國 立 交 通 大 學

網路工程研究所

碩 士 論 文

建構於雲端環境之多人線上遊戲動態資源分配機制

Dynamic Resource Allocation for MMOGs in Cloud Computing Environments

研 究 生：翁振芳

指導教授：王國禎 博士

中 華 民 國 一 百 年 六 月

建構於雲端環境之多人線上遊戲 動態資源分配機制

Dynamic Resource Allocation for MMOGs in Cloud Computing Environments

研 究 生：翁振芳　　Student：Chen-Fang Weng

指導教授：王國禎　　Advisor：Kuochen Wang

國 立 交 通 大 學
資 訊 學 院
網 路 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

建構於雲端環境之多人線上遊戲
動態資源分配機制

學生：翁振芳　　　指導教授：王國禎 博士

國立交通大學網路工程研究所

摘 要

大型多人線上遊戲是指數十萬的玩家同時上網進行遊戲。而運行大型多人線上遊戲時所消耗的 CPU、記憶體以及網路頻寬等資源主要在客戶端玩家。我們將大型多人線上遊戲與雲端計算結合。在雲端計算環境下，我們利用虛擬伺服器來取代傳統實體伺服器。利用 multi-server 的遊戲架構，我們把虛擬遊戲世界切割成數個地圖區域，每個地圖區域由至少一個虛擬伺服器負責運行遊戲以及客戶端玩家間的訊息傳遞。我們根據每個虛擬伺服器的 CPU、記憶體以及網路頻寬，利用類神經網路以及適應性類神經模糊系統來預測及決定該虛擬伺服器執行何種資源分配機制。這些資源分配機制包括：(1)可支援周圍虛擬伺服器；(2)解除被周圍虛擬伺服器支援的狀態或釋放支

援本身的次要虛擬伺服器；(3)維持現有狀態；(4)須要請求周圍的虛擬伺服器支援；(5)在本身及周圍一個虛擬伺服器之間新增一個次要伺服器進行支援。根據我們的研究發現，適應性類神經模糊推論系統比起類神經網路有較低的均方根誤差，亦即有較好的學習效率。因此我們選擇適應性類神經模糊推論系統來實作上述五項資源分配機制。就資源分配機制而言，我們的方法比 deep-level partitioning 的存取時間快 16.7%。

關鍵詞：適應性類神經模糊推論系統、類神經網路、雲端計算、資源分配、負載預測。

Dynamic Resource Allocation for MMOGs in Cloud Computing Environments

Student：Chen-Fang Weng　　Advisor：Dr. Kuochen Wang

Department of Computer Science

National Chiao Tung University

Abstract

A massively multiplayer online game (MMOG) has hundreds of thousands of players who play in the game concurrently. The players consume a great deal of CPU, memory and network bandwidth resources in MMOGs. We combine MMOGs with cloud computing environments. We use virtual machine servers (VMSs) in cloud computing environments instead of traditional physical game servers. By using a multi-server architecture, we divide a game world into several zones, and each zone consists of at least a VMS to execute game processes and exchange game information among players in the zone. In addition, we design an artificial neural network (ANN) and also an adaptive neural fuzzy inference system (ANFIS) to predict the load of each zone and decide a resource allocation policy to be performed by the VMS. These policies include (1) this VMS is sufficient to support adjacent VMSs; (2) this VMS will release the resources which has been supported by adjacent VMSs or a secondary VMS which supports this VMS; (3) this VMS will remain in the current state; (4) this VMS requires adjacent VMSs to support it; (5) a secondary VMS will be created between this VMS and an adjacent VMS. Experimental results show that the mean square error of the ANFIS-based load prediction is lower than that of the ANN-based load prediction. Therefore, we incorporate the ANFIS prediction method along with the five resource allocation policies to the MMOG cloud. In terms of average access time, the proposed ANFIS-based resource allocation method is 16.7% better than the deep-level partitioning (DLP) method.

Keywords: ANFIS, ANN, cloud computing, load prediction, 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 Grant NSC99-2221-E-009-081-MY3 is also gratefully acknowledged. Finally, I thank my family for their endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Resource allocation discusses how to make available resources be allocated more efficiently. Resource allocation includes two topics [5]: (1) basic allocation decision — it determines which resources can be used by which objects; (2) contingency mechanism — it chooses which tasks will be sacrificed when the system is overloaded, and it chooses which tasks will enter when the system is idle. There are two methods for resource allocation [5]: (1) real-time resource allocation — it can allocate resources dynamically when needed; (2) pre-allocation — it predicates the loads by historical data then assigns the available resources. In addition, there are two types of resource allocation [5]: (1) centralized approach — all the available resources are controlled by one central device. It is easier to control and deploy all the resources in a system, but the central device might be the bottleneck in the system; (2) distributed approach — all the devices can make some decisions for resources by themselves, but it is hard to implement.

A massively multiplayer online game (MMOG) is a multiplayer video game which is capable of supporting hundreds of thousands of players simultaneously. There are two kinds of actions for each player: (1) independent action — players only concern about the items, equipments or status in the game; (2) interaction — players interact to the game server or communicate with other players [3]. Most MMOGs are MMORPG (massively multiplayer online role, such as Lineage [21] and WOW [22]. There are three architectures for MMOGs: (1) Client-server architecture — a game server governs all the players in this game. However, the client-server architecture has poor scalability. (2) Peer-to-peer architecture — a player is regarded as a peer who shares their game information by itself. However, peer-to-peer architecture is hard to implement for the security problem. (3) Multi-server architecture — it enhances the client-server architecture. The game world is divided into several zones and each zone is governed by a server. However, load unbalancing might still occur in game servers when some zones with crowed players exist in the multi-server architecture. Therefore, we propose a cloud-based dynamic resource allocation method to resolve the multi-server load unbalancing problem by flexible allocating of resources.

There are four resource components in MMOGs [4]: (1) authentication component, (2) storage component, (3) computation component, and (4) communication component. Since communication and computation components consume most of resources, we only consider these two components in this paper.

The rest of this paper is organized as follows. In Chapter 2, we discuss an MMOG load balancing mechanism, deep-level partitioning (DLP) and its multiple hotspots problem [19]. In Chapter 3, we design an artificial neural network (ANN) and also an adaptive neural fuzzy inference system (ANFIS) to predict MMOG load and decide a resource allocation policy which can be performed by a VMS. In Chapter 4, we describe a simulation environment and discuss simulation results. Finally, concluding remarks and future work are given in Chapter 5.

Chapter 2

Related Work

2.1 Existing load balancing methods for MMOGs

In zonal MMOGs, a virtual game world is divided into several zones (or microcells). Each zone consists of a game server to execute game processes and to exchange game information among players in the zone. Ahmed et al. [24] proposed a server pool concept in the virtual game world. A server pool contains several game servers. Each game server serves a zone. The server pool operates load distribution of the game servers which are in it. Moreover, the authors proposed a buffer region between two zones. The buffer region can share game messages between two adjacent zones which are in different server pools. In this approach, since it will transfer zones between game servers, the cost of inter-server communication may increase.

Wang et al. [25] proposed a method for MMOGs to find less loaded game servers. They defined a threshold *W*. If the server loading *S* is larger than *W*, the game server is regarded as overloading. They used two kinds of lists in their approach, *select list* and *candidate list*. In the first list, they select an overloading game server *A* in the select list, and they chose the servers which are adjacent to A in the candidate list. Then if there is an overloaded game server B in the candidate list, B will be inserted in the select list. And the game servers which are adjacent to B will be inserted in the candidate list. If there are less loaded game servers in the select list, the least loaded game server C will be inserted in the select list. And the game servers which are adjacent to C will be inserted in the candidate list. In this method, the more the number of game zones is, the higher the complexity of the algorithm is.

Carlos Eduardo et al. [26] proposed to use a KD-tree to divide a virtual game world into several zones. Each node of the KD-tree represents a game zone. In this approach, a game zone *A* (we mark node *A* in the KD-tree) contains two subzones *B* and *C* (we mark nodes *B* and *C* in the KD-tree). And nodes *B* and *C* are the children of node *A* in the KD-tree. Each node contains two kinds of values. One value is the load of its children, and the other is the capacity of its children. When a game server is overloaded, it will readjust the load of the server using the KD-tree method. The limitation of this approach is that the game zones should keep a rectangle shape.

2.2 A dynamic resource prediction method for MMOGs

Vlad Nae et al [28] proposed a load model for MMOGs. The load model includes CPU load model, memory load model and network load model. In addition, they used the neural network, average, moving average, last value and exponential smoothing to predict the load of CPU, memory and network based on real game traces from RuneScape. They showed that the prediction error of the neural network based method is lower than that of the other prediction methods. And the neural network based method is faster than the other methods. They also found that the dynamic resource provisioning is more efficient than static resource provisioning.

