An Experimental Cloud Resource Broker System for Virtual Application Control with VM Allocation Scheme

Seong-Hwan Kim[1], Dong-Ki Kang[1], Ye Ren[1], Yong-Sung Park[1], Kyung-No Joo[1], Chan-Hyun Youn[1], YongSuk Park[2]

[1] Department of Electrical Engineering and Grid Middleware Research Center, KAIST, Daejeon, Korea
{s.h_kim, dkkang, catherine, miracle0318, eu8198, chyoun}@kaist.ac.kr
[2] Samsung Electronics Inc.
victorious.park@samsung.com

Abstract. The high computing application needs suitable platforms that enable users to build customized solutions and provide significant resource to users in time. In addition, not only QoS should be guaranteed to users but also the resource operation cost has to be minimized. Optimizing the VM Provisioning scheme which provides adequate amounts VM instance to users in response to fluctuation of workload can be a part of solution. Also VM Allocation scheme which places the request to proper VM instance can be the methodology for achieving objective. In this paper, we propose an Adaptive Cloud Resource Broker System which provides VM Allocation scheme to maximize the utilization of VM resources for minimizing the cost. By using our proposed cloud resource broker system, it enables the minimization of service operation cost and VM instance selection based on the request type and user SLA requirement. To evaluate the performance of the proposed system, service completion time and resource operation cost are measured.

Keywords: Cloud Computing; High Computing; VM Allocation; Request Placement; Virtual Application

1 Introduction

Cloud computing offers a feasible solution to computation-intensive and data-intensive applications that need distributed execution. With on-demand flexible computing capacity with enormous resources from the public cloud resource providers [1], we can process the high computation applications like scientific applications (i.e. various high computation applications in the field of chemistry and biology needs a lot of computing resources so as to calculate the complex formulation in time).

To provide desirable cloud service to users with deadline constraint and cost constraint, the main problems of cloud infrastructure interlocking are optimized resource provisioning and request placement scheme [2]. In particular, users negotiate with cloud resource providers on their required QoS and on the corresponding price to reach a Service Level Agreement (SLA) [3]. The issue of satisfying users' different QoS needs to be well addressed.

In this paper, we propose the Adaptive Cloud Resource Broker System in order to support the previous described applications efficiently and minimize cost for high computing by interlocking the cloud infrastructure. Also we propose Adaptive Cloud Resource Broker which bridges the application policy on the users' side and the resource policy on the cloud resource providers' side to make system abstraction to users and make consensus to provide suitable Quality of Services guarantee. The adaptive cloud resource broker enables service level classification and provides different QoS guaranteed service to users from different domains through an adaptive policy-based resource scheduling mechanism. By provisioning the virtualized computing resources, the adaptive resource cloud broker can handle the users' requests fluctuation. And achieve good resource utilization and high cost-performance with VM Allocation scheme.

For an example case to evaluate our proposed manager, we will use the Virtual Application (in short VApp) with MapChem chemical simulation package. The VApp is application which is working on Cloud VM Instance and communicate with the mobile device in point to point about screen data and user input in point to point manner to implement thin client. But this architecture can't use the on-demand capacity of cloud service (almost infinity with sufficient budget). To serve high computing application and scientific application in the VApp with on-demand cloud resource, we use Cloud Resource Broker to manage and provisioning the VM instances. And MapChem is an integrated application for collaborative pharmaceutical research project involving several organizations. We build the experimental Adaptive Cloud Resource Broker System to serve MapChem application with cloud based service management. We show the performance comparison of SLA-based scheduling in terms of cost-performance efficiency.

2 Architecture of Adaptive Cloud Resource Broker System

An adaptive cloud resource broker integrates the functionality of the resource management system and the cloud infra service system. As shown in the Fig. 1., System is designed as layered architecture including Service User Layer(SUL), Workflow Management Layer(WML), Resource Management Layer(RML) and Cloud Resource Layer(CRL).

