國立交通大學

資訊科學與工程研究所

碩 士 論 文

在軟體定義網路下雲端資料中心之
應用程式感知資源分配機制

Application-aware Resource Allocation for

SDN-based Cloud Datacenters

研 究 生：洪維藩

指導教授：王國禎 教授

中 華 民 國 102 年 6 月

在軟體定義網路下雲端資料中心之應用程式感知資源分配機制

Application-aware Resource Allocation for SDN-based Cloud Datacenters

研 究 生：洪維藩　　　Student：Wei-Fan Hong

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

June 2013

Hsinchu, Taiwan, Republic of China

中華民國 102 年 6 月

在軟體定義網路下雲端資料中心之

應用程式感知資源分配機制

學生：洪維藩　　　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

在雲端資料中心，由於資源需求量的變動非常大，有效地分配與管理資源，並同時滿足每一個應用程式的服務水準協議是一個非常重要的研究議題。在本論文中，我們提出了一個應用程式感知資源分配的機制（App-RA），預估在軟體定義網路下雲端資料中心每個應用程式所需的資源，從而分配適當數量的虛擬機器（VMs）給每個應用程式。就我們所知，我們所提的應用程式感知資源分配機制（App-RA）是第一個可以適用在各種不同的應用程式之感知資源分配機制，可以讓各個不同的應用程式中滿足不同的服務水準協議、且達到有效的分配資源及省電。本應用程式感知資源分配機制（App-RA）是基於類神經網路去預估未來所需的資源（CPU、記憶體、GPU、硬碟 I/O、網路

頻寬)，並且利用目前時間戳記當作輸入的參數之一，使資源預估變得更為準確。我們為不同類型的應用程式提出兩個分配虛擬機器的演算法，並利用動態調整虛擬機器之分配閾值（VM allocation threshold）來避免違反服務水準協議。除此之外，我們採用基於軟體定義網路OpenFlow網路的CICQ交換器，在網路層針對不同類型的應用程式封包進行妥適排程，最後，模擬結果表示，我們所提的應用程式感知資源分配機制僅比最佳解多了 9.21%的耗電，即比起適用於非圖形應用的代表性感知資源分配方法省下了 104.58%的耗電。除此之外，我們的機制對不同應用程式的SLA違反率皆低於4%。

關鍵詞：服務水準協議、應用程式感知、資源分配、雲端資料中心、軟體定義網路。

Application-aware Resource Allocation for SDN-based Cloud Datacenters

Student: Wei-Fan Hong   Advisor: Dr. Kuochen Wang

Department of Computer Science

National Chiao Tung University

Abstract

In cloud datacenters, since resource requirements change frequently, how to assign and manage resources efficiently while meeting service level agreements (SLAs) of different types of applications is an important research issue. In this paper, we propose an Application-aware Resource Allocation (App-RA) scheme to predict resource requirements and allocate the appropriate number of virtual machines (VMs) for each application in SDN-based cloud datacenters. To the best of our knowledge, the proposed App-RA is the first application-aware resource allocation scheme that adapts to all types of applications. The App-RA can meet SLAs, allocate resources efficiently, and reduce power consumption for each application in cloud datacenters. The proposed App-RA adopts the neural network based predictor to forecast the requirements of resources (CPU, Memory, GPU, Disk I/O and bandwidth) for an application. In the proposed App-RA, we have designed two algorithms which allocate appropriate numbers of virtual machines and use the VM allocation threshold to avoid SLA violations for five different types of applications.

In addition, we adopt an SDN-based OpenFlow network with CICQ switches to appropriately schedule packets for different types of application in the network layer. Finally, simulation results show that the power consumption of the proposed App-RA is only 9.21% higher than that of the best case (oracle) and the power consumption of App-RA is 104.58% better than that of EAACVA, which is a representative resource allocation method for non-graphic applications. Furthermore, the SLA violation rate of the proposed App-RA is less than 4% for all applications.

Keywords: service level agreement, application-aware, resource allocation, cloud datacenter, software define network.

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 NSC101-2219-E-009-001 and by the Inventec under Contract 101C179 is grateful acknowledged. Finally, I thank my family for their endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In cloud datacenters, resource requirements changes frequently. Therefore, dynamic allocating and managing resources to meet the SLA of each application is an important research issue. The objective of dynamic resource allocation is to satisfy the SLA while minimizing the power consumption in cloud datacenters.

In order to satisfy the SLA of each application, resource prediction is a fundamental technology. A resource prediction tool is used to predict resource requirements, and then we can allocate resources in advance to avoid SLA violation. In existing resource allocation schemes, most of them adopt a neural network based prediction, which has been proved its prediction accuracy, so we also adopt a neural network based predictor.

There are two types of resource allocation: server-aware resource allocation and application-aware resource allocation. The server-aware resource allocation, which detects loading of a server and allocates VMs for all applications in the server, cannot assign different SLAs to different applications. In contrast, application-aware resource allocation, which detects loading in an application and allocates VMs to the application, can assign different SLAs to different applications.

In the cloud computing, even if we predict and allocate network bandwidth for each application in advance, the network may still congest. In order to resolve this problem, we adopt an SDN-based OpenFlow [1] network with a CICQ switches to schedule packets for different applications in the network layer. OpenFlow is an open standard to allow researchers to run experimental protocols in realistic networks and is currently deployed in large-scale datacenters, like GENI [1].

In this paper, we propose an Application-aware Resource Allocation (App-RA) scheme to predict resource requirements and allocate an appropriate number of VMs for each application in SDN-based cloud datacenters. The rest of the paper is organized as follows. In Chapter 2, we review existing resource allocation mechanisms, compare application-aware and server-aware resource allocation schemes for cloud datacenters, and introduce the OpenFlow. In Chapter 3, we describe the proposed App-RA scheme. In Chapter 4, we evaluate our proposed App-RA scheme using CloudSim, and we compare the power consumption of the proposed App-RA with EAACVA, which is the best available related work for non-graphic applications. Finally, we conclude this thesis and outline future work in Chapter 5.

Chapter 2

Related Work

2.1 Resource allocation architecture

In order to satisfy SLAs for each application, researchers have proposed resource allocation schemes used in cloud datacenters. As shown in Figure 1, there are two types of resource allocation: server-aware resource allocation and application-aware resource allocation. In the next section, we will describe their differences.

