An Adaptive Resource Provisioning Method Using Job History Learning Technique in Hybrid Infrastructure

Jieun Choi*, Yoonhee Kim†
*Dept. of Computer Science, Sookmyung Women's University,
Seoul, Republic of Korea
Email: {jechoi1205, yulan}@sookmyung.ac.kr

***Abstract*—Cloud computing technology enables scientists to dynamically expand their environments for scientific experiments. However, to maximize performance and satisfy user requirements it is difficult to quickly provide hybrid resources suitable to application characteristics. In this paper, we design a resource provisioning model based on application characteristic profiles and job history analysis in hybrid computing infrastructure consisting of cluster and cloud environments. In addition to the multi-layer perceptron machine learning method, error back-propagation technique is used to analyze job history to re-learn the error of the output value. Also, we propose an adaptive resource provisioning method for horizontal/vertical scaling of VMs in accordance with the state of the system. We experiment CPU-intensive applications according to the proposed model and algorithms, in a hybrid infrastructure. The experimental results show that using the proposed method, we satisfy user-specified SLA (cost and execution time) and improve the efficiency of resource usage.**

***Keywords*—Adaptive resource provisioning, multi-layer perceptron learning, job history, hybrid infrastructure, cloud.**

I. Introduction

Cloud computing enables on-demand resource provisioning and scalable resource management. It provides benefits such as economy of scale, elasticity, flexibility and customization to the specification of scientists and HPC communities. In recent research, it is difficult to quickly provide hybrid resources suitable to application characteristics in integrated infrastructure such as cloud computing environments. The resources selected by user are limited in supporting the various scientific applications which requires high performance and high throughput computing. A resource provisioning method, which can provide suitable resources according to application characteristics and experimental environment and satisfy user requirement in advance, is needed. Currently, there are research works [15∼17] about resource provisioning which use a variety of statistical and reasoning techniques. However, they only provide resource provision without consideration of the application characteristics and hybrid infrastructure. It is necessary to develop a resource provisioning method using analysis of job history. Also, efficient resource management method by scaling VM dynamically is needed.

In this paper, we suggest a resource provisioning model based on application characteristic profiles and job history analysis in hybrid computing infrastructure consisting of cluster and cloud environments. In addition to the multi-layer perceptron machine learning method, error back-propagation technique is used to analysis of job history to re-learn the error of the output value. Also, we propose an adaptive resource provisioning method for horizontal/vertical scaling of VMs in accordance with the state of the system. We experiment CPU-intensive applications according to the proposed model and algorithms in a hybrid infrastructure. The experimental results show that using the proposed method, we satisfy user-specified SLA (cost and execution time) and improve the efficiency of resource usage. The main research content and methods of this paper is organized as follows.

- Propose a supervised learning model using multi-layer perceptron and error back-propagation method for analysis of job history in hybrid computing.
- Generate input and output data for the proposed model using job history in hybrid infrastructure consisting of cluster (SGE: Sun Grid Engine [8]) and cloud (OpenStack [9]) and show accuracy of the model.
- Suggest an adaptive resource provisioning method consisting of three algorithms based on job history learning.
- Verify the performance of the proposed algorithm through comparative real experiments and simulation.

The rest of this paper is structured as follows: Sect. II discusses related works while Sect. III introduces the cloud resource provisioning model based on job history learning technique. In Sect. IV, adaptive resource provisioning algorithms are discussed in detail, while experiments and results are presented in Sect. V. Sect. VI concludes the paper and discuss future work.

II. Related Works

HPC communities (DIRAC [8], HTCaaS [1], Condor-G [9], gUSE/WS-PGRADE [10]) recently integrate cloud with traditional distributed computing environments such as grid, cluster, desktop grid. There are a few studies that address the need of simultaneously controlling heterogeneous computing infrastructures. Mateescu et al. [11] described the concept of Elastic Cluster combining of traditional HPC, Grid and Cloud computing to achieve effective and predictable execution of HPC workloads. In [11], they noted that no single infrastructure is the best solution from all points of view.

