KIPS Tr. Comp. and Comm. Sys.
Vol.3, No.7 pp.219~230 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2014.3.7.219

# Cost Efficient Virtual Machine Brokering in Cloud Computing

Dong-Ki Kang[†] · Seong-Hwan Kim[†] · Chan-Hyun Youn[††]

## ABSTRACT

In the cloud computing environment, cloud service users purchase and use the virtualized resources from cloud resource providers on a pay as you go manner. Typically, there are two billing plans for computing resource allocation adopted by large cloud resource providers such as Amazon, Gogrid, and Microsoft, on-demand and reserved plans. Reserved Virtual Machine(VM) instance is provided to users based on the lengthy allocation with the cheaper price than the one of on-demand VM instance which is based on shortly allocation. With the proper mixture allocation of reserved and on-demand VM corresponding to users' requests, cloud service providers are able to reduce the resource allocation cost. To do this, prior researches about VM allocation scheme have been focused on the optimization approach with the users' request prediction techniques. However, it is difficult to predict the expected demands exactly because there are various cloud service users and the their request patterns are heavily fluctuated in reality. Moreover, the previous optimization processing techniques might require unacceptable huge time so it is hard to apply them to the current cloud computing system. In this paper, we propose the cloud brokering system with the adaptive VM allocation schemes called A3R(Adaptive 3 Resource allocation schemes) that do not need any optimization processes and kinds of prediction techniques. By using A3R, the VM instances are allocated to users in response to their service demands adaptively. We demonstrate that our proposed schemes are able to reduce the resource use cost significantly while maintaining the acceptable Quality of Service(QoS) of cloud service users through the evaluation results.

**Keywords : Cloud Broker, Virtual Machine Allocation, Adaptive Resource Management, Resource Reservation**

# 가격 효율적인 클라우드 가상 자원 중개 기법에 대한 연구

강 동 기[†] · 김 성 환[†] · 윤 찬 현[††]

## 요 약

클라우드 컴퓨팅 환경에서, 클라우드 서비스 사용자는 클라우드 자원 제공자로부터 가상화된 컴퓨팅 자원을 사용할 시간만큼 구매하여 할당받는다. 일반적으로 아마존, 고그리드 및 마이크로소프트와 같은 대형 클라우드 자원 제공자들은 자원 과금 정책을 온디맨드와 예약형 기반 가상 자원의 두 가지로 구분하여 제공한다. 예약형 기반 가상 자원은 상대적으로 장기간 할당을 가지므로 단위 시간당 자원 사용 비용이 온디맨드 가상 자원과 비교하여 더 저렴하다. 이러한 과금 정책 특성을 기반으로 클라우드 서비스 사용자의 서비스 요구 사항을 고려하여 적절한 자원 할당을 수행함으로써 클라우드 서비스 제공자는 자원 할당 비용을 효과적으로 절감할 수 있다. 이를 위해서, 기존의 가상 자원 할당 기법들은 서비스 사용자의 요구사항 특성을 미리 예측하여 최적의 자원을 할당하는 방법들을 제안하였다. 그러나 실세계에서는 다양한 클라우드 서비스 사용자가 존재하고 서비스 요구사항이 동적으로 변하기 때문에 정확한 예측을 하기 어려우며, 최적화된 할당을 위한 연산 시간이 추가 오버헤드가 되어 자원 관리 성능을 떨어뜨릴 수 있다. 이를 해결하기 위해, 본 논문에서는 적응적 자원 할당 기법을 제안하여 요구사항 예측 및 최적화 기법을 수행하지 않으면서도 서비스 요구사항에 효과적으로 대응하여 자원을 제공할 수 있도록 한다. 실험 결과를 통해 제안된 기법이 자원 사용 비용을 크게 절감하면서도 클라우드 서비스 사용자의 QoS를 만족함을 보인다.

**키워드 : 클라우드 브로커, 가상 머신 할당, 적응적 자원 관리, 자원 예약**

※ 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).
※ This research also was supported by 'The Cross-Ministry Giga KOREA Project' of The Ministry of Science, ICT and Future Planning, Korea[GK13P0100, Development of Tele-Experience Service SW Platform based on Giga Media].
† 준 회 원 : 한국과학기술원 전기및전자공학과 박사과정
†† 종신회원 : 한국과학기술원 전기및전자공학과 교수

Manuscript Received : May 2, 2014
First Revision : June 26, 2014
Accepted : June 27, 2014
* Corresponding Author : Chan-Hyun Youn(chyoun@kaist.ac.kr)

## 1. Introduction

Recently, in the emerging cloud computing environment, physical computing nodes in the datacenter are provided to the cloud service users as outsourcing virtual resources. Cloud service users do not need considering any setup process for the erection and maintenance of computing resources. The major benefits of cloud computing are lower resource use cost (for resource installation and management) and saving resource provisioning time. Cloud service users just purchase virtualized resources from the certain cloud resource provider (or service provider) whenever their job is required to process and, return the allocated resources to the cloud resource provider after their job is finished. Typically, most cloud resource providers such as Amazon EC2 and GoGrid support three types of resource payment plans to the cloud service users, i.e., on-demand VM, reserved VM and spot VM instances [1, 2, 3].

First, on-demand VM(OVM) instance is provided to users based on the short-term basis whenever they need to process their jobs. The billing time unit of on-demand VM instance has not fine-grained granularity, which is the VM instances are provided to users on a predefined hourly basis (or daily one) from the cloud resource provider. In response to the actual processing time of job, the allocation time of on-demand VM instance is increased proportionately. The cost of on-demand VM instance is more expensive than other VM instance types, so cloud service users prefer to purchase the on-demand VM instance when they need to process the fluctuated and unpredictable job demands.

Second, in contrast to the on-demand VM instance, reserved VM(RVM) instance is allocated to cloud service users on long-term basis regardless of the actual processing time of submitted job (i.e. minimum billing time unit of reserved VM instance is much longer than the on-demand VM instance, it is commonly from a month to an year). Cloud resource providers grant the advantages in view of the resource use cost to cloud service users which using reserved VM instance since the constant benefit can be guaranteed quite a while. Therefore the resource use cost per certain time unit of the reserved VM instance is much cheaper than the one of on-demand VM instance.