2.3 Multiple hotspots problem for MMOGs

In the deep-level partitioning (DLP) method, as show in Figure 1, designed for zonal MMOGs, there is a load threshold T_m, and it defines S_i as the load of game server *i*. When $S_i \geqq T_m$, game server *j*, which is an adjacent game server *i* and $S_j < T_m$, will support game server *i*. In this way, the overloading problem can be resolved.

However, the DLP method becomes inefficient when there are some contiguous zones with crowded players (multiple hotspots problem), as shown in Figure 2.

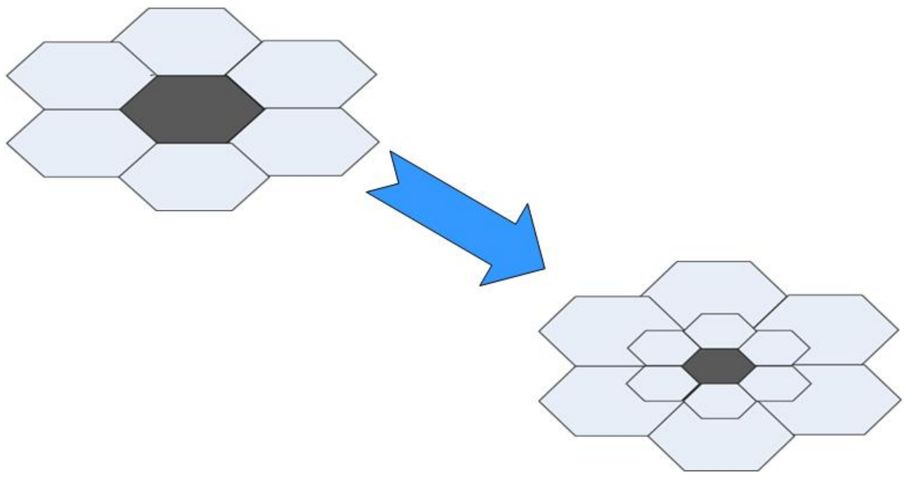


Figure 1: Deep-level partitioning for MMOGs.

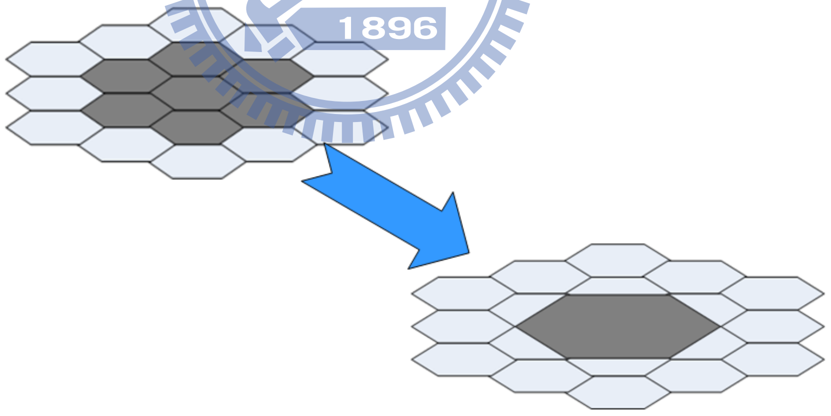


Figure 2: Multiple hotspots problem.

Chapter 3

Dynamic Resource Allocation for MMOG Clouds

3.1 Cloud computing environments

Most MMOGs mainly employ a multi-server architecture. In the multi-server architecture, the game world is divided into several zones, and each zone consists of a game server. In our design, we combine MMOGs with cloud computing environments. We use virtual machine servers (VMSs) instead of physical game servers. There are two advantages for using VMSs: (1) We can easily allocate appropriate VMSs resources from physical game servers; (2) It is easy to migrate players from a VMS to other VMSs. Figure 3 show our MMOG architecture in a cloud computing environment.

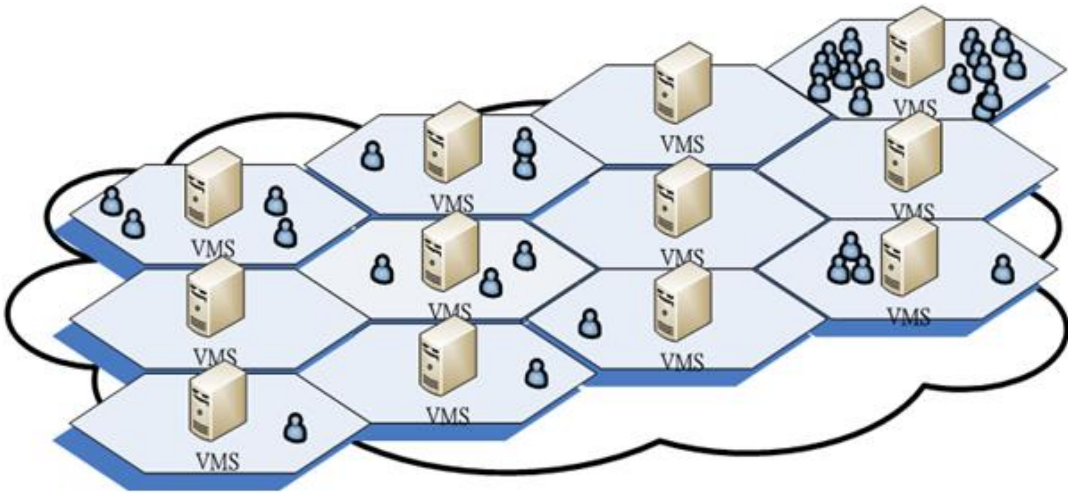


Figure 3: MMOG architecture in a cloud computing environment.

3.2 Defining resource allocation policies

We divide a zone, which is served by a VMS, into two areas, area *A* and area *B*, as shown in Figure 4. Area *A* is in the central part of the zone, which occupies 40% of a zone. Players in area *A* is always handled by this VMS. Players in area *B* can be supported by adjacent VMSs (deep-level partitioning, DLP [19]) or the proposed secondary VMSs (SVMSs) when this VMS is overloaded.

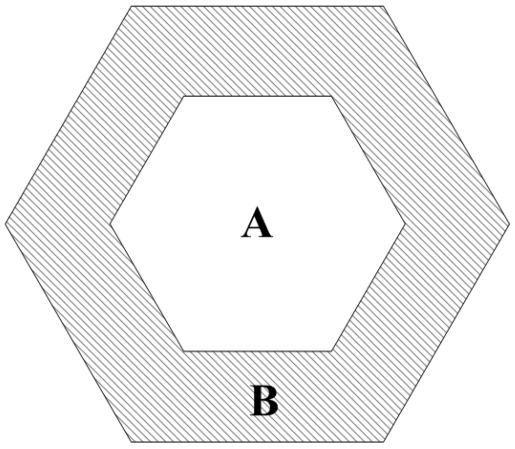


Figure 4: Two areas in a game zone.

For players in area *B*, they have higher probabilities to move to adjacent zones.

Therefore, we will create a SVMS between two adjacent overloaded VMSs to resolve the multiple hotspots problem, as shown in Figure 5. In addition, using the SVMS method can avoid the high cost of supporting the entire game zone by using an extra VMS.

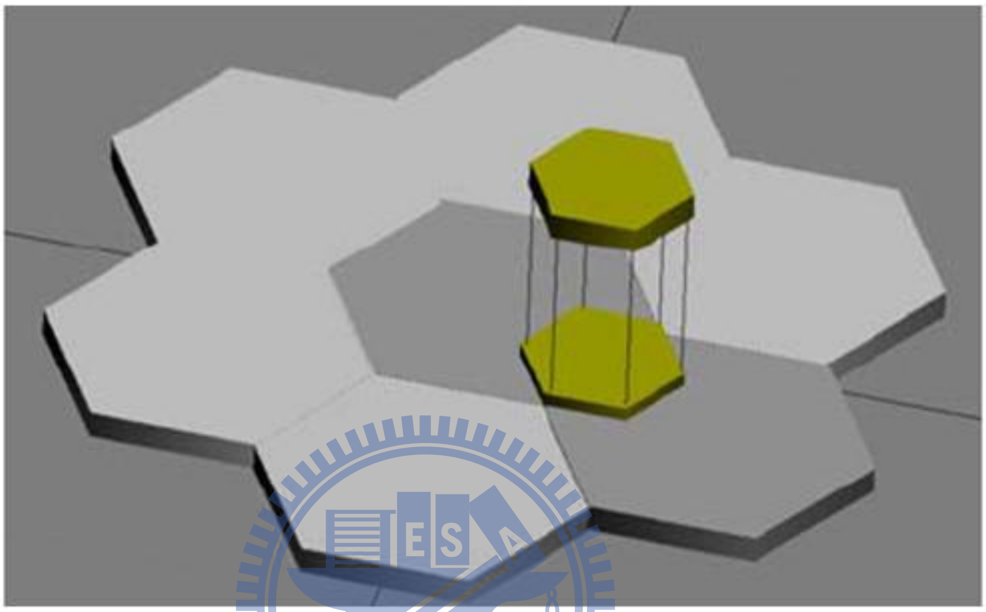


Figure 5: Creating an SVMS between two adjacent VMSs.

