國立交通大學

資訊科學與工程研究所

碩 士 論 文

雲端大型多人線上遊戲下基於玩家行為之資源分配

Player Behavior-based Resource Allocation for MMOG Clouds

研 究 生：賴寬嶧

指導教授：王國禎　教授

中 華 民 國　102　年　7　月

雲端大型多人線上遊戲下基於玩家行為之資源分配
Player Behavior-based Resource Allocation for MMOG Clouds

研 究 生：賴寬嶧　　Student：Kuan-Yi Lai

指導教授：王國禎　　Advisor：KuoChen Wang

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2013

Hsinchu, Taiwan, Republic of China

中華民國 102 年 7 月

雲端大型多人線上遊戲下基於玩家行為之資源分配

學生：賴寬嶧　　指導教授：王國禎 博士

國立交通大學資訊科學與工程研究所

摘 要

現今的大型多人線上遊戲(MMOG)已有超過數以千萬的用戶，其中較受歡迎的遊戲可能有超過一萬人同時在線上。因此，為了解決遊戲伺服器所產生的大量負載變動，許多大型多人線上遊戲的營運商，企圖將他們的遊戲服務放到雲端平台上運行，以便善用雲端計算的優點。遊戲伺服器的資源 (CPU、記憶體、網路頻寬) 需求量和玩家的關注區域 (Area of Interest) 內有多少玩家，以及該玩家正在做什麼的行為有很大的關係。資源分配不足 (under-allocation) 會導致玩家好的遊戲體驗下降，使得玩家離開遊戲並刪除帳號。為了保證玩家有好的遊戲體驗，現在大型多人線上遊戲的營運商大多都是採用超額配置 (over-allocation) 資源這種策略。然而，過度的超額配置資源會導致

資源整體的使用率下降。為了解決這個問題，我們提出一個雲端大型多人線上遊戲下基於玩家行為之資源分配 (PB-RA)方式。我們透過類神經網路來預測未來地圖上各種行為的玩家人數，根據量測不同玩家行為所產生的負載給定一個更精確的伺服器整體資源需求量，讓我們在分配資源時可以更有效率。實驗結果證明，我們所提出的基於玩家行為之資源分配方式，相對於只考慮玩家人數來分配資源的方法減少74%的資源超額分配，相對於考慮玩家互動來分配資源的方法減少50%的資源超額分配。此外，資源分配不足發生的次數相對於考慮玩家互動來分配資源的方法也不超過 1.05 倍。據我們所知，目前並沒有大型多人線上遊戲的資源分配方式有考慮到玩家行為。

關鍵詞：雲端計算、大型多人線上遊戲、玩家行為、資源分配。

Player Behavior-based Resource Allocation for MMOG Clouds

Student: Kuan-Yi Lai Advisor: Dr. Kuochen Wang

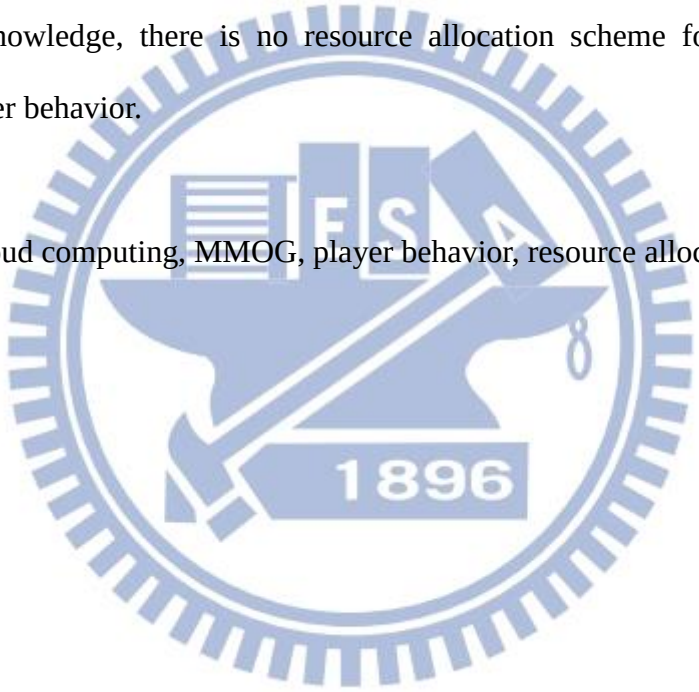
Department of Computer Science

National Chiao Tung University

Abstract

Today's Massively Multiplayer Online Games (MMOGs) have more than tens of millions subscriptions and the popular one may have over 10,000 active concurrent players. Therefore, many MMOG operators attempt to run their game services in clouds to take advantages of cloud computing, such as on-demand self-service and resource pooling characteristics, in order to handle large load variation in game servers. The amount of resource (CPU, memory, and network bandwidth) requirements for a player is related to how many players are in his Area of Interest (AoI) and what kind of player behavior. Resource under-allocation leads to degradation of game experience of players and may trigger player quitting and account closing. To guarantee the better players' experience, resource over-allocation is the most commonly used resource allocation policy adopted by MMOG operators. However, resource over-allocation often results in low overall resource utilization. To address this deficiency, we propose a dynamic *Player Behavior-based Resource Allocation* scheme for MMOG clouds, called PB-RA. We predict the number of players for each behavior type in one map through a neural network-based predictor,

and measure loads generated from different player behavior types. As a result, we can predict total resource requirements more accurately for players with different behavior types in the map. That is, we can allocate resources more efficiently. Experiment results show that the proposed PB-RA can reduce 74% and 50% of resource over-allocation compared to the method that only considers number of players and the method that considers interaction of players as well, respectively. Moreover, in terms of the number of resource under-allocation events, the proposed PB-RA is no more than 1.05 times compared with the method that considers interaction of players. To the best of our knowledge, there is no resource allocation scheme for MMOGs that considers player behavior.

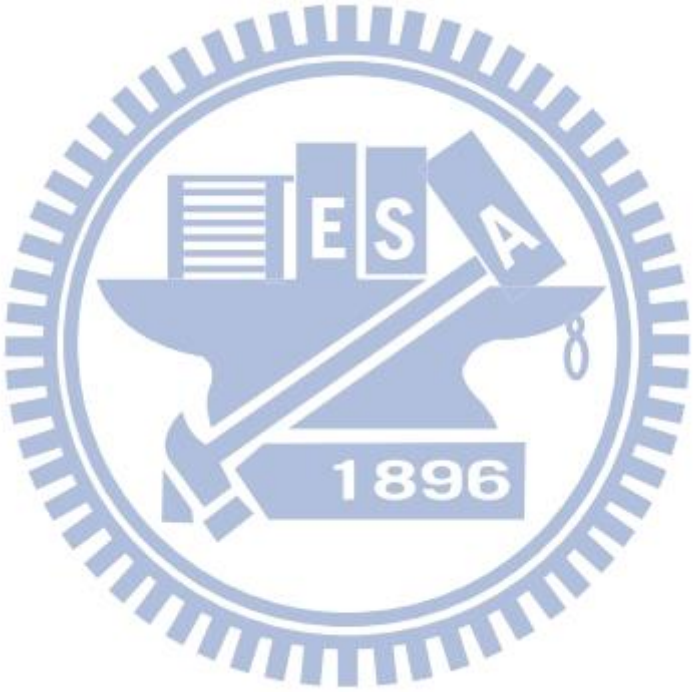


Keywords: cloud computing, MMOG, player behavior, resource allocation.

