Enhancing a Strategy of Virtualized Resource Assignment in Adaptive Resource Cloud Framework

Dong-Ki Kang, Seong-Hwan Kim, Ye Ren,
Byung-Sang Kim, Woo-Joong Kim
Yu-Sik Kim, Chan-Hyun Youn
Korea Advanced Institute of Science Technology
Daejeon, Korea
+82-42-350-5495
{dkkang, s.h_kim, Catherine, b.s.kim, w.j.kim,
yusiky326, chyoun}@kaist.ac.kr

Chang-Sung Jeong
Korea University
Seoul,
Korea
+82-02-3290-3229
csjeong@korea.ac.kr

ABSTRACT
Recently, cloud computing has been emerged to offer new business model for service providers and computing services for many customers in various fields. Both the profit maximization for cloud service providers and the SLA guaranteeing for cloud service customers are most important issues in the research of cloud computing. Many researches about cloud computing have been focused on VM management to satisfy both the cost minimization and acceptable QoS assurance. However, in previous studies, only VM allocation scheme that maps the single request to single instance has been considered, therefore the unnecessary extra capacity of resources has been dissipated. In this paper, we design an Adaptive Resource Cloud Framework(ARCF) with group VM allocating scheme called Multi Requests to Single VM(MRSV) to solve the problems of previous researches. Through the performance evaluation, we showed the effective resource cost saving of the proposed scheme.

Categories and Subject Descriptors
C.2.4: Distributed Systems - Distributed applications

General Terms : Management, Economics, Theory

Keywords: Cloud Resource Management, VM allocation, Cloud Service Broker, Service Level Agreement

1. INTRODUCTION
Cloud computing has been proposed as a new paradigm which can offer a feasible solution to solve the limitation of restricted amount of resources and reduce the cost for purchasing, maintaining and management of resources[1]. Especially, cloud services are provided to cloud service customers as Infrastructure as a Service (SaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In addition, Cloud computing is able to offer an efficient solution to both the computation-intensive and the data-intensive applications by using customized resources on demand [2].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICUIMC(IMCOM)'13, January 17–19, 2013, Kota Kinabalu, Malaysia.
Copyright 2013 ACM 978-1-4503-1958-4…$15.00.

To provide desirable cloud service to customers - this means that the virtual resource service with acceptable QoS is guaranteed and the resource operation cost of cloud service provider is minimized - the important issues of cloud are how virtual machine (VM) is allocated to cloud service users efficiently and the resource provisioning is planned according to the expected customers' requests adaptively[3]. To do this, many researches of VM management scheme in cloud environment have been studied in terms of the performance and the cost. In general, cloud service customers negotiate with cloud service providers on their required SLA and the corresponding cost of VM instance[4]. As the cost of VM instance is increased, the performance of the VM instance processing is also increased. Based on this characteristics, many previous researches have focused on finding the optimal point between the performance of VM instance and the cost of resource operation in accordance with the customers' requests [5,6,7].

Many cloud service providers such as Amazon and Google proposed the several resource allocation policies to cloud service customers[8,9]. In particular, the VM instance is allocated to customers on fixed unit time basis of resource usage. That is, if a VM instance once is allocated to customer, then the customer occupies the VM instance until the end of the usage unit time in regardless of the completion of the request. If the request of the customer needs low computing capacity and it derives short processing time, then the dissipation of running resource might be occurred. However, in traditional researches on cloud computing, the VM allocation scheme is only considered as a mapping function of single customer's request to the single VM instance. In this case, after the completion of the single request, the VM instance cannot be used anymore even though it has still extra available duration.

In this paper, we design the Adaptive Resource Cloud Framework and propose a new VM allocation of Multi Requests to Single VM (MRSV) scheme in the comparison of the conventional Single Request to Single VM (SRSV) scheme. In the proposed scheme, we consider the scheme for the aggregated requests using the single VM instance in order to minimize unnecessary resource occupation while guaranteeing QoS to the users' SLA. In addition, we introduce the decision scheme of resource type by using similar degree function in order to find the desirable VM instance type corresponding to the users' SLA. As an example to evaluate the performance of the proposed schemes, MapChem chemical application is examined to the cloud testbed environment.