We define five resource allocation policies based on the load level of each VMS, as shown in Table 1:

1. *This VMS can support adjacent VMSs*. When the load of this VMS is light, this VMS will inform adjacent VMS that it has redundant resource to support those adjacent VMSs with high or heavy load.
2. *This VMS releases the area which is supported by adjacent VMSs or SVMSs.* If there are some areas of this VMS which has high load, have been supported by adjacent VMSs or the SVMSs, this VMS will take back these areas and support them by itself. If it is a SVMS, the SVMS will be released

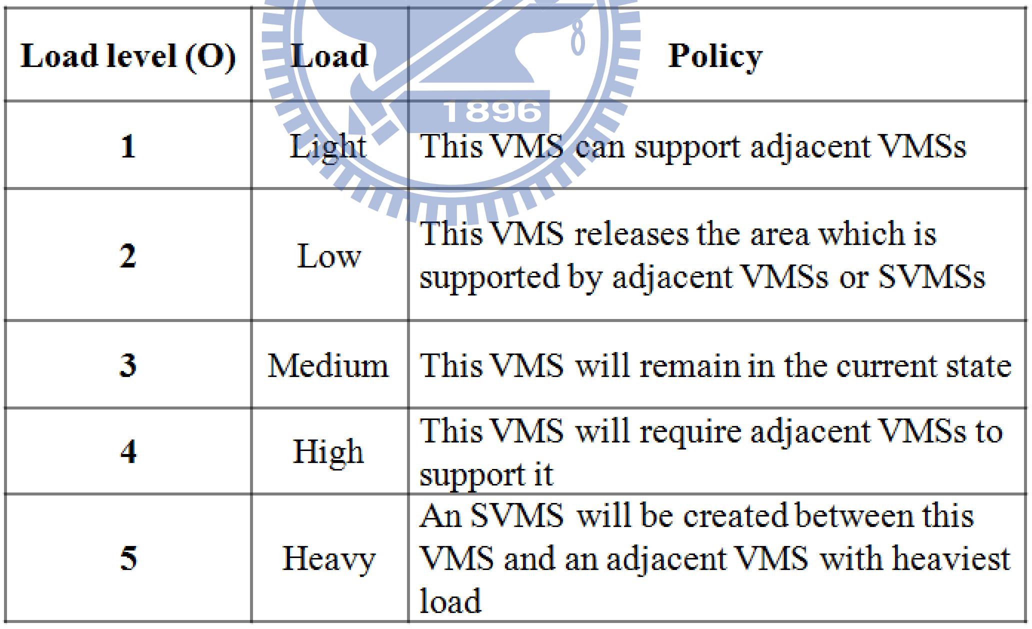
when the SVMS does not support any area between this VMS and the adjacent VMSs.

3. *This VMS will remain in the current state.* This VMS will not change its resource allocation policy even if this VMS has supported adjacent VMSs, or has been supported by adjacent VMSs or SVMS.

4. *This VMS will require adjacent VMSs to support it.* Since this VMS becomes overloaded, it informs adjacent VMSs that it needs to be supported. And an adjacent VMS with lightest load can support this VMS.

5. *An SVMS will be created between this VMS and an adjacent VMS with heaviest load.*

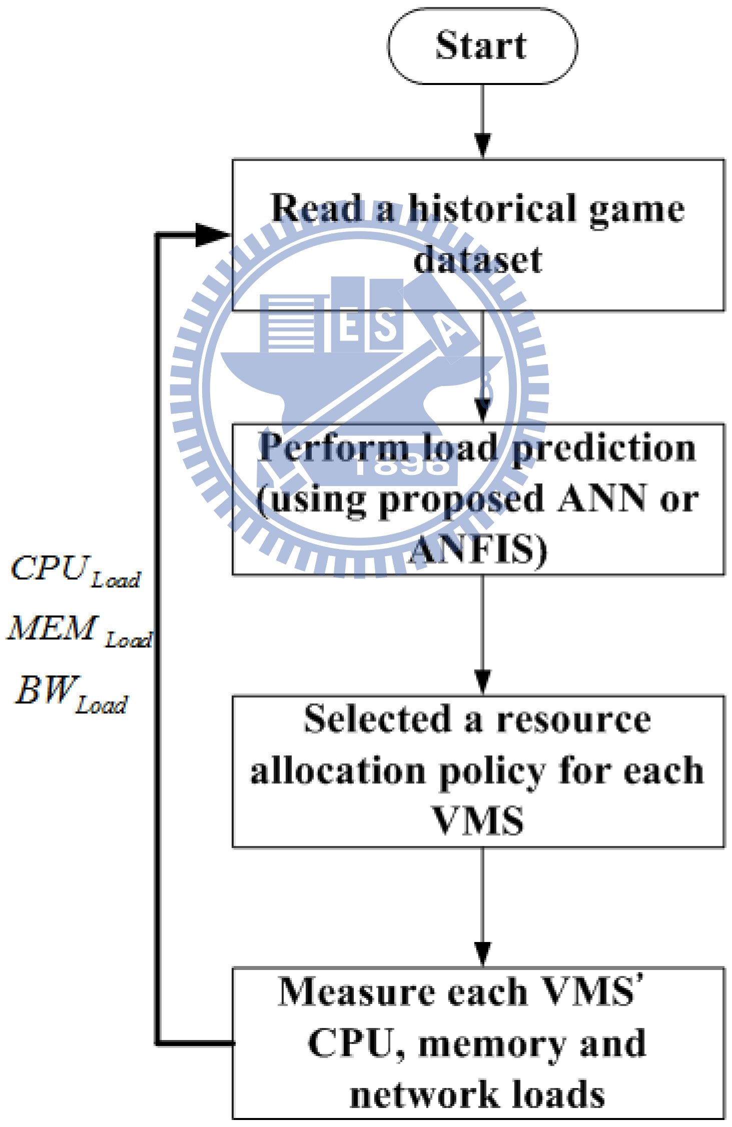
Table 1: VMS based resource allocation policies.



Load level (O)	**Load**	**Policy**
1	Light	This VMS can support adjacent VMSs
2	Low	This VMS releases the area which is supported by adjacent VMSs or SVMSs
3	Medium	This VMS will remain in the current state
4	High	This VMS will require adjacent VMSs to support it
5	Heavy	An SVMS will be created between this VMS and an adjacent VMS with heaviest load

3.3 **Proposed dynamic resource allocation scheme**

In this section, the proposed dynamic resource allocation scheme for MMOGs in cloud computing environment is described in Figure 5. First, we collect MMOG game traffic which includes CPU, memory and network bandwidth load from Lineage [21] for each VMS and obtain a historical game dataset. Then we analyze the historical game dataset to predict the load level of each VMS using the proposed artificial neural network (ANN) or the proposed adaptive neural fuzzy inference system (ANFIS). According to the predicted load level, the VMS can execute a selected resource allocation policy. Finally, we measure the VMSs' CPU, memory and network loads and then add these data to the historical game dataset.


Start
Read a historical game dataset
Perform load prediction (using proposed ANN or ANFIS)
Selected a resource allocation policy for each VMS
Measure each VMS' CPU, memory and network loads
CPU_{Load}
MEM_{Load}
BW_{Load}

3.4 Proposed artificial neural network based load prediction

We define the loads for CPU, memory and network, as follows:

$$CPU_{Load} = \frac{CPU\_usage}{CPU_{VMS}}$$

$$MEM_{Load} = \frac{MEM\_usage}{MEM_{VMS}}$$

$$BW_{Load} = \frac{BW\_usage}{BW_{VMS}}$$

There are three layers, as shown in Figure 7, *input layer*, *hidden layer* and *output layer* in an artificial neural network (ANN), the input layer is composed of CPU_{load}, MEM_{load} and BW_{load}, which are from the historical game dataset (we use x_1, x_2 and x_3 to represent CPU_{load}, MEM_{load} and BW_{load}). The hidden layer contains ten neurons and the output layer contains one neuron.