Acknowledgements

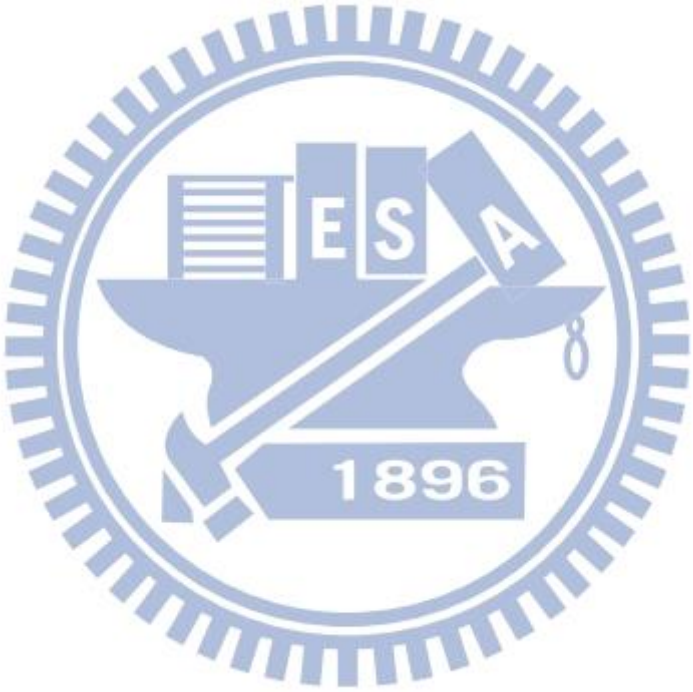
Many people have helped me with this thesis. I deeply appreciate my thesis advisor, Dr. Kuochen Wang, for his intensive advice and guidance. I would like to thank all the members of the *Mobile Computing and Broadband Networking Laboratory* (MBL) for their invaluable assistance and suggestions. The support by the National Science Council under Grants NSC99-2221-E-009-081-MY3 and NSC101-2219-E-009-001 is gratefully acknowledged. Finally, I thank my family for their endless love and support.

Contents



List of Figures

List of Tables



Chapter 1

Introduction

Players may interact with one another in various forms via avatars in the Networked Virtual Environment (NVE) through Internet. Massively Multiplayer Online Games (MMOGs) which is a famous example in NVEs brought the market capitalization exceeded $12 billion in 2012 [21]. World of Warcraft (WoW) has always been popular among MMOGs. In 2009, WoW achieved 12 million subscribers worldwide which is equivalent to the population of a small country, the monopoly exceeded 50% in the MMOG market [1], as shown in Figure 1. Therefore, a successful MMOG may attract a huge number of players and bring great business opportunities. The traditional MMOG is a client-server architecture which has a powerful server to handle all computations generated from a huge number of players. Due to the population of MMOG players grows explosively, the powerful machine becomes a bottleneck. MMOG operators gradually change the architecture from the client-server architecture to a distributed architecture.

More and more game companies run their games in clouds to take advantages of cloud computing, such as on-demand self-service and resource pooling characteristics. It is wasteful if game operators did not use cloud resources efficiently. Thus, an efficient resource allocation solution is necessary. Some existing dynamic resource allocation policies for MMOGs were proposed [7], [8], [9], [10]. However, among those resource allocation methods, some of them only concern number of players and some take interactions of players into consideration additionally. In MMOGs, players may perform different behavior types in the virtual world. The gap between the loads

generated by various behavior may vary large.

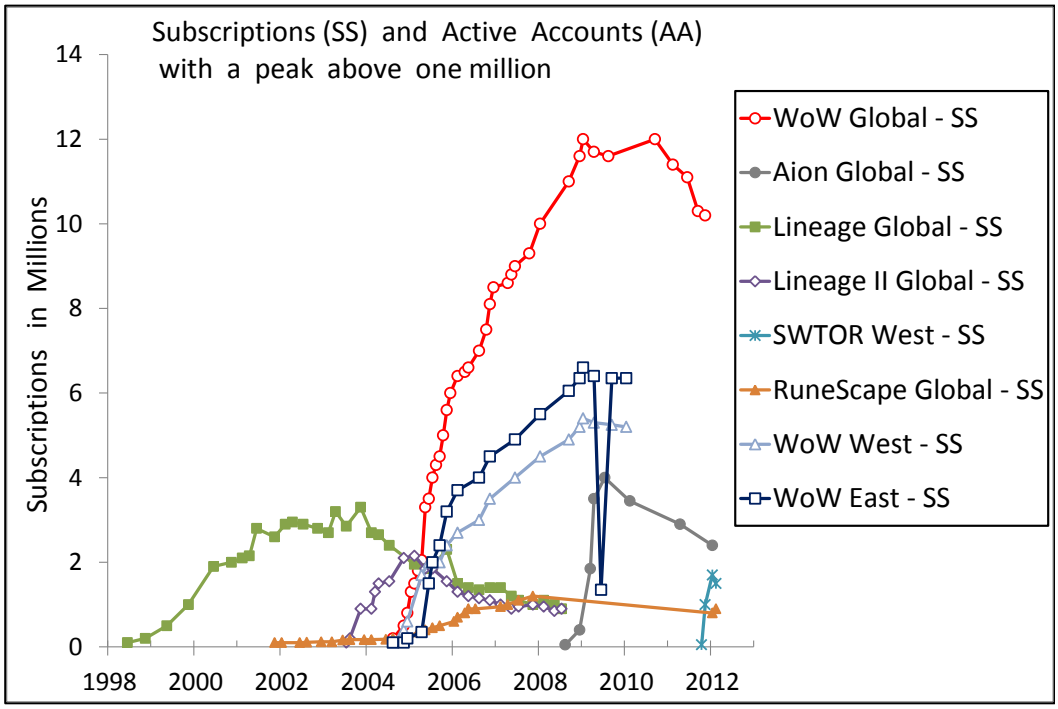


Figure 1. The subscriptions of popular MMOGs [1]

In this paper, we propose an efficient *Player Behavior-based Resource Allocation* (PB-RA) scheme for MMOG clouds. We measure loads generated from different types of player behavior, and use a neural network-based predictor to forecast number of players in different behavior types. According to the prediction results, we calculate how much resources the game server needed in a map by our load model. Different from existing dynamic resource allocation methods for MMOGs, our approach is under the premise of avoiding under-allocation events and reduce resource over-allocation. To the best of our knowledge, there is no resource allocation scheme for MMOGs that considers player behavior.

The rest of this thesis is organized as follows. We review related dynamic resource allocation methods for MMOGs in Chapter 2. Preliminaries of dynamic resource allocation for MMOGs are described in Chapter 3. Chapter 4 shows the load model for our design approach. Chapter 5 presents the proposed PB-RA method.

Chapter 6 describes how we collect game data, set up experiment parameters, and discuss experiment results. Finally, some concluding remarks and future work are given in Chapter 7.

Chapter 2

Related Work

To handle a large number of computations generated by game server, MMOG operators have changed the system architecture from client/server to multi-server which is the well-known architecture in MMOG clouds. In MMOG clouds, it partitions a game world into many zones where each zone has at least one virtual machine (VM) to support the computation it needs [11]. Hence, the computation load not just focuses on a single VM. Existing MMOG cloud issues focus on two parts, which are *load balancing* and *resource allocation*, as shown in Figure 2. There are two types of load balancing: *zone-based* and *player-based*. The zone-based method partitions a game zone into many microcells and adjusts these microcells to achieve load balancing [3], [4], [5]. The advantage of the zone-based method is easier to add VMs to each zone so that the zone-based method can reduce the communication cost between microcells. The player-based method partitions a zone based on the locations of avatars. The player-based method can maintain map completeness [4], [6]. The experiment results by Ahmed et al. show that the zone-based load balance method is better than the player-based method [4].

