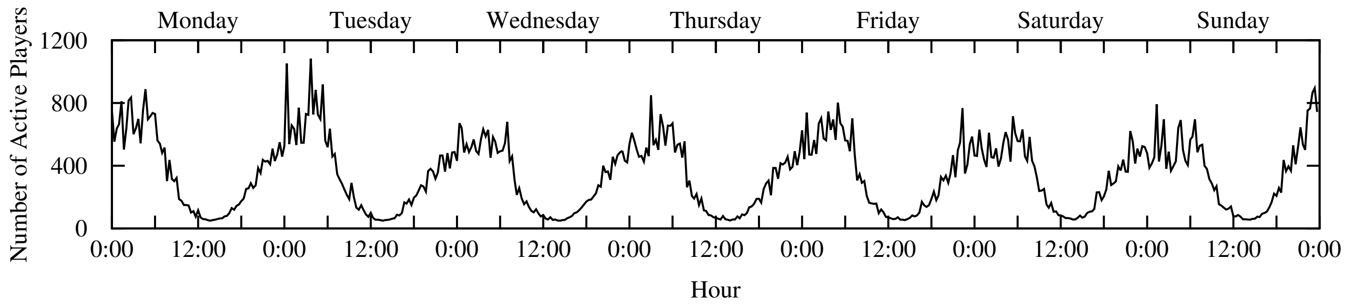
E. Results

In the simulations, each game session represents a play request in cloud gaming. We assume that all the game sessions have the same computational resource requirements and each virtual machine is able to run up to 4 game instances simultaneously (a normal GPU generally can support $3 \sim 5$ game instances concurrently [42]). Figure 3 shows the average numbers of active players as a function of time over different days of the week for the entire WoWAH dataset, DotAlicious dataset and WoT dataset. It is easy to see the diurnal workload pattern in all the datasets (The blackout period on Thursday morning in the WoWAH dataset is due to weekly maintenance downtime scheduled by the operator). According to how the number of active game sessions changes, the 24 hours of a day can be divided into two periods: a climbing period (e.g., from 7:00 to 23:00 in the WoWAH dataset) and a declining period (e.g., from 23:00 to 7:00 of the next day in the WoWAH dataset). The number of active players generally increases during the climbing period and decreases during the declining period.

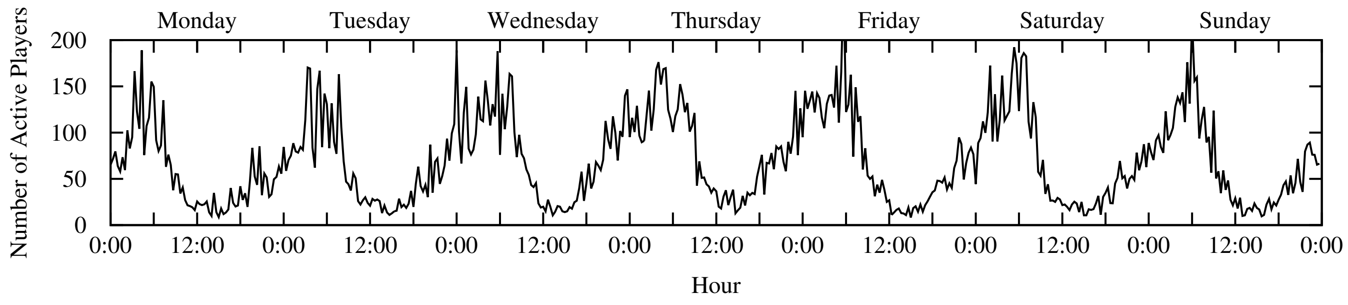
Figure 4 shows the instantaneous numbers of virtual machines used by different dispatching algorithms over a sample day for the WoWAH dataset, the DotAlicious dataset and the WoT dataset. Similar performance trends are observed for other days. The "Optimal" curve represents the minimum number of virtual machines required for accommodating all the active game sessions, which is calculated by dividing the total number of active game sessions by the virtual machine capacity (i.e., 4 game sessions on each virtual machine). As can be seen from Figure 4, the numbers of virtual machines used by First Fit (FF) and Best Fit (BF) dispatching are very close to Optimal during the climbing period. This is because there are more play request arrivals than departures in the climbing period. In this case, no matter how the play requests are assigned, all the active virtual machines are almost fully occupied by the continuously arriving play requests. However, in the declining period, First Fit and Best Fit occupy many more virtual machines than Optimal. It implies that the active virtual machines under First Fit and Best Fit dispatching are not fully utilized in the declining period. This is because there are more player departures than arrivals in the declining period. If the players on the same virtual machine do not leave the system at the same time, the "leftover" game sessions would cause many virtual machines to run at low levels of resource utilization so that these virtual machines cannot be shut down timely. As indicated in Figure 4(a), the area between the "Optimal" curve and the First Fit (or Best Fit) curve represents the "wasted" running hours of virtual machines under First Fit (or Best Fit) dispatching. It can be seen that



(a) WoWAH dataset



(b) DotAlicious dataset



(c) WoT dataset

Fig. 3. Average number of active players as a function of time over different days of the week

there exist a large number of wasted running hours for both First Fit and Best Fit dispatching.