Finally, spot VM instance is provided to users based on the dynamic pricing policy in contrast to the cases of on-demand and reserved VM instance those are based on the fixed pricing policy [14]. Cloud service users set the bid price in advance, and the spot VM instance is allocated to the user if the cost of spot VM instance (in general, the cost of spot VM instance is cheaper than the on-demand and even the reserved VM instance) is under the bid price of the user. The cost of spot VM instance is adjusted dynamically depending on the demand of users. When the cost of spot VM instance is more expensive than the maximum bid price of user, then the allocated spot VM instance is released from the user without any notification and it is allocated to the another user who proposes the higher bid price than the cost of spot VM instance. That is, the cloud service user uses the spot VM instance lower resource use price at the expense of the reliability of job processing.

Consequently, the resource budget can be differentiated according as the VM instance allocation plans, therefore the cloud service provider should set up the accurate plan with the consideration of cloud service users' request demand pattern and trend.

In this context, previous studies about VM instance allocation have been focused on the determination the number and the ratio of each VM instance type by using optimization techniques with several prediction methods [1, 4, 5]. However, there are several problems to be addressed in them.

First, they have assumed that the demand of cloud service users obeys the uniform or normal distribution in their experimental evaluation. However, in the real world, the mixed demands from cloud service users are not based on the simple distribution model and they are represented by the complex and dynamic shape [12]. In addition, since various cloud service users synthesize demands, it is difficult to predict the actual demands exactly in advance; therefore, the evaluation results of existing researches with simple demand distribution might be inconsistent with the real statistics.

Second, in optimization approaches such as integer or linear programming, the process to obtain a solution can be computationally intensive and it is infeasible to be applied to the current cloud computing system when the size of required factors and constraints are large.

In order to solve above problems, in this paper, we propose the adaptive VM allocation schemes to reduce the

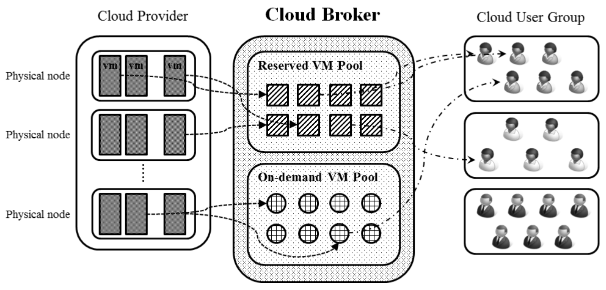


Fig. 1. Cloud Brokering System

resource use cost significantly while guaranteeing the acceptable QoS of cloud service users without the demand prediction and optimization techniques. To achieve the simplicity, we just considered both the on-demand VM instance and reserved VM Instance with the exception of the spot VM instance. Proposed schemes are adopted to the cloud brokering system which interconnects the cloud service provider and the group which is comprised of cloud service users who have a common jobs and shared contents. Through the various experimental results from our implemented cloud brokering system, we demonstrate that the proposed schemes are able to correspond to the uncertainty of the demand effectively and guarantee the required QoS of users while reducing the resource use cost significantly.

This paper is organized as follows: In section 2, cloud brokering system and the related environment are introduced. The adaptive VM allocation schemes are presented in section 3. Section 4 describes the experimental environment, cloud brokering testbed and shows the evaluation results. Finally, conclusions are discussed in section 5.

## 2. Cloud Brokering System and Assumptions

The cloud brokering system between the cloud service provider and cloud user groups is shown in Figure 1. It has resource pools to provision on-demand and reserved VM instances in order to provide the proper VM instances to cloud service users efficiently. It purchases reserved and on-demand VM instances responds to user demands. Both reserved VM instance and on-demand VM instances are able to be shared by cloud service users who are in same user group. The user group stands for the gathering of users who have common jobs and contents to process. For example, we can take the smart

Table 1. GoGrid, pricing policy [3]

| Product | RAM | Cores | Hourly | Monthly Prepaid | Annual Prepaid |
|---|---|---|---|---|---|
| X-Small | .5GB | 0.5 | $0.04(1) | $18.13 (0.63) | $181.25 (0.52) |
| Small | 1GB | 1 | $0.08(1) | $36.25 (0.63) | $362.50 (0.52) |
| Medium | 2GB | 2 | $0.16(1) | $72.50 (0.63) | $725 (0.52) |
| Large | 4GB | 4 | $0.32(1) | $145 (0.63) | $1,450 (0.52) |
| X-Large | 8GB | 8 | $0.64(1) | $290 (0.63) | $2,900 (0.52) |
| XX-Large | 16GB | 16 | $1.28(1) | $580 (0.63) | $5,800 (0.52) |
| XXX-Large | 24GB | 24 | $1.92(1) | $870 (0.63) | $8,700 (0.52) |

care scenario in the hospital. If the medical attendant would like to treat a patient in order to diagnose his brain tumor in the hospital, the process is performed as a collaboration among the MRI(Magnetic Resonance Imaging) medical specialist who takes a photo of the patient's brain, the image developer who processes the MRI result based raw data to construct the 3D brain images and the brain medical specialist who analyzes the brain status. In such an this environment, allocated VM instances do not have to be dedicated to certain users, but rather shared sequentially by users as many as possible preferably. In previous environment in which cloud service users each process their jobs independently, the reserved VM instance allocation is commonly irrational since the resource dissipation is occurred due to the fact that the allocation time of reserved VM instance is much longer than the required processing time of jobs. However, in above environment in which the service collaboration is established among users, the multiple jobs of distributed users can be aggregated to the common VM instance. In this case, the reserved VM allocation is a reasonable approach able to reduce the resource use cost since the utilization ratio of reserved VM instance reaches to the enough level.

