國立交通大學

資訊科學與工程研究所

碩 士 論 文

基於熱點預知之雲端大型多人線上遊戲之動態資源配置

Dynamic Resource Provisioning with Hotspot Anticipation for MMOG Clouds

研 究 生：張晏誌

指導教授：王國禎　教授

中 華 民 國　102　年　7　月

基於熱點預知之雲端大型多人線上遊戲之動態資源配置

Dynamic Resource Provisioning with Hotspot Anticipation for MMOG Clouds

研 究 生：張晏誌　　Student：Yen-Chih Chang

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

基於熱點預知之雲端大型多人線上遊戲之動態資源配置

學生：張晏誌　　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

近年來，有許多類型的遊戲改為網路服務的經營模式。大型多人線上遊戲(MMOG)成為世界上最受歡迎的遊戲服務類型。為了節省營運成本及簡化遊戲伺服器的管理，有許多遊戲服務供應商將他們的服務與雲端計算科技結合。在大型多人線上遊戲的虛擬世界中，玩家經常藉由成群結隊的行動，來達成某些遊戲任務或擊敗遊戲中的魔王。這樣的行為模式可能造成虛擬世界中的”熱點”。在熱點中的玩家，常需頻繁地互動，產生大量的負載，以至於造成服務品質的下降。若有太多的熱點同時存在同一台伺服器中，遊戲的流暢度將會受到影響。為了解決這個問題，傳統的方式會藉由高估單一遊戲地圖的負載量，

分配遠超過需求的資源來維持遊戲品質，但此會造成遊戲資源上的浪費。在本論文中，我們提出一個基於熱點預知的動態資源配置方案(NN-Player+DRP-HA)，並且使用一個有限狀態機來表示玩家狀態及其可能的狀態轉換。我們結合玩家的狀態和類神經網路(neural network)對下一個時間點玩家數量進行預測，我們可以計算出地圖上熱點造成的潛在負載，並且分配適當的計算資源來消化這些負載。實驗結果顯示，我們提出的方法可以在不造成嚴重的資源過度配置的情況下，降低資源配置不足的機率。與現存的另一代表性之動態資源配置方法(NN-Player+DRP)比較，我們提出的方法可以以控制 CPU 過度分配的比率不超過一台虛擬機器容量的前提下，將 CPU 資源分配不足的次數從 2.16% 降低至 0.42%（改進了 80%）。

關鍵詞：雲端計算、動態資源配置、熱點預知、大型多人線上遊戲、資源配置不足。

Dynamic Resource Provisioning

with Hotspot Anticipation for MMOG Clouds

Student: Yen-Chih Chang Advisor: Dr. Kuochen Wang

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

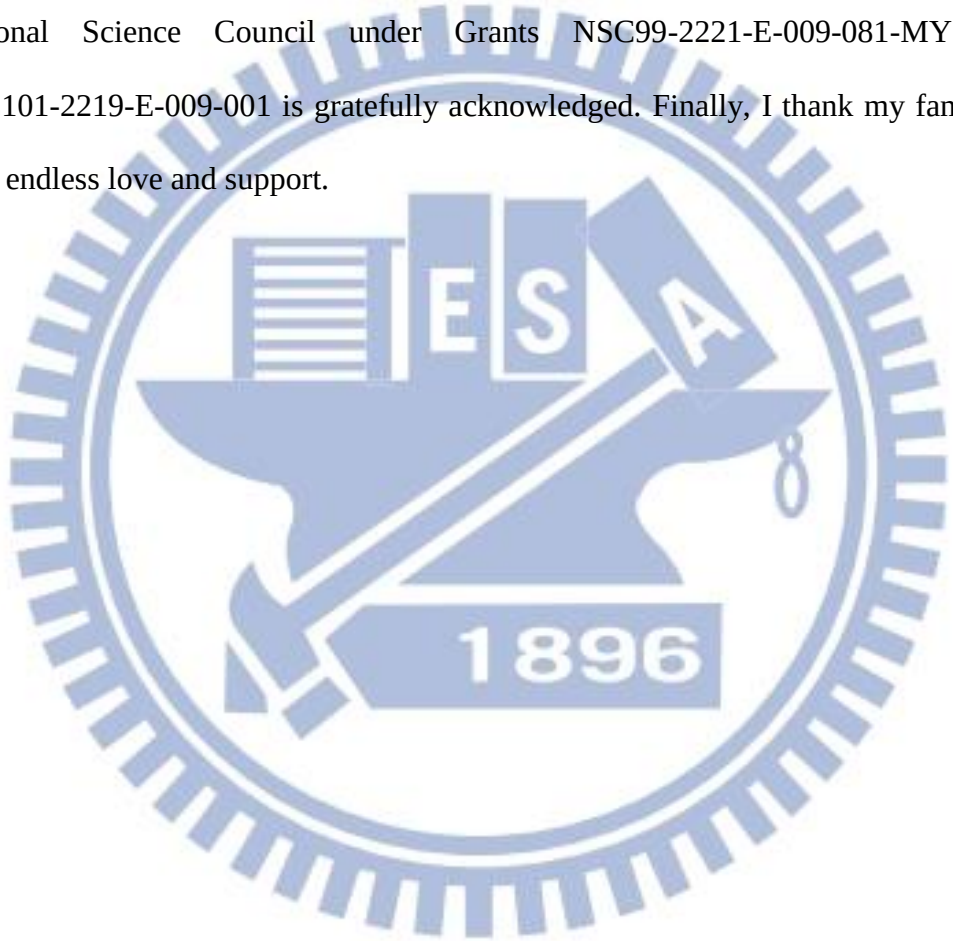
Recently, there are various kinds of games that have been served via the internet. An example of such games is MMOG (Massively Multiplayer Online Game) that has become the most popular game service in the world. For saving operating cost and simplifying management of servers, gaming service providers are combining their online services with cloud computing technology. In MMOG virtual environments, avatars are often acting as a group to help one another to achieve certain goals or defeat bosses, which may become a *hotspot* in a virtual world. Frequent interactions between avatars in hotspots may generate lots of workload and may cause decrease of quality of service (QoS). The latency of gaming service will increase when there are too many hotspots in a single server, which may harm the quality of experience (QoE) enormously. To address this problem, in general, games operators over-allocate resources to game zones, which may cause the waste of gaming resources. In this paper, we propose a *dynamic resource provisioning with hotspot anticipation* scheme, called NN-Player+DRP-HA that employs a vector based model to monitor the movement of avatars in a virtual world. Furthermore, we use a finite state machine to

represent possible avatar states and state transitions. By combining the state of each avatar in a game zone with a neural network (NN) predictor, we may figure out potential workload produced by hotspots, and then allocate appropriate computing resources to support the game zone. Experimental results support that the proposed NN-Player+DRP-HA scheme can avoid most of under-allocation events with an acceptable over-allocation rate. Compared with a representative dynamic resource provisioning method, called NN-Player+DRP, the proposed NN-Player+DRP-HA reduces the probability of under-allocation events from 2.16% to 0.42% (80% improvement) in terms of CPU capacity of a VM, under the premise of controlling the CPU over-allocation rate within the CPU capacity of one VM.

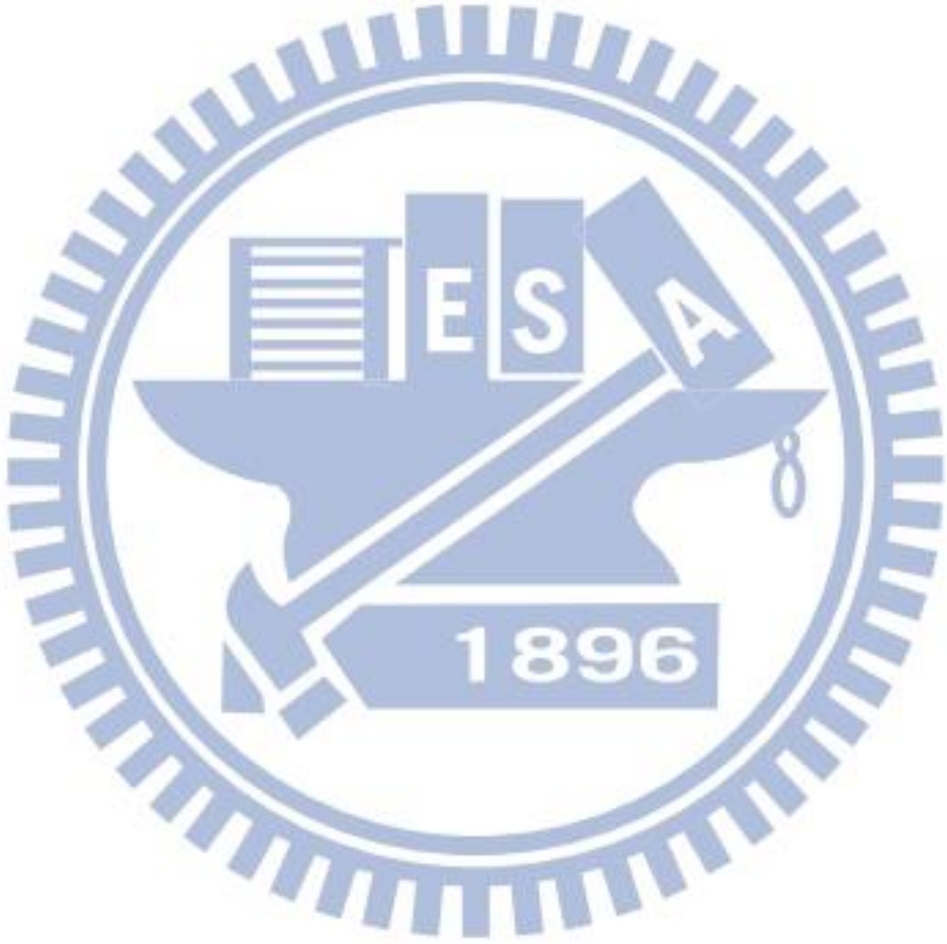
Keywords: Cloud computing; dynamic resource provisioning; hotspot anticipation; MMOG; under-allocation

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.