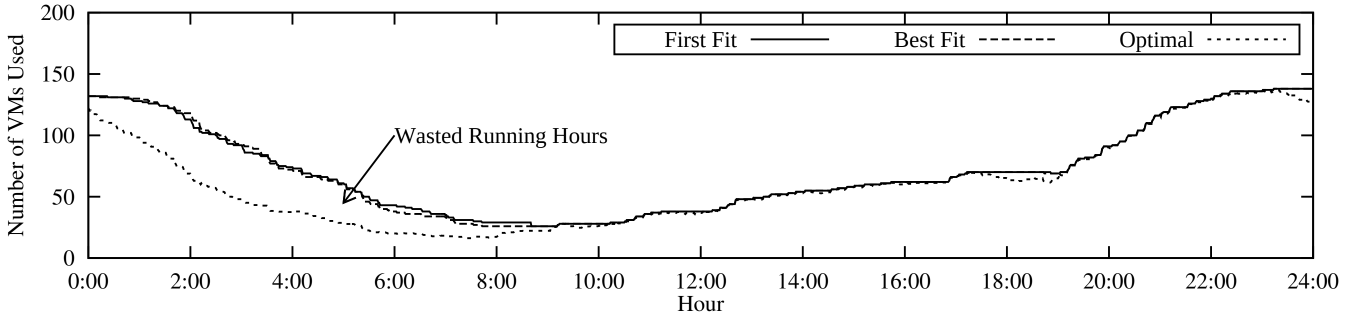
IV. Ending Time based Play Request Dispatching

In this section, we propose an efficient play request dispatching algorithm called ET that can greatly reduce the wasted running hours compared to First Fit and Best Fit. The basic idea is to assign play requests according to the predicted ending times of game sessions, and pack those game sessions that have "close" ending times into the same virtual machine. In this way, the game sessions assigned to the same virtual machine are expected to complete at around the same time. Therefore, each virtual machine can keep a high level of resource utilization during its running hours and can be shut down timely after all of its game sessions end. The detailed dispatching algorithm is described in Algorithm 1.

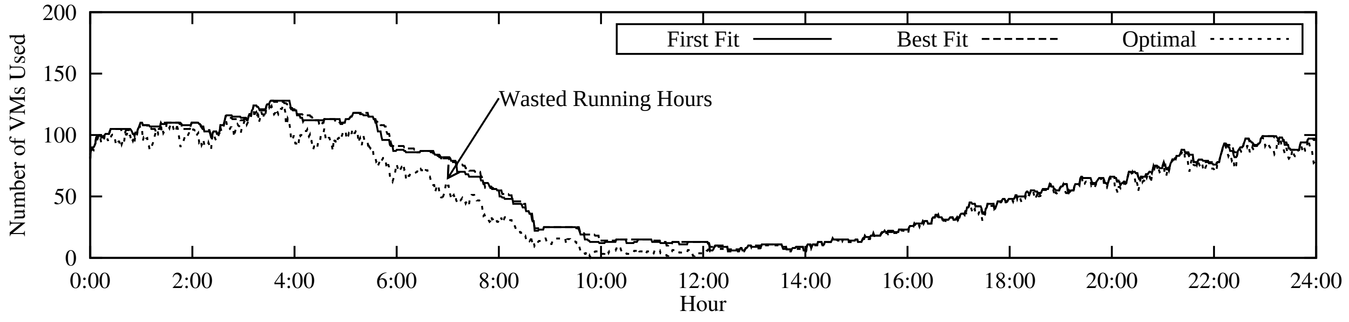
Let r denote the play request to be dispatched. We first predict the ending time of r's game session, which is denoted by $e(r)$ (line 1). If there is no active virtual machine that has enough computational resources to serve r, a new virtual machine needs to be started and the play request r is assigned to the new virtual machine (lines 3-4). Otherwise, r would

Algorithm 1 Ending Time based Play Request Dispatching (ET)

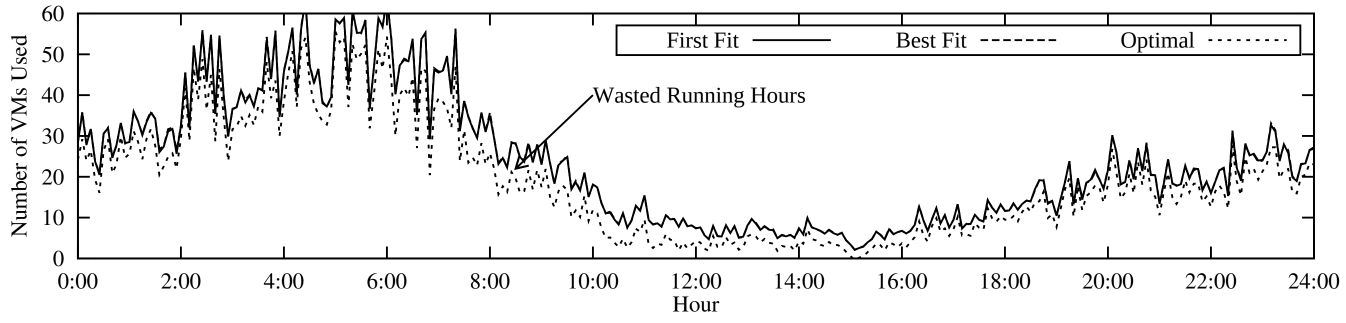
```
1: Predict $e(r)$ for the play request $r$
2: Let $V$ denote the set of active virtual machines that have
   enough computational resources to run the game instance
   for $r$
3: **if** $V = \emptyset$ **then**
4:     Start a new virtual machine and assign $r$ to the new
       virtual machine
5: **else**
6:     Let $V' \subseteq V$ denote the set of all the virtual machines
       $v$ such that $T(v) > e(r)$
7:     **if** $V' \neq \emptyset$ **then**
8:         Assign $r$ to any of the virtual machines in $V'$
9:     **else**
10:        Assign $r$ to the virtual machine $v^*$ such that $T(v^*)$
           is the latest (largest) among all the virtual machines
           in $V$
11:    **end if**
12: **end if**
```



(a) WoWAH dataset



(b) DotAlicious dataset



(c) WoT dataset

Fig. 4. Performance of First Fit and Best Fit dispatching for a sample day

be assigned to one of the active virtual machines that have sufficient spare computational resources. For each active virtual machine v, let $T(v)$ denote the *expected* shutdown time of v, which is defined as the latest predicted ending time of all the game sessions currently running on v. Among all the active virtual machines that can accommodate r, we first consider the virtual machines whose expected shutdown times are later than $e(r)$ (lines 6-8). The rationale behind is that if r is assigned to a virtual machine whose expected shutdown time is later than $e(r)$, the total running hours of the virtual machines will not be increased after the assignment. This is because the virtual machine serving r needs to keep active at least until its expected shutdown time no matter whether r is assigned to it or not. If all the active virtual machines that can accommodate r have their expected shutdown times earlier than $e(r)$, r is then assigned to the virtual machine that has the latest expected shutdown time. The motivation here is to minimize the increase of the total running hours after the assignment (line 10).