We implemented Openstack-based cloud system to evaluate the performance of the proposed schemes. Through several experiments in terms of resource operation cost, VM instance

performance, and the SLA violation, we demonstrate that the proposed schemes are able to get significant reduction of resource operation cost while guaranteeing QoS to the users' SLA.

2. ADAPTIVE RESOURCE CLOUD FRAMEWORK

In this section, we introduce our proposed Adaptive Resource Cloud Framework(ARCF) with VM allocation of Multi Requests to Single VM(MRSV) scheme to solve the dissipation of extra capacity in the resource.

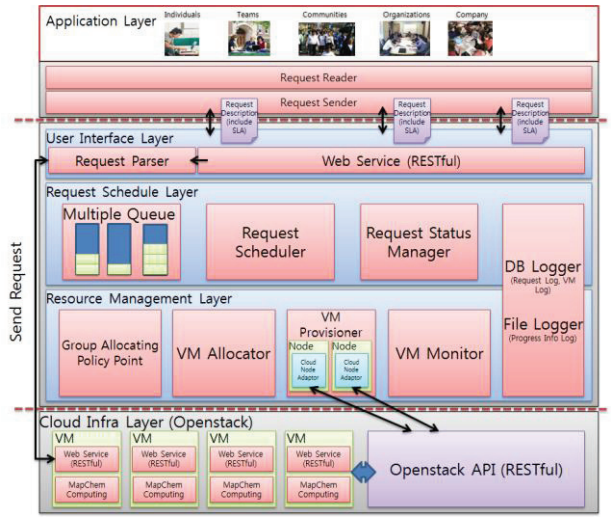


Figure 1. A Model Architecture for Adaptive VM Allocation in Cloud

2.1 Architecture of the Adaptive Resource Cloud Framework

Figure 1 shows the architecture of the proposed Adaptive Resource Cloud Framework(ARCF), which consists of Application Layer, User Interface Layer, Request Schedule Layer, Resource Management Layer, and Cloud Infra Layer. The ARCF integrates the functionality of the resource management system to support the arbitration between cloud service customers and cloud service providers. Ultimately, the objectives of ARCF is to provide the acceptable QoS to cloud service customers. The more detailed description of components in ARCF is described as below,

Customers Application Layer (CAL): Many customers in CAL can submit their request to the ARCF simultaneously. The request reader accepts the request from the cloud service customers and the request sender sends the requests to the User Interface Layer(UIL) as a JSON type description based on RESTFul web service.

User Interface Layer (UIL): When the requests from the Customer Application Layer are arrived at the ARCF as a JSON type description, firstly the request parser parses the request to the separated objects in order to enable them to be processed on the allocated VM instances.

Request Schedule Layer (RSL): In this component, the Request Scheduler is responsible to handle submitted requests in the multiple-queue according to their priority to classify each service level. If the SLA included in the request is very sensitive to the performance, then the high priority tag is attached to the request. These kinds of request are processed preferentially. The Resource Status Manager collects the information of resource status, and transmits it to the Request Scheduler. In general, the resource status is represented to the vector consisting of required hardware components such as CPU, Memory, and I/O Bandwidth, etc.

Resource Management Layer (RML): Resource Management Layer is composed of Group Allocating Policy Point, VM Allocator, VM Provisioner, and VM Monitor. In RML, the VM allocator guarantees the satisfactory service level required based on the SLA description from customers while it minimizes the resource operation cost by the proposed VM allocation scheme from the Group Allocating Policy Point. Group VM allocation scheme called MRSV is described in the next section B in detail. In VM Monitor, the monitoring and recording the information of provisioned VM instances are performed periodically and the request processing data in database is used for managing the ongoing works efficiently. To determine the desirable VM resource type and VM instance to allocate the request, the VM Allocator considers the information of SLA description, previous historical processing result, and current VM utilization. Resource Provisioner adopts the resource provisioning scheme in order to make decision how many VM instances should be prepared to handle the expected requests with acceptable QoS while the resource operation cost is minimized.