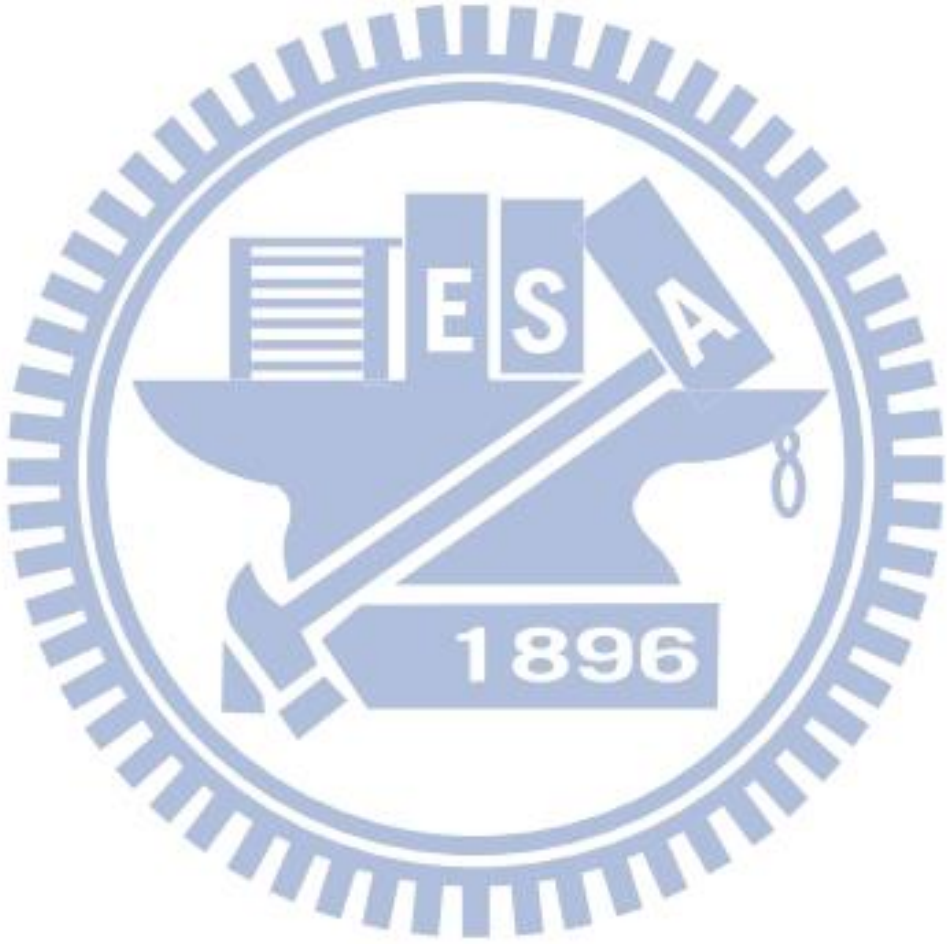


Contents



List of Figures

List of Tables



Chapter 1

Introduction

The Massively Multiplayer Online Game (MMOG) has played an important role in this generation of the internet. According to existing research [1], over half of the internet users are MMOG players, which attract a lot of companies to join the MMOG market. To support a large scale of players, game operators have invested a lot of money to operate and manage gaming servers.

Nowadays, more and more gaming service providers are combining their online services with cloud computing technology. It reduces the initial construction cost of MMOG servers and consolidates computing resources to simplify the server management. Furthermore, cloud computing technology offloads the graphics computation to the server side, which means gaming clients could be low-end devices, so called thin-clients. That is, players may use their hand-held devices to access gaming services via the internet as long as the devices are able to display streaming video data. The user experience using thin-clients will be identical to that using computers or game consoles. The most important thing is cloud computing technology enables gaming service providers to provide resources dynamically, and it creates an opportunity to save more operating cost of gaming servers.

Although cloud computing technology is beneficial to game applications, here comes one problem. Online games are delay-sensitive applications. For those requests from the client side, servers have to respond them in a short while; otherwise, the response time may be unacceptable to players. Note that response time is the most affecting factor to gaming experience of players [2]. In other words, the occurrence of

game delay impacts on the player loyalty directly, which may cause the losing of subscribers. For this reason, if gaming service providers want to leverage the dynamic resource provisioning model for saving operating cost, they must monitor the resource allocating status in real-time and predict the resource requirement in the near future. Therefore, gaming service providers can allocate enough computing resources to every zone to avoid the delay caused by under-allocation, and make sure that response time of the whole system is acceptable.

In MMOGs, for arousing interest and increasing interaction opportunities between players, operators may create some stochastic events, such as a wild boss [3] appearing suddenly. These events will attract many players to come, and further form groups. In such a situation, avatars in the same group are in one another AoI (Area of Interest), indicating that one's action will affect other avatars. Thus, updating status of avatars may frequently happen in this area, which becomes a "hotspot" in the virtual world. If there are hotspots existing in a zone, numerous status update tasks may produce in a single server, which may reduce QoS tremendously. In general, game operators manage resources in the way of over-estimating resource requirements of players to maintain their good gaming experience, which cause the waste of server resources.

In this paper, we discuss the impacts of interactions between avatars on the workload of cloud servers. We then propose a dynamic resource provisioning with hotspot anticipation to avoid the delay caused by resource under-allocation, and reduce the resource over-allocation as well. The remainder of this paper is organized as follows. We introduce the background and review of related work in Chapter 2. Chapter 3 presents the proposed dynamic resource provisioning with hotspot anticipation for MMOG clouds. Chapter 4 evaluates the experiment results of the proposed design. Finally, concluding remarks and future work are given in Chapter 5.

Chapter 2

Background and Related Work

2.1 **Background**

2.1.1 Zones and Replications

In a zone-based game world, a complete virtual world is composed by several zones. Each zone is served by an individual server which can be a physical one or a virtual one. The size and shape of a zone are decided according to the zone management policy. In a zone-based game world, it is easy to determine the serving host of each avatar. However, the zone-based approach suffers from several defects. First of all, there exist problems in inter-zone communications when two avatars, belonging to different zones, need to communicate. In this situation, a huge amount of network traffic and computation tasks could decline the quality of service. Secondly, it limits the computation resources allocated to a zone. In general, game operators use a fix-sized machine to supply resources for a zone. From the point of view for management, it is easy to maintain. However, if there are numerous requests coming suddenly, this model is unable to guarantee QoS. Moreover, because of the lack of flexibility, the resource allocation of MMOG is usually in over-allocation state due to the lack of flexibility; and that may leads to low resource utilization, further results in much unnecessary costs to service operators.

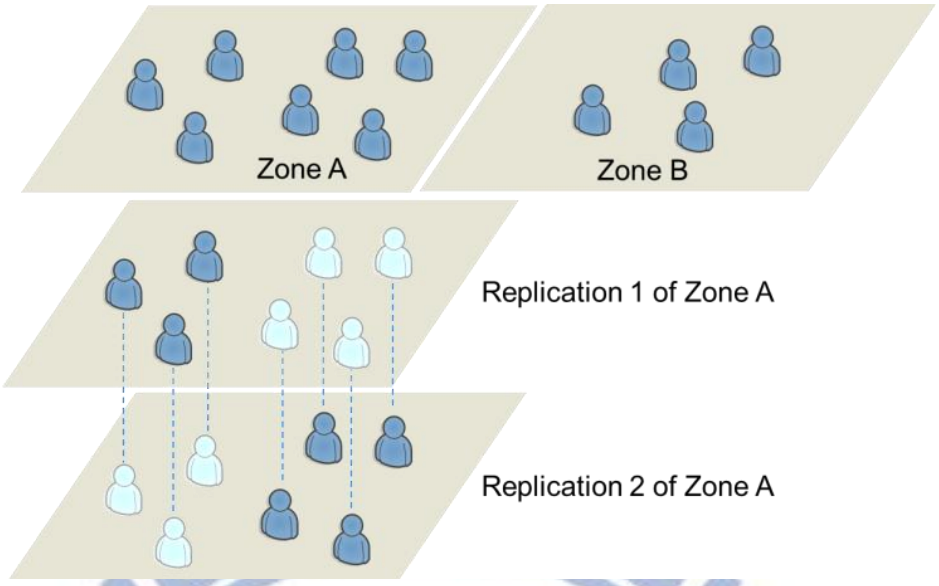


Figure 1: An example of zones and replications.