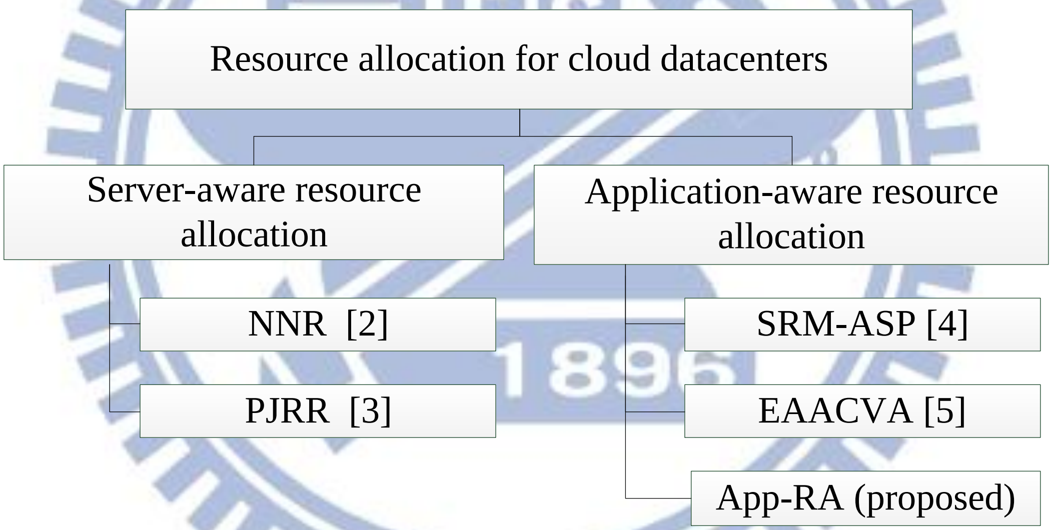

Figure 1. Classifications of existing resource allocation methods.

2.1.1 Server-aware resource allocation schemes

In this section, we introduce existing server-aware resource allocation schemes. In the past, server-aware resource allocation schemes were used in cloud datacenters. However, it cannot assign different SLAs to each application. Because there are always different applications in cloud datacenters, only setting a single SLA cannot satisfy different requirements for different applications. Thus, it is difficult to satisfy the SLA of each application in cloud datacenters. Nowadays, application-aware resource allocation has become the major scheme in cloud datacenters.

In [2], it presents a scheme which predicts workload using neural network and allocate VMs in cloud datacenters, but it mainly focuses on resource prediction. In [3], it introduces a resource prediction scheme using a neural network and uses a resource allocation schemes to save power in cloud datacenters, but it does not consider an SLA for each application in cloud datacenters.

2.1.2 Application-aware resource allocation schemes

We review existing application-aware resource allocation which detects an application's loading and allocates VMs to the application. Application-aware resource allocation can satisfy different SLAs for different applications, its prediction still accurate if an application migrates to another server, and it still can allocate appropriate VMs to the application. To fully utilize the advantage we mentioned above, we adopt application-aware resource allocation for the proposed App-RA. Table 1 compares the differences between application-aware and server-aware resource allocation in cloud datacenters.

Table 1. Comparison of application-aware and server-aware resource allocation schemes for cloud datacenters.

Issues	**Server-aware resource allocation [2] [3]**	**Application-aware resource allocation(proposed App-RA)**
SLA	It only sets a single SLA for all applications	It can set different SLA for different applications
Resource prediction	When an application migrates to another server, prediction will lose accuracy because it predicts a server's loading rather than an application's loading	When an application migrates to another server, prediction is still accurate
Resource allocation	It only allocates VMs according to server's loading	It allocates appropriate VMs to an application according to resource requirements of the application

In [4], it proposes an application-aware resource allocation scheme to dynamically manage resources, and it satisfies SLA and allocates resources efficiently for web applications. In other word, it determines how many VMs should be allocated to satisfy SLA. However, it only adapts to web applications. In [5], it proposes an application-aware resource allocation scheme with minimum power consumption. It runs benchmarks to measure how many VMs and power required in each application. While the resource allocation scheme is based on benchmark process which takes a lot of time, [5] cannot dynamically adjust numbers of VMs for different applications. In other words, it is a static scheme rather than a dynamic scheme. In addition, it does not consider graphic applications.

As shown in Table 2, the proposed App-RA can adapt to all types of applications, meet different SLAs for different applications, adjust resources if an application violates its SLA, adjust bandwidth provisioning in an SDN-based datacenter network for all applications, and provide resource prediction for each application.

Table 2. Comparison of proposed App-RA with related work.

Approach	**SRM-ASP [4]**	**EAACVA [5]**	**App-RA (proposed)**
Suitable for applications	Only for web applications	For non-graphic applications	All types
Different SLAs for different applications	Yes	Yes	Yes
SLA violation handler	Yes (Response time)	No	Yes (Response time)
Bandwidth provisioning	No	No	Yes
Resource prediction	No	No	Yes (Neural network based)

2.2 SDN-based datacenter network

In recent years, researchers have proposed an SDN-based datacenter network which combines with cloud datacenter. In [6], it proposes a practical virtualization cloud datacenter in the SDN network and it defines an APP-ID (a 24 bits label, which can be stored in the IP header) which is used to identify an application in the SDN network. However, it does not consider resource allocation in cloud datacenters.

In [7], it proposes an OpenFlow-based flow level bandwidth provisioning scheme for CICQ switches. Because it schedules packets at flow level, network delay can be decreased by scheduling if the flow requires high bandwidth. Based on this observation, the proposed SDN-based cloud datacenter network schedule packets at application level and network delay can be decreased by scheduling if an application requires low network response time in SLA.

Chapter 3

Application-aware Resource Allocation for SDN-based Cloud Datacenters

In this chapter, we introduce the proposed App-RA, which predicts resource requirements and allocates an appropriate number of virtual machines (VMs) for each application in SDN-based cloud datacenters.

3.1 Application-aware resource prediction

In the proposed App-RA, we propose an application-aware resource prediction to predict resource requirements for each application. In Figure 2, we adopt a neural network to predict the resource requirements (CPU, memory, GPU, hard disk I / O and bandwidth utilization) for each application, and we also use these five types of resource requirements as input factors for the neural network. When the neural network training completes, it can be used to predict resource requirements.

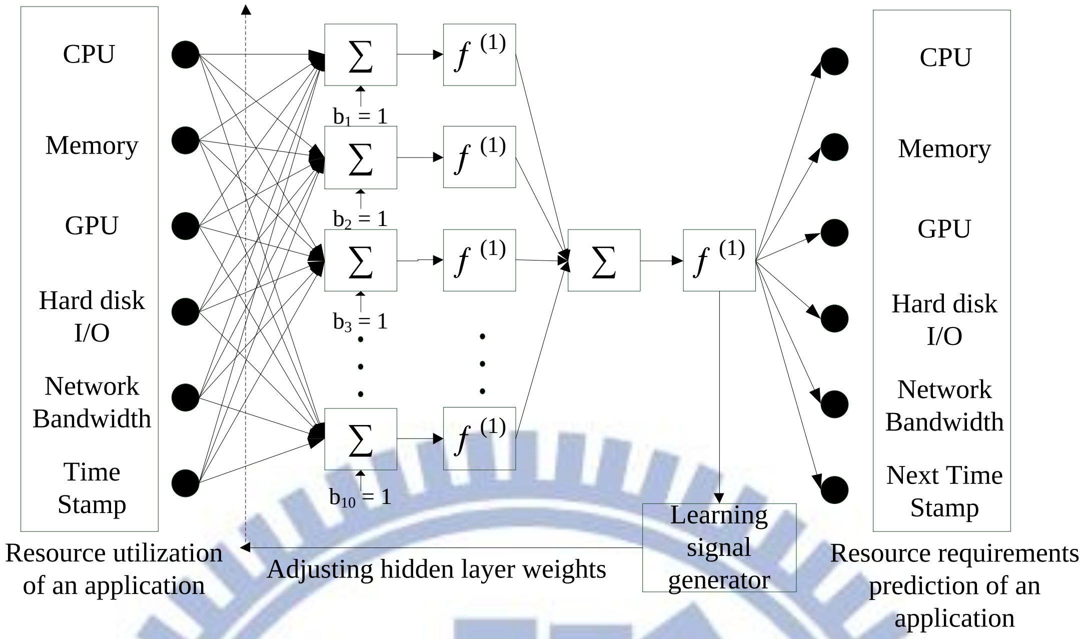

Figure 2. The proposed App-RA resource prediction scheme with neural network based prediction.