V. Ending Times Prediction

It is intuitive that the performance of the ET dispatching algorithm is dependent on how well the ending times of the game sessions are predicted. In this section, we present the evaluations of several classes of prediction approaches. Instead of predicting ending times directly, we focus on predicting the game session lengths. This is equivalent because the ending time of a game session can be calculated by the session length.

We first define the prediction problem. With the consideration of the diversity of user habits, we perform the prediction of the session lengths separately for each player. Suppose a player has t past game sessions which are denoted by s_1, s_2, ..., s_t, where s_i is the i^{th} game session in order of arrival. Denote by $r(s_i)$ the actual session length of s_i. Our goal is to predict the length of next game session (denoted by s_{t+1}) when a new play request of the player arrives.

A. Prediction Algorithms

Quantitative prediction can generally be divided into two classes: time series and explanatory models [43]. Time se-

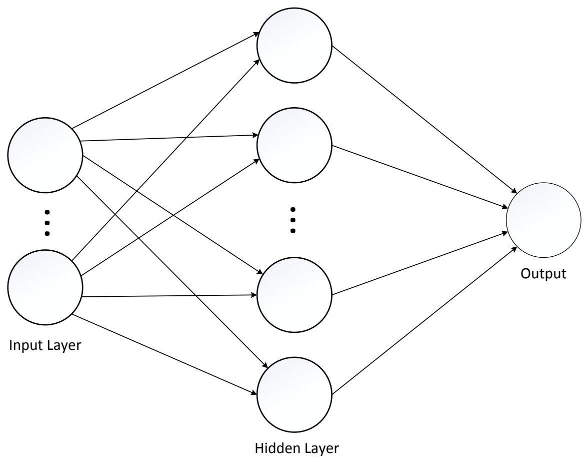


Fig. 5. Example of Multilayer Perception Model

ries prediction uses only information on the variables to be predicted. For example, time series may predict the length of next session by a function of past session lengths. In contrast, explanatory models assume that the variable to be predict exhibits an explanatory relationship with one or more other variables. For example, in our case, the length of the next game session may be predicted according to many factors such as session start time and player's level.

In this paper, we evaluate both the time series prediction algorithms and the explanatory prediction algorithms. The time series algorithms to be assessed include Last Value, Average, Moving Average, Sliding Median, and Exponential Smoothing. For the explanatory model, the neural-network based approach is assessed.

1) Time Series Prediction Algorithms: The predicted next session length $p(s_{t+1})$ by different time series algorithms are given below.

- **Last Value**: $p(s_{t+1}) = r(s_t)$
- **Average**: $p(s_{t+1}) = \frac{1}{t}(r(s_t) + r(s_{t-1}) + \cdots + r(s_1))$
- **Moving Average**: $p(s_{t+1}) = \frac{1}{k}(r(s_t) + r(s_{t-1}) + \cdots + r(s_{t-k+1}))$, k is the moving window size.
- **Sliding Median**: Sort $\{r(s_t), r(s_{t-1}), ..., r(s_{t-k+1})\}$ and label the sorted list as $\{r_1, r_2, ..., r_k\}$ $(r_1 \leq r_2 \leq, \cdots, \leq r_k)$. If k is an odd number, $p(s_{t+1}) = r_{\frac{k+1}{2}}$. Otherwise, if k is an even number, $p(s_{t+1}) = (r_{\frac{k}{2}} + r_{\frac{k}{2}+1})/2$.
- **Exponential Smoothing**: $p(s_{t+1}) = \alpha \cdot (r(s_t) + (1-\alpha)r(s_{t-1}) + (1-\alpha)^2 r(s_{t-2}) + \cdots + (1-\alpha)^{t-1} r(s_1)$, where $0 < \alpha < 1$ is the aging factor.

2) Neural-Network based Approach: The artificial neural network is a representative explanatory model, which has been proven to be a very attractive tool for predictions [44]. Neural networks are composed of a multitude of neurons representing simple processing elements that operate in parallel [45]. The advantage of using neural networks for prediction is that they are able to learn from history data to catch hidden and strongly non-linear dependencies, even when there is a significant noise in the training set. In recent years, neural networks have become very popular for prediction and forecasting in many domains, including finance, medicine, and environmental science. Therefore, for the explanatory model, we adopt neural networks to predict the session lengths.

Figure 5 shows a commonly used model for neural networks [44]. It consists of one input layer, one output layer and one hidden layer. The neural network is established after an offline training phase using history data. Given an input, the established neural network can be used to generate an output for prediction. In this paper, we choose this model for game session length prediction.

In the construction of a neural network, the selection of appropriate inputs is important. For our problem, the following features are selected as the inputs for the neural networks: *session start hour*, *session start day*, *player's level* and *game map*.[1] These are the only four features available from the datasets that change with different game sessions of a given player. Since the neural network requires history data for offline training, we only consider constructing neural networks for the players who have at least 100 game sessions. For every such player, we construct a new neural network for online prediction after every 100 game sessions of the player. At each construction, the first 90% of the past game sessions of this player are used to train the neural network and the last 10% of the past game sessions of the player are used to validate the neural network[2]. The constructed neural network is then used to predict the lengths of the next 100 game sessions of the player. Note that the first 100 game sessions of each player (including the players who have less than 100 game sessions in total) cannot be predicted by the neural network in the above approach. For such sessions, we simply use the Average algorithm of time series prediction.