To avoid the above problems, in most cases, cloud gaming systems adopt the replication scheme. As we shown in Figure 1, in this scheme, each machine serves a different set of avatars in the same virtual world. Between servers, they perform synchronization periodically and maintain “shadow entities” for those avatars served by other servers. Thus each avatar still can communicate with ones who are served by other servers.

2.1.2 Load Management

Rapid elasticity and on-demand self-service are characteristics of cloud computing, which enables the provision of cloud resources on demand whenever they are required. Tenants are charged accord to their usages. Therefore, reducing the resource usages is as important as reducing the response time for gaming service providers.

In cloud computing environments, in general, service providers monitor their QoS to decide whether resizing is necessary. For example, when QoS is descended to a certain threshold or there is too much computing resource in idle mode, resizing can balance the cost and quality of service. A resizing procedure includes powering on/off

VMs, synchronization of VMs and redirection of player connections. All of these actions will consume computing resources. Therefore, threshold settings must take these effects into consideration.

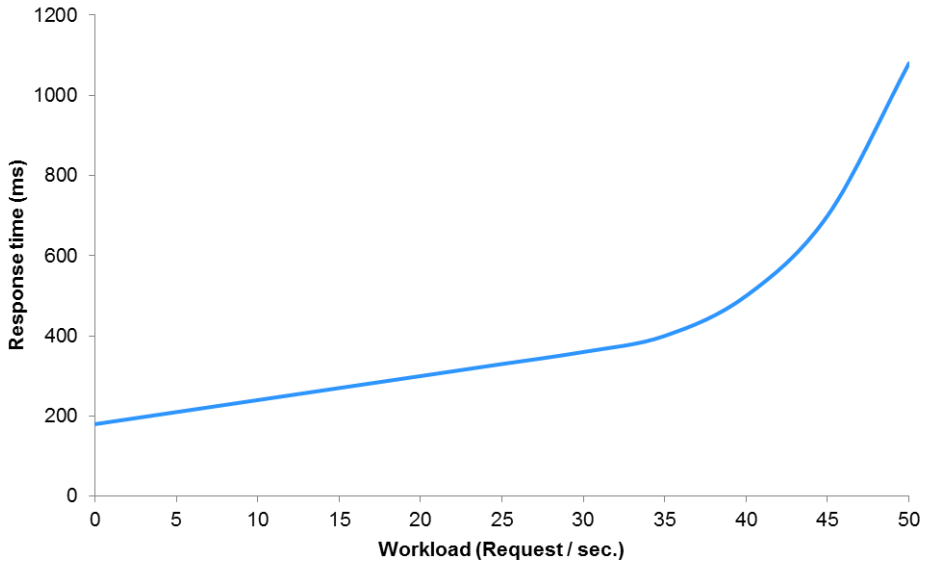


Figure 2: Relation between workload and response time.

According to existing research [4], relation between workload and response time demonstrates non-linear growth. Figure 2 is an example of this relation using a 2.2GHz processor. It shows that with heavier workload, response time grows faster.

2.2 **Related Work**

In this section, we will review related work. The related work can be classified into three categories: avatar mobility, dynamic resource provision and AoI management. The classification tree of related work is shown in Figure 3.

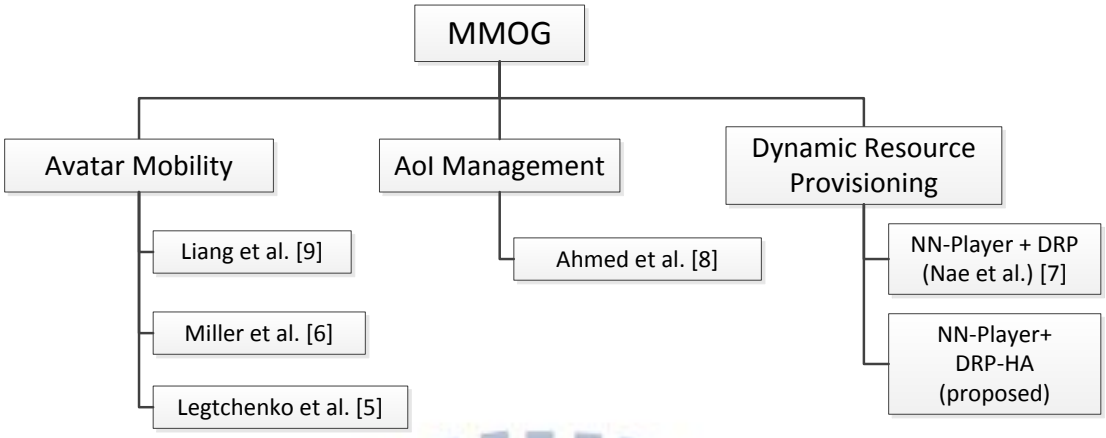


Figure 3: Classification of related work.

2.2.1 Avatar Mobility

In MMOGs, or NVEs (Network Virtual Environments), movement of avatars is a kind of factors that contributes to server load, which produces message exchanging events, status updating events and so on. Therefore, many researches are trying to predict or model the movement of avatars. Legtchenko et al. proposed a mechanism to predict and model avatar movement, which called “Blue Banana” [5]. This approach improves latency of message transfering in a peer-to-peer overlay by foreseeing movement of avatars in the near future. They introduced three modes of avatar mobility: travelling, exploring and in hotspot. In travelling mode, avatars move fast and straight. In exploring mode, avatars move slowly and trajectories are winding. Avatars in hotspot are acting like in exploring mode but the difference is that they have other avatars in their AoI [5].

Miller et al. proposed another research about avatar movement [6]. They measured avatar movement in World of Warcraft battlegrounds, and modeled their observation. They found that hotspot a model is more likely to represent the avatar movement in WoW.

2.2.2 Resource Provisioning

In general, game operators use a simple policy to manage their computing resources. In the beta phase of MMOGs, they estimate the basic resource requirement of their service, and build servers according to this estimation. In the running phase of MMOG services, they observe the utilization of those servers; if overloading occurs in some servers, operators will buy new machines to maintain QoS in an acceptable level. However, to achieve better resource utilization, operators may port their gaming services into cloud platforms; therefore, monitoring or even predicting workload of game servers become necessary.

Nae et al. [7] proposed a serial of research for dynamic resource provisioning in MMOGs. They derived several load models for MMOGs to describe the workload of CPU, memory and network bandwidth in different situations.

2.2.3 AoI Management

In virtual world environments, AoI (area of interest) defines a visual range of a player. When there's an event taking place in this range, the effect of this event must show on the player's screen. In virtual worlds, each avatar is able to observe changes in its AoI. To save computing resources, servers only update the events which take place within one's AoI. Ahmed et al. [8] proposed a dynamic AoI management approach which didn't partition the world map into zones. They partition the world map according to the distribution of AoI. Hence, it's not necessary to deal with inter-zone communications.

2.3 **Summary**

Table 1 is a comparison table which shows the differences between related work and the proposed DRP-HA. Our objective is proposing a dynamic resource allocation mechanism with hotspot anticipation for MMOG environments. In addition, we also

consider interactions between avatars.

Table 1: Comparison of related work on dynamic resource provisioning.

Approach	Description	Architecture	Load prediction	Resource Provisioning
Nae [7]	Resource provisioning for MMOGs	Multi-servers	Neural netowrk	Based on number of players
DRP-HA (Proposed)	Resource provisioning for MMOG clouds	Gaming cloud	Neural network	Based on number of players and hotspot detection

Chapter 3

Proposed Dynamic Resource Provisioning with Hotspot Anticipation

3.1 System Overview

As we described in previous chapters, our objective is to propose a dynamic resource allocation approach to avoid under-allocation of cloud computing resources, which considers the overhead of extra load that are produced by frequent interactions between avatars in a hotspot. According to the research proposed by H. Liang et al., avatars move slowly and chaotically within the hotspots, fast and straight between hotspots [9]. In our approach, we leverage this property to determine if avatars are moving toward hotspots.

Figure 4 is the system architecture of our proposed scheme (DRP-HA). Zone servers allocate VMs in a cloud datacenter which simulates a virtual world. History Storage stores and provides history data of number of avatars to NN Predictor. NN Predictor is a neural network (NN) based predictor that predicts the number of avatars in the next time slot by history data. The DRP-HA module is used to decide to power on/off VMs according to the current number of VMs, current positions and moving speeds of avatars and the prediction results of NN Predictor.