In addition, because the number of users changes all the time in the internet, we also apply *Time Stamp* as an input factor for the neural network to make prediction more accurate. In the next section, we introduce the proposed *Application-aware Resource Allocation* (App-RA) scheme based on application-aware resource prediction to assign an appropriate number of VMs so as to meet the SLA of each applications.

3.2 **Application-aware resource allocation**

In this section, we introduce the proposed application-aware resource allocation scheme to allocate an appropriate number of VMs for each application. The objective of the proposed App-RA is to meet SLAs, allocate resources efficiently, reduce power consumption for each application and adapt all types of applications in cloud datacenters.

We allocate VMs with different capacity levels, as show in Table 3. Because power-on a VM requires the basic resource of an operation system, using large and medium VMs can save resources to several many small VMs are needed. In other words, if an application has high resource requirements, the application may use large or medium VMs.

Table 3. Different capacity levels of VMs for cloud datacenters [8].

Capacity	CPU (core)	Memory (MB)	Bandwidth (KB/s)
Small	1	512	1000
Medium	2	1024	2000
Large	4	2048	4000

As show in Figure 3, it shows the flowchart of the proposed App-RA for an application. The function of Algorithm 1 is used to power-on and power-off VMs, and Algorithm 2 is used to adjust the VM allocation threshold.

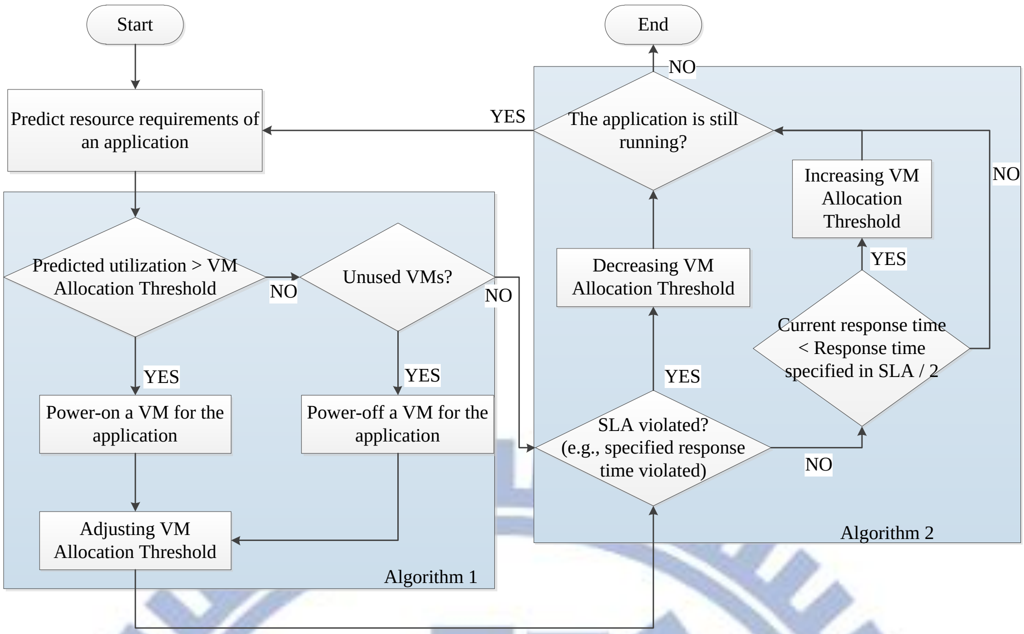

Figure 3. Flowchart of the proposed App-RA.

In a cloud datacenters, we may need to adjust number of VMs to meet each application. Algorithm 1 shows how to decide when to power-on or power-off VMs for an application. If the predicted utilization is greater than the VM Allocation Threshold of an application, we power on a VM to support this application. If the current utilization is less than the sum of all VMs' capacity and the reserved resources (maximum utilization – VM Allocation Threshold) in an application, we power off a VM to save power for this application in the cloud datacenter.

Algorithm 1 Power-on and power-off VMs for an application

X_i = the number of VMs currently used by application i

Maximum utilization = X_i * 100%

Unused resources = Maximum utilization - Predicted utilization

If Predicted utilization > VM Allocation Threshold **then**

$X_i = X_i + 1$;

VM Allocation Threshold = VM Allocation Threshold + VM capacity

End if

If Unused resources > (VM capacity + (Maximum utilization – VM Allocation Threshold)) **then**

$X_i = X_i - 1$;

VM Allocation Threshold = VM Allocation Threshold – VM capacity

End if

We also need to adjust the VM Allocation Threshold to meet the SLA of each application. Algorithm 2 shows how to dynamically adjust the VM Allocation Threshold. When the SLA (for example, response time) is violated, the VM Allocation Threshold is decreased according to an SLA weight (W_{SLA}) which is a value between 0 and 1. If W_{SLA} approaches 0, it means more resources are reserved for an application which can result in decreasing the SLA violation of the application, and vice versa.

When the response time is lower than half of the response time specified in the SLA, the VM Allocation Threshold will be increased according to a power consumption weight (W_P) which a value between 1 and 2. If W_P approaches 2, it represents that very few resource are reserved for an application which can reduce power consumption, and vice versa.

Algorithm 2 Adjustment of VM Allocation Threshold for an application

X_i = the number of VMs currently used by application *i*

W_{SLA} = SLA weight ($0 < W_{SLA} < 1$)

W_P = Power consumption weight ($1 < W_P < 2$)

If $X_i >= 1$ **then**

    If Current response time > Response time specified in SLA **then**

        VM Allocation Threshold = VM Allocation Threshold * W_{SLA};

    Else if Current response time < (Response time specified in SLA / 2) **then**

        VM Allocation Threshold = VM Allocation Threshold * W_P;

    End if

End if

3.3 **Proposed SDN-based datacenter network design**

In cloud computing, even if we predict and allocate network bandwidth for each application in advance, the network may still congest. In order to resolve this problem, we adopt an SDN-based OpenFlow [1] network with CICQ switches to schedule packets from different applications in the network layer. For example, the video streaming application needs more bandwidth, but the search engine application should have a high scheduling priority, so the packets of the search engine application can be sent earlier to avoid increasing network delay. We propose the following scheduling strategy in order to solve the above problem:

1. The controller maintains a bandwidth provisioning table for different types of applications and sends it to CICQ switches.
2. The switches decide packet scheduling priorities based on the bandwidth provisioning table from the controller.