B. Prediction Results

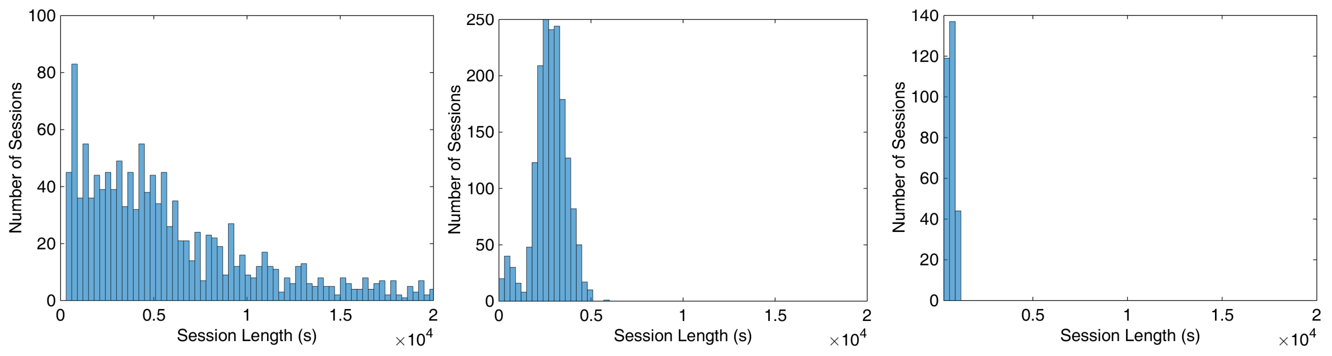
We use the WoWAH, DotAlicious and WoT datasets to evaluate the prediction algorithms. We separately apply the prediction algorithms to each player in the datasets and compare the prediction errors produced by each algorithm. For the WoT dataset, the identifiers of players in each match are not given. So, we do not differentiate the game sessions by players in the training and prediction. Suppose a player has n sessions which are represented by s_1, s_2, ..., s_n. Let $r(s_i)$ denote the actual session length of s_i. Let $p(s_i)$ denote the predicted session length of s_i. The prediction error for the player is defined by

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|p(s_i) - r(s_i)|}{r(s_i)}$$

Table I shows the average prediction errors produced by different algorithms for each dataset. As can be seen, the neural-network based prediction achieves the lowest prediction error compared to the time series algorithms for all the three datasets. Therefore, we shall adopt the neural-network based approach to predict the ending times of game sessions in evaluating our proposed request dispatching algorithm.

[1]Some of these features may not be provided by all the datasets.

[2]For the number of neurons in the hidden layer, we have varied the number for each neural network and achieved good prediction results of training with 10 neurons for most of the neural networks.



(a) A sample WoWAH player, Variance = 5082 (b) A sample DotAlicious player, Variance = 892 (c) A sample WoT player, Variance = 199

Fig. 6. Distribution of game session lengths

TABLE I
AVERAGE PREDICTION ERRORS

Prediction Algorithms	Average Prediction Error		
	WoWAH	DotAlicious	WoT
Average	0.672	0.273	0.242
Moving Average ($k = 30$)	0.580	0.270	0.181
Moving Average ($k = 10$)	0.550	0.263	0.116
Last Value	0.780	0.395	0.112
Sliding Median	1.770	0.440	2.210
Exp Smoothing ($\alpha = 0.75$)	0.794	0.354	0.107
Exp Smoothing ($\alpha = 0.5$)	0.738	0.324	0.115
Exp Smoothing ($\alpha = 0.25$)	0.698	0.229	0.116
Neural Network	0.520	0.188	0.074

We also observe from Table I that the prediction errors produced by the algorithms for the WoWAH dataset are much larger than the other two datasets. This implies that the session lengths for the DotAlicious and WoT datasets are more predictable. This is possibly because DotA and WoT are match-based games. The duration of a match is determined by all the team members together and thus the variance of the match lengths would be small. On the other hand, WoW is not a match-based game. The game session length is highly dependent on the players' willingness, which could vary greatly and be affected by many factors. Figure 6 shows the distributions of the game session lengths of some sample players from different datasets. It can be seen that the session length variance of the player from the WoW dataset is much larger than the variances of the players from the DotAlicious and WoT datasets.

VI. Experimental Evaluations

We conduct extensive experiments using the WoWAH, DotAlicious and WoT datasets to evaluate the benefit of the proposed play request dispatching algorithm.

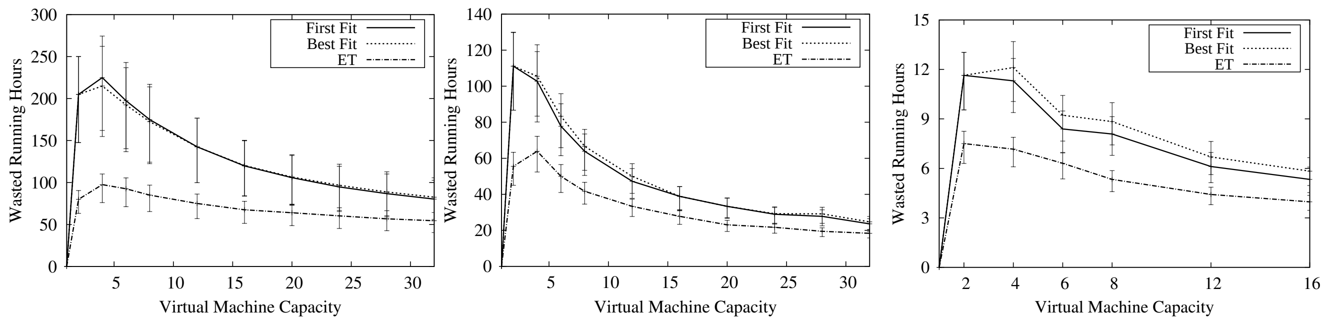
A. *Performance with Perfect Prediction*

We first consider an ideal situation in which the ending times of game sessions are perfectly predicted. This represents a yardstick (upper bound) on the performance of request dispatching. We examine the impacts of virtual machine capacity, the heterogeneity in play requests, and the heterogeneity in virtual machine types.

