A Science Gateway Cloud With Cost-Adaptive VM Management for Computational Science and Applications

Seong-Hwan Kim, Dong-Ki Kang, Woo-Joong Kim, Min Chen, *Senior Member, IEEE*, and Chan-Hyun Youn, *Member, IEEE*

***Abstract*—For the processing of scientific applications in cloud computing, the important challenge is to find an optimized resource scheduling method that guarantees cloud service users' service-level agreement while minimizing the resource management cost. To overcome this problem, in contrast to previous solutions that focus on a few specific applications, we design and implement a unified scientific cloud framework called science gateway cloud, which is a broker between users and providers and is able to process various scientific applications efficiently upon heterogeneous cloud resources. In particular, we design a cost-adaptive resource management scheme that reduces the resource management cost significantly without any degradation of performance based on the long-term payment plans of cloud resource providers. Furthermore, this system allows us to parallelize divisible scientific applications to improve the processing performance. Through the division policy for workflow scheduling, we show that both deadline assurance and cost minimization can be satisfied concurrently. Finally, we demonstrate that our proposed system significantly outperforms conventional cloud systems through various experimental and simulation results.**

***Index Terms*—Cost-adaptive policy, resource management, science gateway, service-level agreement (SLA), workflow scheduling.**

I. Introduction

MANY computational science fields (i.e., computational physics, computational chemistry, bioinformatics, etc.) related to the big-data computing [1], [2] require massive computation resources to solve various problems with methods of simulation, modeling, and numerical analysis. While generating massive streams of data continuously, enormous parallel or distributed supercomputing could be required in proportion to their computational complexity [3], [4].

As a substitute for traditional expensive supercomputing, cloud computing can deliver cost-efficient and scalable computing services, which are provided to cloud users in on-demand fashion; this is becoming an alternative computing paradigm for scientific application processing. Cloud users are able to share computing resources, such as computing components (i.e., CPU, memory, and network), software utilities, databases, and so on, for scientific application processing cost-effectively by leasing virtual machine (VM) instances on a pay as you go basis. However, despite its many advantages, there are several obstacles that prevent cloud computing from achieving successful deployment for scientific application processing.

Advanced scientific applications generally have diverse computing requirements, such as CPU intensive, memory intensive, network intensive computation, and so on [5], [6]. Therefore, it is difficult to derive a resource scheduling scheme which is a panacea for all kind of scientific applications. Moreover, we may observe that there are conflicts among scheduling objectives, such as application completion time and resource cost [7], [8]. The improvement of one objective might bring down the performance of other objectives. In addition, the professionals engaged in computational science might suffer from the complexity of distributed computing system (e.g., heterogeneous resource management, pipeline flow control, service deployment, and intermediate data control) in the cloud for solving their problems. To overcome the above issues to achieve the optimal performance and cost-efficient resource scheduling for scientific application processing on the cloud, the concept of the science gateway is proposed.

A science gateway is an automatic execution environment that can be generally applied for different types of science applications in a common interface to do task scheduling, task execution, and visualization, building a highly tuned computation farm over geographically distributed resources [9]. The important issue of the science gateway is to orchestrate tasks over a distributed set of resources with minimized costs while guaranteeing service level agreements (SLAs), which are the contracts regarding quality-of-service (QoS) constraints with users. However, several previous studies on the science gateway have not considered this issue or could not resolve this issue sufficiently. In addition, they focused on a specific scientific

Manuscript received March 06, 2015; revised July 20, 2015; accepted August 04, 2015. Date of publication February 03, 2016; date of current version March 10, 2017. This work was supported in part 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], in part by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology under Grant 2012-0020522, and in part by the MSIP under the ITRC support program [NIPA-2014 (H0301-14-1020)] supervised by the NIPA (National IT Industry Promotion Agency), and in part by BK21 Program. (Corresponding author: Chan-Hyun Youn.)

S.-H. Kim, D.-K. Kang, W.-J. Kim, and C.-H. Youn are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305–701, South Korea (e-mail: s.h_kim@kaist.ac.kr; dkkang@kaist.ac.kr; w.j.kim@kaist.ac.kr; chyoun@kaist.ac.kr).

M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: minchen@ieee.org).

Digital Object Identifier 10.1109/JSYST.2015.2501750

application, so they are not suitable for various scientific applications generally.

Yao *et al.* [10] proposed a science gateway supporting mobile device to compose and execute scientific applications of bioinformatics in the cloud environment. This system supports the collaboration of researchers on an experiment on genome bioinformatics through mobile devices. However, only the basic functions, such as service repository, service composition, service execution, and user interface, were considered to execute the scientific application. An SLA required by a user with dynamic resource scheduling and provisioning on scientific applications was not considered for executing the scientific application.

CloudBLAST [11] provides BLAST service, which is a bioinformatics tool to find regions of local similarity between nucleotide or protein sequences with a parallelization process in Apache Hadoop in the cloud environment to control the total execution time. This gateway only shows the concept of controlling the QoS through process parallelization for genome bioinformatics. This system does not provide a direct algorithm to guarantee the user SLA or minimize the cost of executing the scientific application.