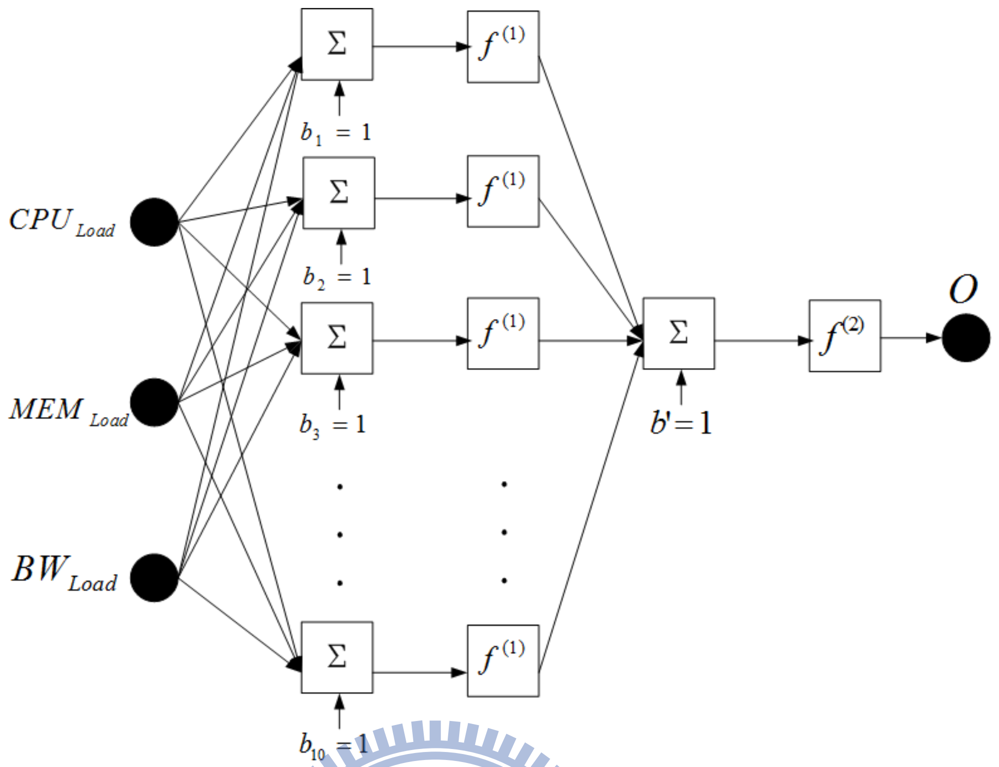


Figure 7: MMOG load prediction using ANN.

Each neuron in the hidden layer must perform the sum up of the weighted inputs (x_1, x_2 and x_3) and compute them by a log-sigmoid function ($f^{(1)}$). Figure 8 shows the first neuron in the hidden layer.

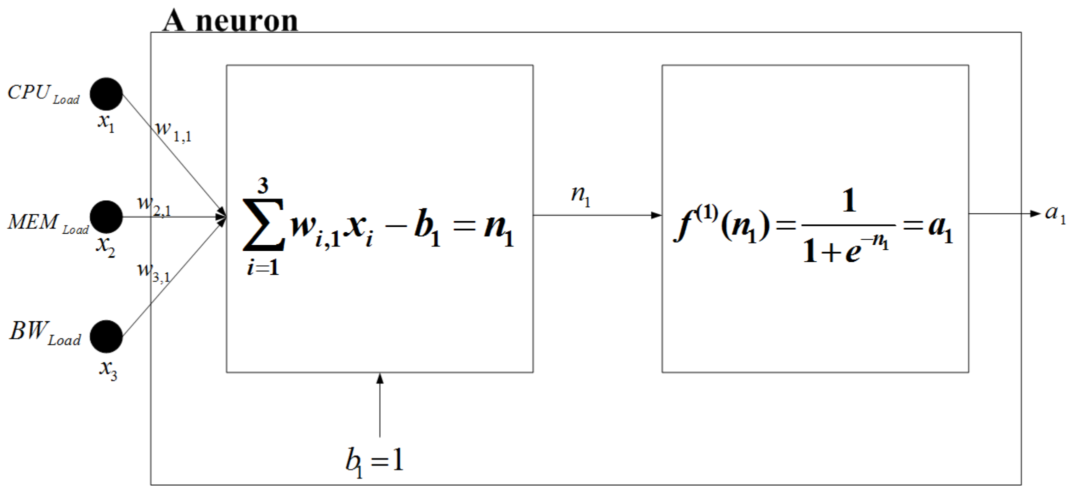


Figure 8: The first neuron in the hidden layer.

For each neuron j ($1 \leqq j \leqq 10$) in the hidden layer, we have

$$\sum_{i=1}^{3} w_{i,j} x_i - b_j = n_j \tag{1}$$

$$f^{(1)}(n_j) = \frac{1}{1+e^{-n_j}} = a_j \tag{2}$$

A neuron of the output layer sums up the weighted a_j, for $j = 1, \ldots, 10$, which are generated by the neurons in the hidden layer and compute them by a linear function ($f^{(2)}$). By this procedure, we can get an output value O which is the load level, as shown in Table 1. Figure 9 shows a neuron in the output layer.

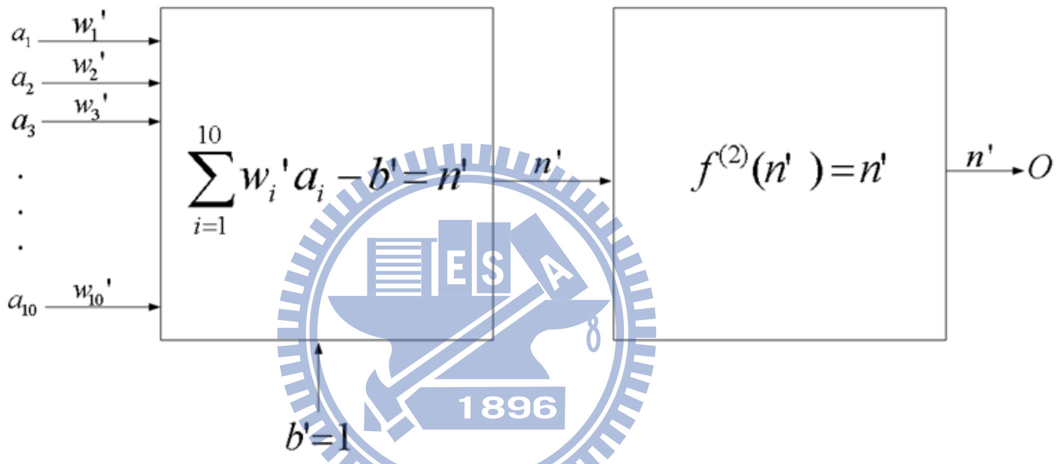


Figure 9: A neuron in the output layer.

In the learning process for an ANN, we define W as a set of all weights, that is, $W = \{w_{i,j} \mid 1 \leqq i \leqq 3, 1 \leqq j \leqq 10\} \cup \{w_k' \mid 1 \leqq k \leqq 10\}$ at first. Then we define B as a set of biases for an ANN, that is, $B = \{b_i \mid 1 \leqq i \leqq 10\} \cup \{b'\}$. We use partial differential mean square error (MSE) equations for weights and biases to adjust weights and biases in the ANN.

$$MSE : E = (d - O)^2 \tag{3}$$

where d is an expected value. $\forall \omega \in W$, ω has a training rate $\Delta\omega$. $\forall \beta \in B$, β has a training rate $\Delta\beta$. We have:

$$\Delta\omega = \frac{\partial E}{\partial \omega} \tag{4}$$

$$\omega_{(t+1)} = \omega_{(t)} - \Delta\omega \tag{5}$$

$$\Delta\beta = \frac{\partial E}{\partial \beta} \tag{6}$$

$$\beta_{(t+1)} = \beta_{(t)} - \Delta\beta \tag{7}$$

where t is an epoch in the ANN [27].

3.5 **Adaptive neural fuzzy inference system based load prediction**

An adaptive neural fuzzy inference system (ANFIS) combines a neural network with fuzzy inference. The ANFIS contains five layers which include *inputs*, *input membership functions*, *fuzzy rules*, *output membership functions* and *defuzzification output*. Figure 10 shows the proposed ANFIS-based MMOG load prediction architecture.