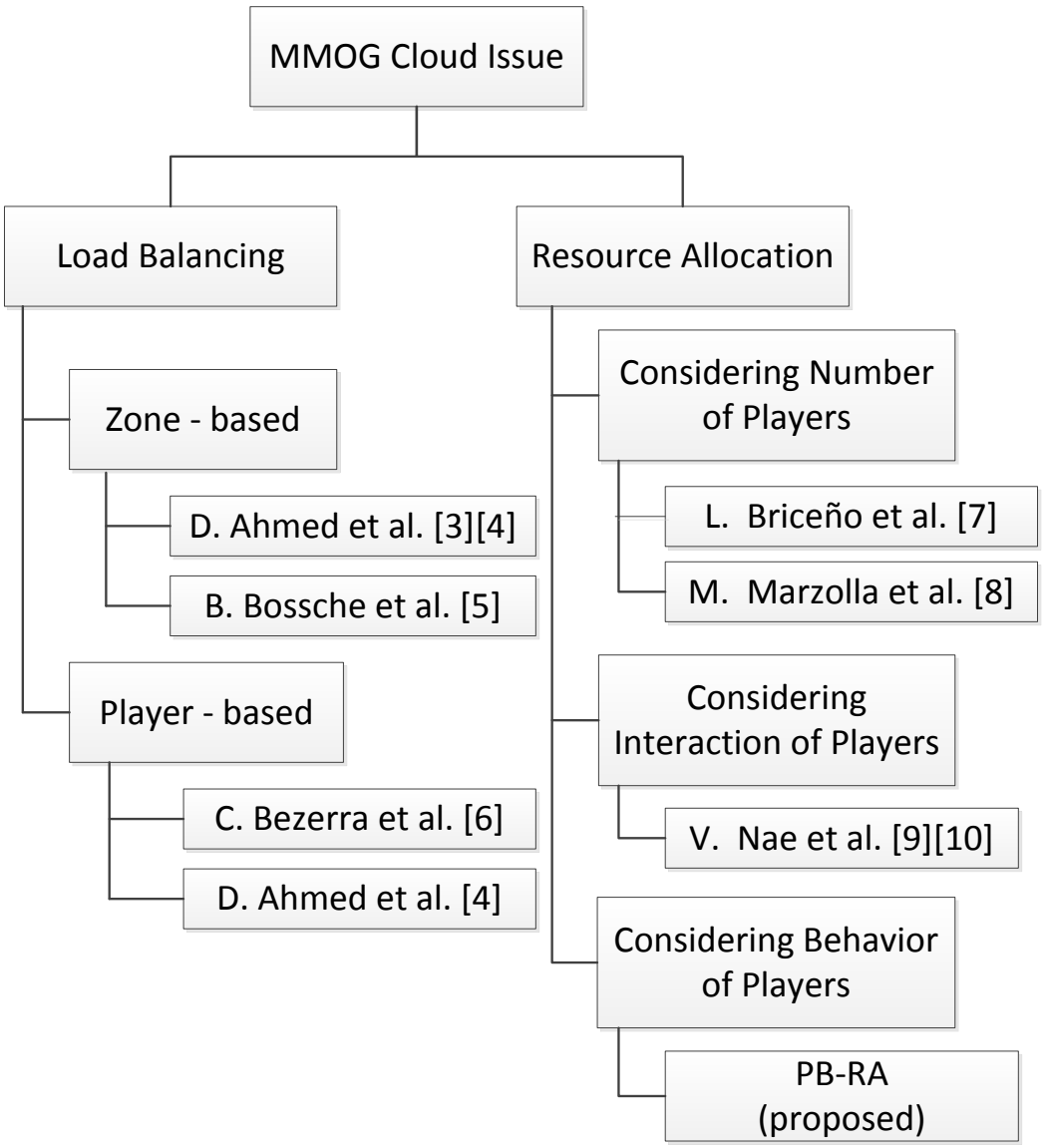


Figure 2. Classification of MMOG cloud issues.

Although load balancing increases a VM's utilization, it may decrease the real-time experience of each player. Bad real-time experience makes players quit the game. To avoid the mentioned situation, MMOG operators focus on resource allocation gradually. We classify resource allocation in three categories which are *considering number of players*, *considering interaction of players*, and *considering behavior of players*, as shown in Figure 2. In the first category, it considers number of players to allocate resource, Briceño et al. create a fair environment which focuses on

maintaining the QoS constraints [7]. It use P2P architecture that has a main server (MS) to handle the operation of the game and secondary server (SS) which is a user's computer. The MS can offload calculation to SSs to guarantee response time will not violate and increase the number of players who can join the game through P2P architecture. In addition, Marzolla et al. describe a framework based on cloud architecture which satisfies the system response time constraints [8]. The main architecture is divided into three layers which are *Gateway layer*, *Cell server layer*, and *Database server layer*. The gateway layer is responsible for the game protocol checking and verification. The cell server layer are composed of many servers to manage the virtual world. The database server layer stores persistent game state information. They use a queuing network performance model to quickly estimate the system response time for different configurations. In [11], it introduces a real-time framework to allocate VMs for developing MMOGs. It automatically distributes the game state among participating servers and supports parallel state update computations. In addition, it also considers the region between overlapping zones.

In order to increase efficiency of resource allocation, the secondary category of resource allocation considers interaction of players to allocate resources. Nae et al. divide player interaction into five degrees such as: $O(n), O(n \cdot \log n), O(n^2), O(n^2 \cdot \log n)$ and $O(n^3)$[9]. In their research, the higher degree of interaction has the higher computations by game servers. They also simulate five prediction algorithms including last value, moving average, exponential smoothing, sliding window median and neural network. Finally, the experimental results show that neural network has the highest accuracy and the lowest complexity characteristics. Furthermore, Nae et al. divide resources into three main types: CPU, memory and network bandwidth [10]. They consider different sizes of player interaction groups to define their load model. The requirements of resources that a game server needed are calculated according to

this load model. In this way, the load can close to the actual situation, and it may increase resources utilization. Therefore, the game operator is able to dynamically adjust the number of resources that a game server needs.

OnLive is a true games company which implements MMOGs on a cloud environment [12]. A user can use any device such as PC, tablet computer, and smart phone, etc., to play games at anywhere with a network connection. Users play games through video streaming just like watching YouTube. OnLive supports at most 4 players cooperating in the same game session. We compare the proposed PB-RA with related work as shown in Table I.

Table I. Comparison of related work.

		OnLive [12]	**M. Marzolla [8]**	**V. Nae [10]**	**PB-RA (proposed)**
Game Scale		Small (at most 4 players)	Large (MMOG)	Large (MMOG)	Large (MMOG)
Cloud Gaming		Yes	No	No	Yes
Load Prediction		N/A	None	Neural Network	Neural Network
Resource Allocation Basis	Number of Players	N/A	Yes	Yes	Yes
	Player Interaction		No	Yes	Yes
	Player Behavior		No	No	Yes

Chapter 3

Preliminaries

3.1 Game zone parallelization

There are three main techniques to partition a virtual game world for parallelizing: *zoning*, *replication*, and *instancing* [11]. The MMOG adopts multi-server architecture which allows multiple VMs in parallel processing the game world instead of client/server architecture which processes the game world by only one powerful server through these three techniques. Thus, the multi-server architecture is more reliable.

Zoning partitions the game world into adjacent areas called zones (there are four zones in Figure 3). A zone usually has the following characteristics. First, avatars can move between zones, but no inter-zone events exist. Second, zones are not necessarily of same shape and size, but obstacles (mountains or sea) usually exist between adjacent zones to avoid too many entries in a zone. Third, it can simply add a VM to a zone to expand the virtual world. However, an excessive density of population in the same zone will generate huge load that may exceed a VM can handle.

Replication distributes copies of a game zone onto different VMs when the zone has a large density of avatars interacting with each other, as shown in Figure 3. The entities (avatars or NPCs/bots) hosted by a VM in a distributed zone called *active entities*. The VM is responsible for the active entities state transferring and behavior processing. The entities which are active on other servers called *shadow entities*. All VMs will synchronize with each other periodically.

Instancing separates a popular area into individual zones. It makes different avatars in different instances cannot see each other even they are located in the same location. That means instances are independent with each other.