To address these issues, we have designed a universal scientific cloud framework called science gateway cloud (SGC), which is described in detailed in Section II. The proposed SGC is able to process various scientific applications efficiently upon heterogeneous cloud resources. In our designed system, both deadline and resource management costs are considered concurrently by using our proposed scheme's cost-adaptive resource management scheme and the workflow scheduling scheme with the division policy. To reduce the resource management cost significantly without any performance degradation of scientific application processing, we designed a VM provisioning scheme with the pool management. Through a long-term payment plan [13] of cloud resource providers, the SGC is able to overcome the problem of cost minimization with a strategy of maximizing its profit. By determining the appropriate amount of resource provisioning over the distribution of arrival requests, it is possible to reduce the cost significantly with an acceptable completion time. We will show this scheme in Section III. Moreover, we propose a workflow scheduling scheme with a division policy that enables the parallelization of scientific applications to satisfy both deadline assurance and cost minimization in Section IV. By using the simplified and efficient integration of cost function model and managing the task affinity over the heterogeneous cloud services, we achieve the significant performance improvement of scientific applications. In particular, through workflow scheduling with the division policy, we show that our proposed scheme works well for scientific applications in practice.

Our contributions to the parallel and distribution computing system field are as follows.

1) We defined an SGC model that supports the generalized common interface for the diverse requested scientific application. Even cloud users who are not familiar with the traditional cloud scientific systems can optimize the processing performance without in-depth understandings of system operation.
2) We adjusted a cloud broker model to the SGC, which can be retained as a third party by economic logic over the cloud echo system. Also, we proposed a cost-adaptive resource management scheme with a VM pool (VMP) management policy, which determine the amount of cloud resources with long-term payment plan to save expenses on resource allocation over the distribution on resource requests.
3) To address the multicriteria optimization problem in the scientific application, we proposed a workflow scheduling scheme with a division policy that compromise the tradeoff between objectives by combining the objectives into one performance metric, while maximizing task affinity over the heterogeneous cloud platform. Furthermore, with the task parallelization policy, the SGC is more tolerable over the unexpected failure and excessive service requirements.
4) We implemented key features for the SGC and evaluated performance through the experiments. The SGC with a cost-adaptive resource management scheme and a workflow scheduling scheme with a task division policy shows good performance characteristics in cost minimization while guaranteeing the processing performance. In addition, we discussed the feasibility and characteristics of each scheme in detail.

II. Model Description of the SGC

For scientific applications, there are various programming models, such as the thread model, the MapReduce model, the MPI model, and workflow [7], [8]. In this paper, we adopt the workflow as the programming model to execute scientific applications, so we assume that each of these scientific applications is represented as a workflow, which is a directed acyclic graph (DAG) composed of a set of nodes and a set of edges (called a scientific workflow).

To execute these scientific workflows in the cloud environment, as mentioned in the introduction, and overcome the problems of existing science gateway systems, we propose the new science gateway which has the following features shown in Table I.

In the next section, we discuss the proposed science gateway; we present its architecture and functional description, including these requirements. Furthermore, we describe the cost-adaptiveness of our science gateway with a resource provisioning scheme and workflow scheduling scheme with the division policy.

A. Architecture and Functional Description

The architecture and procedure of the proposed science gateway are shown in Figs. 1 and 2, respectively.

1) Workflow Scheduling Manager: The workflow scheduling manager is mainly responsible for managing and executing the scientific workflow. It provides a graphic user interface (GUI) for scientific application services for users. The applications are deployed by an administrator with additional QoS-related information such as the average execution time of each

TABLE I
REQUIREMENTS FOR OUR PROPOSED SYSTEM [3]–[6]

Requirement	Description
SLA-based dynamic workflow scheduling with parallelization	A dynamic workflow scheduling mechanism with parallelization is needed to guarantee the SLA required by users with the minimum resource and cope with the dynamic characteristic of cloud resource such as temporary performance degradation
Integrated cloud access management	An integrated interface to access the heterogeneous multiclouds in the programmatic way is needed to lease or release virtual machine in each cloud
Cloud resource provisioning	A cloud resource provisioning to lease VMs of long-term payment plan with resource demand prediction in advanced is needed to reduce the VM leasing cost

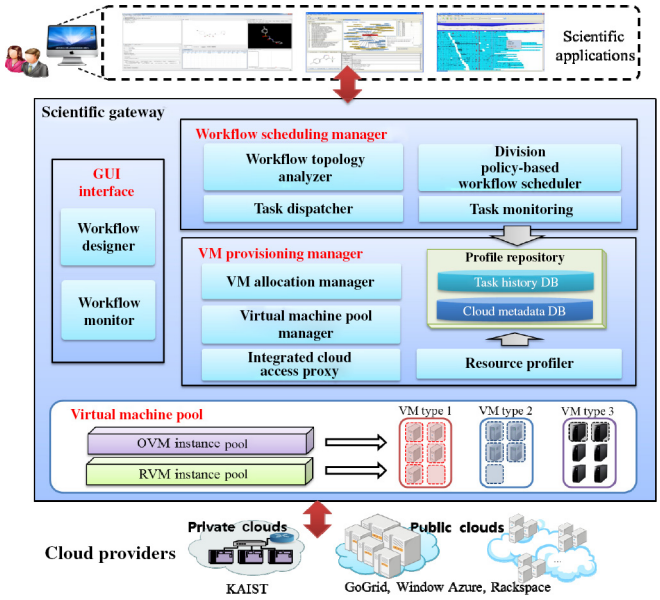

Fig. 1. Architecture of the proposed SGC with cost-adaptive resource management scheme.