As shown in Table 4, an example bandwidth provisioning table is provided, as follows; however, the actual bandwidth provisioning maybe based on the charge of each application.

Table 4. Bandwidth provisioning for different types of applications.

Type of an applications	**Bandwidth provisioning**
Search engine	10
3D Game	8
Social networking	6
Video	4
Message, Mail	2

First, we need to modify a OpenFlow controller to support our method. We add an APP-ID (24 bits) label of each application to the OpenFlow packet header [6], and the controller can identify each application. The controller decides bandwidth provisioning table for each application, and modify the flow table in switches. Second, we modify the OpenFlow switches to support our method. In Figure 4, we use a CICQ (Combined-Input-Crosspoint-Queued) switch [7] to handle packet scheduling for each application. The CICQ switch is a kind of crossbar switches with a small exclusive buffer at each crosspoint.

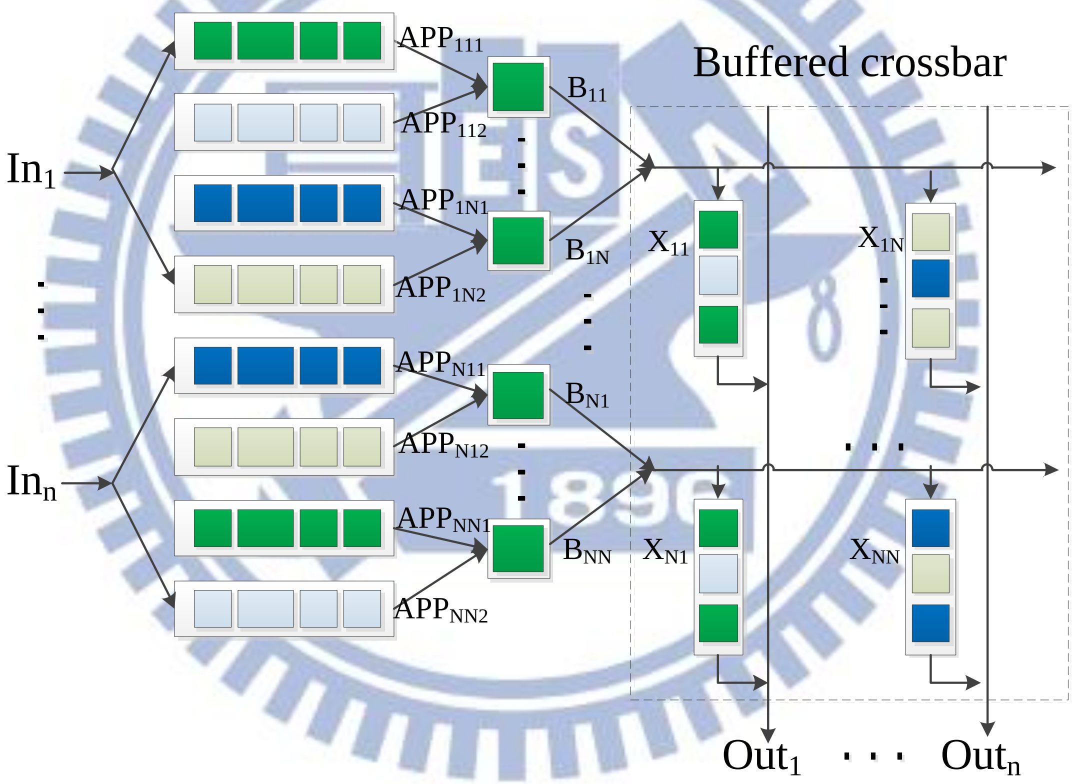

Figure 4. Using a CICQ switch to adjust the bandwidth provisioning for each application.

As shown in Figure 4, we propose a packet scheduling method to forwarding packet scheduling for each application.

1. In *application scheduling*, input port In_i receives packets and put them into corresponding input buffers (APP_{ijk}'s). *Application scheduling* selects packets with minimum execution time [7] and sends them to the corresponding buffer (B_{ij}).
2. In *input scheduling*, packets are sent to output buffers (X_{ij}'s) according to the FIFO order in B_{ij}.
3. In *output scheduling*, packets are sent to output port Out_j according to the bandwidth provisioning table.

Chapter 4

Evaluation and Discussion

4.1 Simulation environment

In section 4.2, we use five different types of applications to evaluate the proposed App-RA in terms of total utilization of an application and response time in the CloudSim [9] simulator. However, since we could not obtain video traffic data in Table 4, APP4 is replaced with a social networking of derive with decreasing traffic data in contrast to APP3 with increasing traffic data. Note that the total utilization of an application is defined as sum of all VMs utilization of an application, and response time of an application is derived from [16]. We simultaneously executed five different types of applications, as described in Table 5. Table 6 shows related simulation parameters.

Because CloudSim does not provide the function for network simulation, we add a network function into our simulation environment to evaluate the proposed App-RA. First, according to the CICQ switch framework, we create corresponding buffers in our simulation environment. Second, we use APP-ID to identify different applications, and put packets to the corresponding buffers. Third, according to our approach, we perform packet scheduling, that we mentioned in section 3.3 in CICQ switches. Finally, after sending a packet in the CICQ switch, we will increase response time for applications which have remaining packets in the CICQ switch.

Table 5. Simulation of five different types of applications.

Application name	Types of an application
APP 1	Search engine
APP 2	3D gaming
APP 3	Social networking
APP 4	Social networking
APP 5	Web

Table 6. Simulation parameters.

Simulator	CloudSim 3.0
Prediction technique	Neural network-based
Prediction tool	MATLAB 7.11.0 (R2010b)
Number of types of applications	5 types
Number of servers	20
Maximum number of VMs	80
Maximum bandwidth	40 Mbps
W_{SLA}	0.9
W_P	1.1
Initial VM Allocation Threshold	Maximum utilization * 0.8
Total utilization of an app.(%)	Sum of all VMs utilization of an application

4.2 Comparison of different resource allocation schemes

In Figure 5, it shows the total utilization of APP1 (search engine service). Because the proposed App-RA is based on the maximum resource requirement (CPU, memory, GPU, hard disk I / O and bandwidth utilization) [17] of an application to adjust number of VMs allocated to the application, we only show CPU utilization for APP 1. From Figure 5, we can know that the APP 1 has high variations of CPU utilization. In Figure 6, it shows the response time of APP 1 (search engine service). We set 100ms as the SLA violation threshold. We execute APP1 for 100 minutes and only 4 times of SLA violation occurred in the proposed App-RA.