1) Impact of virtual machine capacity: First, we evaluate how the virtual machine capacity (in terms of how many game instances can run on a virtual machine concurrently) would affect the performance of the ET algorithm. The virtual machine capacity is varied in the range from 1 to 32 The recent GPU technology is able to support up to 32 game instances on a single board with multiple GPUs [46])[3]. We record the wasted running hours (i.e., the additional running hours compared to the optimal running hours) produced by different dispatching algorithms on every day over the entire WoWAH, DotAlicious, and WoT datasets. Figure 7 shows the average wasted running hours for one day together with the 90th and 10th percentile results. It can be seen that for most of the virtual machine capacities, ET can save more than 50% wasted running hours on average compared to First Fit and Best Fit dispatching for the WoWAH dataset. In the extreme case where the virtual machine capacity is only 1 (i.e., each virtual machine can run one game instance only), it is intuitive that all the dispatching algorithms would occupy exactly the same number of virtual machines at any time and there is no wasted running hour at all. On the other hand, when a virtual machine can accommodate multiple game instances, the percentage of reduction in wasted running hours by our ET algorithm compared to First Fit and Best Fit gradually decreases with increasing virtual machine capacity. This is because given a set of play requests, when the virtual machine capacity is larger, it becomes more difficult to find sufficient number of play requests that share similar ending times to fill up a virtual machine to its capacity. Similar performance trends can also be observed in the results for the DotAlicious and WoT datasets.

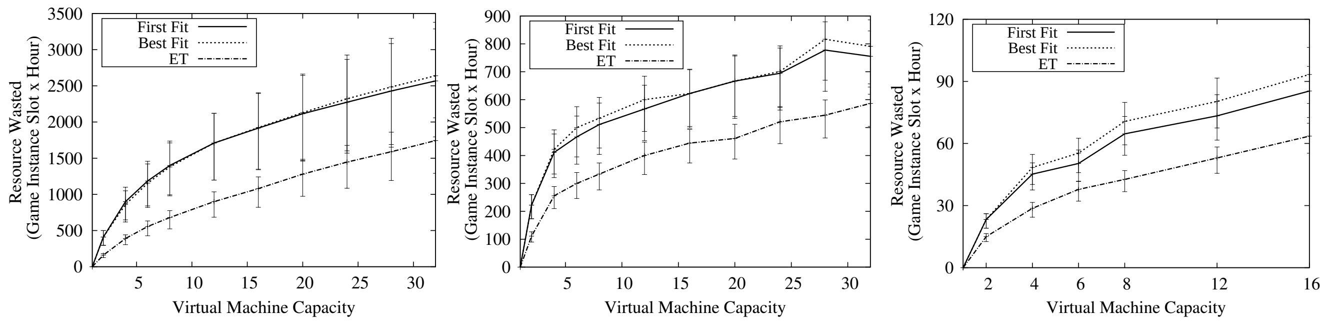
It should be noted that a wasted running hour of a high-

[3]The maximum virtual machine capacity for the WoT dataset is set to 16 because WoT dataset has much less game sessions at peak hours than the other two datasets.



(a) WoWAH dataset (b) DotAlicious dataset (c) WoT dataset

Fig. 7. Daily wasted running hours for different virtual machine capacities



(a) WoWAH dataset (b) DotAlicious dataset (c) WoT dataset

Fig. 8. Daily absolute resource waste for different virtual machine capacities

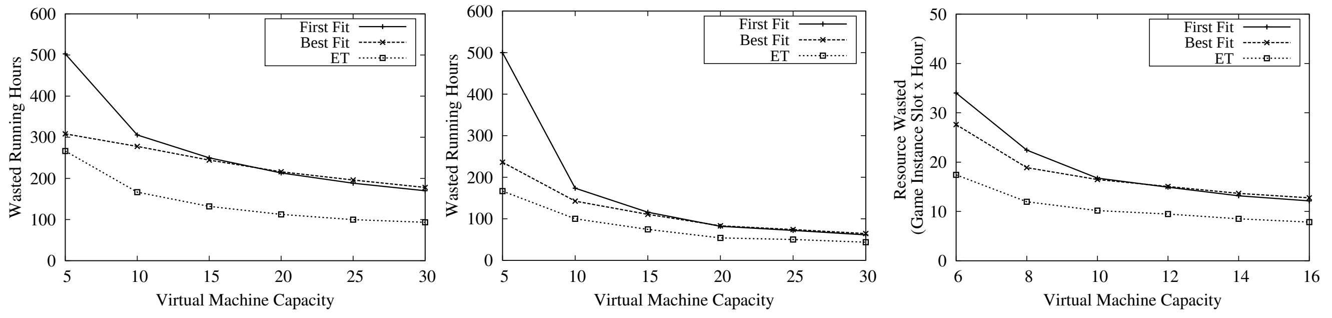
capacity virtual machine actually wastes more resources than a wasted running hour of a low-capacity virtual machine. We also compute the absolute amount of resource waste in terms of the game instance slots (which is given by the product of wasted running hours and the virtual machine capacity) on every day for the WoWAH, DotAlicious and WoT datasets. Figure 8 shows the average absolute resource waste for one day together with the 90th and 10th percentile results. It can be seen that as the virtual machine capacity increases, the resource waste cut by our ET algorithm compared to First Fit and Best Fit dispatching also increases.

2) Impact of heterogeneity in play requests: Now, we evaluate the impact of heterogeneity in the resource demand of play requests. To do so, we assume the games served in the system have five types of resource demands 1, 2, 3, 4, and 5 respectively. We randomly assign a type of resource demand to each play request. Figure 9 shows the average daily wasted running hours produced by different dispatching algorithms with different virtual machine capacities for the WoWAH, DotAlicious and WoT datasets. It can be seen that for most of the virtual machine capacities, ET can save around 50% wasted running hours compared to First Fit and Best Fit for the WoWAH dataset and can save 20%$\sim$30% wasted running hours for the DotAlicious and WoT datasets.

Figure 10 shows the average daily absolute resource waste for the WoWAH, DotAlicious and WoT datasets. It can be seen that as the virtual machine capacity increases, the resource waste cut by ET algorithm compared to First Fit and Best Fit dispatching again increases. The above results indicate that the ET dispatching algorithm can still reduce resource waste significantly with multiple game types.