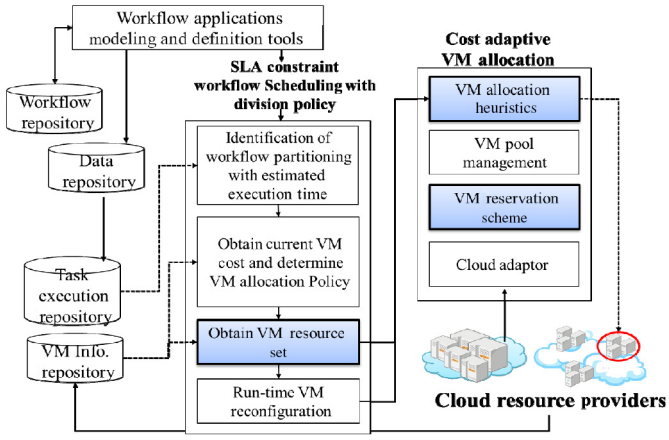

Fig. 2. Procedure of the proposed SGC to solve complicated scientific applications in the heterogeneous cloud platform.

task in the application. Using the GUI interface, a user can compose the workflow requests with a set of available tasks including their dependency and SLA requirement such as deadline. After parsing the user's submitted workflow request, the workflow scheduling manager partitions the entire workflow as simple tasks with its dependency and maps each task on a VM instance based on the division policy-based scheduling scheme. It executes the scheduled workflow dynamically using VM instances provided by VM provisioning manager. Finally, the result of the workflow execution is shown by GUI Interface.

2) VM Provisioning Manager: The VM provisioning manager operates a VMP and allocates a VM for a VM request. The VMP is defined as a logical VM container that stores a set of various types of VMs. It composes the reserved VMs (RVMs) of long-term payment plan on the VMP with resource demand prediction in advanced based on the resource provisioning scheme. Thus, the VM provisioning manager allocates the RVMs of VMP for a VM request. If there are no available RVMs in VMP, it allocates an on-demand VM (OVM).

3) Integrated Cloud Access Proxy: The integrated cloud access proxy provides the integrated interface to access the heterogeneous multiclouds and lease or release VM of different policies (i.e., OVM and RVM) on each cloud in the programmatic way.

Based on these functions, we propose and develop a resource provisioning scheme and a workflow scheduling scheme with a division policy in the workflow scheduling manager and the VM provisioning manager, respectively, in order to implement the cost-adaptive SGC. The resource provisioning scheme in the VM resource provisioning manager reduces the VM leasing cost, determining the amount of RVM which will be provisioned in VMP based on the demand of VM instance. Eventually, this organized VMP is used by the workflow scheduling manager to execute the scientific workflow. This scheme will be described in detail in Section III.

The workflow scheduling scheme with a division policy in the workflow scheduling manager decreases the cloud resource usage cost to execute the scientific workflow while guaranteeing the SLA (i.e., deadline) required by the user with the parallelization of the divisible task. Two examples of scientific workflows that include divisible tasks are presented in [14]–[17] as depicted in Fig. 3. The first example is the computational chemistry solution including quantitative structure–activity relationship (QSAR) analysis which calculates the toxicity and properties of chemical compounds using the similarity of chemical structure [14]. The second example is the bio-computing solution of next-generation sequencing (NGS) for genome analysis which aligns or maps low-divergent sequences against a large reference genome, such as the human genome [15]–[17]. In these cases, reducing the execution time of the divisible task through parallelization on several VMs, the science gateway can admit scientific workflow requests including high performance requirements. Also, scientific workflow processing can be more reliable on guaranteeing the performance requirement because the performance degradation caused by the dynamic characteristics of cloud performance can be compensated by parallelization. However, this parallelization increases the cloud resource usage cost to execute the scientific workflow, so a

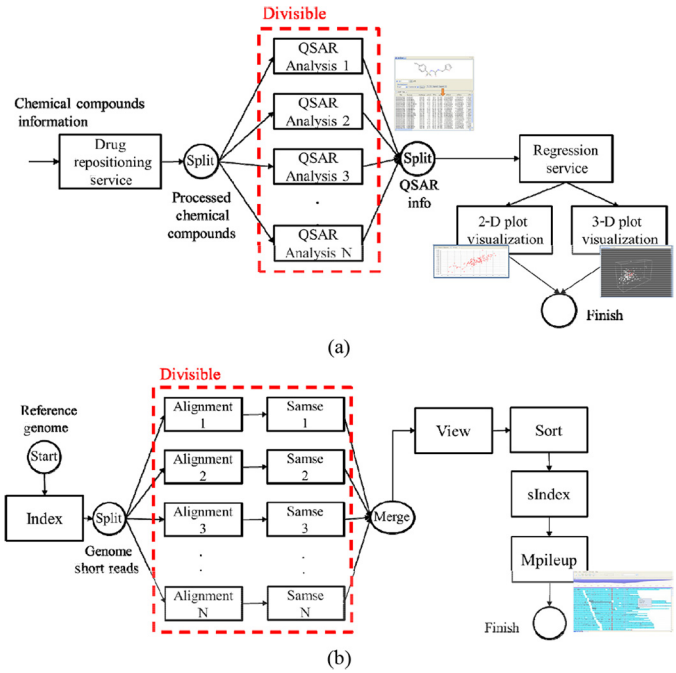

Fig. 3. Procedure of the scientific applications in topological expression (workflow) with diverse computing requirements. (a) Workflow topology of the computational chemistry solution with parallelized QSAR model analysis [14]. (b) Workflow topology of the NGS solution for genome sequencing analysis with high-throughput alignment method, BWA software package [15]–[17].

sophisticated technique to control parallelization considering the tradeoff between performance and cost is needed. This scheme resolves this problem. Furthermore, it considers the penalty cost which is the cost being proportional to the violated duration from the deadline. Through various penalty cost functions, it provides various levels of QoS. Section IV will explain this scheme in detail.