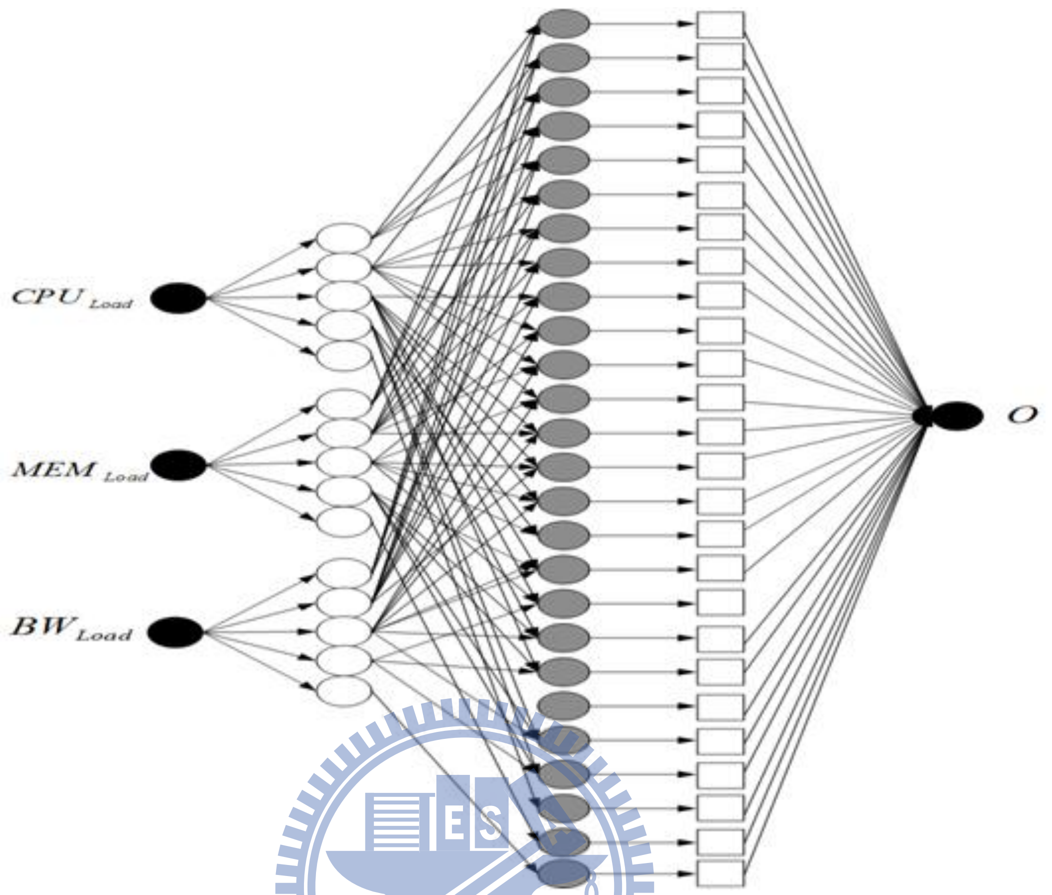
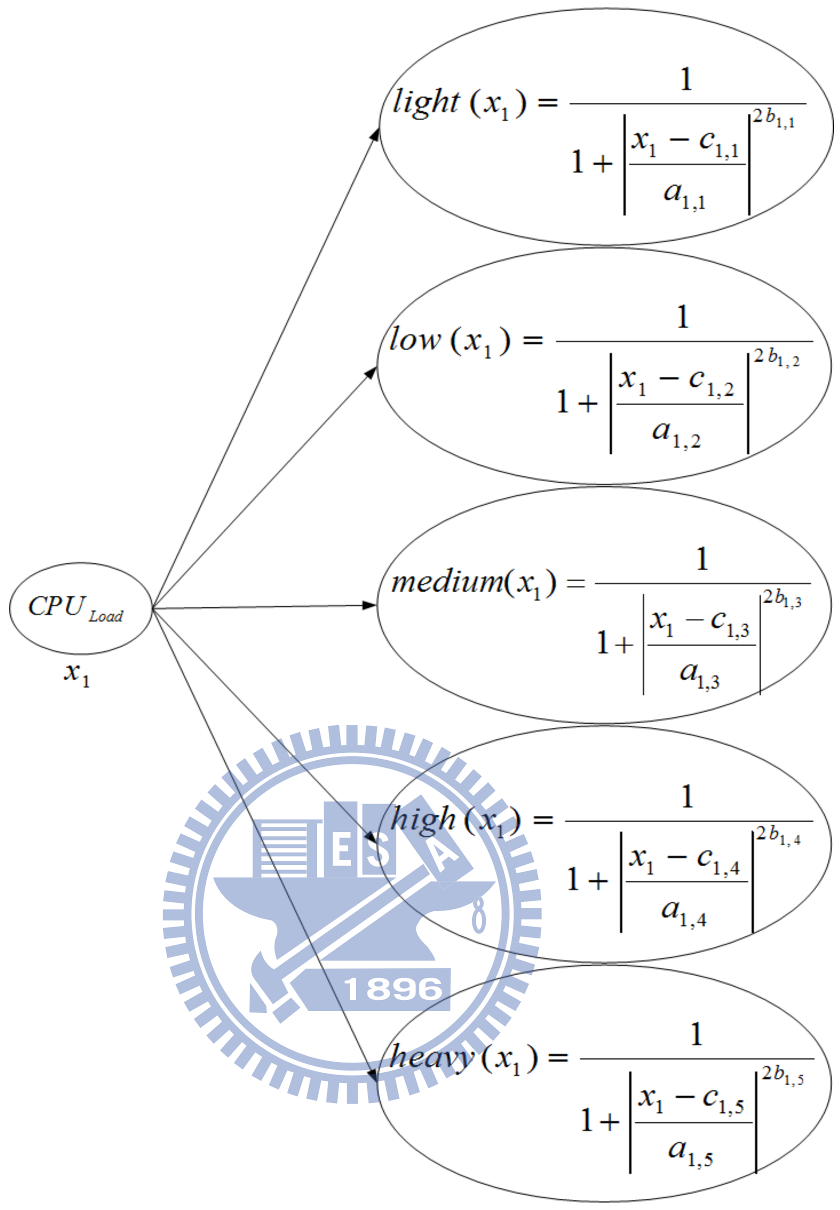


Figure 10: Proposed ANFIS-based MMOG load prediction architecture.

The inputs for ANFIS contain CPU_{load}, MEM_{load} and BW_{load} which are defined in section 3.4. We use x_1, x_2 and x_3 to represent CPU_{load}, MEM_{load} and BW_{load}. For each input, we define five generated bell-shaped input membership functions which are based on an MMOG. We illustrate the five input membership functions of CPU_{load} in ANFIS, as shown in Figure 11.



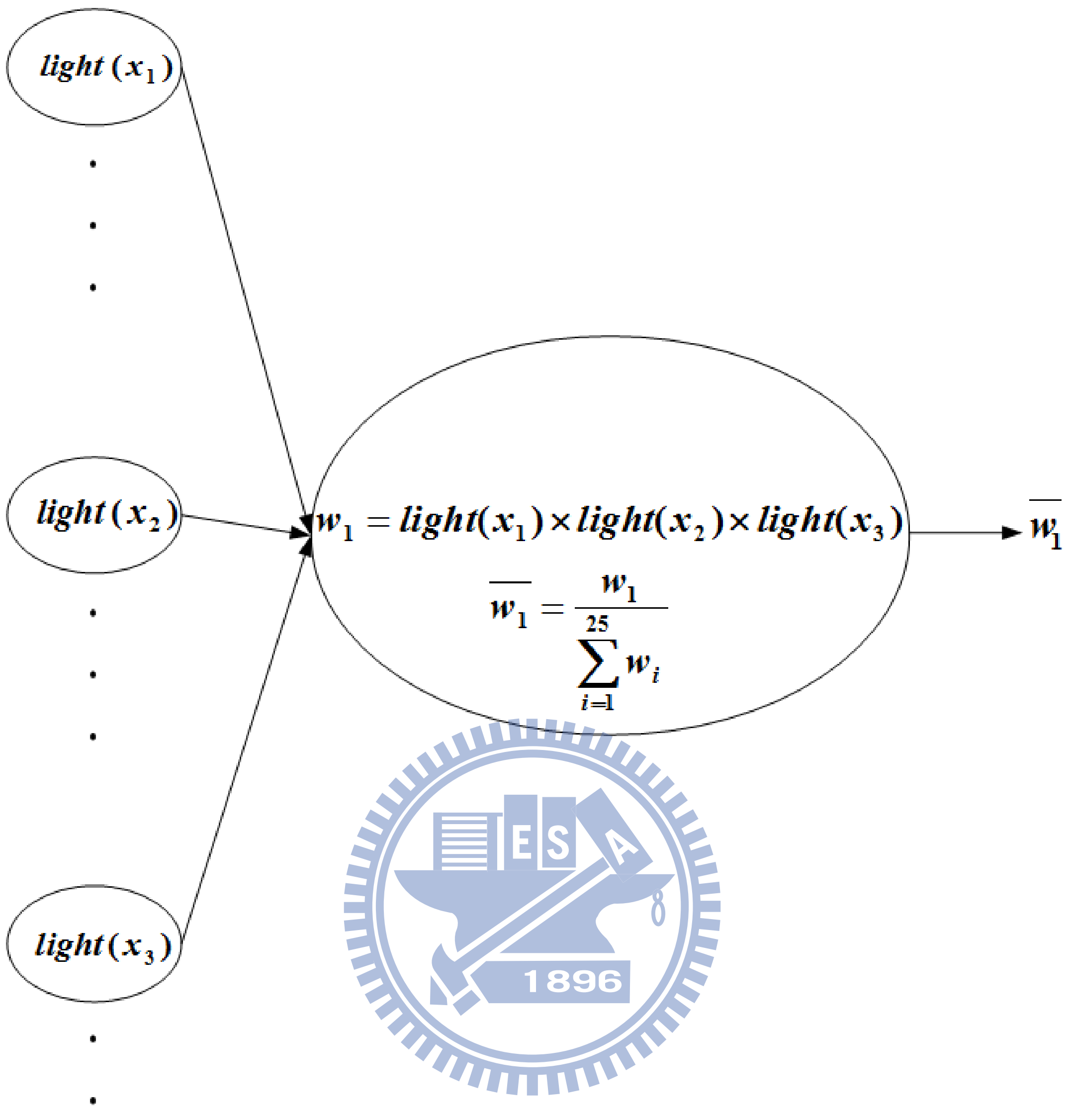
CPU_{Load}
x_1
$light(x_1) = \frac{1}{1+\left|\frac{x_1 - c_{1,1}}{a_{1,1}}\right|^{2b_{1,1}}}$
$low(x_1) = \frac{1}{1+\left|\frac{x_1 - c_{1,2}}{a_{1,2}}\right|^{2b_{1,2}}}$
$medium(x_1) = \frac{1}{1+\left|\frac{x_1 - c_{1,3}}{a_{1,3}}\right|^{2b_{1,3}}}$
$high(x_1) = \frac{1}{1+\left|\frac{x_1 - c_{1,4}}{a_{1,4}}\right|^{2b_{1,4}}}$
$heavy(x_1) = \frac{1}{1+\left|\frac{x_1 - c_{1,5}}{a_{1,5}}\right|^{2b_{1,5}}}$