Cloud Infra Layer(CIL): Physical machines are located in CIL. We establish the cloud platform OpenStack to provide the VM instances to the customers in order to evaluate the proposed scheme. To enable the communication between VM instance and the cloud service customers, the RESTFul web service is operated continuously in VM instance similar to the UIL.

2.2 Group VM Allocating Policy

In contrast to previous researches about VM allocation in cloud environment, in this paper, we propose the group VM allocating policy called Multi Requests to Single VM(MRSV) scheme. We call traditional VM allocation scheme Single Request to Single VM(SRSV)[5,6,7] to compare the characteristics of the proposed MRSV scheme. In the SRSV scheme, the VM allocator just maps one request to the single VM instance as shown in Figure 2. This scheme might cause the dissipation of unnecessary resource usage operation. In general, once a VM instance is generated, it should be used for a period of certain time based on the VM allocation policy from cloud service provider. That is, if the processing time of request from the specific customer is shorter than the predetermined minimum unit time for usage of the allocated resource, then the dissipation is occurred insomuch the remaining usage time of the resource.

Therefore, the cost is given as shown in Eq(1),

$$C_{waste}(REQ, R) = C(R)(t^{*}_{u\_unit}(R) - t_{proc}(REQ, R)) \quad (1)$$

where t_{proc} is the processing time of the inserted request from customer, $t^{*}_{u\_unit}(R)$ is the power of the minimum unit time for usage of the allocated resource. As the processing time of the request is decreased, then the ratio of resource dissipation is increased, because the unused portion of the allocated resource is also increased. Figure 3 shows the proposed grouping VM allocation scheme MRSV.

By adopting our scheme, we can reduce the dissipation of the allocated resource and furthermore, minimize the additional delay

of the generation of resource allocation for each request from customers. In MRSV scheme, the process is as follows,

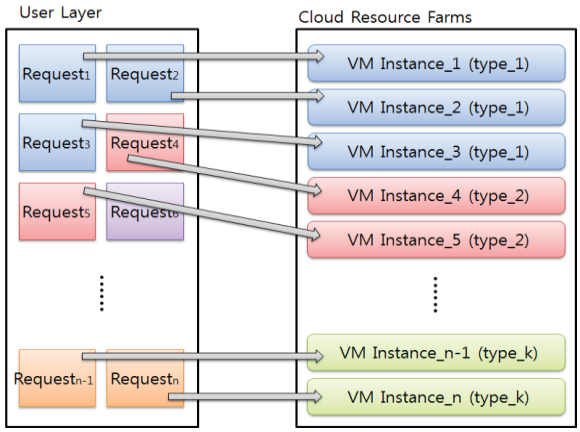


Figure 2. Single Request to Single VM(SRSV)

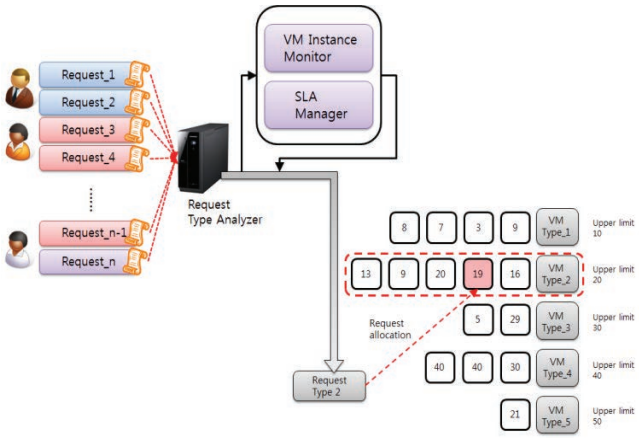


Figure 3. Multi Requests to Single VM(MRSV)

1) When the request from a certain cloud service customer is arrived at the RML, it should be allocated to the resource that is determined in the request type analyzer. In order to decide the accurate resource type for the inserted request, we use simple resource mapping function by using similar degree.