978-1-5090-3651-6/16 $31.00 © 2016 IEEE
DOI 10.1109/FAS-W.2016.27

Moca et al. [12] presented a fault-tolerant and trust-aware scheduler, which allows to execute Bag-of-Tasks applications on elastic and hybrid distributed computing infrastructures. In previous study [22], we conducted a study on the job distribution ratio in a hybrid infrastructure using the HTCaaS. With the accelerated increase in the use of cloud infrastructure, there is a corresponding increase in dynamic resource provisioning techniques that provide required resources. In this paper, we suggest a resource provisioning model based on application characteristic profiles and job history analysis in hybrid computing infrastructure consisting of cluster and cloud environments.

In this section, we introduce several related works with a focus on cloud resource provisioning based diverse reasoning technique, and adaptive resource management.

A. Cloud Resource Provisioning Methods based Reasoning Technique

In cloud environment, there are many research[12∼14] that have considered resource provisioning methods and which use different statistical and reasoning techniques. Kim et al. [13] suggested a fuzzy logic driven VM provisioning scheduling with a precise evaluation of resource availability using resource evaluation. In [13], their method considered the availability of only used VMs, since it cannot select suitable VM for application. Rao et al. [14] proposed a distributed learning mechanism that facilitates self-adaptive virtual machines resource provisioning. The mechanism which uses a proposed reinforcement learning algorithm evaluates the requests and replies with feedbacks. This method however, can have an overhead of user feedback, thus it is not an appropriate method for HPC and HTC applications. Sukhija et al. [15] presented a portfolio-based selection of robust dynamic loop scheduling algorithms using multi-layer perceptron learning. However, it is not a suitable method because it considered only grid computing without cloud computing.

B. Adaptive Resource Management

With Cloud computing, it is easy to build various scientific application that can be managed and controlled for scalability and flexibility. There are many research works [15∼18] that have discussed adaptive resource management methods in cloud environments. Auto-scaling technique is currently being studied as an effective resource management approach. It is categorized into horizontal scaling and vertical scaling. Horizontal scaling adds or removes the number of VM, while vertical scaling expands and reduces the amount of resources (CPU, Mem, Disk)in a given VM. These research works proposed auto-scaling methods using prediction models for future resource usage of applications. Samuel et al. [16] proposed a forecast model to satisfy web applications' SLA such as web service response time and throughput. They used three reasoning techniques such as Linear Regression, Neural Network, and Support Vector Machine. Similarly, Bashar et al. [17] utilized Bayesian Networks to predict resource usage. Nikravesh et al. [18] applied Hidden Markov model and compared their method with Amazon CloudWatch [19]. They determined VM scaling according to prediction models. In this paper, we propose an adaptive resource provisioning method based on application characteristics and analysis of job history using both horizontal and vertical scaling.

III. Cloud Resource Provisioning based on Job History Learning Technique

In this section, we introduce that cloud resource provisioning model based on scientific application job history and application characteristics. It uses multi-layer perceptron learning technique from Weka [20].

Multi-layer perceptron (MLP) maps sets of input data onto a set of appropriate outputs. It consists of multiple layers of nodes (neuron) in a directed graph with activation function. MLP is a model that can solve the non-linear separation problem (XOR) of single-layer perceptrons. In this paper, we propose a cloud resource provisioning model based on application-aware job history learning. To design a MLP model for cloud resource provisioning based on job history data, we need to define the categories of the job history data. The job history data consists of four categories as follows

- *Application Profiles:* includes application name, application type, average CPU utilization, memory utilization, the number of tasks, and input file size.
- *System Status:* consists of machines' specification and current resource usage.
- *VM Information:* has vCPU, Mem, Disk, and VM cost per hour, the number of used VM according to the used VM types.
- *Historical Data:* is total execution time, total cloud resource cost, job distribution ratio.

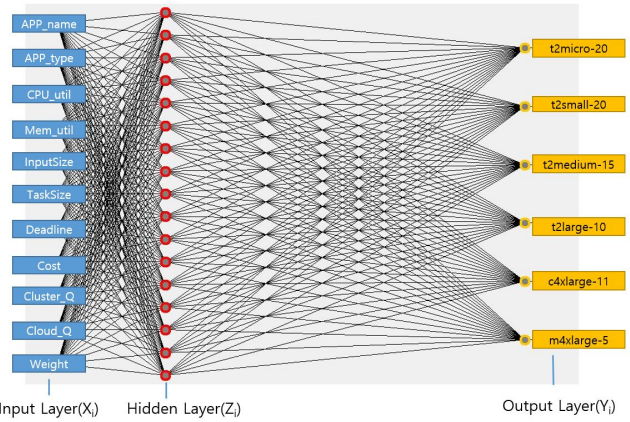

Fig. 1: Resource provisioning model based on job history using MLP

In the model, job history data consisting of various attributes such as application name, application type, resource utilization, input size, task size, deadline, cost limit, cluster queue status, cloud queue status, and job distribution ratio are non-linearly separable. Also, it utilizes error back-propagation technique which is applied to analysis of job history to re-learn the error of the output value.