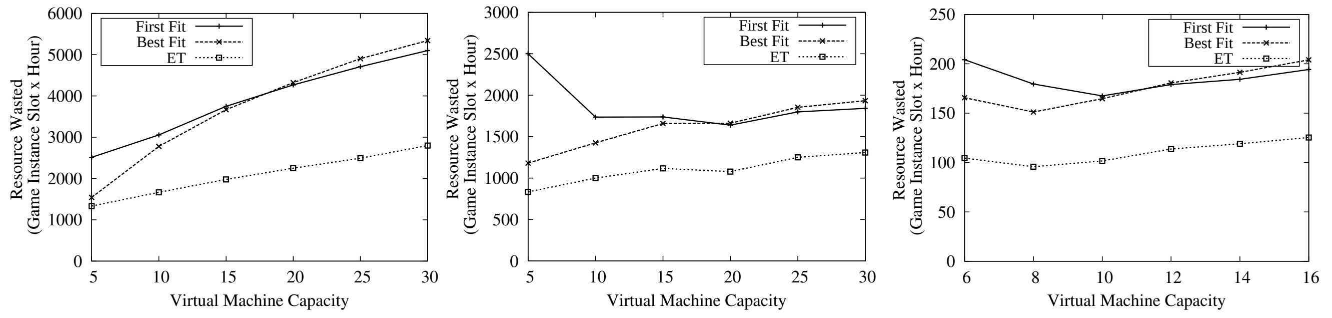
3) Impact of heterogeneity in virtual machine instances: Next, we examine the impact of heterogeneity in the capacities of virtual machine instances. The public cloud provider generally provides multiple types of virtual machine instances with different capacities so that the customers can choose appropriate ones according to their requirements. For example, Amazon offers both light virtual machine instance types like t2.small and powerful instance types like m3.large. In order to evaluate the heterogeneity in virtual machine instances, we assume there are two types of virtual machine instances: small instance and large instance. The number of small instances that can be used is limited by a number N while the supply of large instances is unlimited. When a new virtual machine is required in the dispatching, we first try to allocate a small instance since it has been shown earlier in Section VI-A1 that small instances generally lead to less resource waste. If the small instances are used up, a large instance is then allocated. By default, the capacities of small instances and large instances are set at 10 and 20 respectively.

Figure 11(a) shows the average daily absolute resource waste with different values of N for the WoWAH dataset. When N is smaller than 200, the resource waste decreases with increasing N. It implies that using small instances wastes less resource than using large instances. When N is larger than 200, the resource wasted keeps unchanged. This is because the maximum number of small instances required to serve all play requests in the WoWAH dataset is around 200. Large instances are seldom used when N is larger than 200. Similar trends can also be found for the DotAlicious and WoT datasets



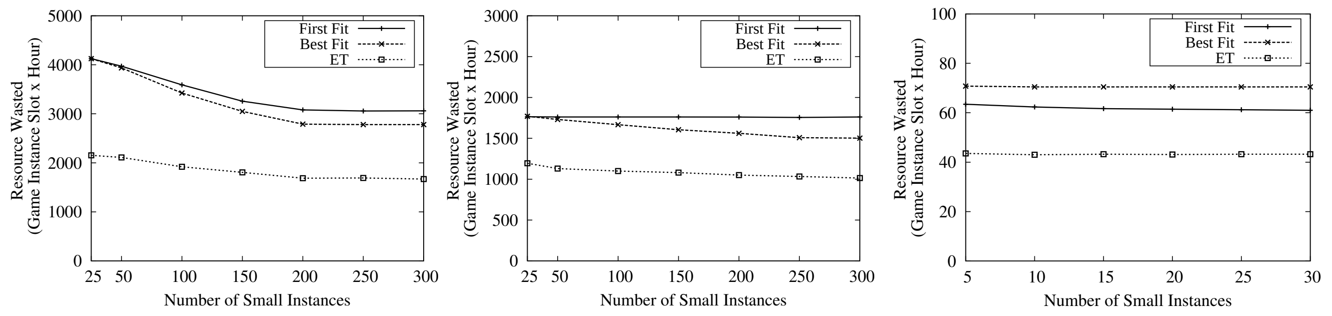
(a) WoWAH dataset (b) DotAlicious dataset (c) WoT dataset

Fig. 9. Daily wasted running hours for heterogeneous play requests



(a) WoWAH dataset (b) DotAlicious dataset (c) WoT dataset

Fig. 10. Daily absolute resource waste for heterogeneous play requests



(a) WoWAH dataset (b) DotAlicious dataset (c) WoT dataset

Fig. 11. Daily absolute resource waste for heterogeneous virtual machines

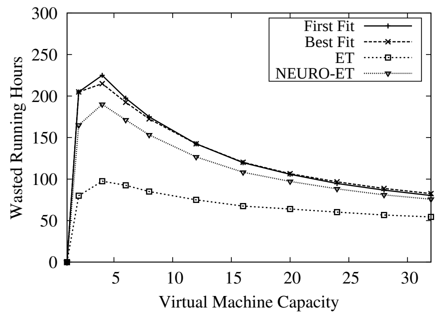
as shown in Figures 11(b) and 11(c). For all datasets, ET can significantly save the resource waste compared to First Fit and Best Fit.

B. Performance with Neural-Network based Prediction

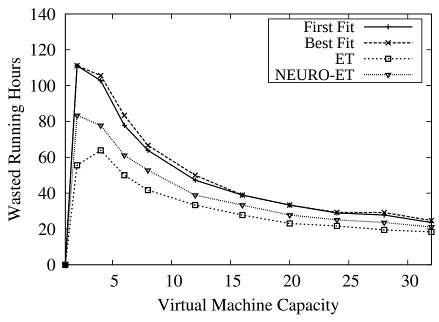
In this section, we present the evaluation of the proposed play request dispatching algorithm with neural-network based prediction for the session lengths. In the simulations, all neural-network related calculations are done using Matlab. The computational time of constructing a neural network for a player is generally within seconds. The computational time of predicting a session length using the neural network is much less than one second. Therefore, the neural-network based prediction is computationally efficient for online prediction. The ET dispatching algorithm using neural-network based prediction is denoted by NEURO-ET.