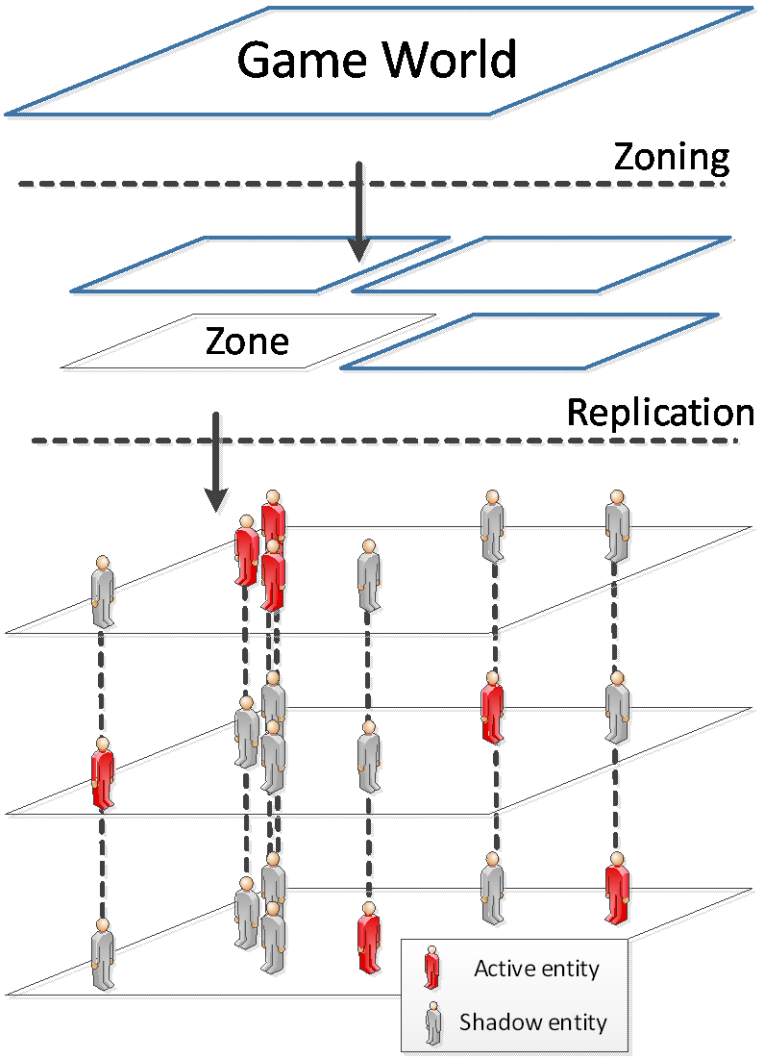


Figure 3. Game zone parallelization [11]

3.2 **Player behavior**

Depending on different online games, players may have different actions in each online game [15]. In general, each Massively Multiplayer Online Role-Playing Game (MMORPG, which is a kind of MMOGs) can generalize in following four kinds of player behavior [15]:

1) Fighting: Avatars combat with other avatars or monsters. In this action, players usually need the highest real-time requirement. If delay time is too long, it may trigger player quit the game.

2) Questing: Avatars explore a map and experience the story of a game. In this action, players need the median real-time requirement.

3) Trading: There are business interactions between avatars, or avatars collect materials for making avatars' equipment. In this action, players need low real-time requirement.

4) AFK: In the Away From Keyboard (AFK), players leave the screen over a period of time (5 minutes in WoW). In this mode, real-time experience is not important.

In Chapter 4, we assign a different weight level (highest to lowest: "Fighting", "Questing", "Trading", and "AFK") to each player behavior in our load model.

3.3 **Area of Interest**

An Area of Interest (AoI) is the area of concern to a player (in MMOG). Each avatar has its own AoI which is defines by its visibility scope. An avatar only can interact (fighting, trading or do nothing) with the entities which are in its AoI scope. In other words, avatars can see avatar *A*'s behavior when they are in the avatar *A*'s AoI. An area with several overlapping AoIs will become a hotspot that may generate large load in MMOGs. We can prevent load increasing by limiting the size of a

hotspot. However, limiting the size of the hotspot would be equal to limit the scope of the interaction between players. This approach will decrease the players' game experience. Therefore, in this paper, we do not consider limiting the size of a hotspot.

Chapter 4

MMOG Load Modeling

There are three main types of resources used for MMOGs: CPU, memory, and network bandwidth [10]. Table II shows definition of notations that are used in our load models. We assume that the number of active avatars per VM in the same game zone is almost the same. We enhance Nae et al.'s load model [10] by considering player behavior. In this way, we can improve resource over-allocation.

4.1 CPU load model for a VM

For modeling the CPU load of a VM in a game zone, we summarize three factors of CPU time which are T_m, T_u, and T_b for an MMOG server. We model the CPU time T_m as the time sending and receiving messages from a VM$_i$ to each client (active avatar) as

$$T_m = AE_i \times t_m.$$

The CPU time T_u spent for processing state updates from the VM$_i$ to the other VMs is

$$T_u = \sum_{j=1, j\neq i}^{N} (AE_j + BE_j) \times t_u.$$

The CPU time T_b spent for computing the different behavior between different active avatars is

$$T_b = \sum_{avatar_k \in \{AE_i\}} AoI_k \times t_b^k$$

Table II. Notation Definition.

Notation	Definition
N	Number of VMs for a game zone
BE_i	Number of NPCs/bots for VM_i, which is one of VMs serving in a game zone
AE_i	Number of active avatars for VM_i, which is one of VMs serving in a game zone
$\{AE_i\}$	The set of active avatars for VM_i
AoI_k	Number of entities for $avatar_k$'s AoI
t_m	Processing time for a VM to process an event message from a client
t_u	Update time of entity states received from/sent to another VM
B	The set of four kinds of possible behavior types
t_b	Computation time of a behavior type, $b \in$ B
t_b^k	Computation time for avatar *k* with behavior type *b*
m_{cs}	The amount of memory needed for a VM to store the state of one avatar
m_{es}	The amount of memory needed for a VM to store the state of an NPC entity
m_{game}	The amount of memory a VM used for running the actual game engine with no game world loaded and no client connected
m_{world}	The amount of memory a VM used with the game world being played
d_{cin}	The amount of data received from a client
d_{cout}	The amount of data sent to a client
d_{updt}	The amount of data exchanged between VMs for updating a single entity state

Thus, the total CPU time spent in one tick is

$$T_C = T_m + T_u + T_b.$$

Finally, we define the CPU load function as:

$$Load_{CPU} = \frac{T_C}{T_{SAT}},$$

where T_{SAT} is a tick saturation threshold.

4.2 **Memory load model for a VM**

The memory load model is less complex than the CPU load model; we can define the memory load function for each VM as follow:

$$Load_{mem} = \frac{(\sum_{i=1}^{N} AE_i \times m_{cs} + BE_i \times m_{es}) + m_{game} + m_{world}}{M_{vm}},$$

where M_{vm} represents the amount of memory available in a VM. Each VM requires the state information of all active avatars and NPCs/bots in the same Game Zone in order to synchronize each VM.

4.3 **Network load model for a VM**

The incoming network bandwidth usage for each VM in a game zone is as follows:

$$D_{in} = AE_i \times d_{cin} + \sum_{j=1, j\neq i}^{N} (AE_j + BE_j) \times d_{updt}.$$

The outgoing network bandwidth usage for each VM in a game zone is defined as follows:

$$D_{out} = AE_i \times d_{cout} + (N-1) \times (AE_i + BE_i) \times d_{updt}.$$

Therefore, we define the overall network load as follows:

$$Load_{NET} = max\left(\frac{D_{in}}{BW_{in}}, \frac{D_{out}}{BW_{out}}\right),$$

where BW_{in} and BW_{out} denote the input and output network bandwidths, respectively.

4.4 **Complete load model for a VM**

According to the above three load model, we derive the overall load for each VM by choosing the maximum load from each individual resource load shown as follows:

$$Load_{VM} = max(Load_{CPU}, Load_{mem}, Load_{NET}).$$

Chapter 5

Proposed Player Behavior-based Resource Allocation

5.1 PB-RA architecture

Figure 4 shows the system architecture for our proposed Player Behavior-based Resource Allocation (PB-RA), which contains two parts: *Main Server* and *Game Zones*. An MMOG has a Main Server and several Game Zones. The Main Server is responsible for storing monitoring history records and according to these records to predict resource requirements for each game zone with the helping of a Neural Network-based predictor to ensure the experiences of players who are in a virtual machine (VM). Each Game Zone is connected to the Main Server to get the information for maintaining the game world. The following is a brief description of each module. *History Storage* stores the game data collected by Zone Handler. *Neural Network-based Predictor* predicts the number of players for each behavior in each zone via the neural network. *Load Model* calculates the server load of next time slot, which will be sent to *Zone Handlers* for adjusting number of VMs. *Communication* facilitates the exchange of messages between Game Zones or forwards VM adjusting information to Zone Handlers. *Video Streamer* is a component which transforms gaming graphics to streaming video. *Zone Handler* is a main controller to communicate with Main Server and to synchronize VMs and to process necessary computations of a game zone. If the overall load exceeds the capacity provided by current VMs, Zone Handler will add a *Mirror VM* to support the Zone Handler.