III. A Proposal for Cost-Adaptive Resource Management in SGC

For scientific application processing, it is important to reduce the resource leasing cost while guaranteeing the SLA of a user's request. From the perspective of the science gateway, an efficient way to reduce the resource leasing cost (i.e., maximizing profit) is to consider the payment plans of cloud resource providers: RVM and OVM plans. To do this, we propose the cost-adaptive SGC in this section. First, we introduce the pricing model and the cost-adaptive resource allocation of the science gateway for profit maximization. Second, the division-based scheduling algorithm of workflow, which is the representative scientific application by the science gateway, is described in detail.

A. Pricing Model for Scientific Computing

In our proposed framework, there are three entities: the SGC, cloud service users, and cloud resource providers. Users submit their scientific applications and data to the science gateway, the science gateway buys or releases cloud resources from/to the cloud resource provider as needed, and then the science gateway picks an appropriate VM instance from its own VMP to execute each task of the scientific applications. Cloud service users pay the SGC for their application processing requests. The SGC makes a profit while satisfying users' requirements as a wholesaler between cloud service users and cloud resource providers. Generally, scientific applications have precedence constraints. Therefore, each task is allocated to their suitable VM instances and executed in order of their starting time based on precedence constraints. Obviously, each task shows different performance in relation to different types of VM instances. Scientific applications may have different levels of importance and urgency. Therefore, users can specify different deadlines for their applications. Since the profit of the science gateway is largely dependent on the arrival density and the structure of scientific application requests, the adaptive pricing model for requests is important to maximize the profit. Variables for SGC are defined in Table II.

TABLE II
Variables for SGC

Notation	Description
t_i	A i^{th} task of scientific application S $i \in \{1,2,\ldots,N\}, N = number\ of\ tasks$ in S
VT_i	i^{th} VM instance type $VT \in \{1,2,\ldots,K\}$.
$R_i(\tau)$	The number of i-type available RVM instances in VM pool at time τ
$A_i(\tau)$	The number of i-type allocated RVM instances in VM pool at time τ
$N_i(\tau)$	The number of i-type leased RVM instances in VM pool from cloud resource provider at time t
$R_{i,j}$	The j^{th} RVM instance of i-type, $\sum_j R_{i,j}(\tau) = N_i(\tau)$
$C_i^{\max}$	The maximum price of i-type VMs to users
$C_i^{\min}$	The minimum price of i-type VMs to users
$C\left(R_{i,j}(\tau)\right)$	The current price of the j^{th} RVM instance of i-type at time τ

Definition 1: (SLA-constrained scientific application) From the users' perspective, they hope the scientific application they submitted can be finished within some specified deadline or budget. For example, if a user specifies deadline constraint D, that is to say, the user wants to run his or her scientific application no later than a specified deadline. Meanwhile, the user hopes that the scientific application should be finished with the least possible cost. In such a case, the scientific application can be described as a tuple $S(\Upsilon, \Phi, D)$, where Υ is the finite set of tasks t_i $(i \in \{1,2,\ldots,N\})$ and Φ is the set of directed edges of the form (t_i, t_j).

For an arbitrary precedence constrained scientific application $S(Y, \Phi, D)$, we assume that we are able to know the estimated completion time of each task of S, then we can obtain the optimized resource management policy of the science gateway for scientific application processing which satisfies the following objective function:

$$\text{Maximize } P[\rho] = \sum_{i \in I} c_{sal}(\rho) \cdot (\bar{r}(S_i(\Upsilon, \Phi, D)) - \delta) - c_{exp} \sum_{i \in I} r(S_i(\Upsilon, \Phi, D))$$

$$\text{Subject to } ECT[S_i] \le D_i \ \forall i$$

where

$$ECT[S_i] = \max_{\forall t_{i,j} \in S_i} \{ft[t_{i,j}]\} \ \forall i. \quad (1)$$

The objective $P[\rho]$ is a profit function of the science gateway with input parameter ρ, which is a policy for determining a price of service sales to user, c_{sal}. Here, c_{exp} $(c_{sal} \geq c_{exp})$ is the expenditure for processing a scientific application, δ is a coefficient, and $c_{sal} \cdot \delta$ means a preference degradation level of service purchasing from the science gateway. Also, $\bar{r}$ is an expected resource requirement in view of a cloud service user for scientific application request S_i (Υ, Φ, D), and r is an actual resource requirement in view of the SGC based on its VMP. Here, ECT is the estimated completion time of a scientific application, and ft is the estimated finishing time of an individual task, while $t_{i,j}$ is the jth task of application S_i. The equality constraint of (1) means that the finishing time of the last task is the completion time of the application.

The cloud resources are categorized into several types, such as small, medium, large, and xlarge with the different amount of computational resources. The various types of VM instances offer different processing capacities, and they are charged for usage in billing time units (BTUs) in proportion to their capacities. Partial-BTU consumption is rounded up to one BTU.

Definition 2. (Cloud VM resource type) A VM type VT_i can be modeled with four parameters, which are time-invariant and continuously capable of being guaranteed by cloud service providers: number of compute units (can be transferred into MIPS) VT_{c_i}, CPU clock rate (Hz) VT_{hz_i}, memory size (GBs) VT_{m_i}, storage space (GBs) VT_{s_i}, and bandwidth (bit rate, bit/sec) VT_{n_i}. The tuple represents a VM instance : $VT_i = \{VT_{c_i}, VT_{hz_i}, VT_{m_i}, VT_{s_i}, VT_{n_i}\}$.