Figure 12 shows the average daily wasted running hours produced by different algorithms with different virtual machine capacities for the WoWAH, DotAlicious and WoT datasets. It can be seen that for the DotAlicious and WoT datasets, NEURO-ET can significantly reduce the wasted running hours compared to First Fit and Best Fit. However, the performance of NEURO-ET for the WoWAT dataset is rather close to First Fit and Best Fit. This is in agreement with the results shown in Table I of Section V-B, which indicate that the session lengths of match-based games are more predictable. The performance of the proposed request dispatching algorithm is dependent on the prediction accuracy. Therefore, the performance of NEURO-ET for the DotAlicious and WoT datasets is much better than that for the WoWAH dataset.

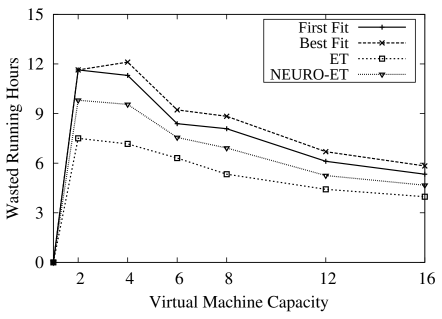
Figure 13 shows the average daily absolute resource waste in terms of the game instance slots. Similarly, we can see



(a) WoWAH dataset

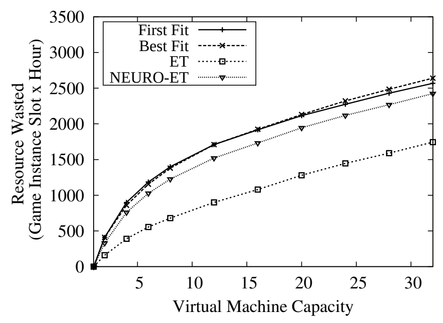


(b) DotAlicious dataset

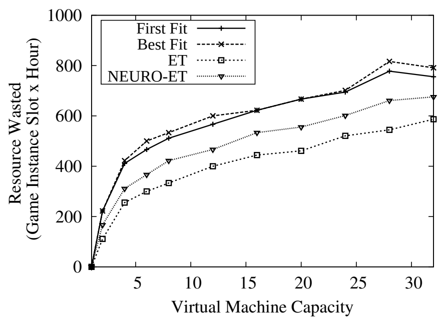


(c) WoT dataset

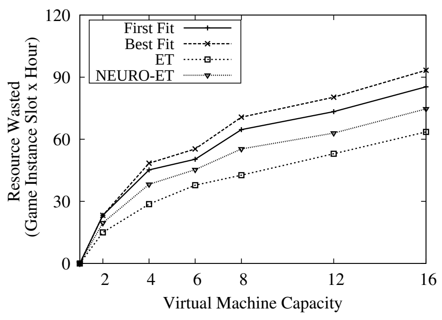
Fig. 12. Daily wasted running hours for different virtual machine capacities



(a) WoWAH dataset



(b) DotAlicious dataset



(c) WoT dataset

Fig. 13. Daily absolute resource waste for different virtual machine capacities

that NEURO-ET can greatly save the resource waste for the DotAlicious and WoT datasets while it achieves less gain for the WoWAH dataset.

VII. Conclusion and Future Work

In this paper, we have proposed a play request dispatching algorithm for cloud gaming systems to optimize their service costs. The proposed algorithm assigns the play requests according to the predicted ending times of game sessions. After assessing several classes of prediction algorithms, we choose a neural-network based approach to predict the session lengths. We use real online game traces to evaluate the proposed dispatching algorithm. The experimental results show that the proposed dispatching algorithm can reduce the resource waste of cloud servers and the benefits are particularly significant for match-based games.

In the future work, we would like to investigate more effective algorithms for predicting the ending times of game sessions. One direction is to use the social structure of the players in online games to help with the prediction. It has been shown that the players are more likely to play games with their “friends” [47]. This may help to predict the ending times more accurately by considering closely related players together. In addition, we have assumed in this paper that there is only one type of dominant resource (e.g., GPU) that can become the bottleneck of virtual machines. However, some games may have different types of dominant resources. Then, different combinations of game instances running on a virtual machine may cause different types of resources to become the bottleneck. We would also like to consider multi-dimensional resource requirements of game sessions in request dispatching in our future work.

Acknowledgment

This research is supported by Multi-plAtform Game Innovation Centre (MAGIC), funded by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority.

References

[1] “Onlive web page.” http://www.onlive.com/.
[2] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, “Gaminganywhere: an open cloud gaming system,” in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013, pp. 36–47.
[3] “Gaikai web page.” http://www.gaikai.com/.
[4] “Distribution and monetization strategies to increase revenues from cloud gaming.”
[5] R. Shea, J. Liu, E.-H. Ngai, and Y. Cui, “Cloud gaming: Architecture and performance,” *IEEE Network*, vol. 27, no. 4, pp. 16–21, 2013.
[6] Y. Li, X. Tang, and W. Cai, “On dynamic bin packing for resource allocation in the cloud,” in *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 2014, pp. 2–11.
[7] Y. Li, X. Tang, and W. Cai, “Dynamic bin packing for on-demand cloud resource allocation,” *IEEE Transactions on Parallel and Distributed Systems (accepted to appear)*, 2015.
[8] Y. Li, X. Tang, and W. Cai, “Let’s depart together: Efficient play request dispatching in cloud gaming,” in *The 13th Annual Workshop on Network and Systems Support for Games*. ACM, IEEE, 2014.
[9] “Streammygame web page.” http://www.streammygame.com/smg/index.php.
[10] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “The brewing storm in cloud gaming: A measurement study on cloud to end-user latency,” in *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE, 2012, p. 2.

[11] M. Claypool, D. Finkel, A. Grant, and M. Solano, "On the performance of onlive thin client games," *Multimedia Systems*, vol. 20, no. 5, pp. 471–484, 2014.

[12] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei, "Are all games equally cloud-gaming-friendly? an electromyographic approach," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games (NetGames), 2012*. IEEE, 2012, pp. 1–6.