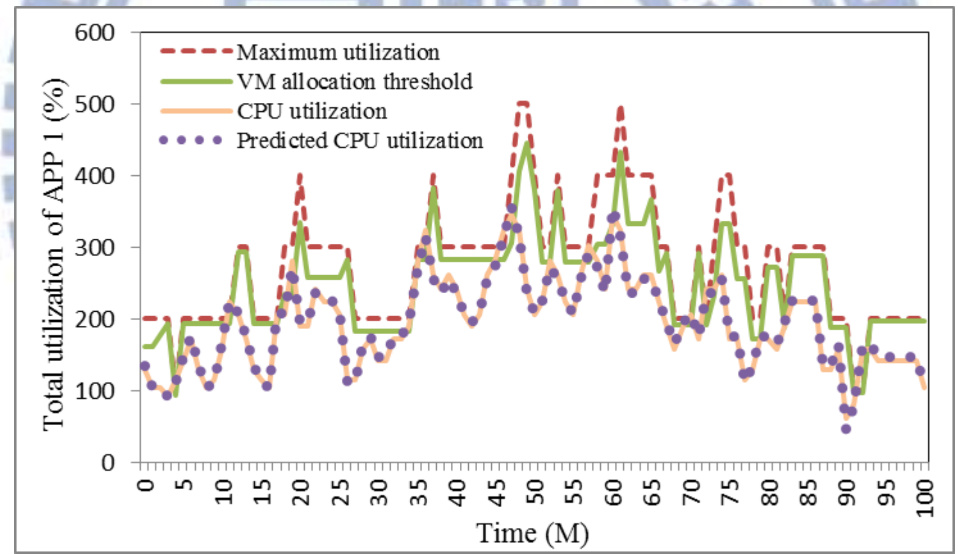

Figure 5. Total utilization of APP 1 (search engine service).

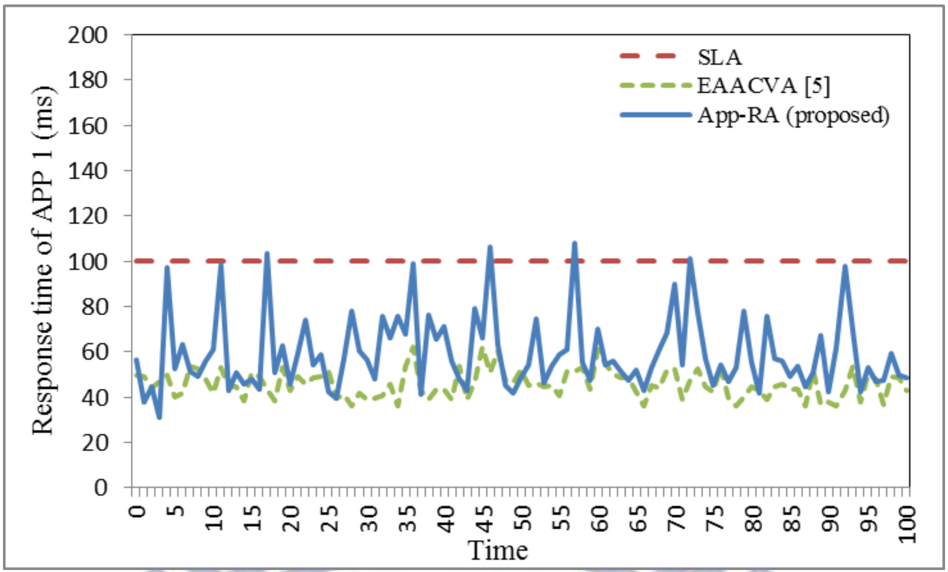

Figure 6. Response time of APP 1 (search engine service).

In Figure 7, it shows the total utilization of APP2 (3D gaming service). Because the proposed App-RA is based on the maximum resource requirement (CPU, memory, GPU, hard disk I / O and bandwidth utilization) [17] of an application to adjust number of VMs allocated to the application, we only show GPU utilization for APP2. In Figure 8, it shows the response time of APP2 (3D gaming service). We set 100ms as the SLA violation threshold. We executed APP2 for 100 minutes and only 4 times of SLA violation occurred in the proposed App-RA. In APP2, EAACVA had a lot of SLA violation because EAACVA did not consider GPU in its design.

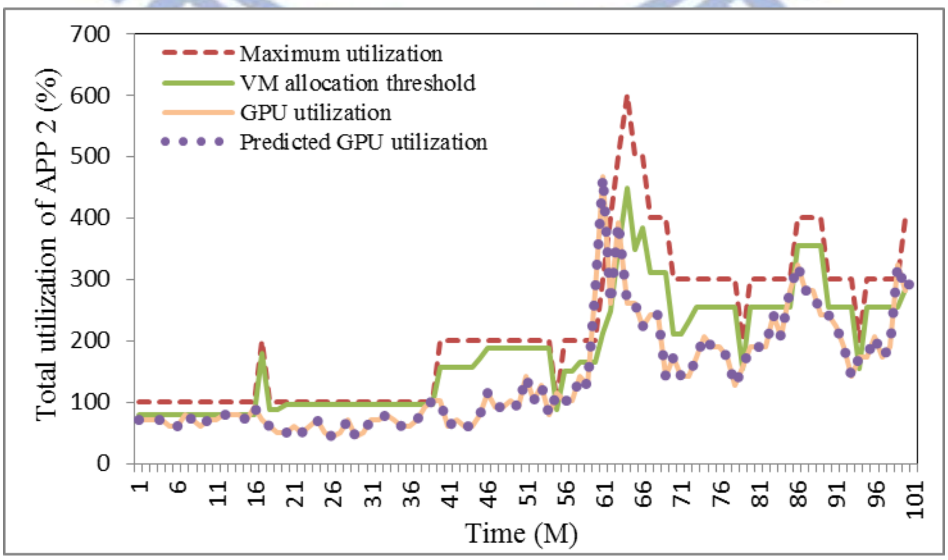

Figure 7. Total utilization of APP 2 (3D gaming service).

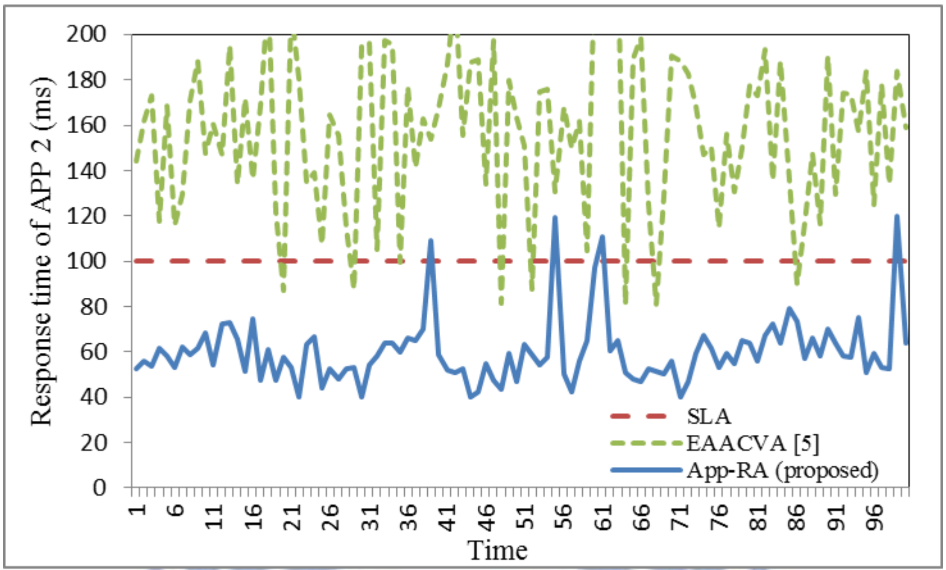

Figure 8. Response time of APP 2 (3D gaming service).