In general, cloud providers have two VM instance payment plans, namely RVM and OVM plans [12], [13]. For an RVM plan, a VM instance is leased for a long BTU (e.g., monthly or yearly) with a low price per allocation time unit. On the contrary, for an OVM plan, a VM instance is allocated for a short BTU (e.g., hourly or daily) with a high price per allocation time unit. The science gateway maintains a certain number of RVMs in its own VMP and adjusts the number of them when the amount of requests drastically changes. Obviously, it is reasonable to prefer an RVM to allocate a task when we can find available RVMs in the VMP since OVM is more expensive than RVM. Because the resources are managed in two contradict payment plans, the SGC should determine the cost within two distinct levels: the lower cost on RVM and the higher cost on OVM. In order to decide the proper cost of service sales c_{sal} in (1) for maximizing the profit of the science gateway, various cost model to select sales cost between $C_i^{\max}$ and $C_i^{\min}$ over resource availability can be adjusted to the SGC. The various cost functions might show different characteristics with respect to fairness among the users and profit of the SGC. Any cost function can be chosen by the system administrator and is agreed by the service users. Because we only consider to measure the minimization of expenditure cost that come from the cost-adaptive resource management scheme represented in Section III-B and do not consider the fairness and maximization of sales cost in (1), we adjust simple cost function in the experiments as follows:

$$C(R_i(\tau))_{simple} = \begin{cases} C_i^{\min}, & R_i(\tau) > 0 \\ C_i^{\max}, & \text{otherwise.} \end{cases} \quad (2)$$

Algorithm 1. VMPM resource provisioning policy

INPUT : *historical data including* $S = \{\forall t_{i,j}\}$ *and* $\forall VT(t_{i,j})$ *during previous time interval* T'

OUTPUT : N_i during current time interval T

01: **For** $VT = 1$ to K
02: **For** $\forall t \in S$
03: $Cls_{VT} = Cls_{VT} \cup \leftarrow t \in S \; if \; VT(t) = VT$
04: **End for**
05: **End for**
06: $n = 0$
07: **For** $\forall Cls_{VT}$
08: sort $tasks$ in Cls_{VT} in order of their starting time
09: **While** available *tasks* exists in Cls_{VT} **do**
10: **For** $\forall t' \in Cls_{VT}$
11: $g_n = g_n \cup \leftarrow t' \in Cls_{VT} \; if \; st(t') \geq ft(t'')$,
12: $t' = first\ task\ in\ Cls_{VT}, t'' = last\ task$ in g_n
13: **End For**
14: remove tasks in g_n from Cls_{VT}
15: $n = n + 1$
16: **End while**
17: **End for**
18: **For** $\forall g_n$
19: $gct(g_n) = ft(g_n) - st(g_n)$
20: $VT = VT(g_n)$
21: **If** $\frac{C(RVM_{VT(g_n),gct(g_n)})}{\sum_{\forall t \in g_n} et(t) \cdot C(OVM_{VT(t)})} < 1$ **then**
22: lease $RVM_{VT(g_n),gct(g_n)}$ from cloud resource provider
23: **End if**
24: **End for**

Fig. 4. Algorithm of VMP management with a VM provisioning scheme.

B. Cost-Adaptive Resource Provisioning in SGC

We describe the VMP manager (VMPM) and its policies in terms of reducing the resource leasing cost. In particular, the VMPM determines the proper number of leasing RVMs from the cloud resource provider to optimize the cost efficiency. The number amount of running RVMs is dependent on the arrival request density of scientific applications and each usage duration.

Algorithm 1 shows the VMPM resource allocation policy to reduce the cost by using RVM leasing. The historical data including all the executed tasks and their allocated VM instance types during the previous time interval T' are the input data for Algorithm 1. We assume that the request pattern in the current time interval T will be the same as that one of historical data in T'. By using Algorithm 1, we can derive the proper amount of RVM, N_i for T.

From lines 01 to 05 in Algorithm 1, we first cluster the tasks in S according to their allocated VM instance type VT. Consequently, all the tasks in S are classified into several clusters Cls_{VT} according to VT. From lines 06 to 18, we make groups that have batches of nonoverlapped tasks. That is, from the arbitrary Cls_{VT}, the first task is picked if its start time st is later than the finish time ft of the last task in the group g_n, and then it is inserted into g_n. This procedure is repeated until we cannot find the available t in Cls_{VT} more. From lines 19 to 24 in Algorithm 1 by using

the group completion time of g_n, $gct\left(g_n\right)$ and allocated VM instance type of g_n, $VT\left(g_n\right)$, we make an RVM description $RVM_{VT(g_n),gct(g_n)}$having the BTU which has the unit size closest to the $gct\left(g_n\right)$. We check the following condition to choose whether to lease $RVM_{VT(g_n),gct(g_n)}$ from the cloud resource provider or not:

$$\frac{C\left(RVM_{VT(g_n),gct(g_n)}\right)}{\sum_{\forall t\in g_n} et\left(t\right)\cdot C\left(OVM_{VT(t)}\right)}<1. \tag{3}$$

The denominator of (3) represents the total cost of execution time et for the tasks in g_n on OVMs. The numerator of (3) represents the cost of RVM for g_n. If (3) is satisfied, it means that the leasing of RVM is more cost efficient than the leasing of OVMs for g_n. As the equation value in (3) is decreased, the cost efficiency by using RVM is increased.

IV. A Workflow Scheduling Scheme With a Task Division Policy

An SLA-constrained scientific application to be executed within user-specified deadline D is defined as a workflow scheduling problem with a deadline constraint. That is, deciding the assigned computing resources-to-be set $R=\{R_1,R_2,\ldots,R_n\}$ and assigned time set $\tau=\{\tau_1,\tau_2,\ldots,\tau_{\mathrm{n}}\}$ according to Petrinet model with intertask dependencies is defined as the problem.