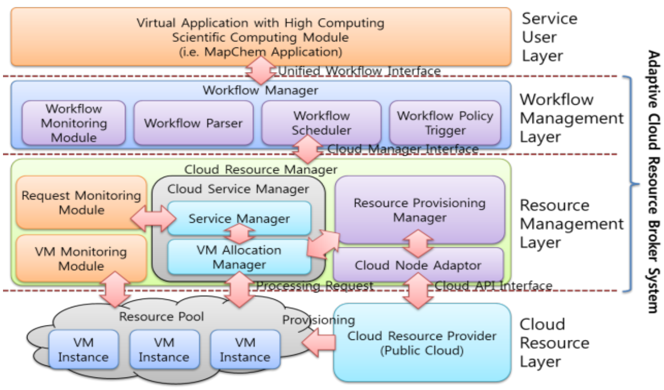


Fig. 1. Layered Architecture of Cloud Resource Broker System

Workflow Management Layer (WML): WML is composed of 4 function which are Workflow Monitoring Module, Workflow Parser, Workflow Scheduler and Workflow Policy Trigger. And Cloud Manager Interface bridges the upper SUL and below RML in unified workflow interface. Workflow is defined as "The automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules" from Workflow Management Coalition (WfMC) [4]. Especially scientific workflow automatically schedule, fetch and processing the scientific process based on workflow description and it is possible to conduct many innovative scientific experiments easily [5]. In the same context, Workflow Manager is where users can compose their scientific workflow according to their practical needs as SLA index with provided high computing application components. In detail, users need to submit their detailed description of application and specific SLA requirements such as the total completion time and the total budget for the each and whole service.

Resource Management Layer (RML): RML is composed of 5 functions which are Request Monitoring Module, VM Monitoring Module, Resource Provisioning Manager, Cloud Node Adaptor and Cloud Service Manager. And Cloud Service Manager is Composed of Service Manager and VM Allocation Manager. The Cloud Resource Manager will provide satisfactory service level with resource management and VM allocation scheme based on requirements description from user. To make a decision in Cloud Resource Manager, it also needs the historical and statistical data. Monitoring and recording of VM instance and Request Processing data in database can be used for managing the ongoing works efficiently. Based on decision from provisioning scheme, Cloud resource computation ability called VM instances is provided to Service User through the Cloud API Interface. To communicate through the Cloud API Interface, Cloud Node Adaptor intermediate each heterogeneous Cloud Resource Providers (i.e. AWS, Azure, etc.) and Cloud Resource Manager. To determine the desirable VM instance type and VM instance number to request, VM Monitor and Request Monitor manages the information of previous historical processing result, SLA, VM utilization and metadata. Resource provisioning manager adopt the resource provisioning scheme to manage the metadata of created VM instance and to make decision how many VM instance should be prepared. Request of creation and termination of VM instance is send to Cloud Node Adaptor. Because creation time of VM is not negligible, so on-demand provisioning is impossible and enough VM instance should be prepared for future request input. Service Manager manages the request (sub-task of workflow) and SLA requirement in queue and schedule the order of processing. When request is fetched from service scheduler, VM Allocation Manager select the proper VM instance to processing based on VM allocation scheme in Section 3. VM allocation scheme basically include the principle of load balancing to process the request efficiently. It could adapt any kind of policy, but in this paper we consider the minimization of cost. After selecting the proper VM instance, processing request is send to the VM instance and after the processing result is return to Service User and statistical data are saved to the database.

In Section 3, VM Allocation Scheme, the VM resource type decision scheme and VM resource allocation method, is introduced so as to provide cloud resource to service users efficiently.

3 Virtual Machine Allocation Scheme

After getting the processing schedule table, to decide which type of VM resource should be allocated to the current sub-task, we adopt the scheme shown in Fig. 2. in the VM Allocation Manager. The particular instance of the requested VM type, which is to be practically allocated to the current sub-task, will be selected by the scheme shown in Fig. 2.