Figure 1 shows the proposed resource provisioning model based on job history using MLP. The input data for the model

includes application profiles, system status, and historical data, while output is the VM information of job history items. Input data for the learning step of learning model uses 156 data and input data for the validation step of learning model uses 9 data.

Learning factor was carried out several times and selected as a parameter in the case shown the highest accuracy as follows. Learning rate: 0.2, momentum: 0.8, the number of learning: 500, the number of allowed consecutive errors: 20, the number of hidden layer: 17, the number of input layer: 11, the number of output layer: 6. In the verification step by attempting to predict, the accuracy of learning model was calculated. The proposed resource provisioning model shows the accuracy of 77.7778%.

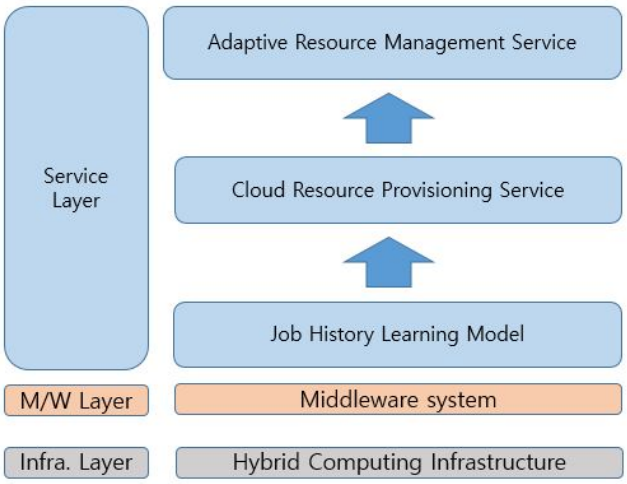

Fig. 2: Service Architecture Model

Figure 2 presents an overview of a service architecture model for an adaptive resource provisioning method using job history learning technique in a hybrid infrastructure. It basically consists of three layers which are Service Layer, Middleware Layer, and Infrastructure Layer. In this paper, hybrid computing infrastructure was targeted thus clusters and cloud environments were deployed on the HTCaaS middleware. Service layer largely consists of two services, Cloud Resource Provisioning Service (CRPS) based on job history learning model and Adaptive Resource Management Service (ARMS). In the CRPS, when the user submits the SLA (cost and deadline) and scientific applications to the system through the middleware, virtual machines are assigned according to the appropriate SLA and system status based on deductions from work history learning model. While performing scientific applications using the inferred resources, the ARMS monitors available resources to provide services and control the number of virtual machines in accordance with the system status.

IV. Adaptive Resource Provisioning Method

In this section, three algorithms that can provision resources in hybrid infrastructure and monitor systems' availability ratio, and scale the VM in accordance with the state of the system are proposed. The assumptions and notations of the proposed algorithms can be seen in table I.

TABLE I: Notations

Notation	Description
$\mathbb{R}$	A user request $\{ App_{name}, Input_{size}, \|Input\|, SLA_d, SLA_c \}$
P	A user policy for SLA $P \in \{ COST, PERFORMANCE \}$
PD	Profiling data for $\mathbb{R}$ = $\{ \mathbb{R}, App_{type}, CPU_{util}, Mem_{util} \}$
R_i	A Resource $R_i \in \{ Cluster, Cloud \}$
SR_i	Current status of R_i = $\{ Spec\_R_i, Util\_R_i \}$
UR	Selected Resource type from user
TR_i	Type of Resource R_i
$\|cR\|$	The number of changed resources
$\|rR\|$	The number of reduced / released resources