2) The SLA Manager handles the information of SLA description included in the request from the cloud service customer to guarantee the desirable QoS(Quality of Service) level. There are three factors of SLA : CPU, Memory, and I/O Bandwidth. In Cloud Infra Layer, there are several VM resource types representing their resource specification (flavor) which means their system capacity. The dynamic status of ongoing VM instances is recorded to database and collected by the VM Instance Monitor and is sent to the SLA Manager.

3) Once the VM resource type that is due to allocation is determined, then the selection of the specific VM instance among the ongoing VM instances in Cloud Infra Layer is performed by the VM allocator. In this phase, two VM allocation policies, request balancing and request consolidation scheme, can be considered[7]. If we adopt the request-balancing scheme, then we are able to provide the cloud service to cloud customers with stable QoS but need considerable VM instances. Otherwise, when we adopt the request consolidation scheme, then we are able to minimize the number of allocated VM instances, but the degradation of the service quality might be occurred. In this paper, we choose the latter scheme, request consolidation scheme in order to minimize the resource operation cost.

4) Since we adopt the request consolidation scheme, the VM allocator should find the VM instance that has the least capacity while it is able to support the inserted request. For example, as shown in Figure 3, if the VM resource type 2 that has the capacity of 20 requests is chosen, and there are 5 ongoing VM instances in the VM resource type, then we select the VM instance processing 19 requests among them. The VM instance processing 20 requests is disable to be selected since it has no extra capacity to process the additional request.

Here, we assume that all the sub jobs of the request from the certain cloud service customer should be processed in the sequential order and they are not parallel computing jobs. The notations of the used variables in this paper are described as follows,

CPU : The resource capacity to process the CPU instruction of the request per a certain unit time. For example, the unit time is 1sec, and the CPU=15, then the resource is able to process the 15 CPU instructions per 1 sec.

Mem: This is similar to CPU. The capacity to process the memory related instruction per a certain unit time

I/O : Data read/write capacity per a certain unit time.

$\mathrm{REQ}_i = \{CP_1^i, CP_2^i, CP_3^i, \ldots, CP_k^i, \ldots, CP_n^i\}$, The request from cloud service customer i. CP_k^i is the *k*th component job included in the request. As mentioned above, it is assumed that all the component jobs are processed in the sequential order, not in parallel.

$\mathrm{CPU}_i, \mathrm{Mem}_i, \mathrm{I/O}_i$: The capacity of each component in resource i.

The processing time of the *k*th component job in the request from user *i* in the *m*th VM instance as shown in Eq. (2),

$$t_{proc}\left(CP_k^i, R_m\right) = \frac{t_{proc}(CP_{k,CPU}^i)}{CPU_m} + \frac{t_{proc}(CP_{k,Mem}^i)}{Mem_m} + \frac{t_{proc}(CP_{k,I/O}^i)}{I/O_m} \quad (2)$$

And the processing time of the request from user i in the m th VM instance is obtained as Eq.(3),

$$t_{proc}(REQ_i, R_m) = \sum_{k=1}^{l} t_{proc}\left(CP_k^i, R_m\right) + t_{init}(R_m) \quad (3)$$

where l means the number of components in the request.

Algorithm 1 Procedure of Resource Type Decision

INPUT :

$[\overline{r_c}, \overline{r_m}, \overline{r_{I/O}}]$ the required resource specification vector for processing the request from the cloud service customer. c, m, i/o means the capacity of CPU, Memory, and I/O Bandwidth, respectively.

$v_{vm\_type_i} = [r_{c_i}, r_{m_i}, r_{I/O_i}], 1 \le i \le n_{vm\_type}$ the resource flavor type of the VM instance *i*. n_{vm_type} is the total number of resource types. V_{vm_type} includes all the VM types those are able to

allocated to current being considered requests, $\forall V_{vm\_type_i} \in V_{vm\_type}$.

c_i, the processing capacity of request allocation to VM instance type *i*.

v_{ij}, the *j*th VM instance having VM type $v_{vm\_type_i}$

VARIABLES :

$d(i)$ is the similarity degree between the resource specification of the VM resource type i and the required resource specification from the cloud service customer.