In order to calculate the utilization of reserved VM instance depending on the user demands, we denote the number of cloud service users is $n$, and the probability of demand occurrence from each user is $p$, then the probability of that the demand is occurred once at least is as follows,

$$C_n^n p^n + C_n^{n-1} p^{n-1}(1-p) + C_n^{n-2} p^{n-2}(1-2) + \dots + C_n^1 p(1-p)^{n-1} = 1-(1-p)^n \quad (1)$$

where $(1-p)^n$ is the probability that the demand is not occurred at all. As the user number $n$ is increased the demand occurrence probability is closed to 1. This is consistent with that the reserved VM instance allocation is useful to the environment in which multiple cloud service users share the allocated resource alternatively.

In the table 1, the resource use cost for both of on-demand VM instance and reserved VM instance in the representative cloud service provider, GoGrid is shown. The number in the each blanket represents the normalized values by the standard of hourly cost. That is to say, if we assume that the resource hourly use cost for on-demand VM instance is 1, then the monthly prepaid cost for reserved VM instance is 0.63, and the annual prepaid cost is 0.52 in all the resource type. According to above table, as the duration of reservation is increased, the resource use cost per time billing unit (e.g., hourly) is decreased. It is important to choose the proper reservation type to maximize the benefit and avoid unnecessary resource dissipation since once the allocated reserved VM instance can not be released for its quite some billing time. However, it is difficult to select optimum reservation plan at all times in practical resource management, the resource allocation scheme is required which is efficient even under the failure of the reservation plan.

## 3. Adaptive Virtual Machine Allocation Schemes

As described in the earlier section, previous researches of resource allocation have been focused on the theoretical optimization approaches for determining the proper ratio of both the on-demand VM instance and reserved VM instance in order to guarantee the user's required QoS and minimize the resource operation and maintenance cost of cloud service provider. However, optimization techniques such as integer or linear programming requires the high processing complexity and the long processing time, it is difficult to adapt these approaches to the current cloud resource management system in practice. Moreover, since the prediction based on the demand history has to be performed to do optimization techniques, if the demand of users is fluctuated and the request pattern has not fixed shape, then the accurate prediction cannot be achieved. Finally we might not be available to guarantee the required QoS from cloud service users. In contrast to the previous approaches, our proposed adaptive VM allocation scheme does not require the additional prediction scheme and is feasible to current cloud computing system since its processing time is acceptable. To provide against the resource adjustment failure for demand we propose the novel scheme solving resource under-provisioning and over-provisioning. These schemes are called A3R(Adaptive 3 Resource management) and they are described as follows.

### 3.1 Adaptive Recycle

In Adaptive Recycle scheme, the running on-demand VM instance (i.e. is more expensive than reserved VM instance) is not released immediately after its running job processing is completed, is continuously maintained until its original termination time in order to accept additional requests from other cloud service users. To do this, on-demand VM instance is stand by in the resource pool temporarily. Since the surplus time of running on-demand VM instance is made the best use of another user's request the resource dissipation can be reduced. The structure of Adaptive Recycle process is shown in figure 2. When the request $aReq(a,r)$ in which $a$ is a task (or

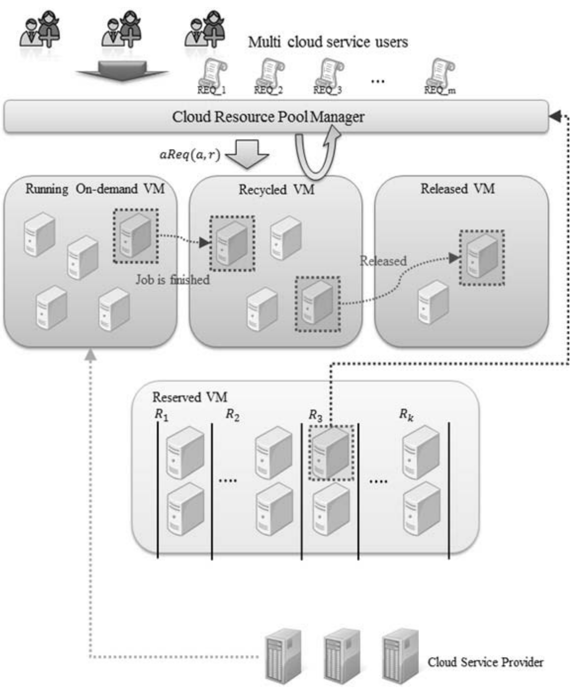


Fig. 2. Adaptive Recycle of on-demand VM instance

task processing time) and $r$ is a required resource type from cloud service user is submitted to the cloud brokering system, the resource pool manager checks whether the available reserved VM instance exists or not to process the submitted job. If there are not available reserved VM instance which is consistent with required VM instance type, then the manager tries to find another available VM instance in the recycled VM pool.

If the manager cannot find the available VM instance even in the recycled VM pool, then newly on-demand VM instance is generated to process the submitted job. After the job processing is finished and the complete message is noticed from the cloud service user to the cloud broker system, then allocated on-demand VM instance is not released immediately but added to the list of recycled VM pool to be reused. Finally, when the allocation time of recycled VM instance is reached to its expiration time, then it is released. The more description is entered into details as follows,

Step 1 : The request description from the cloud service user includes the required VM instance type, and the job processing time like as $REQ_i(VM_k, Task_i)$ where $REQ_i$ is the request of the cloud service user $i$, $VM_k$ is the VM instance type $k$, and $Task_i$ is the job processing time in the required resource. First, when the cloud resource pool manager accepts the request from the cloud service user, the cloud resource pool manager searches the available VM instance having resource type among the reserved VM instance set preferentially. If the cloud resource pool manager finds the proper VM instance, then it provides the VM instance to the cloud service user and go step 4. Otherwise, go step 2.

Step 2 : The cloud resource pool manager searches the available recycled VM instance consistent with required resource type $k$ secondly. If the cloud resource pool manager finds the suitable reserved VM instance, then it provides the VM instance to the cloud service user and go step 4. Otherwise, even in recycled VM instance pool if it cannot find the proper resource, then it should generate the newly on-demand VM instance which has the VM resource type $k$ from the cloud service provider. Go step 3.