In Figure 9, it shows the total utilization of APP 3 (social networking service). Because the proposed App-RA is based on the maximum resource requirement (CPU, memory, GPU, hard disk I / O and bandwidth utilization) [17] of an application to adjust number of VMs allocated to the application, we only show CPU utilization for APP 3 [14]. From Figure 9, we can know that the total utilization of APP3 keeps increasing, because more user login to this application increasing as time goes on. In Figure 10, it shows the response time of APP 3 (social networking service). We set 100ms as the SLA violation threshold. We executed APP 3 for 100 minutes and only 4 times of SLA violation occurred in the proposed App-RA.

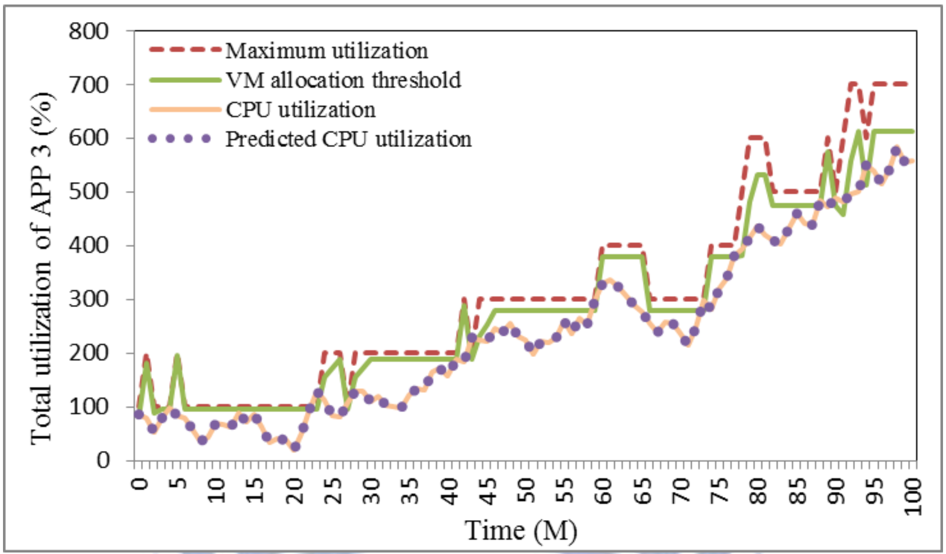

Figure 9. Total utilization of APP 3 (social networking service).

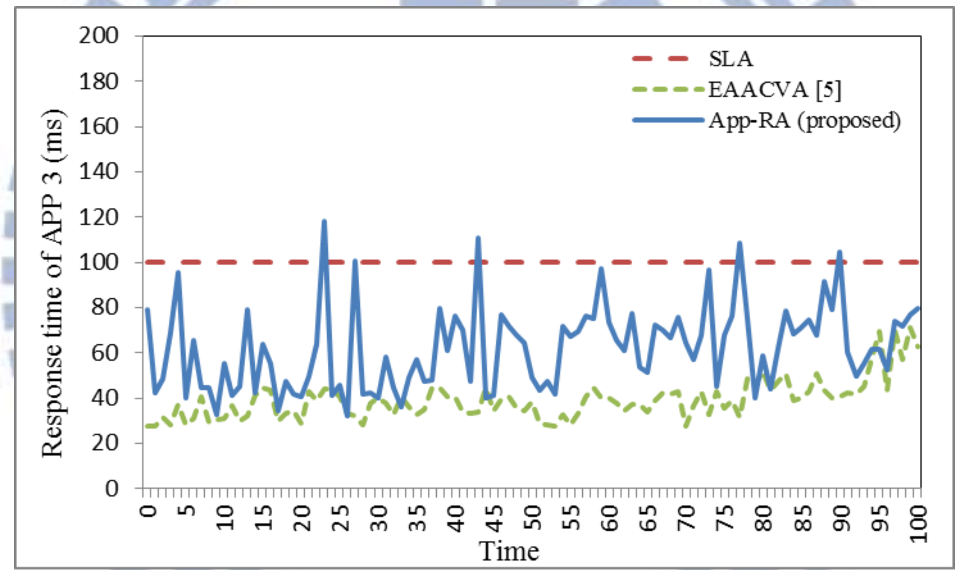

Figure 10. Response time of APP 3 (social networking service).

In Figure 11, it shows the total utilization of APP 4 (social networking service). Because the proposed App-RA is based on the maximum resource requirement (CPU, memory, GPU, hard disk I / O and bandwidth utilization) [17] of an application to adjust number of VMs allocated to the application, we only show CPU utilization [14] for APP 4. From Figure 11, we can know that the total utilization of APP 4 keeps decreasing, because more user logout as time goes on. In Figure 12, it shows the response time of APP 4 (social networking service). We set 100ms as the SLA violation threshold. We executed APP 4 for 100 minute and only 4 times of SLA violation occurred in the proposed App-RA.

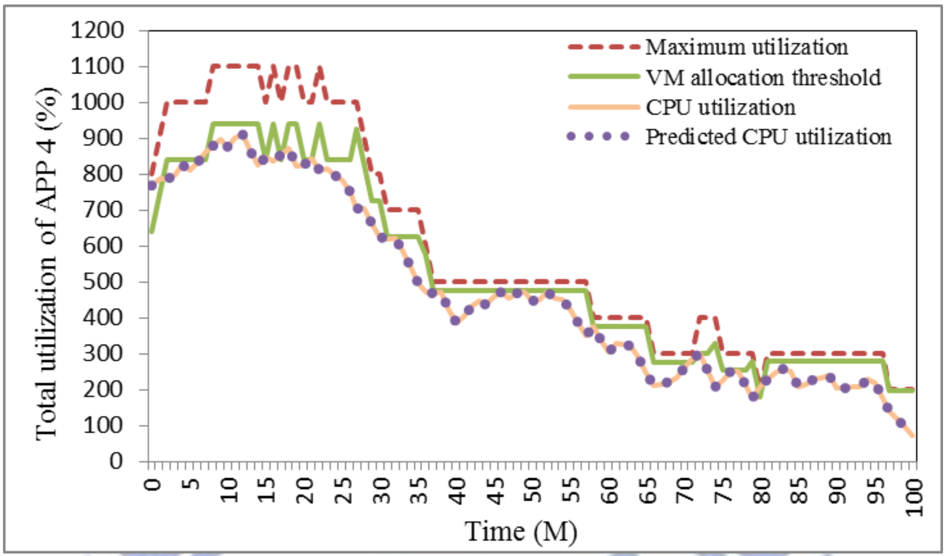

Figure 11. Total utilization of APP 4 (social networking service).