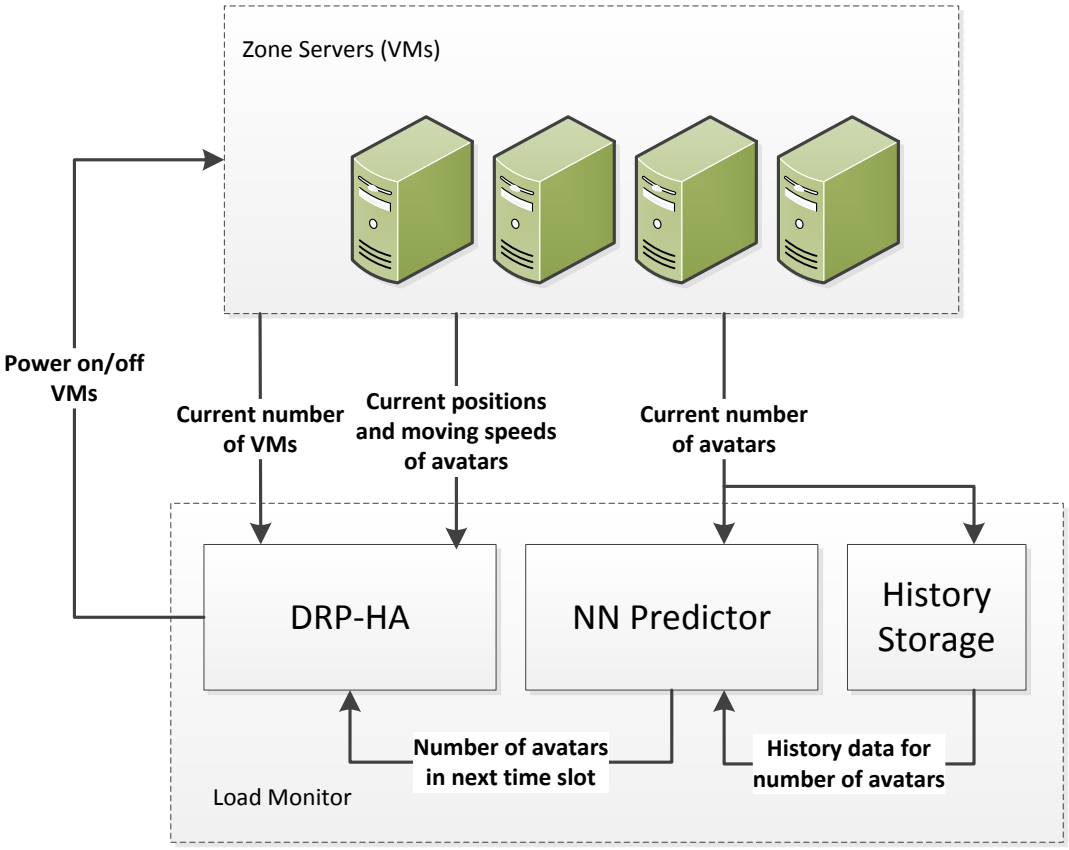


Figure 4: The system architecture of DRP-HA.

3.2 **Avatar Mobility**

We define a scheme to monitor the movement of avatars. As shown in Figure 5, the circle presents the AoI of avatar *a*. And we use a vector V_a to represent the moving direction and speed of *a*. D_a is the distance that *a* can moves within a time slot. As we mentioned in Chapter 2, the VM scaling procedure takes a few minutes to finish. Therefore, the length of a time slot must longer than the length of the VM scaling procedure; otherwise, the gaming system may result in under-allocation.

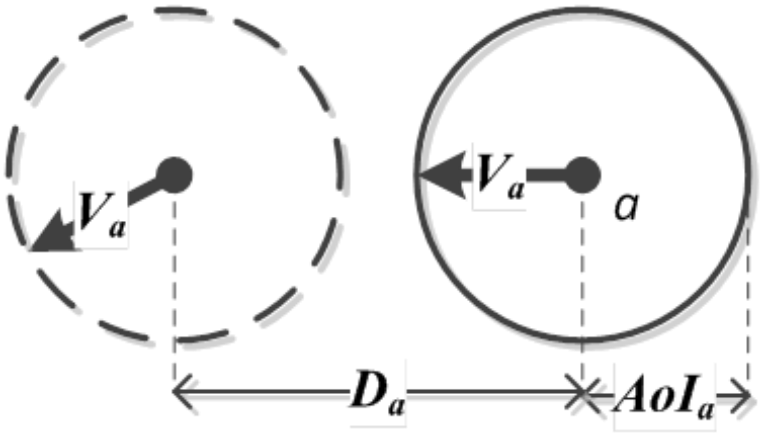


Figure 5: An illustration of the movement of avatar a in a time slot.

Our main objective of dynamic resource provisioning is to avoid under-allocation. If avatars move into a hotspot, it results in extra loading, which is the main concern of our design. As we mentioned, avatars are move more quickly and straightly between hotspots, this property enables us to predict whether avatars will move into a hotspot in the next time slot.

Therefore, we first define a hotspot h, as shown in Figure 6. A hotspot is a set of AoIs such that at least two avatars must be in each other's AoI. We use following two statements to define a hotspot. In following description, h_a denotes the hotspot of avatar a.

$$\{AoI_a \in h_a \mid \exists\ avatar\ b \neq avatar\ a, b\ is\ in\ AoI_a\}$$

$$\text{if } h_a = h_b, h_b = h_c, then\ h_a = h_c$$

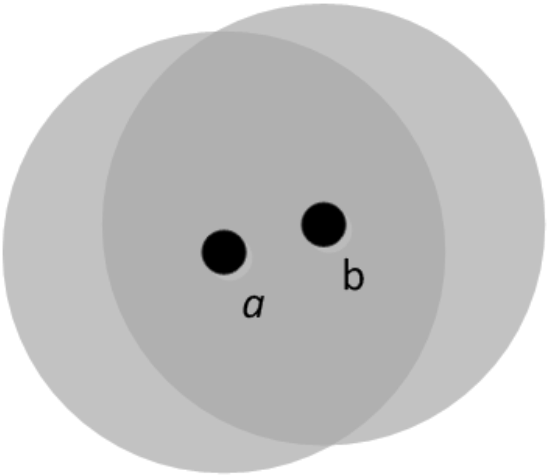


Figure 6: Definition of a hotspot.

3.3 **Finite State Machine Model**

Next, we use a finite state machine (FSM) to represent the state transition of an avatar, as we shown in Figure 7. There are three states of players. If the original state of an avatar *a* is State (I), which indicates that there was no any avatar in *a*'s AoI. In such situation, two possible events may occur. Event (a) means avatar *a* is moving toward a hotspot *h* and *a*'s speed is fast enough to reach hotspot *h* within a time slot. In other words, D_a is close to $|V_a| * t$, where *t* is the period of a time slot. Event (a) transfers *a*'s state from State (I) into State(II). Otherwise, if this avatar forms a hotspot while traveling (Event (f)), *a*'s state will be transferred to State (III).

If the original state of an avatar *a* is State (II), it indicates that avatar *a* may enter a hotspot *h* within a time slot (Event (c)) and its state will be transferred to State (III), as shown in Figure 8. Load Monitor, as shown in Figure 4, calculates resource requirements based on the assumption that avatar *a* is in hotspot *h*. If Event (b) occurs, which means avatar *a* changes its direction, avatar *a* will no longer reach any hotspot in the next time slot, *a*'s state will be transferred to State (I).

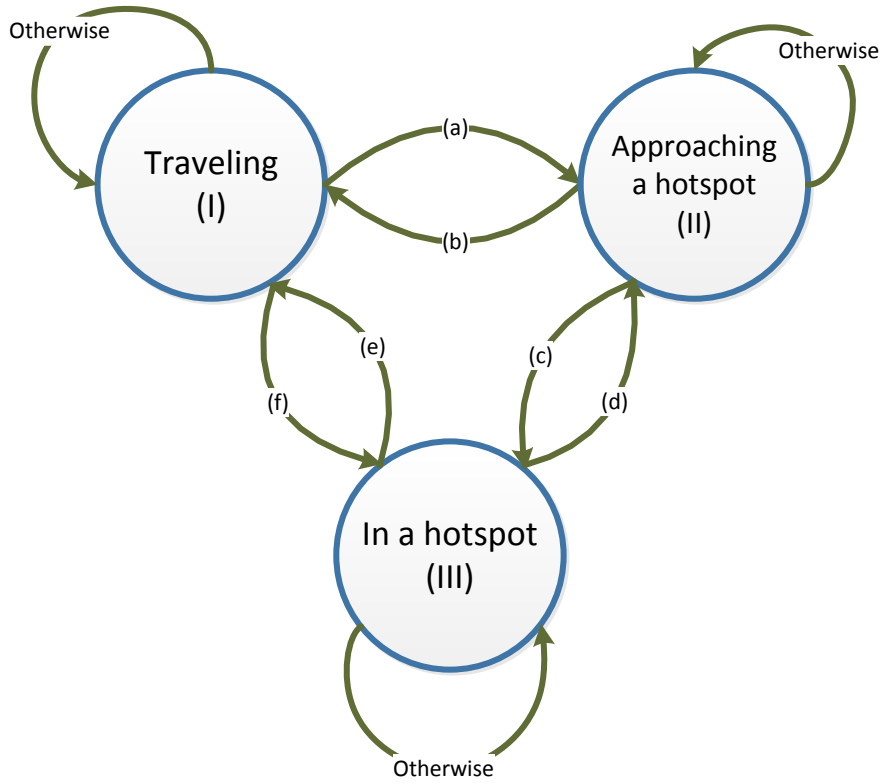


Figure 7: The state transition of avatars in DRP-HA.