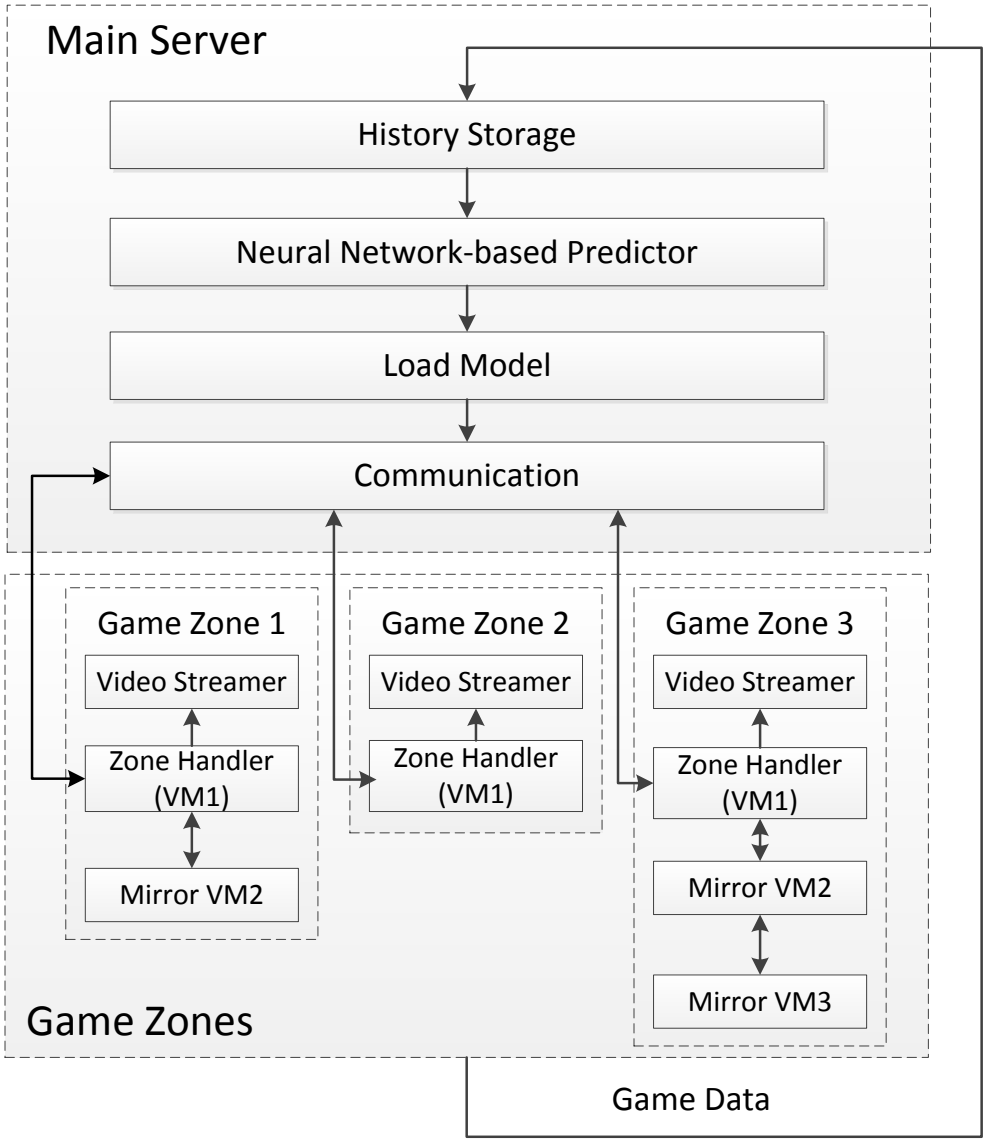


Figure 4. Proposed PB-RA Architecture.

5.2 **Neural network-based predictor**

A neural network (NN) is a simulation of biological neural network [14]. It simulates human's brain by mathematical statistics techniques. The neural network connects many simple artificial nodes which called "neurons." Neural networks have following features: parallel process, fault tolerance, associative memory, solution

optimization, etc [14].

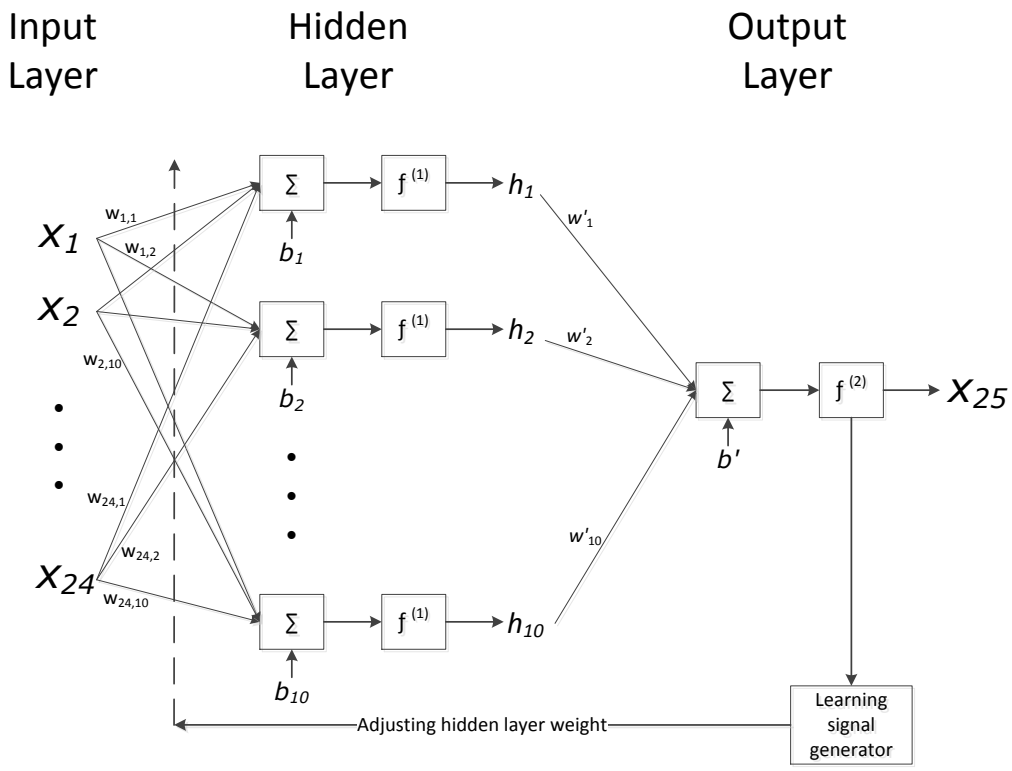


Figure 5. ANN-based predictor.

In this paper, we choose a NN-based predictor because that predictor is based on the most suitable prediction method that was observed in [9], [10], [17]. We ran our NN-based predictor on MATLAB which is a high-level language and interactive environment for numerical computation, visualization, and programming [16]. Figure 5 shows such a network with 24 network inputs and one network output. We input the number of players for each behavior from time slot 1 to time slot 24 and predict the number of players for each behavior at time slot 25. The neural network has a hidden layer, which has 10 neurons. Neurons in the hidden layer and the output layer perform calculations based on the following equations, respectively:

$$h_j = f(\sum_{i=1}^{24} x_i w_{i,j} + b_j)$$

$$x_{25} = f(\sum_{i=1}^{10} h_i w_i' + b')$$

where $f(x) = \begin{cases} \frac{1}{1+e^{-x}} & for\ a\ hidden\ layer\ neuron\ \ (log\text{-}sigmoid) \\ x & for\ an\ output\ layer\ neuron\ \ (linear) \end{cases}$

Note that h_i is the output of the hidden layer, x_i is an input, $w_{i,j}$ is a weight modifying x_i, and b_j is the bias of the hidden layer. x_{25} is the output of the predictor, h_i is an input of the output layer, w_i' is a weight modifying h_i, and b' is the bias of the output layer. In addition, $f(x)$ is either a log-sigmoid transfer function for the hidden layer neurons, or a linear transfer function for the output layer neurons. We used this neural network-based predictor to predict the number of players in each behavior.