Algorithm 1 Resource Provisioning Algorithm

Input: a Request $\mathbb{R}$, a user policy P, userResource UR
Set PD , $SystemStatus$
Set $InputData_i = \{ PD, SystemStatus \}$
VM_{type} and $|VM| \leftarrow MLP(InputData_i)$;
if R_i == Cloud **then**
  $TR_i \leftarrow VM_{type}$;
  $|R_i| \leftarrow |VM|$;
else
  $TR_i \leftarrow R_i$;
  $|R_i| \leftarrow AvailableR_i$;
end if
Output: Resource Provisioning RP
  = {(TR_i , $|R_i|$) | i = 0,1,$\cdots$, N−1, $R_i \in UR$}

Algorithm 2 Monitoring Algorithm

Input: a result of resource provisioning RP , a user policy P
Default SF, HJS, DV , SD = false ;
$SF \leftarrow DetectSF()$;
$HJS \leftarrow DetectHJS()$;
if $SF == true$ **then**
  $|cR| \leftarrow (|R_i| - |rR|)$;
  RESCHEDULE($|cR|, waitingJob$);
  $DV \leftarrow CompareEFT(EFT, SLA_d)$;
end if
if $HJS == true$ **then**
  Increase $JobPriority$;
  Release $|reducedResource|$;
  $|cR| \leftarrow (|R_i| - |rR|)$;
  RESCHEDULE($|cR|, waitingJob$);
  $DV \leftarrow CompareEFT(EFT, SLA_d)$;
end if
if $DV == true$ **then**
  if $P == PERFORMANCE$ **then**
    $SD \leftarrow true$;
  else
    $Update(SLA_d)$;
  end if
end if
Output: Scaling Decision SD ,
  Amount of reduced resource $|rR|$

Algorithm 1 describes a resource provisioning based on application characteristic profiles and job history analysis in hybrid computing infrastructure including cloud environments.

In algorithm 1, a user submits a request $\mathbb{R}$ and then sets the profiling (PD), system status (SystemStatus) information and input data ($InputData_i$) of the learning model (lines 1-2). It finds a suitable VM type and the number of VM according to input data by using multi-layer perceptron model (line 3). If the resource is cloud, the VM provisioning result by learning model is selected (lines 4-6). While, other resources are provisioned by the availability of the resource (lines 7-9).

Algorithm 2 shows available resources monitoring algorithm. Algorithm 2 monitors system failure (SF), higher priority job submission (HJS), and user-defined deadline violation (DV). If system failure happened, waiting jobs of failed resources are rescheduled (lines 4-8). If new jobs' priority are higher than current job, current jobs' priority is increased and the released resource size is calculated (lines 9-15). In spite of rescheduling, if deadline violation is detected then scaling decision (SD) is decided according to user policy.

Algorithm 3 Adaptive Resource Scaling Algorithm

Input: Scaling Decision SD, Scaling Type ST,
Amount of reduced resources $|rR|$,

1: **while** SD **do**
2: **Switch** (ST)
3: **case** $HORIZONTAL$**:**
4: $TR_i \leftarrow VM_{type}$;
5: $|R_i| \leftarrow |VM| + |rR|$;
6: **case** $VERTICAL$**:**
7: $TR_j \leftarrow NewVMtype(VM_{type})$;
8: $|R_j| \leftarrow |rR|$;
9: $TR_i \leftarrow VM_{type}$;
10: $|R_i| \leftarrow |VM| - |rR|$;
11: **end switch**
12: **end while**

Output: Adaptive Resource Provisioning ARP
$= \{(\ TR_i\ ,\ |R_i|\ )\ |\ i = 0,1,\cdots, \mathrm{N}-1,\ R_i \in UR\}$

Algorithm 3 describes the adaptive resource scaling method which applies horizontal or vertical scaling according to previous selected scaling type while scaling decision is true. In horizontal scaling, added number of VMs is calculated (lines 3-6). Vertical scaling should calculate changed VM size of not only the original VM but also new VM (lines 6-10). The output of algorithm 3 is the results of the proposed adaptive resource provisioning (ARP) method.

V. Experiments

Experiments that validated our adaptive resource provisioning method are presented in this section. First, we describe the system and target applications, and subsequently present the result of experiments.