$w_c, w_m, w_{I/O}$ represent the weighted values for the CPU, Memory, and I/O Bandwidth of a resource in order to reflect the portion of each resource component.

$wl(v_{ij})$, the workload of request to v_{ij} such that $0 \le wl(v_{ij}) \le c_i$ for all v_{ij}

OUTPUT :

v_{sel}, a selected VM instance in order to process the current request from cloud service customer.

BEGIN :

For $1 \le i \le n$ **then**,

Find the best resource type that has the closest similarity degree with the required resource specification from the cloud service customer by calculating the following formula as,

$d(i) = \sqrt{w_c(\overline{r_c} - r_{c_i})^2 + w_m(\overline{r_m} - r_{m_i})^2 + w_s(\overline{r_s} - r_{s_i})^2}$

$min(d(i))$ is the resource type which is suitable to process the request.

For chosen VM type $v_{vm\_type_i}$, find $v_{sel} = max\left(wl(v_{ij})\right)$ with constraint that $wl(v_{ij}) < c_i$.

END

In our proposed MRSV scheme, the whole processing time of multi requests of the set I (so, requests from the customer i ∈ I) in the mth resource is given by Eq.(4),

$$t_{proc}(REQ_I, R_m) = \sum_{i \in I} \sum_{k=1}^{N_{comp}(i)} t_{proc}(CP_k^i, R_m) + t_{init}(R_m) + t_{CS}(I) \quad (4)$$

where $T_{cs}(I)$ means the context switching time to process the request set I and $T_{init}(R_m)$ is the initialization time for the generation VM instance.

In SRSV scheme, the whole processing time of requests from the customer i in the multi VM instances is obtained as Eq.(5),

$$MAX_{i \in I}(\sum_{k=1}^{N_{comp}(i)}(t_{proc}(CP_k^i, R_{user_i}) + t_{init}(R_{user_i}))) \quad (5)$$

where R_{user_i} means the VM resource allocated customer *i*.

All the VM instances have the minimum unit time for the usage of resource. In MRSV, the usage time of mth resource that has the request set I to process is as shown in Eq.(6),

$$t_{usage}(I, R_m) = \left\lfloor \frac{\sum_{i \in I} \sum_{k=1}^{N_{comp}(i)} t_{proc}(CP_k^i, R_m) + t_{init}(R_m) + t_{cs}(I)}{t_{u\_unit}(R_m)} \right\rfloor \quad (6)$$

Similar to MRSV case, in SRSV, the usage time of each resource which has the request set I to process is as shown in Eq.(7),

$$t_{usage}(I, R_{user_i}) = \left\lfloor \frac{\sum_{k=1}^{N_{comp}(i)} \left(t_{proc}(CP_k^i, R_{user_i}) + t_{init}(R_{user_i})\right)}{t_{u\_unit}(R_{user_i})} \right\rfloor \quad (7)$$

After all, we are able to calculate the saving cost of our proposed MRSV scheme with compare to existing SRSV scheme is given by Eq.(8),

$$t_{usage}(I, R_m) C(R_m) - \sum_{i \in I} t_{usage}(I, R_{user_i}) C(R_{user_i}) \quad (8)$$

The performance of both MSRV and SRSV schemes depends on the additional delay caused by the context switching for multi requests processing and the generation of VM instances, respectively. However, our proposed MRSV scheme has better resource saving than the existing SRSV scheme in any case. In order to map the requests to the accurate VM resources, the VM allocator finds the VM resource type that has the most similar degree value with the SLA description included in the request in views of CPU, Memory, and I/O Bandwidth. The algorithm description of the procedure of VM resource type decision is presented in Algorithm 1.