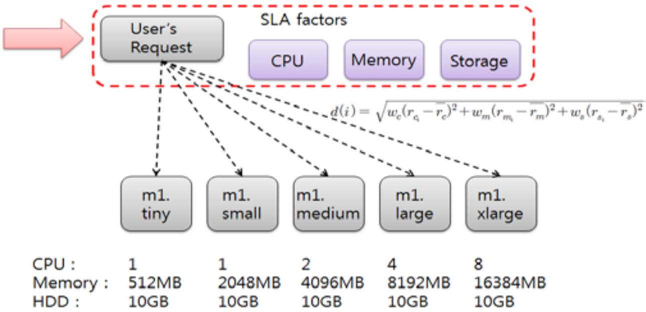


Fig 1. Procedure of VM Type decision

System uses 2-tire SLA definition which are SLA1 and SLA2 to give user the abstraction from complexity of system information. SLA1 is user familiar service requirement which include deadline and total cost. And SLA2 is systematic requirement like CPU, Memory, Storage, Network, etc. The Similarity Degree Function is calculated as the Euclidean distance between two vectors each of which contains SLA2 parameters. Because three SLA parameters reflect the resource capability, we can infer that a smaller similarity degree indicates more similar resource amount between the sets of request and supply from an overall view. So if there is the more VM type, the more proper and similar VM type we can use. VM type decision procedure firstly start from transforming the SLA1 into SLA2 to decide the resource specification to allocate optimized resource supply when the user's request with SLA1 is turned in the Cloud Resource Manager. To transform the SLA1 to SLA2, we need to define the SLA conversion function, however this it is an another complex problem so it will be studied in future work. So in this paper, we only focus on SLA2 mapping process. There are several VM types which are prepared in advance. By using the Similarity Degree Function as mentioned earlier, user's request can be allocated to the most proper VM instance. By selecting the type of resource which has the most similar resource capability with the expected one, we are more likely to get the expected execution result, which is, to comply with the resource parameters specified in SLA2 as much as we can so that we can guarantee the QoS requirements specified in SLA1.

After the resource type decision, to allocate user's request to VM instance with maximization of VM instance utilization we allocate multi request to each VM. Because there can't be infinite set of VM types which capacity matches to each request, there can be resource over supply that we can't use. When the VM type of user's request is selected, the VM allocation module check out the current processing number of request in each VM instances. The upper limit threshold of allocation capacity to each VM instances is the value obtained from many empirical tests that achieving the maximization of resource utilization where the point that performance degrades dramatically for additional request handling. The goal of “Policy that gives first priority to choosing the instance is that select the most large amount of workload of processing requests among all instances and also the amount of the being processed requests is under the instance’s upper limit of request amount” is to utilize the resource capability to the fullest of each VM instance but also still achieve the user’s requirement.

The description of the procedure of resource type decision and VM instance selection is shown in below Table 1. in more detail.

Table 1. Procedure of Resource Type Decision and VM Instance Selection

INPUT :

$[\overline{r_c}, \overline{r_m}, \overline{r_s}]$ the required resource specification for executing cloud service.

$vt_i = [r_{c_i}, r_{m_i}, r_{s_i}]$, the resource type of each VM instance.

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

v_{ij}, the VM instance which VM type is vt_i

VARIABLES :

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

w_c, w_m, w_s represent the weighted values for the CPU, memory, and storage of a resource .

$wl(v_{ij})$, the workload of request to v_{ij} and $0 \leq wl(v_{ij}) \leq c_i$ for all v_{ij}

OUTPUT :

v_{sel}, the selection of a VM instance to execute the request.

BEGIN :

Calculate the similarity degree using the following formula referenced from

$d(i) = \sqrt{w_c(\overline{r_c} - r_{c_i})^2 + w_m(\overline{r_m} - r_{m_i})^2 + w_s(\overline{r_s} - r_{s_i})^2}$ and find the $min(d(i))$ among all the values of similarity degree.

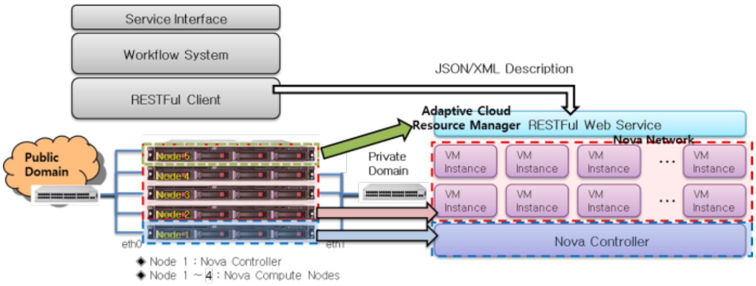
For selected VM type vt_i, find $v_{sel} = max(wl(v_{ij}))$ with constraint that $wl(v_{ij}) < c_i$.