Figure 11: Input membership functions of CPU_{Load} in ANFIS.

Table 2: The premise parameters of input membership functions of CPU_{Load} (x_1).

Load level	j	$a_{1,j}$	$b_{1,j}$	$c_{1,j}$
light(x_1)	1	0.125	2	0
low(x_1)	2	0.125	2	0.25
medium(x_1)	3	0.125	2	0.5
high(x_1)	4	0.125	2	0.75
heavy(x_1)	5	0.125	2	1

We define $S_1 = \{a_{i,j}, b_{i,j}, c_{i,j} \mid 1 \leqq i \leqq 3, 1 \leqq j \leqq 5\}$ as a set of premise parameters of j^{th} input membership function of x_i in ANFIS. For illustration, the premise parameters of input membership functions of CPU_{Load} (x_1) is shown in Table 2. Premise parameters will be adjusted by the training process of ANFIS. Next, we set up twenty five fuzzy rules based on MMOG data, as shown in Table 3. Fuzzy rules are in if-then forms. The conditional statements are expressions of the input membership functions. And we choose the product to be the AND method. Figure 12 shows an example that "*if x_1 is light and x_2 is light and x_3 is light.*" And we normalize the products from all the fuzzy rules.



$light(x_1)$
$light(x_2)$
$light(x_3)$
$w_1 = light(x_1) \times light(x_2) \times light(x_3)$
$\overline{w_1} = \frac{w_1}{\sum_{i=1}^{25} w_i}$
$\overline{w_1}$

Table 3: The 25 fuzzy rules.

Rule	If			Then
	CPU_{Load}	MEM_{load}	BW_{load}.	Output membership function
1	*light*	*light*	*light*	f_1
2	*light*	*light*	*low*	f_2
3	*light*	*low*	*light*	f_3
4	*low*	*light*	*light*	f_4
5	*low*	*low*	*light*	f_5
6	*low*	*light*	*low*	f_6
7	*light*	*low*	*low*	f_7
8	*low*	*low*	*low*	f_8
9	*low*	*low*	*medium*	f_9
10	*low*	*medium*	*low*	f_{10}
11	*medium*	*low*	*low*	f_{11}
12	*medium*	*medium*	*low*	f_{12}
13	*low*	*medium*	*medium*	f_{13}
14	*medium*	*low*	*medium*	f_{14}
15	*medium*	*medium*	*medium*	f_{15}
16	*medium*	*medium*	*high*	f_{16}
17	*high*	*medium*	*medium*	f_{17}
18	*medium*	*high*	*medium*	f_{18}
19	*medium*	*high*	*high*	f_{19}
20	*high*	*medium*	*high*	f_{20}
21	*high*	*high*	*medium*	f_{21}
22	*high*	*high*	*high*	f_{22}
23	*heavy*	-	-	f_{23}
24	-	*heavy*	-	f_{24}
25	-	-	*heavy*	f_{25}

Each rule corresponds to an output membership function. The output membership functions are linear combination functions with inputs x_1, x_2 and x_3. An output membership function for a rule *k* is as follow:

$$f_k(x_1, x_2, x_3) = p_k x_1 + q_k x_2 + r_k x_3 + s_k \qquad (8)$$

where $1 \leqq k \leqq 25$ since we set up twenty five fuzzy rules. We define $S_2 = \{p_k, q_k, r_k, s_k \mid 1 \leqq k \leqq 25\}$ as a set of consequent parameters. The consequent parameters will be adjusted by the training process of ANFIS.

Finally, the output value O in ANFIS is the load level of each VMS. O is the summation of all the output membership functions f_k, k = 1 to 25, with normalized weight $\overline{w_k}$, which is generated by the k^{th} fuzzy rule:

$$O = \sum_{k=1}^{25} \overline{w_k} f_k \qquad (9)$$

There are two steps in the learning process of ANFIS. In the first step, we use a least square estimator (LSE) to adjust the consequent parameters in S_2. X is a vector which is composed of consequent parameters.

$$X = [p_1, q_1, r_1, s_1, p_2, q_2, r_2, s_2, \ldots, p_{25}, q_{25}, r_{25}, s_{25}]^T \qquad (10)$$

B is a vector which is composed of the expected value of each epoch.

$$B = [d^{(1)}, d^{(2)}, \ldots d^{(t)}]^T \qquad (11)$$

where $d^{(1)}$, $d^{(2)}$, … , and $d^{(t)}$ mean that there are t epochs and $d^{(n)}$ is the expected value of n epoch. We expand equations (8) and (9) and express them as a matrix equation as follows:

$$AX = B \qquad (12)$$

where A is a coefficient matrix. And we use the LSE to adjust the consequent parameters.

$$X = (A^T A)^{-1} AB \qquad (13)$$

In the second step, we use partial differential input membership functions for premise parameters, such as the learning process in the ANN [31].

Chapter 4

Simulation Results

4.1 Simulation setup

We collected game traffic which includes CPU, memory and network loads from Lineage. We used MATLAB as our prediction tool. Both of ANN and ANFIS are APIs (application programming interfaces) in MATLAB [29] to predict the load level of game traffic. We used *CloudSim* [30] as our simulation tool. We used 16 VMSs to deploy in the game world. The number of players in each VMS is 40 ~ 100. The details of the simulation setup are shown in Table 2.

Table 4: Simulation setup.

Game data	**from Lineage [21]**
Prediction tool	**MATLAB API (*nntool*)**
Simulation tool	**CloudSim**
CPU speed of a VMS	**10000 MIPS**
Memory space of a VMS	**4 GB**
Network bandwidth of a VMS	**20 MB/s**
Number of VMSs	**16**
Number of players	**40 ~ 100 players in each VMS**

4.2 Game data collection

We installed Lineage in our game server. The CPU utilization, memory consumption and network bandwidth usages will be recorded at every minute in our gaming experiment. Figure 13 shows the collected game data from Lineage [21]. CPU and network bandwidth usages are proportional to players in Lineage [21]. Also, players' actions affect CPU utilization and network bandwidth usage. The memory usage increased slowly as the players increased.