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

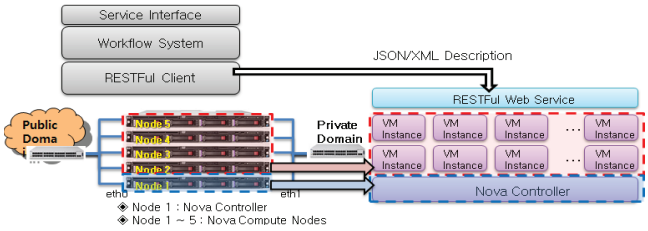
We consider the MapChem service which is a kind of chemical application program as an application service in order to evaluate the performance of the proposed scheme MRSV in comparison of the traditional scheme SRSV. In addition, we also evaluate the resource type decision algorithm under the cloud-based environment. In this section, we establish the open-source cloud platform called OpenStack[10] that supports a variety of hypervisors such as XEN, KVM. We use the component called Nova in the OpenStack that provides computing service to cloud customers and manages VM allocation, image registration, and resource type definition, etc. The Nova controller operates as a front-end machine in the OpenStack and attached Nova compute nodes practically support the computing services for request from cloud service customer. The detailed configuration of our experimental environment is described below in table 1, there are 4 types of requests in the Mapchem application service such as SDF50, SDF100, SDF200, SDF400. The number of sdf file means the required size of computation for chemical compounds. Therefore as the number of sdf file is increased, the workload of Mapchem application is also increased.

Table 1. MapChem based Experimental Environment

Variables of Experimental Environment
- 2 Schemes : Single Request to Single VM(SRSV) Multi Requests to Single VM(MRSV)
- 5 Machines : 1 Nova Controller (16Core 2.4G, 16GB RAM) 4 Nova Compute nodes(16Core 2.4G, 16G RAM)
- O/S and S/W : Ubuntu 12.04, Java 1.6 version
- VM Instance Type : Tiny(Core1, 1GB RAM) Small(Core2, 2GB RAM) Medium(Core4, 4GB RAM)
- Mapchem request type : SDF50, SDF100, SDF200, SDF400

In general, the role of the cloud controller node is to manage the operations between compute nodes and service customers in clouds. Namely, all actual works are processed in Nova computing nodes and the results are reported to the cloud service customers via Nova controller node. In Openstack–based cloud environment, each Nova computing node has the public and private IP addresses, respectively. If we want to access the Nova computing nodes, then we need to do via Nova controller that has the access point to Nova computing nodes or use the public IP address of the nodes.

Figure 4 shows the physical machine specification of the testbed environment in detail. In addition, we applied a test scenario for performance evaluation using three VM types in Nova computing nodes.



	Node 1	Node 2	Node 3	Node 4	Node 5
Functions	Nova Controller, Network, Volume, API, Scheduler and Image Services Nova Compute	Nova Compute	Nova Compute	Nova Compute	MapChem Broker
H/W Specification	HP, Xeon E5620 2.4G, Core16, Mem 16G, HDD 1T				
S/W Specification	Ubuntu 12.04, OpenStack Essex (VM: Ubuntu 12.04, JAVA 1.6, Apache Tomcat 7)				Ubuntu 12.04, JAVA 1.6, Apache Tomcat 7, Mysql
IP address	Eth0 (143.248.6.161)	Eth0 (143.248.6.162)	Eth0 (143.248.6.163)	Eth0 (143.248.6.164)	Eth0 (143.248.6.165)
	Eth1 (192.168.3.1)	Eth1 (192.168.3.2)	Eth1 (192.168.3.3)	Eth1 (192.168.3.4)	Eth1 (192.168.4.62)

Figure 4. Cloud based Experimental Environment

As a result, Figure 5 shows the resource allocation cost of both the proposed MRSV and SRSV schemes. Preliminary, we assume that the cost of MRSV is cheaper than that of MRSV schemes. As mentioned before, in MRSV scheme, since the utilization of VM instance is maximized by processing the multi requests as many as possible in the single resource, the operation cost of resource can be more reduced in comparison of SRSV scheme. Furthermore, as the workload of Mapchem application is decreased, the cost of MRSV is also decreased since the possibility of request group allocation is increased when the resources are not busy.