END

4 Experimental Result

We evaluate the proposed scheme in Section 4 in terms of performance, cost, and cost-performance. We compare the performance results of our proposed scheme under the Cloud based testbed with that of the traditional scheme under the physical machine based environment.

In this paper, to mimic the cloud resource provider, we establish the open-source Cloud platform called OpenStack which supports a variety of hypervisors such as XEN, KVM, etc. We use the component naming Nova in the OpenStack which provides computing service to users and manages VM life cycle. OpenStack platform operate front end machine called Nova controller node.



	Node 1	Node 2	Node 3	Node 4	Node 5
Functions	Nova Controller Nova Compute	Nova Compute	Nova Compute	Nova Compute	Adaptive Cloud Resource Manager
Specification	H/W: HP, Xeon E5620 2.8G, Core8, Mem 16G, HDD 1T S/W: Ubuntu 12.04, OpenStack Essex				

Fig. 2. Experimental environment

Cloud controller node manages the operations between Nova computing nodes. And actual computing works are processed on Nova computing nodes and their status are reported to the Nova controller node. Nova controller node is easily accessible through the public IP address, anywhere in the Internet environment. But for convenient management for assessment to the VM instance, it is helpful to locate adaptive cloud resource manager in same private network to access the nova virtual network between Nova Network module and VM instances. In summary, the requests from users are submitted to the our system through the public IP address, and out system request resources to the Nova controller node through the public/private IP address and actual Job request is send to the provided VM instance through the nova virtual network.

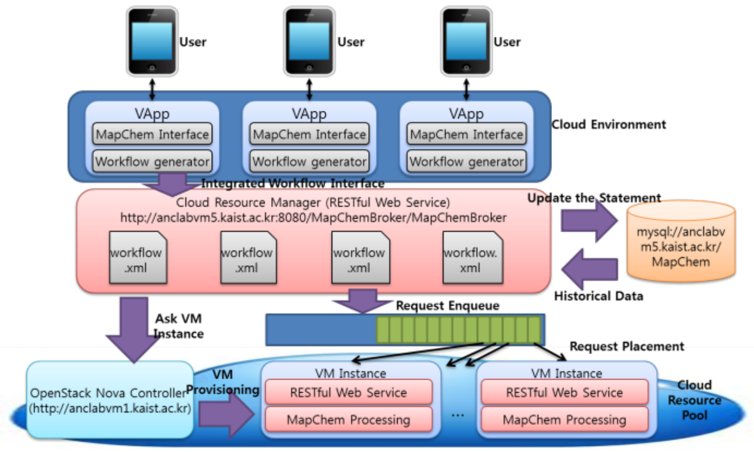


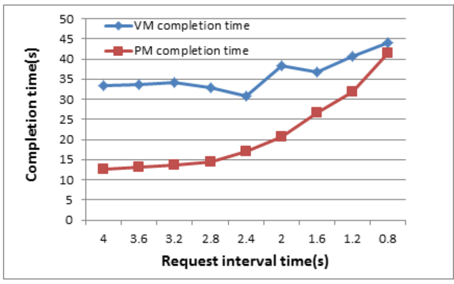
Fig. 3. Experimental testbed implementation

In this paper, to evaluate the performance of adaptive VM allocation scheme, we implement the system except for Workflow management layer. We just send workflow description liked request to the system through the dummy client. Each request has only one sub-task and it is going to the resource management layer directly without manipulation from workflow manager. And resource management operate the scheme in algorithm 1, to maximize the resource utilization, minimize the cost and achieve the deadline that user required.

Table 2. H/W environment for 3 experimental cases

Hardware Environment
5 Bare machine 8 Core 2.8GHz 16GB RAM
Cloud Platform with same Computing Resource

We make experiment that compares the performance of different resource provisioning schemes – the physical resource provisioning and the visualized resource provisioning which bear the equal resource capability. There are two cases of target environments. As shown in Table 2., the first case is 5 bare machine with 8 Cores 2.8GHz and 16GB RAM memory. The second case is using VM instances in Nova cloud resource with same amount of computing resource.