A. Hybrid Infrastructures

HTCaaS[1]aims to expedite exploring large-scale HTC or MTC problems for diverse computing environments such as Supercomputers, Grids, Clusters and Clouds. It can conceal diversity of integrating heterogeneous resources from users, and enable users to effectively submit a large number of tasks immediately. It applies a *pilot-based* multi-level scheduling technique which aids the decoupling of resource assignment from resource binding. The general pilot (or agent) itself is a normal batch job which is submitted by HTCaaS system and is assigned into the resources by the local batch scheduler. Then it execute '***pulling and executing***' sub-jobs as well as coordinating the launch and monitoring procedures.

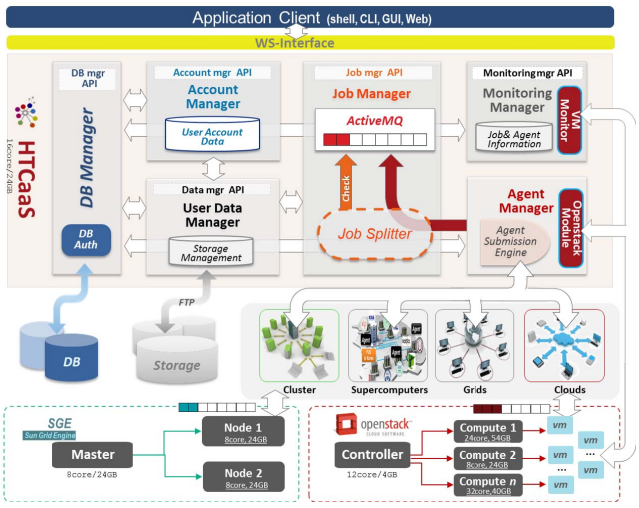

Fig. 3: Hybrid Infrastructures for Scientific Applications using HTCaaS

The computing resources consist of local cluster and private cloud resources using HTCaaS. The local cluster uses a Sun Grid Engine (SGE) [2] which is a batch-queuing system supported by Sun Microsystems. OpenStack [5] is an open source software that provides large pools of compute, storage and networking resources used for cloud environments. Our OpenStack cloud environment is made up of 1 Intel(R) Core(TM) i7-4930K CPU @ 3.40GHz Controller with 12 cores of CPU and 4GB of RAM and 1 Intel(R) Core(TM) i7-4930K CPU @ 3.40GHz compute node with 8 CPU cores and 24GB RAM and 2 Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz compute nodes with each 24 and 32 CPU cores and 40 and 54GB RAM. In the experiments, 6 flavor such as t2.micro, t2.small, t2.medium, t2.large, c4.xlarge, m4.xlarge is used for VM and its' cost is configured according to Amazon EC2 [4] as shown in Table 2. Each VM was created identically using Ubuntu 12.04 Server image.

TABLE II: VM information

Name	vCPU	Mem(GB)	Disk(GB)	Cost($) per hour
$t2.micro$	1	1	5	0.02
$t2.small$	1	2	5	0.04
$t2.medium$	2	4	5	0.08
$t2.large$	2	8	10	0.16
$c4.xlarge$	4	8	10	0.24
$m4.xlarge$	4	16	10	0.33

B. Target Application

We address High Throughput Computing (HTC) and Many Task Computing (MTC) applications consisting of millions or billions tasks [22] to be dealt with comparatively short per task

execution time. For our experiment, we use two applications, PYTHIA and Autodock.

PYTHIA[6] is a common tool for Monte Carlo (MC) simulations for events generation in high-energy physics. It makes up a consistent set of physics models as a library and also has a set of utilities and interfaces to external programs [24]. PYTHIA is mainly a pure CPU-intensive application with small size of in/output files. PYTHIA uses average 94.59% CPU utilization and 0% memory utilization.

AutoDock[7] on the other hand, is a suite of automated docking tools to anticipate how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure. The AutoDock job we use is considered a CPU-intensive application having in/output data files with small sizes. The AutoDock uses average 99.12% CPU utilization and 0.7% memory utilization.

C. Experimental Results

The proposed resource provisioning method satisfies user requirements such as cost and execution time because it uses resources inferred from job history data. In this experiment, we compare our method (LB) to the method of using resources according to the cost-minimum (CM) and performance-maximum (PM) policies. Figures 4 and 5 show the experimental results for Autodock and PYTHIA respectively.