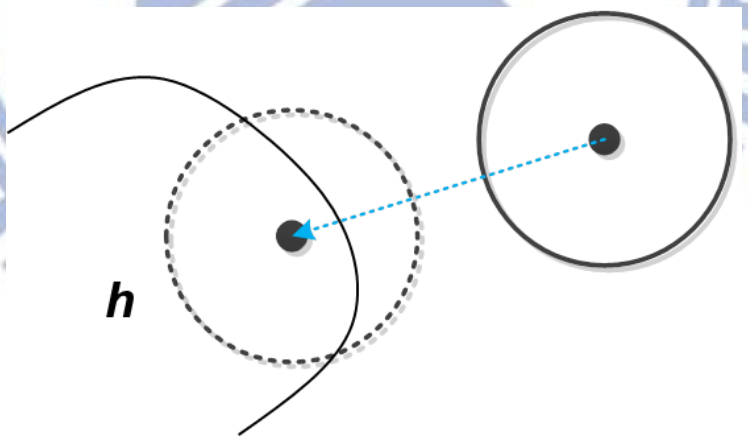


Figure 8: The situation of state (II).

If the original state of avatar *a* is State (III), it indicates that avatar *a* is in a hotspot *h*, and *a* can observe the actions produced by other avatars in *h*. If event (d) occurs, which means avatar *a* leaves the original hotspot *h*, but it still may reach a hotspot within a time slot, then *a*'s state will be transferred to State (II). On the other

hand, if it will not reach any hotspot (Event (e)), *a*'s state will be transferred to State (I).

We adopt these three states to determine the growth of members in hotspots. In the beginning of every time slot, we use the information we mentioned above to perform a state update procedure to decide whether the resizing VM scaling of this zone is necessary. The flowchart of the proposed DRP-HA scheme is shown in Figure 9, and we briefly describe the algorithm of updating avatar states, as shown in algorithm 1. According to the result of the above algorithm, we can know that if there is a new AoI joins a hotspot, if there are members leaving a hotspot, or if a hotspot will continue to exist in the next time slot.

The flowchart of NN-Player+DRP-HA is shown in Figure 9. In Step 1, neural network predictor outputs the number of avatars in the next time slot. In Step 2, Load Monitor detects the moving speed of each avatar, and updates their states by algorithm 1, which can anticipate the status of each hotspot in a game zone. For example, we detect the joining/leaving of avatars for each hotspot. Therefore, we can anticipate the percentages of interactions involving avatars and NPCs, which denotes as p_{ci} and p_{ei}, and the number of avatars and NPCs involving interactions, which denotes as IC_h and BE_h for each hotspot h. In Step 3, we use a Load Model to calculate a predicted load in the next timeslot by p_{ci}, p_{ei}, IC_h and BE_h, which we will describe in detail in next section. Finally we determine if the current number of allocated VMs is appropriate, in other words, if the allocated VMs meets the demand of current workload of the gaming cloud, and the over-allocation rate is less than 100. The formula we used for evaluating over-allocation rate will introduce in Chapter 4.

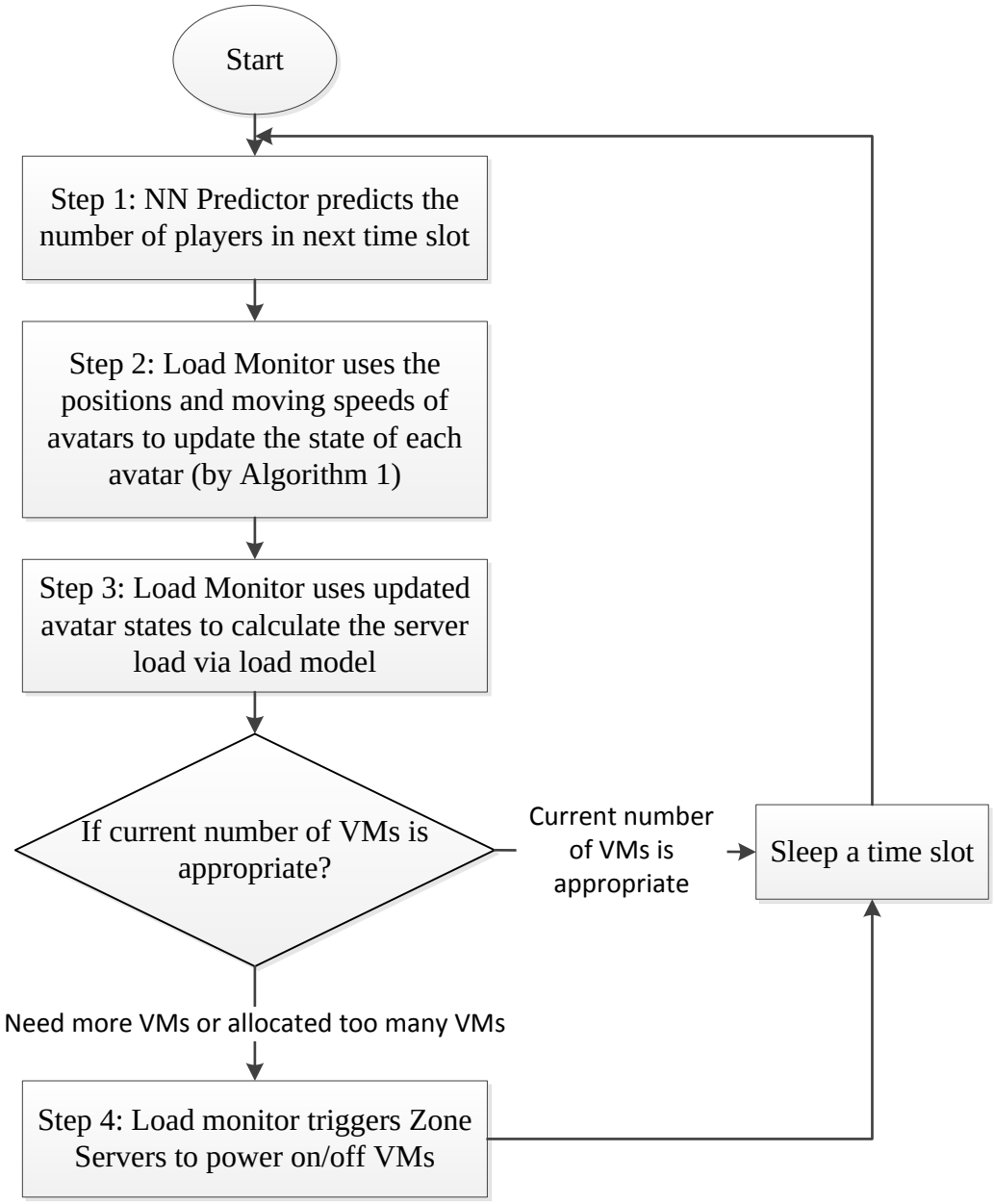


Figure 9: The flowchart of the proposed DRP-HA scheme.

Algorithm 1: Update the states of avatars

p_a' = (x', y'). The position of $avatar_a$ in previous time slot
p_a = (x, y). The current position of $avatar_a$
$\overrightarrow{V_a} = \alpha\vec{i} + \beta\vec{j}$ is the moving vector of $avatar_a$
$state_a$ = the current state of $avatar_a$
$hotspot_a$ = the hotspot that $avatar_a$ is belonged to

for *a* = 1 to *number of players* **do**
  if $state_a$ is "*traveling*" **then**
    if $|p_a - p_a'| = |\overrightarrow{V_a} * t| * (1-\varepsilon)$ **then** //where ε is a small number
      if *in_hotspot*$(x+\alpha, y+\beta)$ **then**
        $state_a$ = "*approaching a hotspot*"
    if there is any other avatars $avatar_b$ in AOI($avatar_a$) **then**
      $state_a$ = "*in a hotspot*", $hotspot_a$ = *new hotspot ID*
      $state_b$ = "*in a hotspot*", $hotspot_b$ = *new hotspot ID*
  else if $state_a$ is "*approaching a hotspot*" **then**
    if *in_hotspot*(x, y) **then**
      $state_a$ = "*in a hotspot*", $hotspot_a$ = *ID of the hotspot covers (x, y)*
    else if *distance*$(p_a', \text{all of avatars}) > |\overrightarrow{V_a} * t|$ **then**
      $state_a$ = "*traveling*"
  else // $state_a$ is "in a hotspot"
    if there are no other avatars in AOI($avatar_a$) **then**
      if *in_hotspot*$(x+\alpha, y+\beta)$ **then**
        $state_a$ = "*approaching a hotspot*"
      else if $|p_a - p_a'| = |\overrightarrow{V_a} * t| * (1-\varepsilon)$ **then** //where ε is a small number
        if *distance*$(p_a', \text{all of avatars}) > |\overrightarrow{V_a} * t|$ **then**
          $state_a$ = "*traveling*"
end

In algorithm 1, we demonstrate the state update procedure in detail. Note that ε is a small number (close to zero), which means if $|p_a - p_a'|$ is very close to $|\overrightarrow{V_a} * t|$, the statement $|p_a - p_a'| = |\overrightarrow{V_a} * t| * (1-\varepsilon)$ is true. The setting of ε is an implementation issue, the system will become more sensitive when ε becomes bigger, but this may cause overestimate of numbers of in-hotspot-avatars, results in higher over-allocation rate of resources. The *in_hotspot(x, y)* function in algorithm 1 helps us to determine if the

given position *(x, y)* is in certain hotspot. In this function, we perform linear search, if the given position is in certain avatar's AoI and this avatar is in a hotspot, *in_hotspot(x, y)* will return true, else, return false.