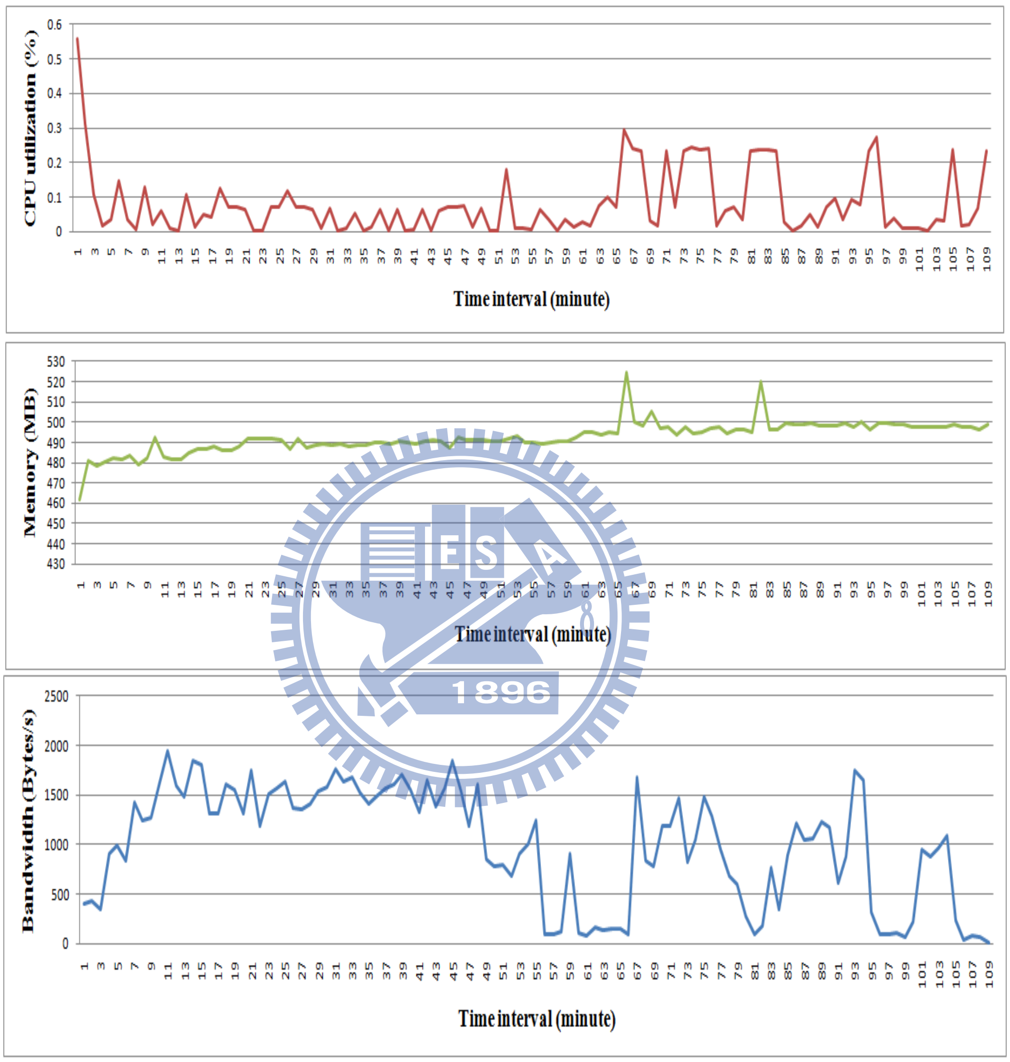


Figure 13: Game data from Lineage.

4.3 **Comparison of load prediction methods**

We compared the mean square errors for the ANN-based load prediction method and the ANFIS-based load prediction method. Since the ANFIS-based load prediction method has fuzzy rules which were based on the game features from Lineage [21], the mean square error is lower than that of the ANN-based load prediction method, as shown in Figure 14. Moreover, the prediction time of the ANFIS load prediction method is much smaller than that of the ANN-based load prediction method, as shown in Figure 15.

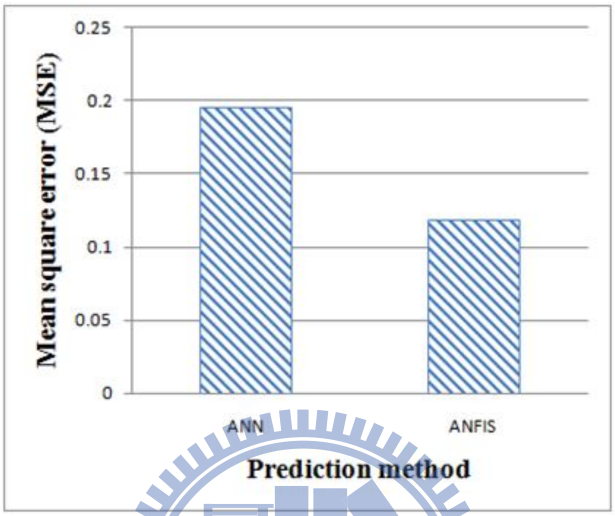


Figure 14: Mean square errors between two load prediction methods.

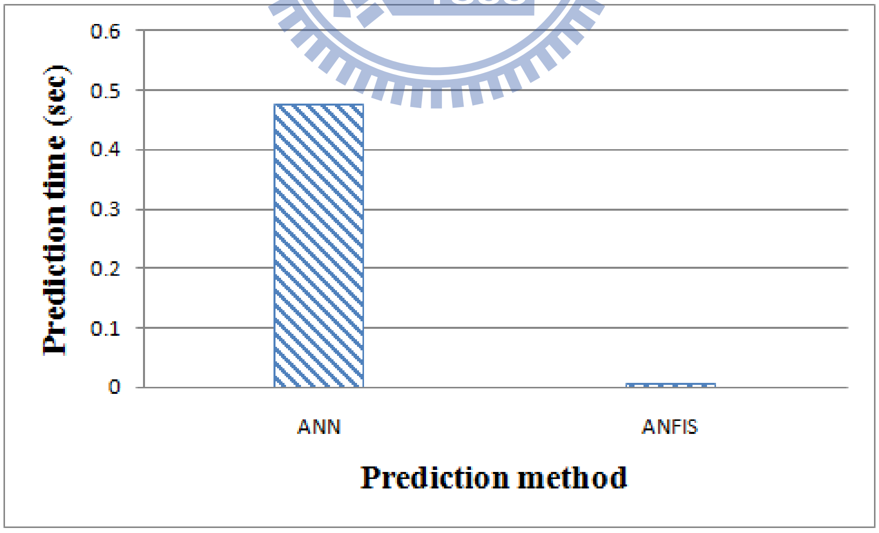


Figure 15: Prediction time between two load prediction methods.

4.4 Comparison of three resource allocation methods

We implemented multi-server, ANFIS-based DLP and DLP+SVMS with ANFIS methods. Experimental results show that the average access time (queuing time + CPU time) of the proposed ANFIS-based DLP+SVMS resource allocation method is 16.7% shorter than that of the ANFIS-based DLP method, as shown in Figure 16. In Figure 17, we show the VMS usages of the three resource allocation methods. The proposed ANFIS-based DLP+SVMS method has the smallest number of VMSs used among the three methods.

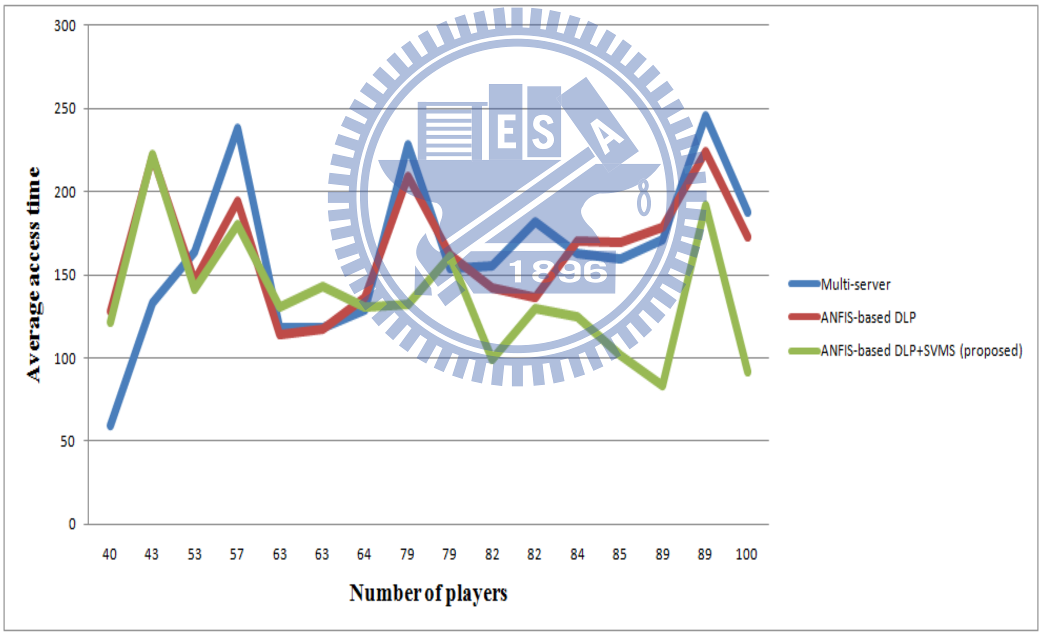


Figure 16: Average access time among three resource allocation methods.