In figure 4, the VM types used are t2.medium(15), t2.micro(20), and c4.xlarge(11) for LB, CM, and PM respectively with user-defined deadline of 12000 seconds and cost limitation of $4.4. And in figure 5, LB uses t2.large(10) relative to CP with t2.micro(20) and PM with c4.xlarge(11) with user-defined deadline(470 seconds) and cost limit($1.7). The results show that the proposed resource provisioning method based on job history learning technique is better than cost-minimum or performance-maximum policy in terms of user-defined deadline and cost limit.

Fig. 4: Comparing SLA satisfaction for Autodock using the proposed learning-based (LB), cost-minimum (CM) and performance-maximum (PM) methods

The concept of the proposed adaptive resource scaling algorithm is demonstrated in a simulation using CloudSim [21] as shown in figure 6 and 7. We consider two scenarios which represent a case of system failure and higher priority job

Fig. 5: Comparing SLA satisfaction for Pythia using the proposed learning-based (LB), cost-minimum (CM) and performance-maximum (PM) methods

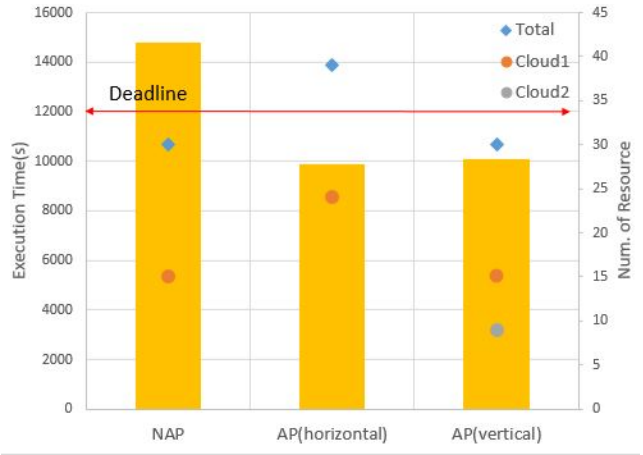

Fig. 6: Scenario 1 : system failure

submission using autodock. We compare the proposed adaptive resource provisioning method (AP) which uses both horizontal scaling and a vertical scaling, to a non-adaptive resource provisioning (NAP) method. The results show that there is deadline violation using the NAP method, because it could not add resources in accordance with the availability ratio of system resources. Meanwhile, the proposed method through VM scaling can satisfy user-defined deadlines by adding VMs according to the amount of required resources. The difference between horizontal scaling and vertical scaling is seen in the total number of resources per VM type and in the overhead of vertical scaling.

VI. Conclusion

In this paper, we design a resource provisioning model based on application characteristic profiles and job history analysis in hybrid computing infrastructure consisting of cluster and cloud environments. In addition to the multi-layer perceptron machine learning method, error back-propagation technique is used to analyze job history and re-learn the error of the output value. Also, we propose an adaptive resource provisioning method for horizontal/vertical scaling of VMs in

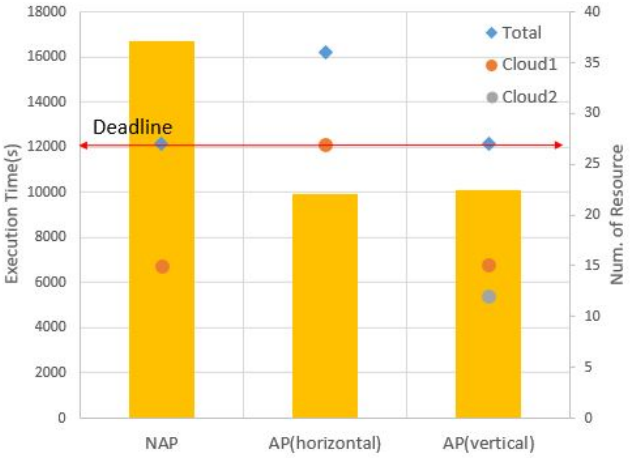

Fig. 7: Scenario 2 : higher priority job submission

accordance with the state of the system. We experiment CPU-intensive applications according to the proposed model and algorithms in a hybrid infrastructure. The experimental results show that using the proposed method, we satisfy user-specified SLA (cost and execution time) and improve the efficiency of resource usage.