Step 3 : The cloud resource pool manager calls for the on-demand VM instance to the cloud service provider when there are no idle (with no running job) reserved VM and recycled VM instances. The generated on-demand VM instance is inserted to the list of running on-demand VM instances and provided to the cloud service user so as to process the $REQ_i$. Now go step 4.

Step 4 : When the request in the running on-demand VM instance is finished, then the VM instance is removed from the list of on-demand VM instance and inserted to the list of recycled VM instance. All the VM instances in the recycled VM instance pool are not released immediately but maintained until reaching to their originally scheduled release time. If the ongoing request is not finished until the next allocation point of the recycled VM instance and it is exceed the originally allocated duration of the VM instance, then the lifetime of the recycled VM instance would be extended. The available duration of the waiting VM instance in the list of recycled VM instance is as follows,

$$t_{remain}(r(REQ_i)) = t_{life}(r(REQ_i)) - \\ (t_{proc}(REQ_i, r(REQ_i)) + t_{gen}(r(REQ_i))) \quad (2)$$

where $r(REQ_i)$ is the VM instance which is allocated to process the request of cloud service user $i$, $t_{remain}(r(REQ_i))$ is the remaining time of the VM instance which processed $REQ_i$, $t_{life}(r(REQ_i))$ is the whole allocation time of VM instance, $t_{proc}(REQ_i,\ r(REQ_i))$ is the processing time of the $REQ_i$ on the resource $r(REQ_i)$, and $(t_{gen}(r(REQ_i)))$ is the VM instance generation (or startup) time of the resource $r(REQ_i)$.

That is, as $t_{remain}$ is increased, the dissipation ratio of the VM instance is also increased. Therefore, in traditional schemes, as $t_{remain}$ is increased, the dissipation problem of resource becomes more serious. In the case of traditional schemes, in order to support $n$ requests, $n$ VM instances are required. That is, in regardless of the actual size of requests, the VM instance with fixed allocation time is allocated to cloud service user.

This is a quite unreasonable dissipation. In our proposed scheme, we can minimize the number of newly allocated VM instances unnecessarily by making the best use of the existing VM instances simply as described earlier. Especially the effectiveness of our scheme is increased when the size of request tends to be small piece. If the job processing time exceeds the originally allocation time of VM instance, then the life duration of VM instance is extended to accomodate the running job automatically.

In addition, in our scheme we can reduce the VM instance generation overhead by reducing the number of newly generated VM instance. Multi requests can be processed in the single VM instance. In this case, the processing time is calculated as follows,

$$t_{proc}(REQ_{i\in I}, VM_k) = \sum_{i\in I} t_{proc}(REQ_i, VM_k) + t_{gen\_delay} \tag{3}$$

$$t_{proc}(REQ_{i\in I}, VM_k) = \sum_{i\in I} t_{proc}(REQ_i, VM_k) + N(I)\, t_{gen\_delay} \tag{4}$$

The equation (3) represents the total processing time of the request set $I$ which requires the identical resource type $k$ in our scheme, and the equation (4) represents same value in traditional schemes. In contrast to traditional scheme in which the total generation delay is proportional to the number of requests, in our scheme the total generation delay is occurred just only once.

## 3.2 Adaptive Replacement

All the VM instances allocated to the cloud service users have their own VM flavor (i.e. CPU, Memory, Storage, etc.) type. It is important to purchase the proper amount of reserved VM instance with the prediction of user's demand about each flavor type. Therefore, the broker should either allocate on-demand VM instance newly or provision additional reserved VM instance when the under-provisioning is occurred by the imprecise prediction of demand. In this case, it is reasonable to utilize idle reserved VM instances having another flavor type to accommodate excess demands rather than either the on-demand VM instance allocation or the additional provisioning in case the demand fluctuation is temporary.

To solve this problem, we propose Adaptive Replacement scheme in which, the VM instance with higher flavor capacity can be allocated to user as a substitute instance instead of the VM instance having lower flavor capacity. For example, the flavor capacity of arbitrary reserved VM instance is $CAP_1$ and the other flavor capacity of arbitrary VM instance is $CAP_2$, if $CAP_1 \geq CAP_2$, the VM instance with the capacity can be substitute for the other VM instance with the capacity $CAP_1$. In opposite case, the VM instance with $CAP_2$ can be substitute for the VM instance with $CAP_1$.

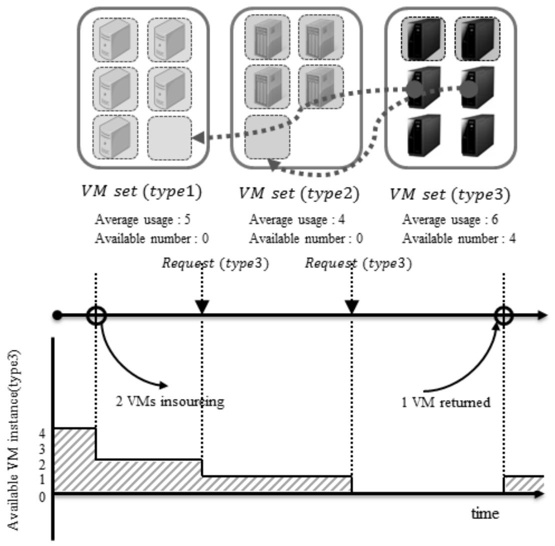


Fig. 3. Adaptive Replacement of reserved VM instance

Figure 3 shows the procedure of the Adaptive Replacment scheme. As shown in Figure 3, we assume that there are three prepared resource flavor types such as $RT_1$, $RT_2$, $RT_3$ in order to provide suitable resources to users according to their request types ($RT_1 < RT_2 < RT_3$). If the average number of resource demand for flavor types $RT_1$, $RT_2$, $RT_3$ is 5, 4, and 6, then the provisioning number of instance for each VM flavor types is also determined 5, 4, and 6, respectively. At the certain epoch, suppose that the transient number of resource use for VM flavor types $RT_1$, $RT_2$, $RT_3$ is 6, 5, and 2, respectively. This means that the demands for flavor types $RT_1$, $RT_2$ exceeds their originally provisioned number of VM instances. However, it is not desirable solution to increase the provisioning number of reserved VM instance since the high resource demand is not persistent but transient, which is returned to the average amount of demand. Also, the newly allocation of on-demand VM instances to support excessive demands is either an unsuitable alternative due to the high price of the instance. In this case, by our proposed Adaptive Replacement scheme, 2 idle VM instances for flavor type $RT_3$ can be supported to process excessive demands instead of increasing the provisioning number of VM instances for flavor types $RT_1$, $RT_2$. This scheme is allowable since the capacity of VM instance for flavor type $RT_3$ is larger than VM instances for flavor types $RT_1$, $RT_2$. To enable the Adaptive Replacement scheme, we should know the expected number of arrived requests during from the epoch of VM instance