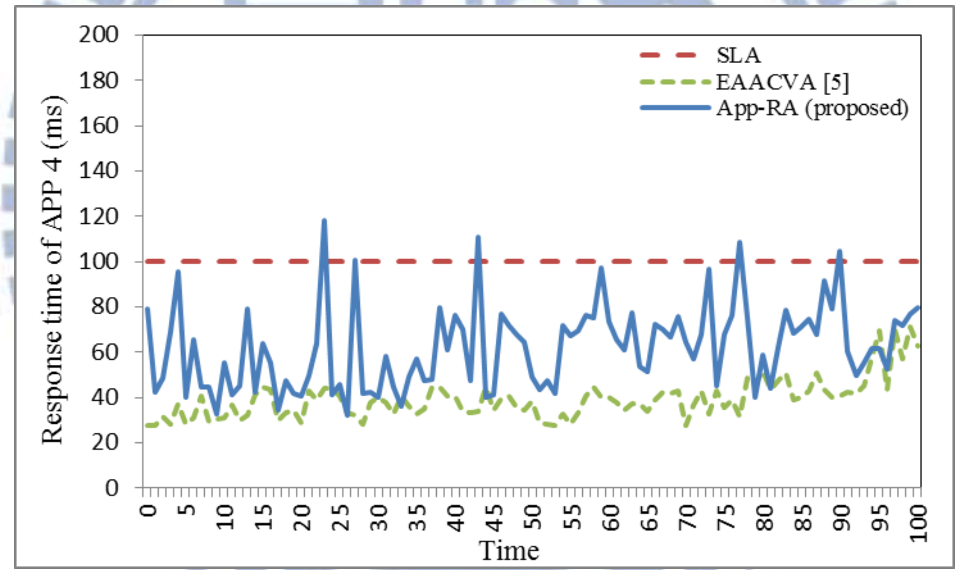

Figure 12. Response time of APP 4 (social networking service).

In Figure 13, it shows the total utilization of APP 5 (web service). Because the proposed App-RA is based on the maximum resource requirement (CPU, memory, GPU, hard disk I / O and bandwidth utilization) [17] of an application to adjust number of VMs allocated to the application, we only show CPU utilization [15] for APP 5. In addition, the traffic source of APP 5 is collected in a lab website. In Figure 14, it shows the response time of APP 5 (web service). We set 100ms as the SLA violation threshold. We execute APP 5 for 100 minute and only 3 times of SLA violation occurred in the proposed App-RA.

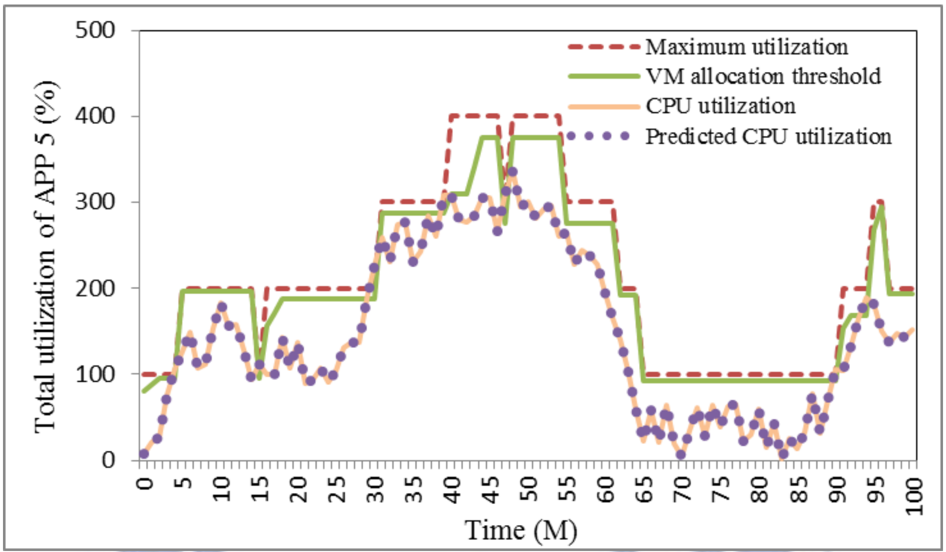

Figure 13. Total utilization of APP 5 (web service).

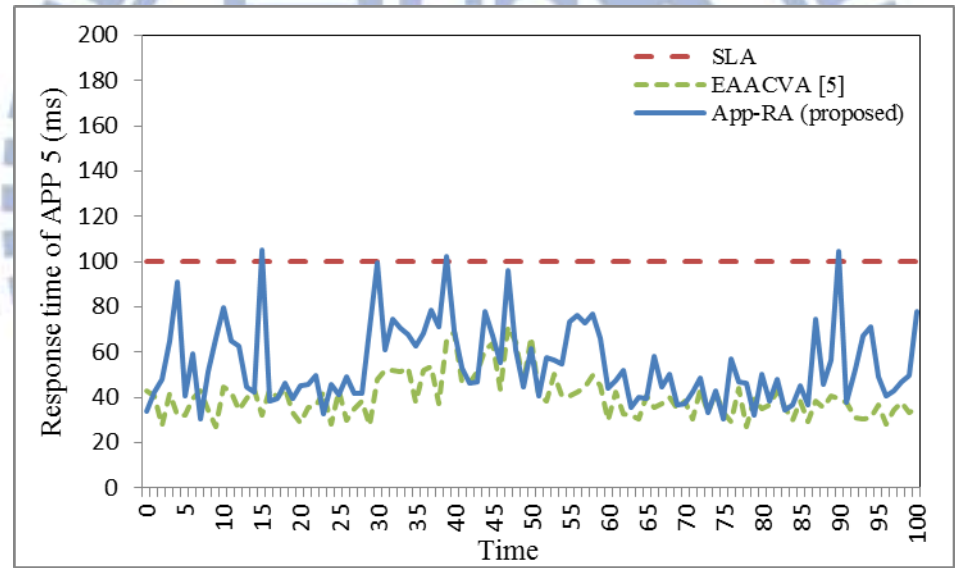

Figure 14. Response time of APP 5 (web service).

4.3 **Comparison of power consumption**

We evaluate the power consumption of power-on/off physical servers and VMs under different loadings [13] among three resource allocation schemes. Note that Google App Engine [11] only guarantees a Monthly Uptime Percentage at least for 95%, and Amazon EC2 [12] guarantees an Annual Uptime Percentage at least for 99%. However, they do not guarantee response time in SLA for each application.

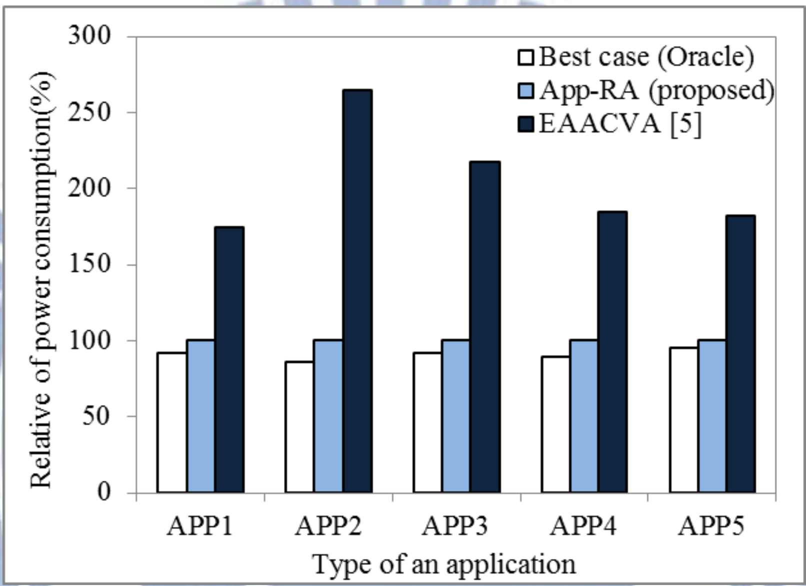

Figure 15. Power consumption comparison of each application among three resource allocation schemes.