In the future, our research work will include further experiments with different types of scientific applications. Furthermore, studies on identifying instances for horizontal and vertical scaling in automatic and ingenious ways according to application types will be investigated.

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 Science, ICT & Future Planning (2015M 3C 4A7065646)

REFERENCES

[1] Seungwoo Rho, Seoyoung Kim, Sangwan Kim, Seokkyoo Kim, Jik-Soo Kim, Soonwook Hwang, HTCaaS: a large-scale high-throughput computing by leveraging grids, supercomputers and cloud, High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, pp. 1341-1342, 2012

[2] SGE, http://www.oracle.com/technetwork/oem/grid-engine-166852.html

[3] Partnership and Leadership for the nationwide Supercomputing Infrastructure, http://www.plsi.or.kr/

[4] Amazon EC2(Elastic Compute Cloud), http://aws.amazon.com/ec2

[5] OpenStack, http://www.OpenStack.org

[6] T. Sjstrand, S. Mrenna, P.Z. Skands, PYTHIA 6.4 physics and manual, 2006

[7] Autodock, http://autodock.scripps.edu/

[8] DIRAC, http://diracgrid.org/files/docs/Overview/index.html?highlight=pilot

[9] James F., Todd T., Miron L., Ian F., Steven T., Condor-G: A Computation Management Agent for Multi-Institutional Grids, Cluster Computing Vol. 5, No. 3, pp. 237-246, 2002.

[10] gUSE/WS-PGRADE, http://guse.hu/about/home

[11] G. Mateescu, W. Gentzsch, Calvin J. Ribbens, Hybrid Computing Where HPC meets grid and Cloud Computing, Future Generation Computer Systems. Vol. 27, No. 5, pp. 440-453, 2011.

[12] M. Moca, C. Litana, G. C. Silaghi, G. Fedak, Multi-criteria and satisfaction oriented scheduling for hybrid distributed computing infrastructures, Future Generation Computer Systems. Ver. 55, pp.428-443, 2016.

[13] Jae-Kwon Kim and Jong-Sik Lee, Fuzzy Logic-driven Virtual Machine Resource Evaluation Method for Cloud Provisioning Service, Journal of the Korea Society for Simulation, Vol. 22, No. 1, pp. 77-86, 2013.

[14] J. Rao, X. Bu et al., A Distributed Self-learning Approach for Elastic Provisioning of Virtualized Cloud Resources, 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 45-54, 2011.

[15] N. Sukhija, B. Malone et al., A Learning-based Selection for Portfolio Scheduling of Scientific Applications on Heterogeneous Computing Systems, Journal of Parallel and Cloud Computing, Vol. 3, No. 4, pp. 66-81, 2014.

[16] AJILA A. Samuel and BANKOLE A. Akindele, Proactive Prediction Models for Web Application Resource Provisioning in the Cloud, International Conference on Transition from Observation to Knowledge to Intelligence, pp. 17-35. 2014.

[17] Abul Bashar, Autonomic scaling of Cloud Computing resources using BN-based prediction models, 2013 IEEE 2nd International Conference on Cloud Networking, pp. 200-204, 2013.

[18] Ali Yadavar Nikravesh et al., Cloud Resource Auto-scaling System Based on Hidden Markov Model(HMM), 2014 IEEE International Conference on Semantic Computing(ICSC), pp. 124-127, 2014.

[19] Amazon CloudWatch, http://aws.amazon.com/cloudwatch/

[20] Weka, http://www.cs.waikato.ac.nz/ml/weka/

[21] Rodrigo N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience, 41(1), pp. 23-50, 2011.

[22] Jieun Choi, Seoyoung Kim, Theodora Adufu, Soonwook Hwang, Yoonhee Kim, A Job Dispatch Optimization Method on Cluster and Cloud for Large-scale High-Throughput Computing Service, 2015 IEEE International Conference on Cloud and Autonomic Computing, Cambridge, MA, USA, Sept. 21-24, 2015

[23] I. Raicu, I. Foster and Y. Zhao, Many-Task Computing for Grids and Supercomputers, In: Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08), (2008) November.

[24] Sjstrand, Torbjrn, Stephen Mrenna, and Peter Skands. "A brief introduction to PYTHIA 8.1." Computer Physics Communications 178.11 (2008): 852-867.