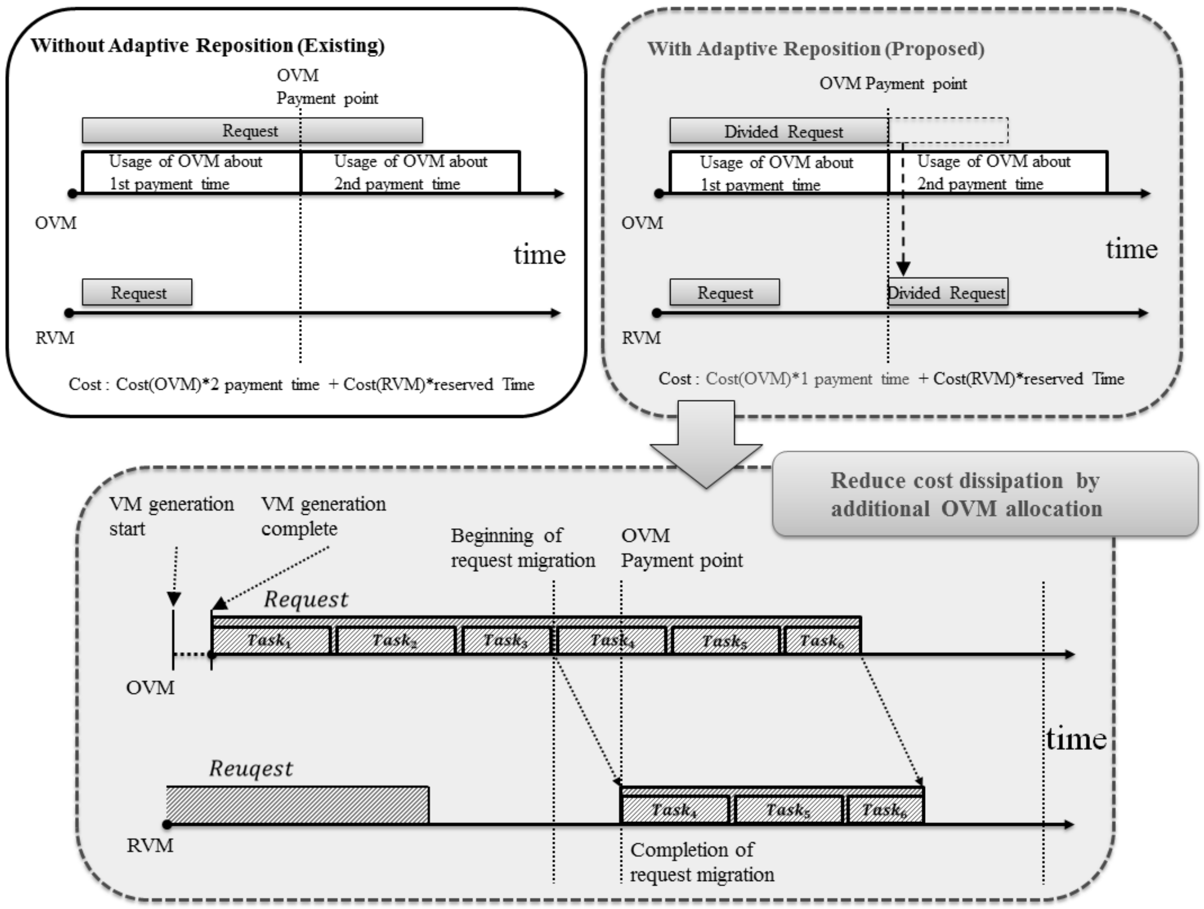

Without Adaptive Reposition (Existing)
OVM Payment point
Request
Usage of OVM about 1st payment time
Usage of OVM about 2nd payment time
OVM
time
Request
RVM
Cost : Cost(OVM)*2 payment time + Cost(RVM)*reserved Time
With Adaptive Reposition (Proposed)
OVM Payment point
Divided Request
Usage of OVM about 1st payment time
Usage of OVM about 2nd payment time
OVM
time
Request
Divided Request
RVM
Cost : Cost(OVM)*1 payment time + Cost(RVM)*reserved Time
Reduce cost dissipation by additional OVM allocation
VM generation start
VM generation complete
Beginning of request migration
OVM Payment point
Request
Task1 Task2 Task3 Task4 Task5 Task6
OVM
time
Reuqest
Task4 Task5 Task6
RVM
Completion of request migration

Fig. 4. Adaptive Reposition from on-demand VM to reserved VM instance

insourcing to returning. If the expected number of request arrival during that period is more than the number of available VM instances, then idle VM instances are not available to be aided for other VM instance types. By using the Adaptive Replacement scheme, we can increase the utilization of reserved VM instances and cover the payment for additional resource allocation due to transitory rise in resource demands.

### 3.3 Adaptive Reposition

Adaptive Reposition scheme is conducted between running on-demand VM instance and reserved VM instance. In this scheme, it is allowable to migrate the running job process from the on-demand VM instance to the available reserved VM instance if the job processing time exceeds the inherent allocation time of on-demand VM instance. Suppose that the inherent allocation time of on-demand VM instance is an hour, and the required processing time of the running job is an hour and 10 minutes, then, the allocation time of the instance has to be extended to two hours. Consequently, the unnecessary 50 minutes is dissipated due to just 10 minutes that is, we have to pay double prices of hourly on-demand VM instance. This is not advisable outcome for us.