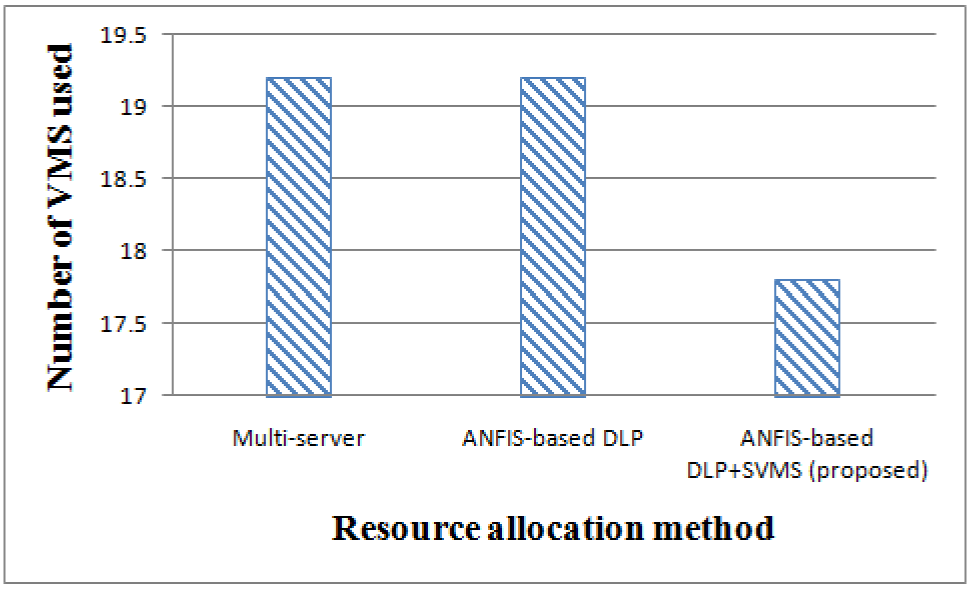


Figure 17: Number of VMSs used among the three resource allocation methods.

Chapter 5

Conclusion

5.1 Concluding remarks

There are two phases in the proposed dynamic resource allocation method: load prediction phase and resource allocation phase. In the load prediction phase, we collected historical game data which includes CPU, memory and network loads from a popular MMOG, Lineage. We have designed and simulated an artificial neural network (ANN) and an adaptive neural fuzzy inference system (ANFIS) to predict an appropriate resource allocation policy to be executed in each game zone. Experimental results show that in the load prediction phase, the mean square error and prediction time of the ANFIS-based load prediction scheme are lower than those of the ANN-based load prediction scheme. In the resource allocation phase, the average access time (execution time plus queuing time) of the proposed ANFIS-based deep-level partitioning (DLP) with secondary virtual machine servers (SVMSs) method is 16.7% shorter than that of the ANFIS-based DLP method. In addition, the proposed method has the smallest number of VMSs used among the three methods.

5.2 Future work

In our current design, we focused only on CPU, memory and network loads in a VMS. In the future, we will include the access time of storage devices in our load prediction. In addition, we will implement and evaluate our proposed load prediction methods and the proposed resource allocation policies in a real cloud computing environment.

References

[1] "Resource allocation," [Online]. Available: http://en.wikipedia.org/wiki/Resource_allocation.

[2] "Resource allocation for network communication," [Online]. Available: http://www.cs.nccu.edu.tw/~lien/VOIP/BBQresrc/.

[3] J. Muller, "On correctness of scalable multi-server state replication in online games," in *Proceedings of the 5th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 1-11, 2006.

[4] C. G. Dickey, "A fully distributed architecture for massively multiplayer online games," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 171, 2004.

[5] K.-W. Lee, B.-J. Ko, and S. Calo, "Adaptive server selection for large scale interactive online games," in *Proceedings of Computer Networks*, vol. 49, pp. 84-102, September 2005.

[6] B. Hariri, S. Shirmohammadi, M. R. Pakravan, and M. H. Alavi, "An adaptive latency mitigation scheme for massively multiuser virtual environments," in *Journal of Network and Computer Applications*, 32(5), pp. 1049-1063, 2009.

[7] W. Kok Wai, "Resource allocation for massively multiplayer online games using fuzzy linear assignment technique," in *Proceedings of the 5th IEEE on Consumer Communications and Networking Conference*, pp. 1035-1039, January 2008.

[8] R. Prodan and V. Nae, "Prediction-based real-time resource provisioning for massively multiplayer online games," in *Proceedings of Future Generation Computer Systems*, vol. 25, pp. 785-793, July 2009.

[9] L. D. Brice, "Robust resource allocation in a massive multiplayer online gaming environment," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, pp 232-239, 2009.

[10] J. Slegers, I. Mitrani, and N. Thomas, "Evaluating the optimal server allocation policy for clusters with on/off sources," in *Proceedings of Performance Evaluation*, vol. 66, pp. 453-467, August 2009.

[11] W. Streitberger and T. Eymann, "A simulation of an economic, self-organizing resource allocation approach for application layer networks," in *Proceedings of Computer Networks*, vol. 53, pp. 1760-1770, July 2009.

[12] R. Stanojevic and R. Shorten, "Load balancing vs. distributed rate limiting: an unifying framework for cloud control," in *Proceedings of the IEEE International Conference on Communications*, pp. 1-6, August 2009.

[13] E. Caron, F. Desprez, D. Loureiro, and A. Muresan, "Cloud computing resource management through a grid middleware: a case study with DIET and Eucalyptus," in *Proceedings of the IEEE International Conference on Cloud Computing*, pp. 151-154, September 2009.

[14] C. Yang, W. Tianyu, and L. Jianxin, "An efficient resource management system for on-line virtual cluster provision," in *Proceedings of the IEEE International Conference on Cloud Computing*, pp. 72-79, December 2009.

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

[16] C. C. Lee, "Design and implement dynamic load balancing mechanism for MMOG," [Online]. Available: http://www.cs.npue.edu.tw.

[17] S. Zhang, S Zhang, X. Chen and X. Huo, "Cloud computing research and development trend," in *Proceedings of the Second International Conference on*

Future Networks, pp 39-97, March 2010.

[18] "Hadoop and Map-Reduce," [Online.] Available: http://dotnetmis91.blogspot.com/2010/04/vs-hadoop-mapreduce.html.

[19] D. T. Ahmed and S. Shirmohammadi "Uniform and non-uniform zoning for load balancing in virtual environments," in *Proceedings of 5th International Conference on Embedded and Multimedia Computing (EMC)*, pp. 1-6, August 2010.

[20] D. T. Ahmed and S. Shirmohammadi, "An auxiliary area of interest management for synchronization and load regulation in zonal P2P MMOGs," in *Proceedings of IEEE International Workshop on Haptic Audio visual Environments and Games*, pp. 36-41, October 2009.

[21] "Lineage," [Online.] Available: http://tw.beanfun.com/lineage/index.aspx.

[22] "World of Warcraft," [Online.] Available: http://www.wowtaiwan.com.tw/main_index.asp.

[23] M.R. Head, A. Kochut, C. Schulz and H. Shaikh, "Virtual hypervisor: enabling fair and economical resource partitioning in cloud environments," in *Proceedings of Network Operations and Management Symposium (NOMS)*, pp 104-111, June 2010.

[24] D. T. Ahmed and S. Shirmohammadi, "A microcell oriented load balancing model for collaborative virtual environments," in *Proceedings of IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pp. 86-91, August 2008.

[25] J. Wang and Z. Yue, "A finding less-loaded server algorithm based on MMOG and analysis," in *Proceedings of IEEE Conference on Intelligent Computation Technology and Automation (ICICTA)*, pp. 96-99, July 2010.

[26] C. Bezerra, J. Comba and C. Geyer, "A fine granularity load balancing technique for MMOG servers using a KD-tree to partition the space," in *Proceedings of*

VIII Brazilian Symposium on Games and Digital Entertainment (SBGAMES), pp. 17-26, June 2009.

[27] V. Nae, A. Iosup and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," in *the IEEE Transactions on Parallel and Distributed Systems*, pp. 380-395, January 2010.

[28] A. K. Jain, M. Jianchang and K. M. Mohiuddin, "Artificial neural network: a tutorial," in *the IEEE Computer*, pp. 31-44, August 2002.

[29] "MATLAB," [Online.] Available: http://www.mathworks.com/.

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

[31] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," in *the IEEE Transactions on Man and Cybernetics Systems*, pp. 665-685, May 1993.