3.4 **Load Model**

To figure out the load caused by hotspots, we leverage the load model proposed by V. Nae et al. in [7]. According to their work, interactions between players only affect CPU utilization; memory utilization and bandwidth utilization are only related to the number of players [7]. Therefore, in the proposed design, we focus on CPU utilization. The load model of CPU utilization which V. Nae et al. proposed is shown below [7]:

$$Load_{CPU} = Load_{update} + Load_{interaction}$$
$$= \frac{t_u}{t_{SAT}}(N + BE) + \left(\frac{t_i}{t_{SAT}} \cdot p_{ci} \cdot f(IC, IC) + \frac{t_i}{t_{SAT}} \cdot p_{ei} \cdot f(IC, BE)\right)$$

Where $Load_{update}$ is the workload for proceeding state updates from the other machines and $Load_{interaction}$ is the workload for computing the interaction between entities. In the above formula, N is the number of players connected to this game session, IC is the number of avatars that involve interactions, BE is the total number of moving bots or NPCs (Non-Player Characters), t_u is the updating time for a single entity, t_i is the computing time for one interaction, t_{SAT} is the saturation threshold to evaluate CPU performance, and p_{ci} and p_{ei} are the percentages of interactions involving avatars and NPCs, respectively. Note that *f(x, y)* is the formula to represent message numbers of each interaction class. For $O(n^2)$ and $O(n^3)$ class interactions, *f(x,y)* is $x \cdot y$ and $x^2 \cdot y$, respectively [7].

In our design, we assume that all the avatars and NPCs are involving the interactions occurring in each hotspot. IC_h is the number of in-hotspot-avatars for

estimating the load of an individual hotspot *h*. The $Load_{interaction}$ in [7] can be reduced into following for calculating the load of hotspot *h*:

$$Load_h = \frac{t_i}{t_{SAT}} \cdot (p_{ci} \cdot f(IC_h, IC_h) + p_{ei} \cdot f(IC_h, BE_h))$$

Therefore, the total load of a zone is the sum of load produced by each hotspot and updating operations of every VM and avatar which not involving any interactions. The following formula is used to estimate the total load of a zone.

$$Load_{CPU} = Load_{update} + Load_{interaction}$$
$$= \frac{t_u}{t_{SAT}} \cdot (N + BE) + \sum_{h=1}^{H} Load_h$$

Chapter 4

Evaluation Results

4.1 Experimental Setup

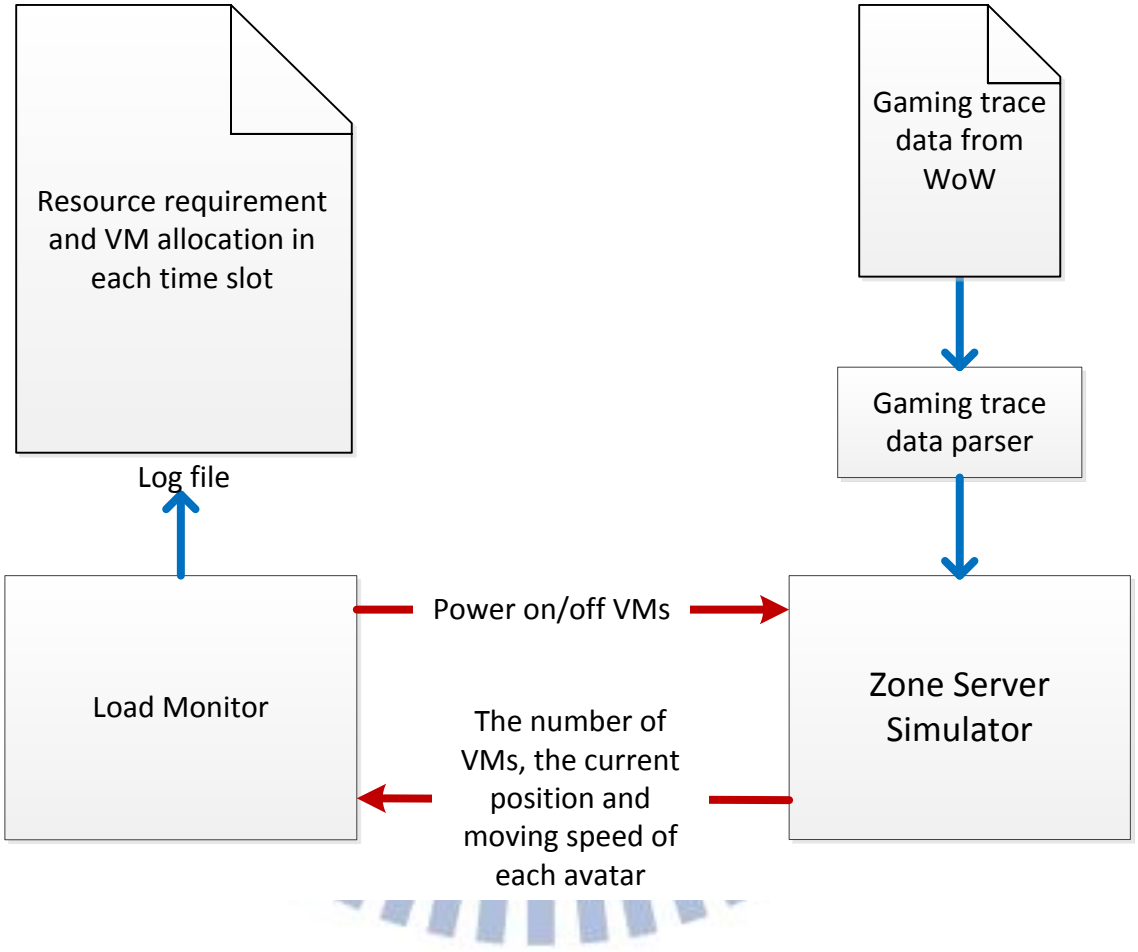


Figure 10: Experimental environment..

To evaluate our proposed approach, we collect avatar information from World of Warcraft (WoW), which is the most popular MMOG these years. Figure 10 is our experimental environment. A gaming trace data parser is used to analyze the records we collected from WoW. These records include the move-in/move-out of avatars in this zone and the playing characters of avatars. These data will be sent to the zone

server simulator, which can simulate the movement of avatars, then we use the load model we mentioned in previous chapter to calculate the resource requirements of every time slot for the zone servers. It also can simulate the power on/off of VMs. The adjustment the number of VMs is controlled by load monitor, which receives the resource requirements and number of current supporting VMs from zone server simulator; determine an appropriate number VMs to support the zone, and generate log files eventually.

Table 2: Simulation parameters.

AoI Range	50 yards
Avatar speed	10 yards/min.
VM capacity	31 Avatars [10]
VM startup time	100 sec. [11]
Time slot	5 min.
Interaction class	$O(n^2)$
VM capacity	31 avatars
Game data	From World of Warcraft (v5.2.0)
Evaluation period	2013/4/22 15:43 to 2013/5/7 15:36
Evaluation map	*Valley of the four winds*

Table 2 shows the parameters used for simulation. The range of AoI is 50 yards and avatar speed is 10 yards per minute. These are approximate values in WoW. For servers, we assume the capacity of a VM is 31 avatars [11], and VM startup time is 100 seconds, which is the startup time of a linux server in Amazon EC2 cloud [12]. The evaluation period is from 2013/4/22 15:43 to 2013/5/7 15:36. We use */combatlog* command, which is a built-in command in WoW, to get the interaction information of avatars in a virtual world. Among all maps in WoW, we choose the *Vlley of the Four Winds* to collect data, since player interactions within the map occur frequently.

4.2 **Definition of Allocation Status**

To evaluate the performance of our proposed scheme, we define a formula α to determine the over/under allocation rate.

$$\alpha = n_{vm}^{(allo)} \times 100\% - \lambda_{usage}, \begin{cases} if\ \alpha \geq 0, over\text{-}allocation \\ if\ \alpha < 0, under\text{-}allocation \end{cases}$$

where $n_{vm}^{(allo)}$ is the number of allocated VMs.