To overcome this problem, the job should be processed on the on-demand VM instance during an hour, and migrated to the available reserved VM instance after the inherent release time of on-demand VM instance.

Then the extension of allocation time is not required and at the same time, the dissipation of the provisioned reserved VM instance is minimized. We called this operation request migration.

Actually to do Adaptive Reposition scheme, there are two necessary conditions. First, the capacity of destination reserved VM instance is larger than or equal to the source on-demand VM instance. Second, the migration of running job has to be conducted before inherent allocation time of reaching to the end point of instance allocation time due to the additional time caused by the processing and communication overhead.

Table 2. Experimental Environment

| Components | Values |
|---|---|
| Testbed pecification (5 machines) | 1 Nova controller<br>4 Nova compute nodes<br>(16 core, 2.4Ghz, 16GB RAM)<br>NIC cards : 1Gbps |
| S/W | Ubuntu 12.04, Java jdk 1.6 |
| VM flavor types | SMALL (core 1, 1GB RAM)<br>MEDIUM (core 2, 2GB RAM)<br>LARGE (core 4, 4GB RAM) |
| Application | MapChem application<br>(sdf 100, 200, 400) |
| Interarrival time of request | 2, 1.9, 1.8, 1.7, 1.6, 1.5, 1.4, 1.3, 1.2, 1.1(sec) |
| Schemes | with A3R<br>without (w/o) A3R - stochastic VM provisioning [13] |
| VM instance price | (RVM)<br>SMALL - 0.04$/hour(36.25$/month)<br>MEDIUM - 0.09$/hour(72.50$/month)<br>LARGE - 0.19$/hour(145$/month) |
| | (OVM)<br>SMALL - 0.36$/hour<br>MEDIUM - 0.15$/hour<br>LARGE - 0.32$/hour |
| Performance metrics | Request completion time<br>(queueing + service time)<br>Resource use cost<br>Utilization of VM instances |

Figure 4 represents the comparison of procedures of both the traditional VM allocation scheme and the proposed Adaptive Reposition scheme. The left side of figure represents the traditional VM allocation scheme (e.g. without Adaptive Reposition) in which, running job is fixed on the on-demand VM instance. Since the job processing time exceeds the first payment duration the inherent allocation time of VM instance has to be extended to twice. Although there is an available reserved VM instance, the additional allocation of on-demand VM instance is occurred. That is, the resource use cost is increased and the utilization of provisioned reserved VM instance is decreased. The right side of figure 4 shows the proposed task re-allocation scheme (with Adaptive Reposition) in which, the running job is migrated from the on-demand VM instance to the reserved VM instance. In contrast to the traditional scheme, if the job processing is not finished and the number of available reserved VM instance is one at least at the first payment point, then the running job is migrated to the target reserved VM instance without any extension of the on-demand VM instance.

We denote that the base payment time is $t_{alloc}$, and the overheads of processing and communication for migration are $t_{comp}$, $t_{comm}$, respectively. Then, the conditions for request migration are as follows,

$$(n+1)t_{alloc} < t_{proc} < (n+2)t_{alloc}, n = 0, 1, 2, \ldots \tag{5}$$

$$n_{avil}\left(RVM_{VM(r(REQ_i))}, epoch_{mig(REQ_i)}\right) > 0 \tag{6}$$

where $n$ is the number of extension of on-demand VM instance allocation. The equation (8) represents the number of available reserved VM instances is one at least when the job on the running on-demand VM instance is migrated.

Additionally, we add one more condition to the request migration. The request migration leads to additional processing and communication overheads increasing not only the undesirable additional delay but also resource use cost. Therefore, if the required cost for request migration is larger than the case without request migration, then it is advisable to keep the running job on the on-demand VM instance. That is, if the completed part of the ongoing job is significantly large compared to the amount of remaining part which is small, then it would be better to process the running job on the on-demand VM instance than the request migration.

The equation (7) represents the constraint for request migration in the Adaptive Reposition scheme. The left clause derives the resource use cost with the request migration and the right clause draws the resource use cost without the request migration. In addition, to maximize the cost efficiency by the request migration, the migration epoch should be coincided with the payment

$$\begin{aligned}&\left(\left\lceil \frac{\sum_{j=1}^{m} t_{proc}\left(task_i^j\right)}{t_{alloc}} \right\rceil t_{alloc} + t_{comp}\left(task_i^{\forall j \in \{m+1, m+2, \ldots, n\}}\right) + t_{comm}\left(task_i^{\forall j \in \{m+1, m+2, \ldots, n\}}\right)\right) C_{on}\left(r\left(REQ_i\right)\right) \\ &< \left\lceil \frac{\sum_{j=1}^{m} t_{proc}\left(task_i^j\right)}{t_{alloc}} \right\rceil t_{alloc} C_{on}\left(r\left(REQ_i\right)\right)\end{aligned} \tag{7}$$

point of the on-demand VM instance. As the difference between them is increased, then the cost inefficiency is also increased. Therefore, the making the right decision about determining the value $m$ that means the migration epoch is the important objective in the Adaptive Reposition scheme. We will study this issue in more detail later on.

## 4. Experimental Environment

In section 4, we described the cloud test bed in order to evaluate the performance of our proposed Adaptive VM instance allocation scheme in the cloud brokering system. We establish the open-source based cloud platform called Openstack [10] that supports a variety of hypervisors such as XEN, KVM, and etc. The nodes playing a role of computing called Nova in Openstack are support to not only provide a computing service to cloud service users but also manage a resource provisioning, VM allocation, image registration and networking management. To evaluate the performance of our proposed cloud brokering system and the adaptive VM allocation scheme, the application called ChemApp is processed on the VM instance. ChemApp is an application in which the new medicine is developed based on the traditional drug lists in the database. If we want to generate the certain medicament, then we can find the related existing medicines that are similar to or replaceable the target medicine. Especially, the astronomical cost for development of novel medicines is dimished significantly by Drug Repositioning of ChemApp.