It is hard to find a schedule to satisfy a user-specified tight deadline due to the finite set of resource types in cloud computing. Unlike workflow scheduling with CPU time competition on cluster computing, on-demand resource provisioning is operated on cloud workflow scheduling. Therefore, coarse-grained resource allocation with a finite set of resource types makes scheduling schemes more sophisticated for better resource utilization. Also, when a deadline is shorter than the earliest possible execution time of a workflow instance or unexpected processing delays occur above a certain level, we cannot guarantee the deadline to the limitation of an expressible quantity measure.

To overcome the problems in the simple management scheme, we utilize two methods: load-rate-based dynamic workflow scheduling with a token control scheme in Petrinet [18] and a task division policy with a profit-cost model [19], [20].

An SLA-constrained scientific application is given in definition 1. Here, $S\left(\Upsilon,\Phi,D\right)$ is transformed with the Petrinet workflow model $S\left(W\left(P,T,A\right),D\right)$, where *P* is place, *T* is transition, and *A* is arc [18], for better status expression and execution control with mathematical tools.

In addition, we define divisible task using the concept of task parallelization. A task t_i in a scientific application, which can be partitioned, as they do not have any precedence relations and all subtasks are identical, is defined as a divisible task [19]. The divisible degree (*dd*) of a divisible task is defined as the maximum number of possible subtasks. Then, the subtasks of divisible task t_i can be expressed as set $\{t_{i,1},t_{i,2},\ldots,t_{i,n}\}$. From the task parallelization, we can reduce the processing time by allocating distributed resources to the subtasks. Although the task division strategy reduces the time required for workflow scheduling, the division is not always carried out as it is not always appropriate to processing time, which can result in increase in the cost. Therefore, we should determine when we have to carry out division based on processing time and required cost. We only consider half-load division to lessen the complexity of scheduling and the management of profiling data. In addition, it is assumed that the load of each subtask is identical. The composition of partial subtasks on task type $\mathrm{tt_k}$ with amount r is defined as subtask type $tt_{k,\{r\}}$, whose $dd\left(tt_{k,\{r\}}\right)=r$. Additional application profiling for all kinds of task bunches is required to measure the execution time of each subtask type $tt_{k,\{r\}}$ on a different resource type VT_i.

When a token forward and backward matrix is already extracted by analysis of workflow topology, an SLA-constrained scientific application in the Petrinet scheme $S\left(W\left(P,T,A\right),D\right)$ can be controlled by moving the token with the multiplication of the token status vector and the token forward/backward matrix.

1) [Phase I] Calculate the Load Rate r(p) for Each Placement: We set the initial marking of the token vector for the first phase as $m=[0\ldots01]$. Then, the token moves through the backward matrix along the Petrinet topology path reversely while investigating each task's load rate. The load rate $r\left(p\right)$ on a place p is the rate of the following transition's relative load compared to the relative load of its critical path

$$r\left(p\right)=\frac{rl\left(p\right)}{cpl\left(p\right)}. \tag{4}$$

Also, the relative load is defined as the average execution time for a task over entire VM types

$$rl\left(p\right)=\mathrm{avr_j}\left(\tau_{VT_j}^{type\left(p^*\right)}\right)=\frac{1}{\mathrm{m}}\cdot\sum_j\tau_{VT_j}^{type\left(p^*\right)}. \tag{5}$$

In addition, the critical path on a task is defined as the set of following tasks, which is composed of the biggest relative load [19]. Then, we can figure out the critical path load

$$cpl\left(p\right)=f\left(x\right)=\begin{cases}\max\limits_{\mathrm{p}^{**}}\ cpl\left(p^{**}\right)+rl\left(p\right), & if\ p^{**}\ exist\\ rl\left(p\right), & otherwise.\end{cases} \tag{6}$$

2) [Phase II] Allocate the Most Cost Efficient Resource With Properly Assigned Sub-Deadline: We set the initial marking of a token vector for the second phase as $m=[10\ldots0]$. Then, the token moves through the forward matrix along the Petrinet topology path while allocating the cheapest VM that can guarantee the estimated sub-deadline (sd). To achieve successful scheduling with guarantee of the entire deadline, a properly assigned sd for each task should be observed. Therefore, we allocate an *sd* rationally, based on the remaining time and the load rate when $\tau\left(\mathrm{M}\right)$ is the execution timestamp with token status m

$$sd\left(p^*\right)=rl\left(p\right)\cdot\left(D-\tau\left(m\right)\right). \tag{7}$$

When the leasing cost per unit time for arbitrary VM type VT_i is denoted as C_{VT_i} and there are known estimated times for task execution times on each VM type

Algorithm 2. Task Division Policy

Input: *workflow topology tp, workflow topology with half division tp', current execution token matrix m, current execution time $T_C(m)$, workflow deadline D, budget B, token forward matrix F*

Output: *Decision of division*

Copy execution token set m onto SV estimation token set m_e and copy current execution time $\tau(m)$ onto temporal current execution time $\tau_t(m_e)$.