λ_{usage} is the actual usages of computing resources.

According to this formula, the allocation status is either over-allocation or under-allocation. We assume that the scale of a zone is adjusted according to supporting number of VMs in an MMOG cloud. Therefore, α will between 0 and 100 if current allocated computing capability is sufficient to support this game zone, and the wasted computing resource is minimized. It is impossible to decrease the over-allocation rate by shutting down supporting VMs in this situation.

4.3 **Simulation Results**

Figure 11, Figure 12, Figure 13 and Figure 14 are the simulation results of our experiments. The results of NN-Load are derived by using a neural network tool provided by MATLAB 7.11.0 (R2010b) to predict server load. The type of the neural network we chose is dynamic time series, which takes *d* past values and *d* past prediction results as input and predicts the next value. In the NN-Load experiment, since the length of a time slot is 5 minutes, we set *d = 24*, which means the inputs are the CPU usages and the prediction result of past two hours and output is the prediction value of next 5 minutes. The NN-Player+DRP experiment uses the same setting of neural network; the difference between NN-Load experiment is that the inputs are 24 past values and 24 past prediction value of player numbers in the next time slot. The predicted number of players is a parameter for load model to compute an estimated load.

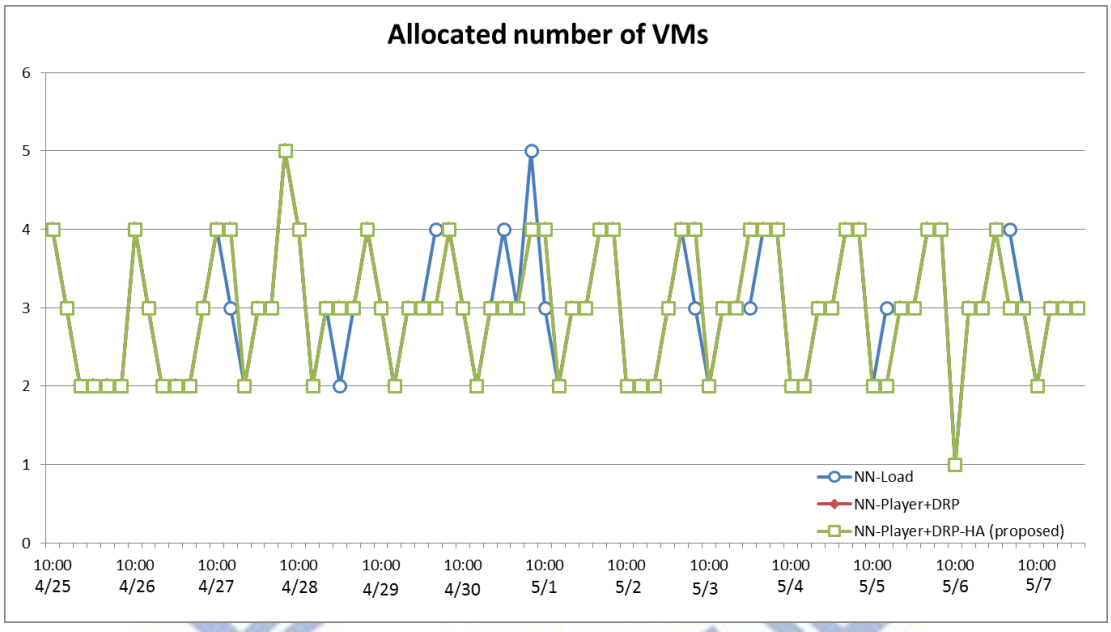


Figure 11: Maximum number of allocated VMs comparison in all time slots for three different approaches.

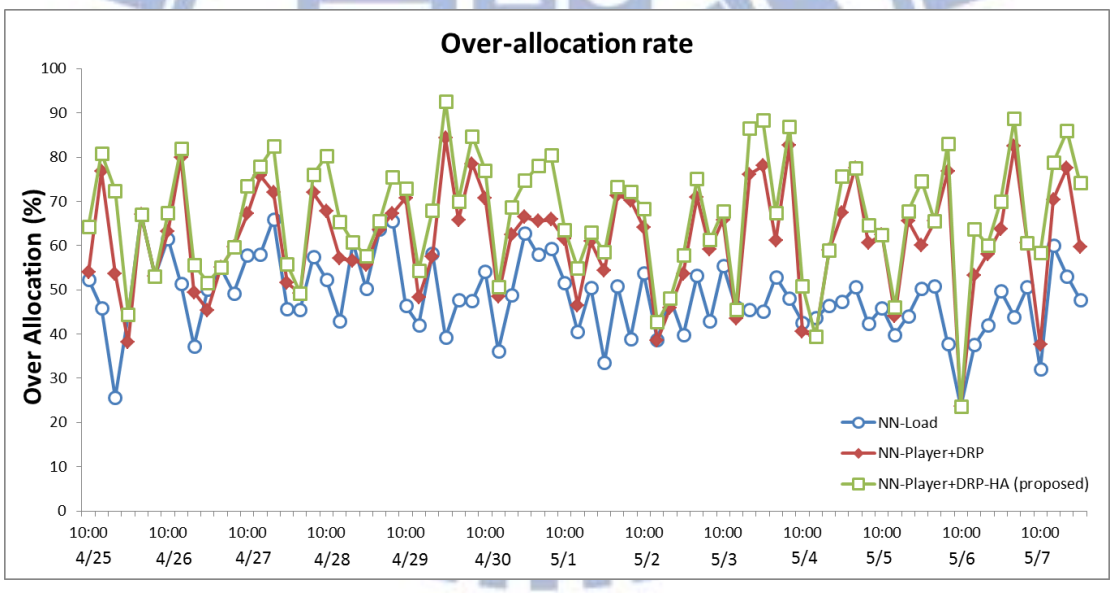


Figure 12: Over-allocation rate comparison for three different approaches.

Figure 11 illustrates the number of allocated VMs of three experiments, the points in this figure we plot is the maximum value of each two hours. It can be observed that our approach allocates the same number of VMs as DRP approach at most of time. Figure 12 is the over-allocation rate comparison for three different approaches. This figure shows that NN-Load approach is more resource efficient, but

the following two figures will show the defects of NN-Load approach.

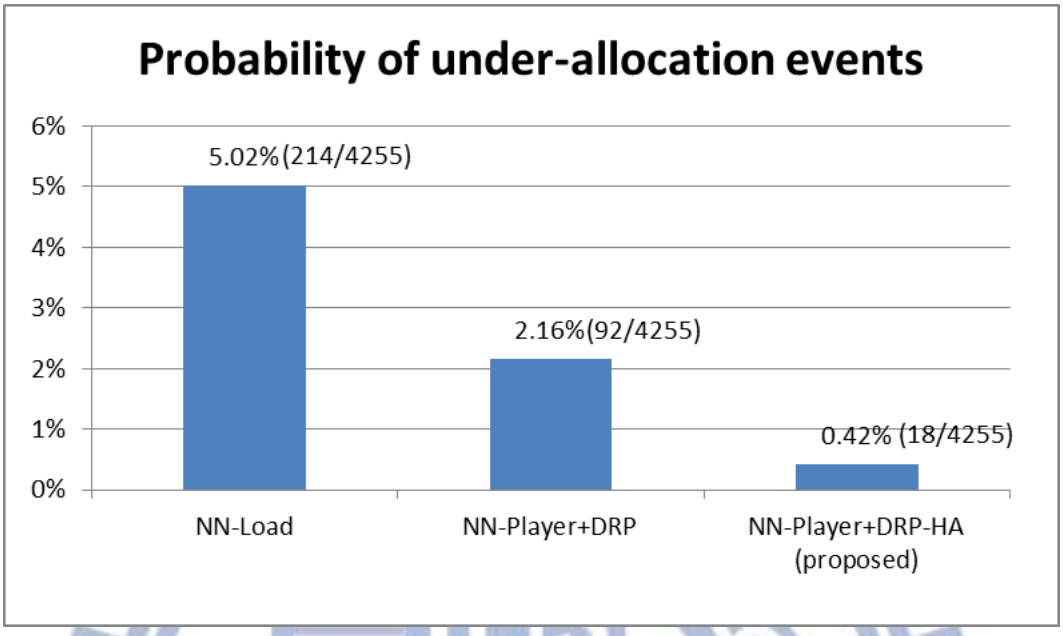


Figure 13: Probability of under-allocation for each experiment.