5.3 **PB-RA flowchart for a zone**

In our approach, each zone initializes a suitable VM size according to statistical results of history data. In general, the VMs are able to meet the requirements of players and guarantee the response time that satisfies the quality of player experiences.

Figure 6 shows the flowchart of PB-RA. At first, Main Server obtains the number of players in this zone and stores the collected data to History Storage. After that, Main Server determines the zone type. If the zone type is one of Dungeon, Raid or Battlefield, the NN-based predictor only predicts the number of players in this zone. In these types of zones, avatars always combat with each other, and they require the highest real-time experience. Therefore, we always assume the player behavior is "Fighting" so that satisfy QoS requirements. If the zone type is Home City or Normal Map, the NN-based predictor will predict the number of players for each behavior. Then we will assign different weights for different behavior types and the priority

order is “Fighting” > “Questing” > “Trading” > “AFK” that is based on the measures in WoW. Next, Main Server derives the maximum load according to our load model described in Chapter 4. Afterwards, Main Server passes the load information to Zone Handler.

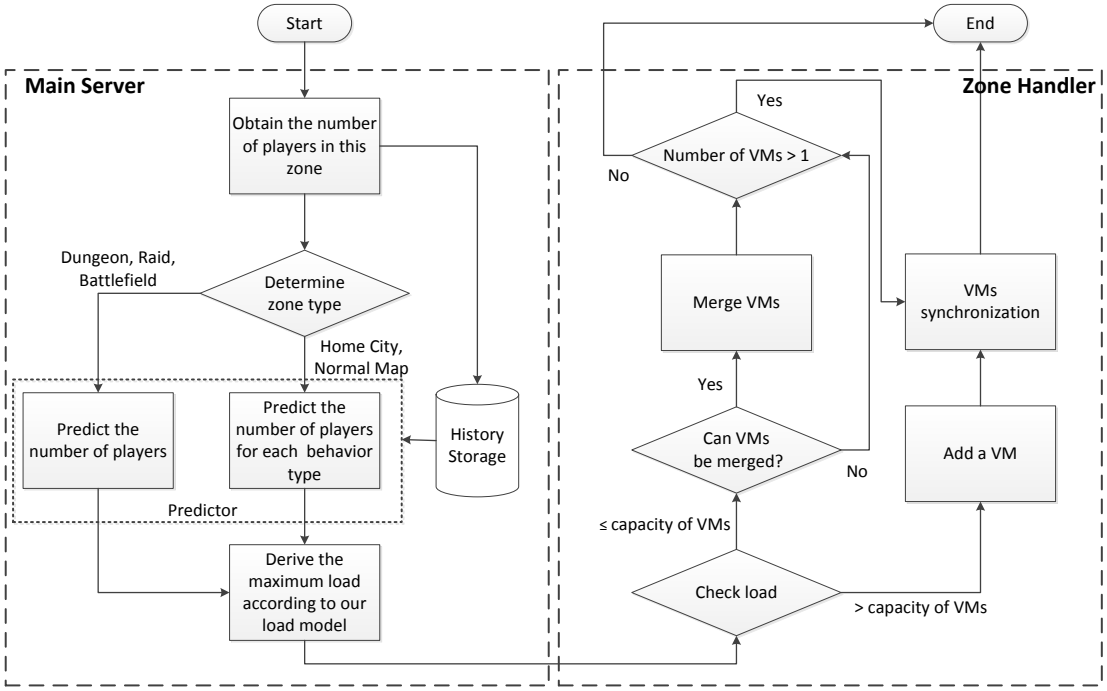


Figure 6. The flowchart of PB-RA.

After receiving the load information form Main Server, Zone Handler checks the load of each VM in the Game Zone. If the predicted load is greater than the capacity of total VMs in this Game Zone, Zone Handler will add a mirror VM to lighten the load of Game Zone. If the predicted load is smaller than the current capacity of VMs, Zone Handler will decide which working VMs can be merged. Finally, if the number of working VMs in Game Zone is greater than two, Zone Handler will synchronize the VMs’ state of avatars and monster information.

Chapter 6

Evaluation

In this chapter, we first describe how to collect game data, show simulation setup and then discuss simulation results.

6.1 **Data collection**

In our simulation, we used World of Warcraft (WoW) as a case study, which is the most famous MMOG in the world; according to statistics, it has 10.2 million subscribers in February 2012 [1]. We collected game data from WoW clients for two weeks, from April 22 to May 7, and the collected data are recorded every five minutes. The details of our game data collection are described in the following. First, we chose the map named *Valley of the Four Winds* as our collecting area, which is a popular map for avatars of level-90. Because this map is abundant in natural resources which are requirements for producing equipment, the avatar density and behavior diversity are higher than the other maps. Next, we distributed 21 WoW clients over this map, and log the state information of each client by typing the */combatlog* command. *Combatlog* is a kind of log, which records combat actions and results within the radius of 200 yards from each WoW client, as Figure 7 shows. *Combatlog* may also includes results of profession activities of nearby characters [22]. Last, we parsed and analyzed all the combat log derive the number of players for each behavior.

Figure 7. The distribution of WoW clients.

6.2 **Simulation setup**

Table III shows our simulation setup. Although parsing the *Combatlog* can derive the number of players for each behavior, the location of each player is still unknown. Therefore, we randomly generated the location for each player at the beginning to distribute avatars uniformly in a game zone and update the direction and mobile distance for each avatar in each timeslot to simulate a real game zone environment. According to the game data we collected in History Storage, we predicted the player behavior for each avatar in the next timeslot by using a neural network-based predictor in MATLAB. We applied the prediction results to our load model in order to predict the number of VMs required in the next timeslot. Then, we used ColudSim which is a famous cloud simulation tool to simulate PB-RA. In our simulation, we assumed one VM can support 31 avatars [18], each VM had 512 MB memory and 10

Mbps bandwidth, and VM startup time was 100 seconds [19]. The radius of AoI was set to 50 yards which is the default vision range in WoW. We implemented our load model introduced in Chapter 4 on CloudSim and follow the flowchart of PB-RA to adjust number of VMs used.

Table III. Simulation setup.

Game data	Collected from World of Warcraft (WoW) (version 5.2.0)
Prediction technique	Neural network
Prediction tool	MATLAB (version 7.11.0)
Evaluation period	2013/04/22 15:43 ~ 2013/05/07 15:36
Cloud simulation tool	CloudSim 3.0
VM startup time	100 sec. [19]
VM capacity	31 avatars [18]
VM memory	512 MB (CloudSim default value)
VM incoming bandwidth	10 Mbps (CloudSim default value)
VM outgoing bandwidth	10 Mbps (CloudSim default value)
Radius of AoI	50 yards in WoW

6.3 **Simulation results**

In our simulation, we collected the game data from WoW clients for two weeks from April 22 to May 7. However, a full week is enough to reflect the ecological cycle because the story events will be reset every week in WoW. Therefore, we only used one week data in our simulation. According to our load models in Chapter 4, we compare the proposed PB-RA with the methods considering number of players and interaction of players [10] in terms of total resource requirement, number of VMs used, resource over-allocation rate, and resource under-allocation rate in a WoW's map which is called *Valley of the Four Winds*. The calculation of the represent the resource over/under allocation rate is as follows:

$$\delta = n_{vm}^{(pred)} \times 100\% - \lambda_{usage},$$

$$\begin{cases} if\ \delta \geq 0, over\text{-}allocation \\ if\ \delta < 0, under\text{-}allocation \end{cases}$$

where $n_{vm}^{(pred)}$ is the predicted number of VMs required and λ_{usage} is the actual usage of resources. We calculated λ_{usage} via using actual number of players for each behavior calculated by our load model introduced in Chapter 4. In Figure 8 and Figure 9, we show the total resource requirements and number of VMs in the *Valley of the Four Winds* of WoW, repectively. In both figures, we can see that the required number of VMs will vary with time. The number of VMs required achieves the minimum at 6:00 a.m. and the maximum at 8:00 p.m. every day. Furthermore, adjusting the number of VMs dynamically is better than that statically. In addition, in terms of the number of VM need in a zone, the proposed PB-RA is 55% lower than the method only considering number of players and 33% lower than the method considering interaction of players [10].