01: **For** workflow topology *{tp, tp'}*
02: **While** m_e is not equal to [0 … 0 1] **do**
03: Let P= $\{p_1, p_2, \ldots, p_n\}$ be places which has SV estimation tokens at current stage i and estimation token set at stage i as m_e^i
04: **For** p = each element in P
05: $\mathrm{sd}(p^*) = rl(p) \cdot T_t(D - m_e^i)$
06: $p^*.temporal\_vm = VT_{\left\{j \middle| \tau_{VM_j}^{type(p^*)} < sd(p^*), min\left(c_{VT_j} \cdot \tau_{VM_j}^{type(p^*)}\right)\right\}}$
07: Let $Tr = \{t_1, t_2, \ldots, t_k\} =^* p$
08: $\tau_{tc}(m_e^i) = \tau_{tc}(m_e^{i-1}) + max_j\left(\tau_{t_j.temporal\_vm}^{type(t_j)}\right)$
09: **End for**
10: $m_e^{i+1} = F \cdot m_e^i$ //token forwarding
11: **End while**
12: **If** $T_{tc}(m_e) - D > 0$ **then**
13: $c_{sv} = \alpha + \beta \cdot SV = \alpha + \beta \cdot (\tau_{tc}(m_e) - D)$
14: **Else**
15: $c_{sv} = 0$
16: **End if**
17: $c_v = \sum_i C_{t_i.temporal\_vm} \times \tau_{t_i.temporal\_vm}^{type(t_i)}$
18: $Pf = B - c_v - c_{sv}$
19: **End for**
20: **Return** $Pf < Pf'$ // If $Pf < Pf'$, return value is true. Otherwise return value is false

Fig. 5. Algorithm of a task division policy in workflow scheduler.

$\tau_{\mathrm{VM_i}}^{type(p^*)}$, we can allocate the most efficient resource guaranteeing the sd for the jth task from leasing VM with the type of $VT_{\left\{i \middle| \tau_{\mathrm{VM_i}}^{type(p^*)} < sd(p^*), \min\left(c_{VT_i} \cdot \tau_{\mathrm{VM_i}}^{type(p^*)}\right)\right\}}$.

If there is no available resource type to guarantee the sd, this may cause deadline violation for the entire workflow. Therefore, we apply a task division strategy based on profit calculation in phase III.

3) [Phase III] Task Division With Profit Calculation: As mentioned before, we should determine when we have to proceed with division based on processing time and required cost to avoid over-provisioning and burdensome management cost.

To comparatively evaluate solutions of the constraint problem only in the objective domain, we consider the service-level violation penalty on SLA constraint. We define the profit-cost model to maximize profit while proceeding with task division

$$Pf = B - c_v - c_{sv}. \tag{8}$$

In the formula above, *Pf* indicates profit, $B$is the budget supplied by the user, C_l is the total cost of leasing VM from the cloud service providers, $c_v = \sum_i c_{uv}^i \cdot \tau_u^i$.Penalty cost c_{sv}, which is caused by service-level violation *SV*, is represented as follows:

$$c_{sv} = \begin{cases} \alpha + \beta \cdot SV, & \text{if } SV > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$SV = \begin{cases} ECT - D, & \text{if } ECT - D > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

In (9) and (10), variable *SV* indicates the degree of SLA violation. There are many models of violation penalty cost, but in this paper, we use the linear violation penalty model in (9) [20]. As shown in (10), *SV* can be described as subtraction deadline D from the estimated completion time ECT.

By estimating the SLA violation SV over the nondivision case and the SV′ overdivision case in a deterministic way, we can compare the profit for making a decision of division. We define the SV estimation token $\mathrm{M_e}$ that is a clone of the current token status *m*, to calculate the penalty cost. Initially, the location of m_e is replicated from the current execution token m. Also, each token saves a temporal execution timestamp $\tau_t(m_e)$, which is replicated from the current execution time $\tau(m)$. By forwarding the SV estimation token with the allocation of the temporal VM over the estimated sd, we can cumulate the $\tau_t(m_e)$ for each transition with the scheme described in phase II until the token reaches the final destination. Then, we can get the estimated execution time ECT from timestamp $\tau_{\mathrm{t}}(m_e)$ in a heuristic way. With the comparison of profit between the nondivision case and division case (half division), we can make a cost-efficient decision. Therefore, we should calculate profit P for a nondivision case and profit P′ for a division case based on the knowledge of SV and SV′, which are, respectively, acquired from SV estimation tokens.

If $Pf < Pf'$, half division is applied and phase II is repeated. If a division case does not yet guarantee the deadline, division can be carried out recursively until the tasks are not further divisible (dd equals 1). Otherwise, the biggest VM is allocated to transition and return to phase II to forward the token to the next place P**.

V. Performance Evaluation and Discussion

A. Test Environments for Performance Evaluation

In this section, to evaluate the performance of the proposed resource management scheme and workflow scheduling scheme which provide efficient resource operation in the SGC, we implemented the SGC with heterogeneous cloud platforms as shown in Fig. 6. The SGC is a broker between users and providers and is able to process enormous scientific applications efficiently upon heterogeneous cloud resources. When a user coordinates the procedure of their scientific application as workflow instance, the user can make contract of workflow execution with the SGC through a negotiation interface. With the agreed service level including the deadline constraint and the cost policy on the contract, the workflow is analyzed in a set of tasks.

As for the cost-adaptive resource management policy proposed in Section III, the SGC can significantly reduce the

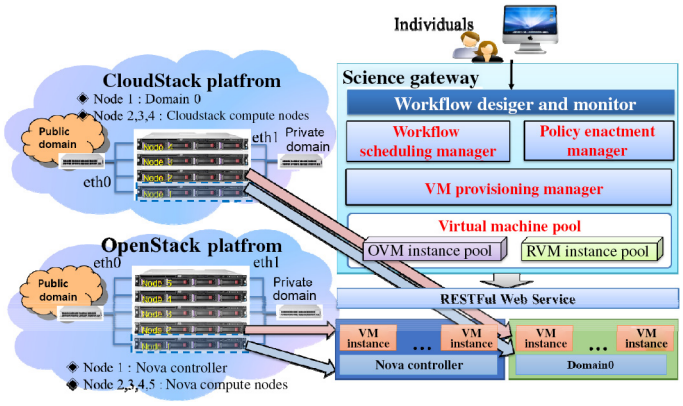

Fig. 6. Test environments for the proposed science gateway system to evaluate cost-adaptive VM management schemes, such as the VM Provisioning scheme and the workflow scheduling scheme with the task division policy.

resource allocation cost without performance degradation in scientific application processing under various VM billing policies among the cloud service providers. In addition, by the scheduling scheme of workflow with task parallelization proposed in Section IV, the SGC allows us to improve the processing performance, and also to guarantee both of deadline assurance and cost minimization.