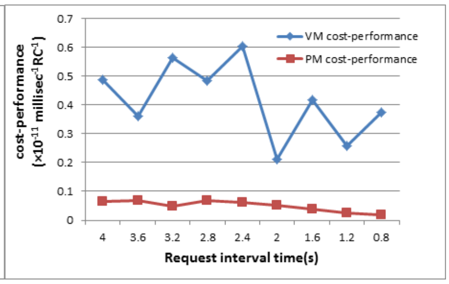


Fig. 5. Request completion time of both schemes (left)

Fig. 6. Cost-performance of both schemes (right)

To evaluate the performance of Cloud Resource Manager and demonstrate the efficiency of our scheme, we use MapChem service as an example of application service. MapChem is an integrated application for collaborative pharmaceutical research. The services provided by MapChem can be used individually, or be composed into various complex sub-tasks in according with user's various needs. The computing intensive workload of such request is time-consuming.

In Fig. 5., for the X-axis, a smaller value of request interval time indicates more input SDF files (Unit of MapChem request); for the Y-axis, the completion time means the total spent time between when the system receives job submission and when it finishes the processing of the last job in the system. We see that as the number of total job requests increases, the total completion time increases as well for both scenarios. Here we see that the PM case can complete processing the same number of job requests faster. However, we still need to consider about the factor of total resource cost. Therefore, we need to calculate the cost for both VM and PM.

In Fig. 6., we see that the cost-performance numeric value of VM scheme is higher than that of the PM scheme. That is to say, the VM resource utilization scheme is much more cost-efficiency than the traditional physical resource utilization scheme. It is proved that the resource virtualization can bring much more efficiency than the traditional utilization of physical computing resources. On a wider view, by resource virtualization, we can achieve better performance with the same resource capacity of equal monetary cost.

5 Conclusion

In this paper, we proposed the Cloud Resource Manager, a cloud broker system for high computing application such as chemical application services development. The VM Allocation is the key module in the Cloud Resource Manager. The Cloud Resource Manager implements a VM resource provisioning and allocation scheme in order to optimize the assignment of the cloud services to virtualized resources in the cloud-based testbed, and also to enhance the efficiency of the utilization of VM resources. Through the experimental evaluation, we show that the VM Allcoation scheme outperforms the physical machine allocation in the view of cost-performance. We also experiment on a cloud-based testbed to test the allocation property of our proposed scheme. Results show that it can allocate the cloud services onto the VM instance pool efficiently to satisfy the fluctuation of job requests while maintain a better resource utilization.

Acknowledgment

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 was equally supported by Integrated dev-environment for personal, biz-customized open mobile cloud service and Collaboration tech for heterogeneous devices on server through the National Research Foundation of Korea(NRF) funded by the Ministry of Knowledge Economy [10039260].

References

1. Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente.: Elastic Management of Cluster-based Services in the Cloud. In: ACDC '09 Proceedings of the 1st workshop on Automated control for datacenters and clouds, pp. 19-24. (2009)
2. Ming Mao, Jie Li, Humphrey, M.: Cloud Auto-scaling with Deadline and Budget Constraints. In: Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on, pp. 41-48. (2010)
3. Saeid Abrishami, Mahmoud Naghibzadeh, Dick Epema.: Cost-driven Scheduling of Grid Workflows Using Partial Critical Paths. In: Delft University of Technology, Parallel and Distributed Systems Report Series. Report NO. PDS-2011-001. (2011)
4. Workflow Management Coalition.b http://www.wfmc.org/
5. Dong Yuan, Yun Yang, Xiao Liu, Jinjun Chen.: A data placement strategy in scientific cloud workflows. In: Future Generation Computer Systems Volume 26, Issue 8, October 2010, pp. 1200–1214. (2010)
6. Boloor, K., Chirkova, R., Viniotis, Y., Salo, T.: Dynamic request allocation and scheduling for context aware applications subject to a percentile response time SLA in a distributed cloud. In: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pp. 464-472. (2010)