The performance of MRSV and SRSV that is $\frac{1}{t_{proc}}$ is shown in Figure 6. In our evaluation, surprisingly, the performance of MRSV is better than the SRSV scheme. In this case, since the processing time of application service is considerably small compared to the overhead of VM instance generation, the effect of the generation delay of SRSV is bigger than the effect of separation of resource capacity and context switching delay of MRSV, the performance of SRSV is degraded more compared to the proposed MRSV scheme. However, in general case, the processing time of request is much longer than the generation time of VM instance, and the resulted graph of performance of MRSV can be shown differently in practice.

In Figure 7, we can check the average waiting time of MRSV and SRSV. The average waiting time of SRSV is significantly longer than the MRSV scheme in the entire workload interval. This result is compatible with the result of Figure 7, as mentioned before, since the generation delay of VM instance in SRSV prevails the effect of the performance degradation of MRSV.

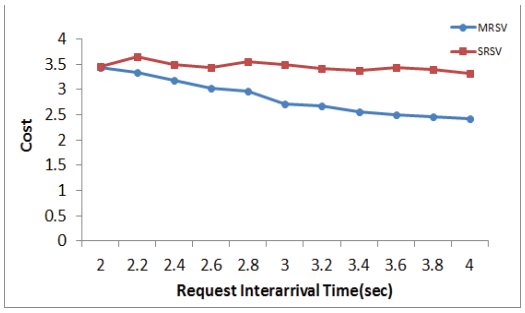


Figure 5. Cost Comparison of Resource Allocation Schemes, MRSV and SRSV, in terms of request interarrival times

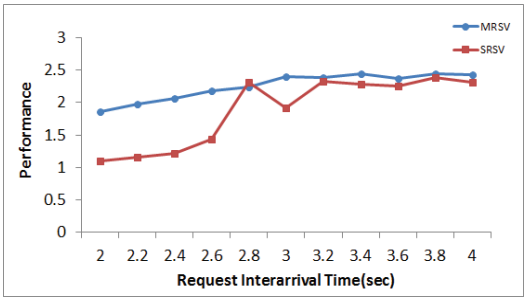


Figure 6. Performance Comparison of Resource Allocation Schemes, MRSV and SRSV, in terms of request interarrival times

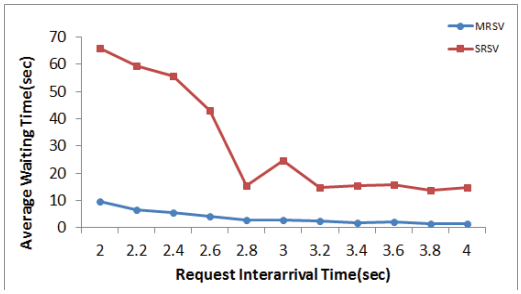


Figure 7. Average Waiting Time of Resource Allocation Schemes, MRSV and SRSV, in terms of request interarrival times

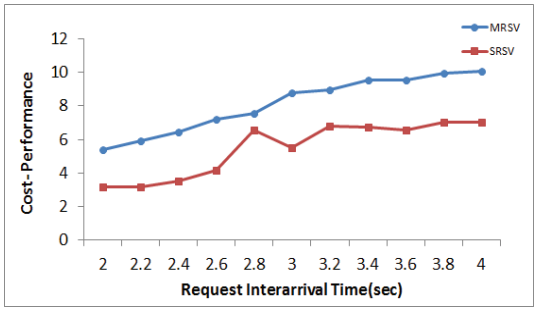


Figure 8. Cost to Performance of Resource Allocation Schemes, MRSV and SRSV, in terms of request interarrival times

Figure 8 presents relative ratio of the cost to performance variation between MRSV and SRSV schemes. The cost to performance can be calculated as $\frac{performance \times (1-w_{cost})}{cost \times w_{cost}}$. In this graph, the weighted factor of cost w_{cost} is 0.5, therefore, the cost

and the performance have the same importance level in our experiment.