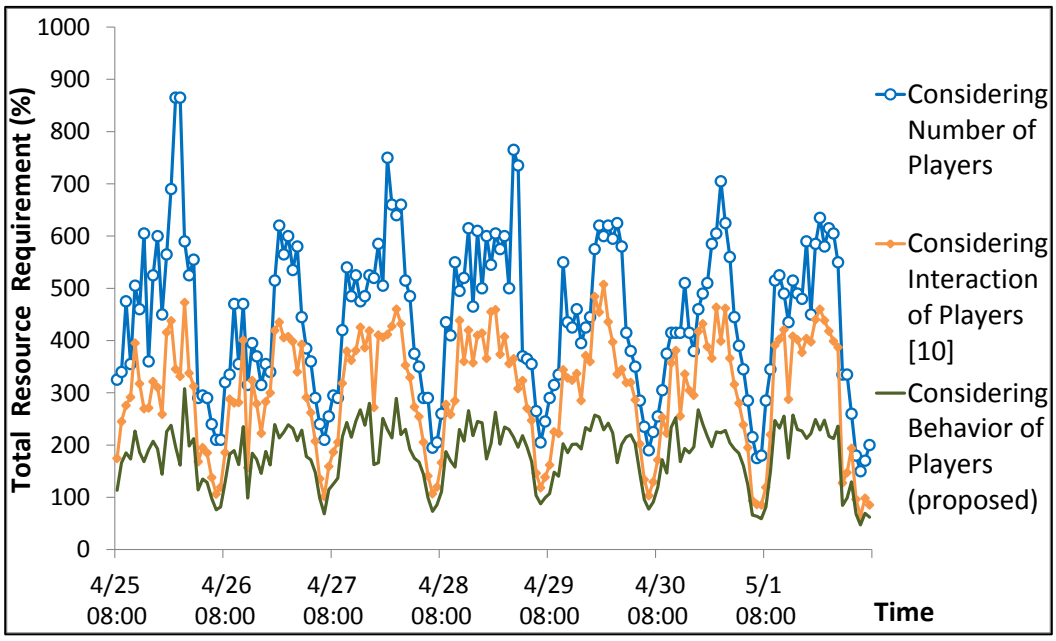


Figure 8. Total resource requirements in Valley of the Four Winds of WoW.

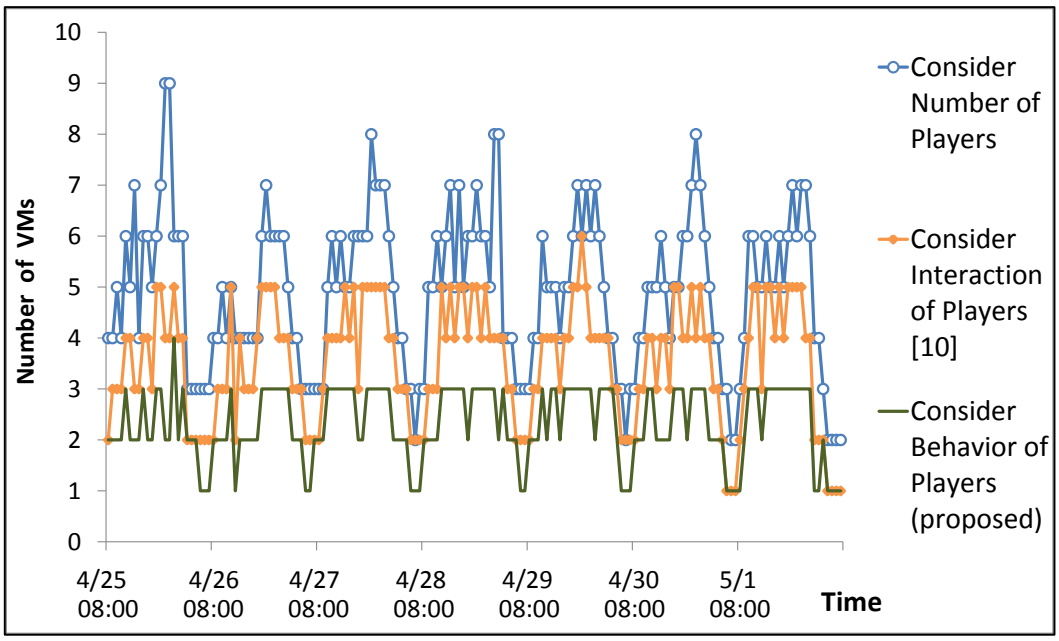


Figure 9. Number of VMs used in Valley of the Four Winds of WoW.

Finally, we evaluate the resource over-allocation rate and resource under-allocation rate of the proposed PB-RA. Resource over-allocation means that the allocated resources exceed the resource requirements of a zone and resource under-allocation means that the allocated resources do not meet the resource requirements of a zone. Figure 10 shows the over-allocation rate of PB-RA is lower than the other methods. The result shows that PB-RA can reduce 74% of resource over-allocation rate compared to the method that only considers number of players and 50% compared to the method that considers interaction of players [10] as well. Figure 11 shows the number of under-allocation events over two weeks for each method. Although the method only considering number of players did not has under-allocation, its over-allocation is too high. The proposed PB-RA's number of resource under-allocation events is no more than 1.05 times compared with the method that considers interaction of players [10]. Table IV shows the average under-allocation rate and the percentage of events with the under-allocation rate

higher/lower than the average under-allocation rate per VM.

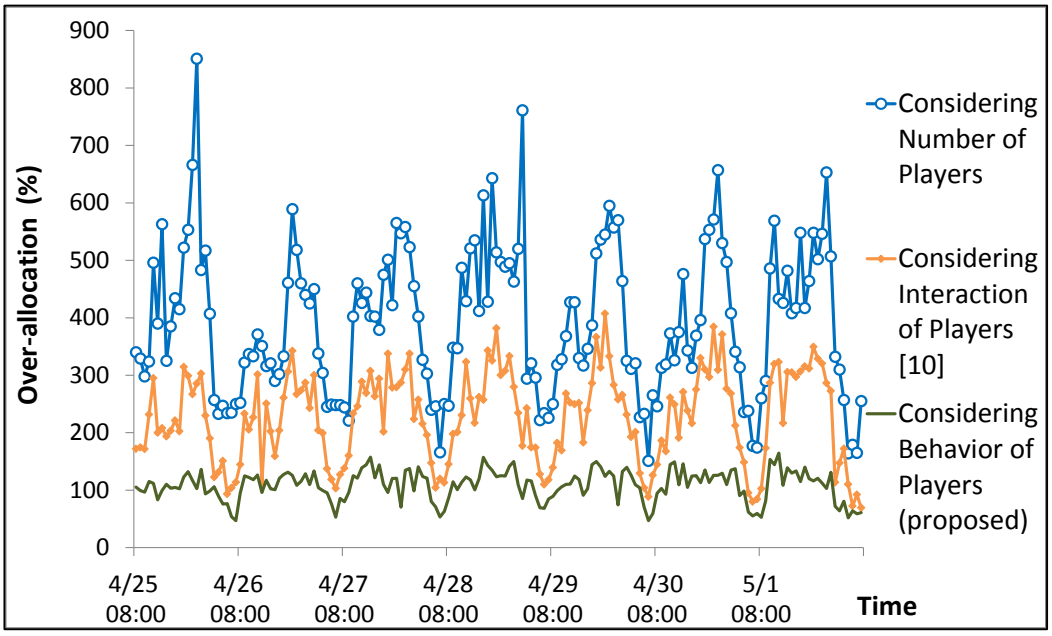


Figure 10. Over-allocation of each method.