The input data for ChemApp is expressed in the file type ‘sdf’ and attributes in sdf file contains the detailed information of medicines. Generally, as the number of medicine entities in the sdf is increased, the processing time for sdf is also increased. In this study, we do experiments based on ChemApp with sdf 100, 200, 400 (e.g. the number of sdf means the number of including medicine entities).

Our cloud brokering system and the adaptive VM allocation scheme are implemented based on Java JDK 1.6 version. Especially, the cloud client module communicates with the cloud brokering system through RESTful web service that is an emerging Resource Oriented Architecture (ROA) paradigm. In addition, there are 3 VM instance flavor types for job processing which are allocated to cloud service users in respond to their application type. VM instance pricing model for each VM flavor type is according to the GoGrid pricing model. As mentioned earlier, the resource use cost of reserved VM instance is more expensive than the on-demand VM instance. Detailed experimental parameters are shown in above the table 2. In part of ‘Scheme’, we compared our proposed A3R scheme to without(w/o) A3R scheme which is the stochastic integer programming based VM provisioning [13].

## 5. Experimental Results

Figure 5 shows the curves of cost comparison between the VM allocation with A3R scheme and w/o A3R scheme. As described earlier, by using A3R scheme, the number of newly generated on-demand VM instance is decreased, existing on-demand VM and reserved VM instance tends to be utilized as much as possible, therefore the resource usage cost is reduced. Additionally, the cost curve of w/o A3R is increased exponentially as the workload from cloud service users is increased (The longer request interarrival time means the lower workload of users).

Contrast to the w/o A3R case, the cost curve of A3R is fluctuated regardless of the workload level. In the A3R approach, the resource use cost is decided based on how much the difference between the actual job processing time and the payment duration of on-demand VM instance is occurred regardless of the workload level. Consequently, the resource use cost of A3R is decreased about 70% on average compared to the w/o A3R throughout all the interval points.

In figure 6, the result curves of request completion time of both the A3R scheme and w/o A3R scheme are drawn in respond to the VM request interarrival time. In this result, the processing performance in the case of A3R is better than the case of w/o A3R throughout all the VM request interval points. By minimizing the generation of newly on-demand VM instances, the number of VM instance startup overhead is decreased, therefore the whole processing time of request is also decreased. The request completion time of w/o A3R is longer than the case of A3R about 300~500 seconds on average, that is the performance achievement of our proposed A3R results in 15% improvement compared to existing approach.

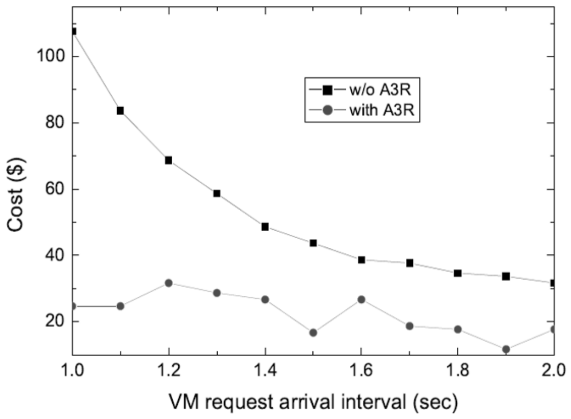


Fig. 5. Cost comparison of VM management schemes with A3R and without A3R

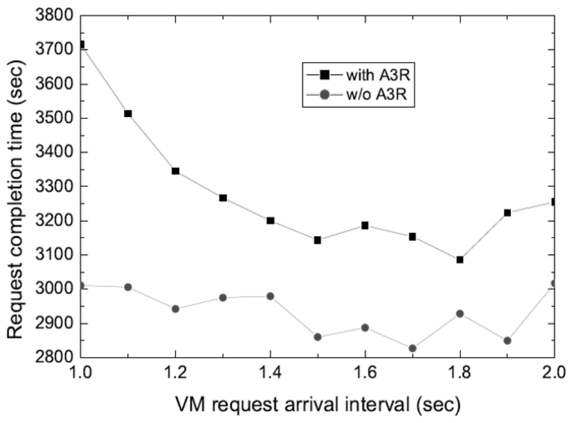


Fig. 6. Request completion time of VM management schemes with A3R and without A3R

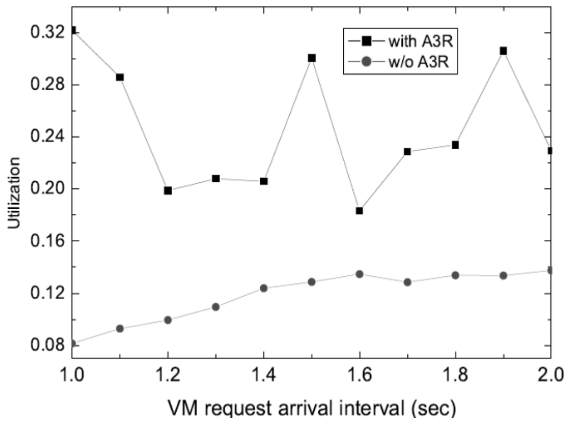


Fig. 7. Resource utilization of VM management schemes with A3R and without A3R

In w/o A3R scheme, some submitted jobs might be stand by due to the startup overhead of on-demand VM instance generation so the undesirable queueing delay is occurred. This leads to degrade the whole performance of the job processing. But in the case of A3R scheme, newly on-demand VM instance is not always generated and allocated to cloud service user whenever the request is submitted to the cloud brokering system, but existing running VM instances tend to allocated to the user, therefore the startup overhead is decreased. However, in general case, this result might be different to our case. If the ratio of startup time to whole job processing time is negligible, then the performance of A3R is similar to the case of w/o A3R. Figure 7 shows the curves of resource utilization of both A3R scheme and w/o A3R scheme. In A3R, the secondhand on-demand VM instances and idle reserved VM instances are utilized to process requests from cloud service users as many as possible, therefore the resource utilization is increased.