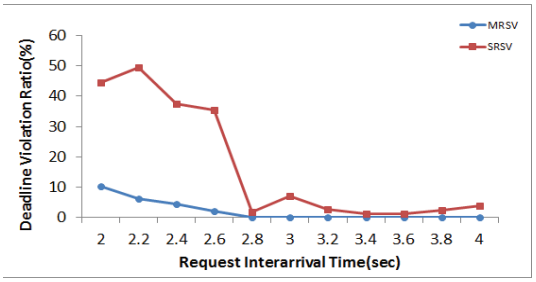


Figure 9. Request Deadline Violation ratio of Resource Allocation Schemes, MRSV and SRSV, in terms of request interarrival times

Finally, Figure 9 shows the deadline violation ratio of both MRSV and SRSV schemes. We assumed that deadline is 1minute and 30sec with consideration of desirable processing time for Mapchem application services. In high workload, the deadline violation ratio of SRSV is much longer than the case of MRSV about 40% because of the generation delay of VM instances. However, after the interarrival time of 2.8sec, the deadline violation ratio of SRSV is decreased dynamically, which is because the generation of VM instance is occurred sparsely.

So far, based on our evaluation graphs, we are able to conclude that our proposed MRSV scheme has improved performance compared to traditional SRSV scheme in cloud environment. In future works, we would investigate and reflect the real distribution of components in whole request from cloud service customer to our system model and find the optimal combination of application services mapping to resources in order to minimize the cost of resource operation and guarantee the acceptable SLA of cloud service customers.

4. CONCLUSIONS

In this paper, we proposed the Adaptive Resource Cloud Framework with group VM allocation scheme in order to minimize the resource operation cost while guaranteeing the acceptable QoS. The proposed group VM Allocation scheme called Multi Requests to Single VM is the key module in our system compared to traditional researches in cloud computing. In MRSV scheme, multi requests are able to be allocated to one VM instance within the acceptable QoS, so the cost of resource operation can be reduced and even the performance can be improved in some special cases. In addition, request mapping to VM instance can be performed efficiently by using simple similar degree function of resource type decision procedure. Through the experimental evaluation, we show that our proposed Group VM Allocation scheme outperforms the Single Request to Single VM scheme in terms of cost, performance such as average waiting time, and the SLA violation ratio. Finally, we conclude that our system can be a reasonable approach to improve the performance of cloud environment in future.

5. ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2012-0020522).

6. REFERENCES

[1] M. Armburst, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing ," University of California at Berkeley, Tech. Rep. EECS-2009-28, 2009.

[2] Y. Han, C. Youn, and D. Kang, "Adaptive Management Framework for Scientific Workflow Applications," in Proceedings of ACM ICUMIC 2012.

[3] C. Hyser, B. McKee, R. Gardner, and B. J. Watson, "Autonomic Virtual Machine Placement in the Data Center," HP Labs Technical Report HPL-2007-189, February 2007.

[4] R. Buyya, C.S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in Proceedings of. 10th IEEE Int. Conference on High Performance Computing and Communications, HPCC 2008, Dalian, China, Sept. 2008

[5] S. Chaisiri, B. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in Proceedings of Services Computing Conference (APSCC) 2009, pp. 103–110.

[6] Mark. C.C.T, Niyato. D, Tham. C.K, and Tham, C.K, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," in Proceedings of AINA, , 2011, pp. 348–355

[7] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in Proceedings of CLOUD'11, Washington, DC, USA, Jul 2011.

[8] Amazon elastic compute cloud. http://aws.amazon.com/ec2/

[9] S. Jian, and J. Xiaojing, "A Study of the Influence of Technical Architecture on the Total Cost of Google Cloud Computing Platform," Telecommunications Science., 1 (2010), pp. 38–44

[10] OpenStack Foundation, http://www.openstack.org/, 2012.