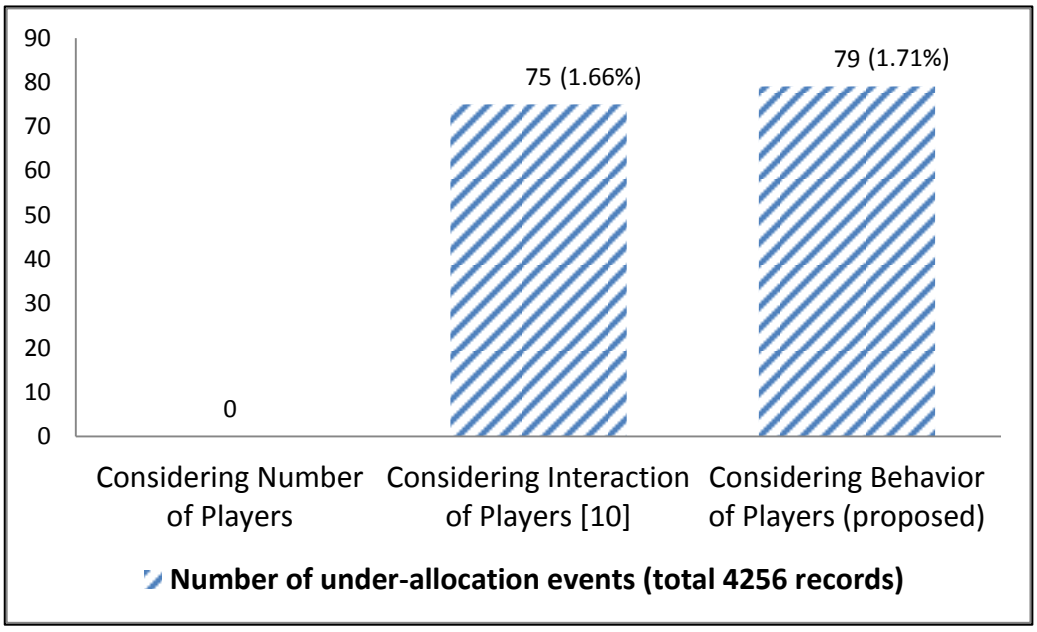


Figure 11. Number of under-allocation events of each method.

Table IV. The detailed information of resource under-allocation rate.

	Average	Higher than average	Lower than average
Considering Number of Players	0	0	0
Considering Interaction of Players [10]	4.08%	35% (26/75)	65% (49/75)
Considering Behavior of Players (proposed PB-RA)	3.97%	35% (28/79)	65% (51/79)

Chapter 7

Conclusion

7.1 Concluding remarks

We have presented a Player Behavior-based Resource Allocation (PB-RA) scheme for MMOG clouds. The behavior of players is classified into four types: Fighting, Trading, Questing and AFK. We used WoW as a case study to illustrate the proposed PB-RA that is applicable to MMOGs and each behavior type results in a different load. We predict the number of players for each behavior type in one map through a neural network-based predictor and measure loads generated from different player behavior types. As a result, we can predict total resource requirements more accurately for players with different behavior types in the map. That is, we can allocate cloud resources more efficiently. Experiment results have show that the proposed PB-RA can reduce 74% of resource over-allocation compared to the method that only considers number of players, and 50% compared to the method that considers interaction of players [10] as well. Moreover, in terms of the number of resource under-allocation events the proposed PB-RA is no more than 1.05 times compared with the method that considers interaction of players [10].

7.2 Future work

In this paper, we collected the game data only from MMOG clients. In the future, we may obtain the real data of game servers to have more accurate evaluation. In addition, we will implement the Video Streamer to achieve real cloud gaming.

References

[1] “MMOGs Subscription,” [Online]. Available: http://mmodata.net/.

[2] “Cloud Gaming,” [Online]. Available: http://en.wikipedia.org/wiki/Cloud_gaming/.

[3] D. T. Ahmed and S. Shirmohammadi, “A Microcell Oriented Load Balancing Model for Collaborative Virtual Environments,” in *Proc. IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, VECIMS’08*, pp. 86- 91, July 2008.

[4] D. T. Ahmed and S. Shirmohammadi, “Uniform and Non-Uniform Zoning for Load Balancing in Virtual Environments,” in *Proc. IEEE International Conference on Embedded and Multimedia Computing, EMC’10*, pp. 1-6, Aug. 2010.

[5] B. V. D. Bossche, B. D. Vleeschauwer, T. Verdickt; F. D. Turck, B. Dhoedt, and P. Demeester, “Autonomic Microcell Assignment in Massively Distributed Online Virtual Environments,” *Journal of Network and Computer Applications*, vol. 32, no. 6, pp. 1242-1256, Nov. 2009.

[6] C. E. B. Bezerra, J. L. D. Comba, and C. F. R Geyer, “A Fine Granularity Load Balancing Technique for MMOG Servers Using a KD-Tree to Partition the Space,” in *Proc. VIII Brazilian Symposium on Games and Digital Entertainment, SBGAMES’09*, pp. 17-26, Oct. 2009.

[7] L. D. Briceño, H. J. Siegel, A. A. Maciejewski, Y. Hong, B. Lock, M. N. Teli, F. Wedyan, C. Panaccione, C. Klumph, K. Willman, and C. Zhang, “Robust Resource Allocation in a Massive Multiplayer Online Gaming Environment,” in

Proc. ACM 4th International Conference on Foundations of Digital Games, FDG'09, pp. 232-239, 2009.

[8] M. Marzolla, S. Ferretti, and G. D'Angelo, "Dynamic Resource Provisioning for Cloud-based Gaming Infrastructures," in *ACM Computers in Entertainment - Theoretical and Practical Computer Applications in Entertainment, CIE'12*, vol. 10, no. 3, pp. 1-20, Dec. 2012.

[9] V. Nae, A. Iosup, S. Podlipnig, R. Prodan, D. Epema, and T. Fahringer, "Efficient Management of Data Center Resources for Massively Multiplayer Online Games," in *Proc. IEEE International Conference on High Performance Computing, Networking, Storage and Analysis, SC'08*, pp. 1-12, Nov. 2008.

[10] V. Nae, A. Iosup, and R. Prodan, "Dynamic Resource Provisioning in Massively Multiplayer Online Games, " *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 380-395, March 2011.

[11] F. Glinka, A. Ploβ, J. Müller-Iden, and S. Gorlatch, "RTF: A Real-Time Framework for Developing Scalable Multiplayer Online Games," in *Proc. ACM 6th SIGCOMM Workshop on Network and System Support for Games, NetGames'07*, pp. 81-86, Setp. 2007.

[12] "OnLive," [Online]. Available: http://www.onlive.com/.

[13] "CloudSim," [Online]. Available: http://www.cloudbus.org/cloudsim/.

[14] "Artificial Neural Network," [Online]. Available: http://en.wikipedia.org/wiki/Artificial_neural_network/.

[15] M. Suznjevic, I. Stupar, and M. Matijasevic, "MMORPG Player Behavior Model Based on Player Action Categories," in *Proc. IEEE 10th Annual Workshop on Network and Systems Support for Games, NetGames'11*, pp. 1-6, Oct. 2011.

[16] "MATLAB," [Online]. Available: http://www.mathworks.com/products/matlab/.

[17] R. Prodan and V. Nae, "Prediction-based Real-time Resource Provisioning for Massively Multiplayer Online Games," *Journal of Future Generation Computer Systems*, vol. 25, no. 7, pp. 785-793, July 2009.

[18] Y. Lee and K. Chen, "Is server consolidation beneficial to MMORPG? a case study of World of Warcraft," in *Proc. IEEE 3rd International Conference on Cloud Computing, CLOUD'10*, pp. 435-442, July 2010.

[19] M. Mao and M. Humphrey, "A Performance Study on the VM Startup Time in the Cloud," in *Proc. IEEE 5th International Conference on Cloud Computing, CLOUD'12*, pp. 423-430, June 2012.

[20] D. T. Ahmed and S. Shirmohammadi, "A Dynamic Area of Interest Management and Collaboration Model for P2P," in *Proc. IEEE/ACM 12th International Symposium on Distributed Simulation and Real-Time Applications, DS-RT'08*, pp. 27-34, Oct. 2008.

[21] "MMOGs market capitalization," [Online]. Available: http://www.thealistdaily.com/news/global-mmo-games-spending-over-12-billion-report/.

[22] "Combat Log," [Online]. Available: http://www.wowwiki.com/Combat_Log/.