[13] M. Manzano, J. A. Hernández, M. Uruenña, and E. Calle, "An empirical study of cloud gaming," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*. IEEE, 2012, p. 17.

[14] H. Ahmadi, S. Khoshnood, M. R. Hashemi, and S. Shirmohammadi, "Efficient bitrate reduction using a game attention model in cloud gaming," in *Proceeding of the IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)*. IEEE, 2013, pp. 103–108.

[15] H. Ahmadi, S. Zad Tootaghaj, M. Hashemi, and S. Shirmohammadi, "A game attention model for efficient bit rate allocation in cloud gaming," *Multimedia Systems*, vol. 20, no. 5, pp. 485–501, 2014.

[16] S.-P. Chuah and N.-M. Cheung, "Layered coding for mobile cloud gaming," in *Proceedings of International Workshop on Massively Multiuser Virtual Environments*. ACM, 2014, pp. 1–6.

[17] M. Hemmati, A. Javadtalab, A. A. Nazari Shirehjini, S. Shirmohammadi, and T. Arici, "Game as video: Bit rate reduction through adaptive object encoding," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2013, pp. 7–12.

[18] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 103–112.

[19] Y. Li and W. Cai, "Consistency-aware partitioning algorithm in multi-server distributed virtual environments," in *Proceeding of the 26th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2012, pp. 798–807.

[20] Y. Li and W. Cai, "Consistency-aware zone mapping and client assignment in multi-server distributed virtual environments," *IEEE Transactions on Parallel and Distributed Systems (accepted to appear)*, 2014.

[21] L. Zhang and X. Tang, "Optimizing client assignment for enhancing interactivity in distributed interactive applications," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1707–1720, 2012.

[22] L. Zhang and X. Tang, "The client assignment problem for continuous distributed interactive applications: analysis, algorithms, and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 785–795, 2014.

[23] 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*, vol. 24, no. 8, pp. 1405–1416, 2014.

[24] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, Jan 2015.

[25] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin packing approximation algorithms: Survey and classification," *Handbook of Combinatorial Optimization (second ed.), Springer*, 2013.

[26] G. Galambos and G. J. Woeginger, "On-line bin packingła restricted survey," *Zeitschrift für Operations Research*, vol. 42, no. 1, pp. 25–45, 1995.

[27] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1979.

[28] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, "Dynamic bin packing," *SIAM Journal on Computing*, vol. 12, no. 2, pp. 227–258, 1983.

[29] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks in operations research and management science*, vol. 4, pp. 445–522, 1993.

[30] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber, "A unified approach to approximating resource allocation and scheduling," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1069–1090, 2001.

[31] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Proceeding of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2009, pp. 327–334.

[32] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of IEEE INFOCOM, 2010*, 2010, pp. 1–9.

[33] A. L. Stolyar and Y. Zhong, "A large-scale service system with packing constraints: Minimizing the number of occupied servers," in *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. ACM, 2013, pp. 41–52.

[34] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proceedings of IEEE INFOCOM, 2011*. IEEE, 2011, pp. 71–75.

[35] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2010, pp. 179–188.

[36] Y.-T. Lee, K.-T. Chen, Y.-M. Cheng, and C.-L. Lei, "World of warcraft avatar history dataset," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 123–128.

[37] Y. Guo and A. Iosup, "The game trace archive," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*. IEEE, 2012, p. 4.

[38] "Mmogdata.net." http://users.telenet.be/mmodata/Charts/Subs-1.png.

[39] T. Henderson and S. Bhatti, "Modelling user behaviour in networked games," in *Proceedings of the ninth ACM international conference on Multimedia*. ACM, 2001, pp. 212–220.

[40] C. V. Hsueh-hua and D. H. Been-Lirn, "Understanding social interaction in world of warcraft," in *Proceedings of the international conference on Advances in computer entertainment technology*. ACM, 2007, pp. 21–24.

[41] Y.-T. Lee and K.-T. Chen, "Is server consolidation beneficial to mmorpg? a case study of world of warcraft," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 2010, pp. 435–442.

[42] M. Yu, C. Zhang, Z. Qi, J. Yao, Y. Wang, and H. Guan, "Vgris: Virtualized gpu resource isolation and scheduling in cloud gaming," in *Proceedings of the 22nd International Symposium on High-performance Parallel and Distributed Computing*. ACM, 2013, pp. 203–214.

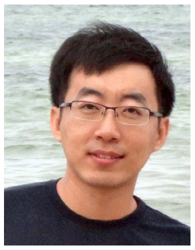
[43] G. C. R. G. E. P. Box, G. M. Jenkins, *Time Series Analysis: Forecasting and Control, 4th Edition*. Wiley, 2014.

[44] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[45] S. I. Gallant, *Neural Network Learning and Expert Systems*. MIT press, 1993.

[46] "Nvidiagrid: http://www.nvidia.com/object/grid-technology.html."

[47] A. Losup, R. V D Bovenkamp, S. Shen, A. Jia, and F. Kuipers, "Analyzing implicit social networks in multiplayer online games," *IEEE Internet Computing*, vol. 18, no. 3, pp. 36–44, May 2014.



Yusen Li is currently a Research Fellow in the School of Computer Engineering at Nanyang Technological University. His research interests include scheduling, resource management issues in distributed systems and cloud computing.

Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an associate professor in the School of Computer Engineering at Nanyang Technological University, Singapore. He has served as an associate editor of IEEE Transactions on Parallel and Distributed Systems. His research interests include distributed systems, mobile and pervasive computing, and wireless sensor networks. He is a senior member of the IEEE.

Wentong Cai is a Professor in the School of Computer Engineering at Nanyang Technological University, Singapore. His expertise is mainly in the areas of Modeling and Simulation and Parallel and Distributed Computing. He is an associate editor of the ACM Transactions on Modeling and Computer Simulation (TOMACS) and an editor of the Future Generation Computer Systems (FGCS). He has chaired a number of international conferences. Most recent ones include CloudCom 2014, SIMUTools 2013, ICPADS 2012, and MACOTS 2011.