However, in this graph, the curve of A3R is fluctuated at certain interval points. This is because that actual job processing time of request is not consistent with the original allocation time of VM instance at those points. As the difference between the actual job processing time and the original resource allocation time is increased, then the dissipation of resource occupation is also increased proportionally.

## 6. Conclusion

In this paper, we propose the adaptive VM allocation scheme in cloud computing in order to solve the problems in traditional VM management scheme with the prediction approaches and optimization techniques. Traditional predictions are inappropriate to foresee the cloud service users' requests including various applications in advance, so they result in undesirable performance. In addition, the required processing time for optimization is unacceptable; it is difficult to apply them to current cloud computing system. Our adaptive resource allocation scheme based on the two payment plans including on-demand and reservation does not need any prediction approaches and optimization algorithm. Moreover, it reduces the resource use cost about 70% and increases the utilization of allocated VM instances about 20% on average compared to existing schemes by using Recycle, Replacement and Repositioning policies. We conclude that our proposed schemes are able to contribute to the improved performance of the future cloud computing system.

## Reference

[1] S. Chaisiri, B. -S. Lee, and D. Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," *IEEE Trans. Services Computing*, Vol.5, No.2, pp.164-177, Apr., 2012.

[2] Amazon EC2 (2013), http://aws.amazon.com/ec2/

[3] GoGrid (2013), http://www.gogrid.com/

[4] S. Chaisiri, B. Lee, and D. Niyato, "Optimal Virtual Machine Placement across Multiple Cloud Providers," *Proc. IEEE Asia-Pacific Services Computing Conf. (APSCC)*, 2009.

[5] C. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary Optimal Virtual Machine Placement and Demand Forecaster for Cloud Computing," *Proc. IEEE Int'l Conf on Advanced Information Networking and Apps. (ICAINA)*, 2011.

[6] S. Son, and K. Sim, "A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations," *IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol.42, No.3, pp.713-728, June, 2012.

[7] R. Buyya, C.S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilites," *Proc. IEEE Int'l Conf on High Performance Computing and Communications. (HPCC)*, 2008.

[8] J. Simarro, R. Vozmediano, R. Montero, and I. Llorente, "Dynamic Placement of Virtual Machines for Cost Optimization in Multi-CLoud Environments," *Proc. IEEE Int'l Conf on High Performance Computing and Simulation. (HPCS)*, 2011.

[9] R. Jeyarani, N. Nagaveni, and R. Vasanth Ram, "Design and implementation of adaptive power-aware virtual machine provisioner(APA-VMP) using swarm intelligence," *Elsvier, Future Generation Computer Systems*, Vol.28, Issue.5, pp.811-821, May, 2012.

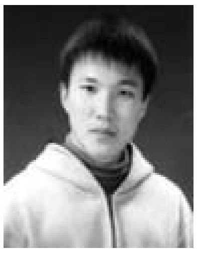
[10] OpenStack Foundation, http://www.openstack.org/, 2013.

[11] D. Kang, S. Kim, Y. Ren, B. Kim, W. Kim, Y. Kim, C. Youn, and C. Jeong, "Enhancing a Strategy of Virtualized Resource Assignment in Adaptive Resource Cloud Framework," *Proc. ACM Int'l Conf on Ubiquitous Information Management and Communication. (ICUIMC)*, 2013.

[12] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Szxena, "Intelligent Workload Factoring for a Hybrid Cloud Computing Model," *Proc, Congress on Services-I,* 2009.

[13] S. Chaisiri, B. Lee, and D. Niyato, "Optimal Virtual Machine Placement across Multiple Cloud Providers," *Proc, IEEE Int'l Conf on Asia-Pacific Services Computing Conference (APSCC),* 2009.

[14] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic Resource Allocation for Spot Markets in Cloud Compputing Environments," *Proc, IEEE Int'l Conf on Utility and Cloud Computing,* 2011.



### Dong-Ki Kang

e-mail : dkkang@kaist.ac.kr

He received a M.S degree in Dept. of Computer Engineering from Chonbuk, National Univ, Jeonju, Korea in 2011. He is currently working toward the Ph.D. degree in Electrical Engineering at Korea Advanced Institute of Technology (KAIST), Daejeon, Korea. His main research interests include Virtualized Resource Management, and Cloud Computing.

### Seong-Hwan Kim

e-mail : s.h_kim@kaist.ac.kr

He received the B.S. degree in Media and Communications Engineering from Hanyang University, Seoul, Korea in 2012. Now he is a Ph.D. candidate in Department of Electrical Engineering at Korea Advanced Institute of Science and Technology(KAIST), Daejeon, Korea and is member of Advanced Network and Computing Laboratory in KAIST. His research interests include mobile cloud computing and cloud collaboration.

Chan-Hyun Youn

e-mail : chyoun@kaist.ac.kr

Chan-Hyun Youn received the B.Sc and M.Sc degrees in Electronics Engineering from Kyungpook National University, Daegu, Korea, in 1981 and 1985, respectively. He also received a Ph.D. in Electrical and Communications Engineering from Tohoku University, Japan, in 1994. Since 2009, he has been a professor at Department of Electrical Engineering in KAIST, Daejeon, Korea. He also was a Dean of Office of Planning Affairs and a Director of Research and Industrial Cooperation Group at former Information and Communications University, in 2006 and 2007. He was a Visiting Professor at MIT in 2003 and has been engaged in the development of Physio-Grid system with Prof. R.G. Mark's Group in LCP (Laboratory for Computational Physiology) of MIT since 2002. He also is a Director of Grid Middleware Research Center at KAIST. Where, he is developing core technologies that are in the areas of mobile cloud, mobile collaboration system, Internet computing workflow management, distributed network architecture, communication middleware, advanced e-Healthcare system, e-Health application services and others. Currently, he is serving the Editor-in-Chief of KIPS (Korea Information Processing Society), and an Editor of Journal of Healthcare Engineering (U.K.), and served head of Korea branch (computer section) of IEICE, Japan (2009, 2010). He is a member of IEEE, KICS and IEICE, respectively.