Figure 15 and Table 6 shows that the power consumption of the proposed App-RA is only 9.21% higher than that of the best case (oracle) and the power consumption of App-RA is 104.58% better than that of EAACVA, which is the best resource allocation method for non-graphic applications. Furthermore, the SLA violation rate of the proposed App-RA is less than 4% for five types of applications.

Table 7 Comparison of power consumption among three resource allocation schemes.

Application	Best case (Oracle)	APP-RA (proposed)	EAACVA [5]
APP1	91.59%	100%	174.59%
APP2	85.66%	100%	264.29%
APP3	91.73%	100%	217.33%
APP4	89.36%	100%	184.79%
APP5	95.60%	100%	181.88%
Average	90.79%	100%	204.58%

4.4 Comparison of the proposed App-RA with different mechanisms

If the proposed App-RA does not have a better prediction tool, the proposed pp-RA still can provide stable SLA for each application. In Table 7, we compare the proposed App-RA with different prediction mechanisms, which are neural network and last value. The power consumption of the proposed App-RA with last value based prediction is 6.98% higher than that of the proposed App-RA with neural network based prediction in five types of applications. The SLA violation rate of the proposed App-RA with neural network based prediction is almost the same with the proposed App-RA with last value based prediction. In other words, using a better prediction tool can reduce power consumption in the proposed App-RA, but the SLA is almost the same under these two prediction mechanisms.

Table 8 Comparison of App-RA with different prediction mechanisms.

Application	App-RA with neural network based prediction		App-RA with last value based prediction	
	Power consumption	SLA violation rate	Power consumption	SLA violation rate
APP1	100%	4%	108.54%	4%
APP2	100%	4%	101.91%	5%
APP3	100%	4%	105.51%	2%
APP4	100%	4%	112.03%	4%
APP5	100%	3%	106.91%	3%
Average	100%	3.8%	106.98%	3.6%

Chapter 5

Conclusion

5.1 Concluding remarks

In this paper, we have presented an *Application-aware Resource Allocation* (App-RA) scheme to predict resource requirements and allocate an appropriate number of virtual machines (VMs) for each application in SDN-based cloud datacenters. To the best of our knowledge, the proposed App-RA is the first application-aware resource allocation scheme that adapts to all types of applications.

Proposed App-RA can meet SLAs, allocate resources efficiently, and reduce power consumption for different types of applications in cloud datacenters. It adopts a neural network based predictor to forecast the requirements of resources. We have designed two algorithms for proposed App-RA to allocate VMs and dynamically adjust VM Allocation Threshold to avoid SLA violation for different types of applications. In addition, we have also presented an SDN-based OpenFlow network with CICQ switch to schedule packets from different types of applications in the network layer.

Finally, simulation results have shown that the power consumption, of the proposed App-RA is only 9.21% higher than the best case (oracle), and is 104.58% better than EAACVA, which is the best resource allocation method for non-graphic applications. In addition, the SLA violation rate of the proposed App-RA is less than 4% for each application.

5.2 **Future work**

In our simulation environment, we have evaluated the proposed App-RA using the CloudSim simulator, and we have added a network function to the CloudSim simulation. In the future, we will implement the proposed SDN-based datacenter network in the Mininet [10] emulator to evaluate its performance in combination the proposed App-RA. In addition, we will deploy proposed App-RA to an operational cloud datacenters for further evaluation.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow: enabling innovation in campus networks,” in *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[2] Md. T. Imamt, S. F. Miskhatt, R. M. Rahmant, M. A. Amin, "Neural network and regression based processor load prediction for efficient scaling of Grid and Cloud resources," in *Proc. IEEE Computer and Information Technology (ICCIT) Conf.*, pp.333,338, 22-24 Dec. 2011.

[3] J. Prevost, K. Nagothu, B. Kelley, M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proc. IEEE System of Systems Engineering (SoSE) Conf.*, pp.276-281, 27-30 June 2011.

[4] V. Cardellini, E. Casalicchio, F. L. Presti, L. Silvestri, "SLA-aware Resource Management for Application Service Providers in the Cloud," in *Proc. IEEE Network Cloud Computing and Applications (NCCA) Conf.*, pp.20-27, 21-23, Nov. 2011.

[5] H. Viswanathan, E.K. Lee, I. Rodero, D. Pompili, M. Parashar, "Energy-Aware Application-Centric VM Allocation for HPC Workloads," in *Proc. IEEE Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW) Conf.*, pp.890-897, 16-20 May 2011.

[6] P. Lin, J. Bi, H. Hu, "VCP: A virtualization cloud platform for SDN intra-domain production network," in *Proc. IEEE Network Protocols (ICNP) Conf.*, pp.1-2, Oct. 30 2012-Nov. 2 2012.

[7] H. Jin, D. Pan, J. Liu, N. Pissinou, "OpenFlow based flow level bandwidth provisioning for CICQ switches," in *Proc. IEEE INFOCOM Conf.*, pp.476-480, 10-15 April 2011.
[8] “Amazon EC2,” [Online]. Available: http://aws.amazon.com/ec2/.
[9] “CloudSim,” [Online]. Available: http://www.cloudbus.org/cloudsim/.
[10] “Mininet,” [Online]. Available: http://mininet.org/.
[11] “Google App Engine Service Level Agreement,” [Online]. Available: https://developers.google.com/appengine/sla?hl=zh-tw/.
[12] “Amazon EC2 Service Level Agreement,” [Online]. Available: http://aws.amazon.com/ec2-sla/.
[13] A. Beloglazov, R. Buyya1, Y. Lee, A. Zomaya, “A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems”, in *Proc. IEEE Advances in Computers Conf.*, Vol. 82, 2011.
[14] “ Social networking service” [Online]. Available: http://en.wikipedia.org/wiki/Social_networking_service#Social_network_hosting_service.
[15] “High CPU Usage - Learn How to Identify the Root Causes and Reduce Your Web Hosting Bandwidth Usage,” [Online]. Available: http://ezinearticles.com/?High-CPU-Usage---Learn-How-to-Identify-the-Root-Causes-and-Reduce-Your-Web-Hosting-Bandwidth-Usage&id=4575916/.
[16] “Response-Time Modeling for Resource Allocation and Energy-Informed SLAs” [Online]. Available: http://www.cs.berkeley.edu/~jordan/papers/sysml07.pdf/.
[17] V. Nae, A. losup, R. Prodan, “Dynamic Resource Provisioning in Massively Multiplayer Online Games” in *Proc. IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 3, March 2011.