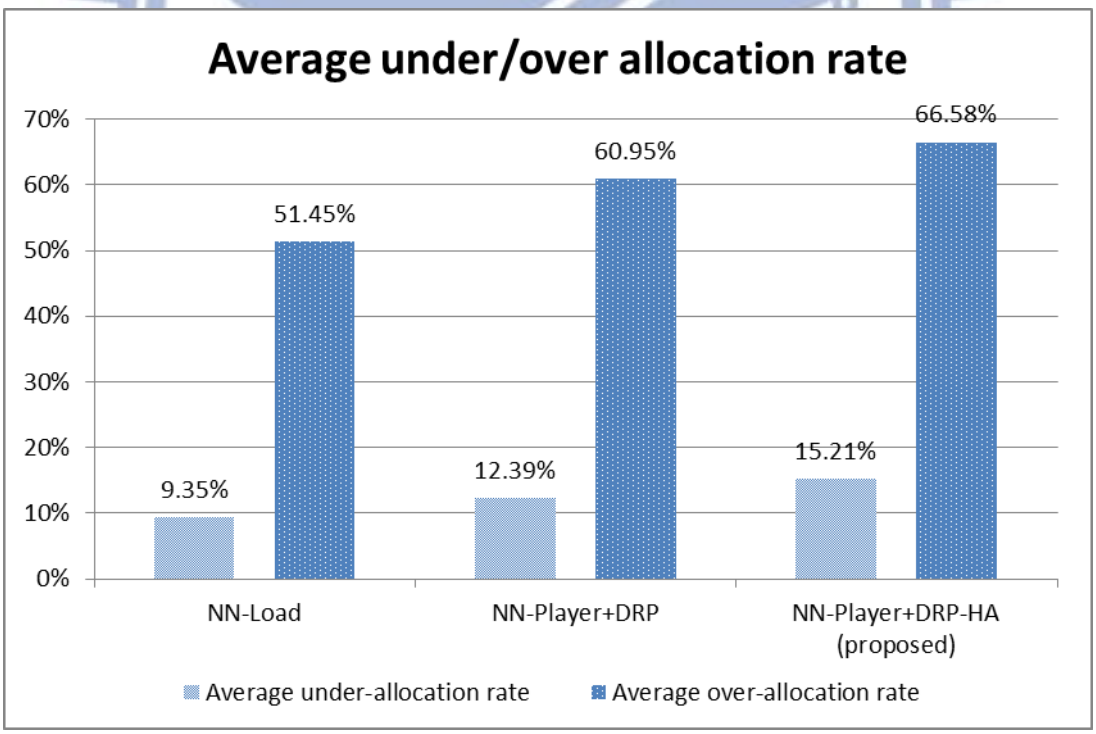


Figure 14: Average rate of under/over allocation.

Figure 13 and Figure 14 show that our proposed scheme decreases the

probability of under-allocation. The results of NN-Load experiment shows that the neural network achieve good accuracy in predicting load. However, it has higher probability to under-allocate resources than other two experimental setups. The result of NN-Player+DRP experiment shows that probability of under-allocation will decrease, since load model will overestimate the resource requirements. However, neural network predictor is based on trial-and-error design strategy, so the prediction results cannot fit the actual value perfectly. Thus, there're still 2.16% under-allocation event occurred. Finally, our proposed DRP-HA scheme can avoid most of under-allocation event, which only 18 events over 4255 records. The average over-allocation rate is 66.58%, which can be handled by single VM.

Chapter 5

Conclusion

5.1 Concluding Remarks

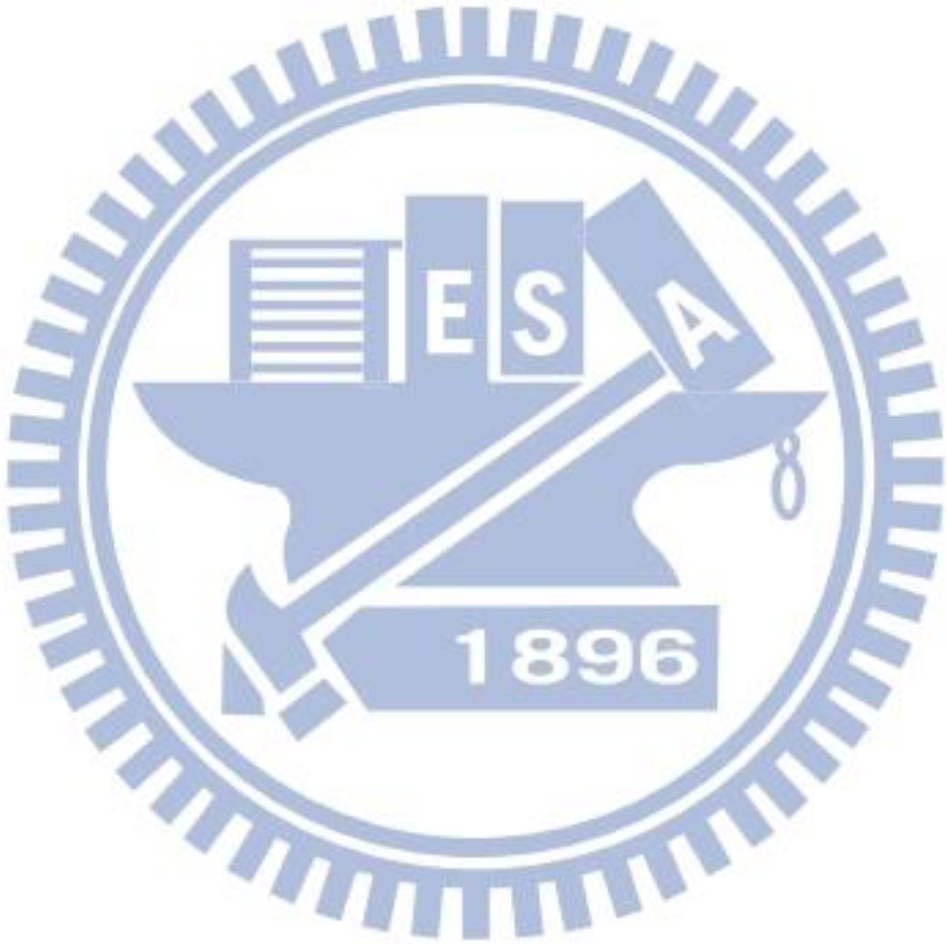
In this paper, we have presented a hotspot anticipation (HA) scheme to enhance dynamic resource provisioning (DRP), called DRP-HA for MMOGs in cloud computing environments. If avatars are aggregated into groups and become hotspots in a virtual world, interactions between avatars in hotspots will cause extra load to the zone server. Our proposed DRP-HA employs a finite state machine model to monitor the movement of avatars in a virtual world. By combining the state of each avatar in a game zone with a neural network (NN) predictor to forecast the number of players in the next time slot (called NN-player+DRP-HA), we may figure out the potential workload produced by hotspots, and then allocate appropriate computing resources to support the game zone.

Experimental results have supported that the proposed NN-Player+DRP-HA scheme can avoid most of under-allocation events with an acceptable over-allocation rate. Compared with a representative dynamic resource provisioning method, called NN-Player+DRP, the proposed NN-Player+DRP-HA reduces the probability of under-allocation events from 2.16% to 0.42% (80% improvement) in terms of CPU capacity, under the premise that the CPU over-allocation rate is within the capacity of one VM.

5.2 Future Work

In this paper, we focus on reducing the under-allocation events in terms of CPU capacity by considering the interaction of avatars. In the future, by taking different

behavior of avatars that result in different loads into account, we may further reduce under/over-allocation rates of the dynamic resource provisioning for MMOGs.



References

[1] Y. Lee; K. Chen; Y. Cheng; C. Lei, “World of Warcraft avatar history dataset,” in *Proc. the Second Annual ACM Conference on Multimedia Systems MMSys’11, San Jose, California, USA*, Feb. 2011, pp. 123-128.

[2] S. Wang; S. Dey, “Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach,” in *Proc. IEEE Global Communications Conference GLOBECOM’09*, Hilton Hawaiian Village, Honolulu, Hawaii, USA, Dec. 2009, pp. 1-7.

[3] “Boss (video gaming),” [Online.] Available: http://en.wikipedia.org/wiki/Boss_(video_gaming)

[4] P. Bod´ık, C. Sutton, A. Fox, D. Patterson and M. Jordan, “Response-Time Modeling for Resource Allocation and Energy-Informed SLAs,” in *Workshop on Statistical Learning Techniques for Solving Systems Problems (MLSys)*, Whistler, Canada, 2007.

[5] S. Legtchenko, S. Monnet and G. Thomas, "Blue Banana: resilience to avatar mobility in distributed MMOGs," in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, vol., no., June 28 2010-July 1 2010, pp.171-180.

[6] J.L. Miller and J. Crowcroft, "Avatar movement in World of Warcraft battlegrounds," in *Workshop on IEEE Network and Systems Support for Games (NetGames)*, vol., no., 23-24 Nov. 2009, pp.1,6.

[7] V. Nae, A. Iosup and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," *IEEE Transactions on Parallel and Distributed Systems*, pp. 380-395, Mar. 2011.

[8] D.T. Ahmed and S. Shirmohammadi, "A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs," *IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, vol., no., pp.27-34, 27-29 Oct. 2008.

[9] H. Liang, I. Tay, M. F. Neo, W. T. Ooi and M. Motani, "Avatar mobility in networked virtual environments: Measurements, analysis, and implications." CoRR, abs/0807.2328,2008.

[10] 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*, June 2012, pp. 423-430.

[11] 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)* , July 2010, pp. 435-442.