As described in Table III, we considered specific experiments' set for the evaluation. We designed the experiment 1 composed of tests 1, 2, and 3, respectively. We adjust various input parameters including workflow service level (e.g., deadline), penalty function parameters, and arbitrary delay factors, which impose excessive penalty burden for resource management. Although the Petrinet-based phased workflow scheduling scheme works appropriately with the profit function-based resource allocation, systematic counter-measure to the radical circumstance is examined for the implementation feasibility. Also, we designed the experiment 2 with test 4 to compare the VM allocation cost efficiency between the static RVM management scheme and dynamic RVM management scheme with log-based analysis of historical data over distinct workload and with combination of system operation policies.

1) Workload Parameter, λ: When the scientific application is composed of the finite set of tasks t_i $(i \in \{1, 2, \ldots, N\})$ and the set of directed edges of the form (t_i, t_j) defined in definition 1, we assumed that the workload of the scientific applications is requested to the SGC in Poisson distribution Pois(λ) with certain expectation on the number of events' occurrence within a fixed interval, λ.

2) Mean of Service Level, μ: The input parameter "service level" denotes certain requirements given to individual scientific application, especially deadline. The "service level" functions as a constraint to scheduling problem and may ask compensation cost for a contract between the user and the broker when service-level violation occurs. The service level in each scientific application is given to distribution of Normal distribution N (μ, σ^2) with expectation value μ, to reflect various service users' requirement.

3) Penalty Function Parameters, α, β: The penalty function is adjusted for billing compensation cost when

TABLE III
PARAMETER SETS AND PERFORMANCE METRICS FOR THE EXPERIMENTS

Parameters	Experiment 1			Experiment 2
	Test 1	Test 2	Test 3	Test 4
Workload parameter, λ (tasks/hour)	(18,12,…,4)	(18,12,…,4)	(18,12,…,4)	(36,24,…,7.2)
Mean of service level, μ [deadline, (s)]	(500,700, 900,1100)	800	800	700
Penalty function parameters, α, β	β=5, α=0	β=(1,3,5,7), α=0	β=5, α=0	β=5, α=0
Degree of failure, γ	1.0	1.0	(1.0,1.2, 1.4,1.6)	1.0
Amount of operational RVM	0	0	0	10
Performance metric				
VM allocation cost (relative-cost)	$c_v = \sum_i c_{uv}^i \cdot \tau_r^i + \mathrm{n_{rvm}} \cdot c_{urvm} \cdot \tau_{total}$			
Service level violation (penalty) cost (relative-cost)	$c_{sv} = \sum_{j\mid \tau_s^j - \tau_{sl}^j > 0} \left(c_{usv}^j \cdot (\tau_s^j - \tau_{sl}^j) + \alpha\right) = \sum_{j\mid t_{sv}^j > 0} \left(\beta \cdot \tau_{sv}^j + \alpha\right)$			
Operation Cost (relative-cost)	$c_t = c_v + c_{sv}$			
Average service level violation (s)	$avr(\tau_{sv}^j)$, where $\tau_{sv}^j > 0$			

service-level violation occurs for failure to keep contract condition. In this experiment, we applied linear function as penalty function defined in (9). When the slope value "β" and the constant value "α" are excessive compared to VM allocation cost, the system should aggressively avoid service-level violation.

4) Degree of System Failure, γ: The degree of system failure γ is designed for generating arbitrary failures on a system with the generation of arbitrary delay on task execution time by multiplying γ on actual execution time of the task. The failure is defined as miss-prediction of task execution time with task profiling data, caused by temporal resource unavailability or unexpected errors on a system. Because of frequent occurrence of the failures in distributed computing environment, it should be appropriately controlled within the scheduling scheme.

5) Amount of Operational RVM: Lastly, the parameter stands for the number of initially managed RVMs for static resource pool management scheme literally. However, in case of dynamic resource managed scheduling, the number of managed RVMs might be controlled with regard to the workload.

To evaluate the cost efficiency on proposing scheme, we define relative cost that does not have monetary meaning in reality, but shows relative performance index for comparison between algorithms or models (11). The unit time cost is amount of payment on specific time period for imposing price on service that is agreed on the contract. Then, the definition of relative cost in the ith contract is described as multiplying unit time cost on the ith contract c_{u}^i by unit time consumption on the ith contract τ_{u}^i and addition of constant value a, where i is the sequence number of resource contract for the execution of specific workflow

$$RC^i = c_u^i \cdot \tau_u^i + a. \tag{11}$$

TABLE IV
SPECIFIC CONFIGURATIONS ON CLOUD TESTBED ENVIRONMENT

		OpenStack platform	CloudStack platform
H/W specification		Intel Xeon E5620 2.40 GHz,Core 16, MEM 16G, HDD 1T, 5 node	Intel Core i7-3770 CPU 3.40 GHz, Core 8, MEM 16G, HDD 1T, 4 node
S/W specification		OS: Ubuntu 14.04, Hypervisor: KVM	OS: CentOS 6.0, Hypervisor: XEN
VM types	Small	Spec: 1 VCPU, 2 GB MEM, 80GB disk Unit time cost: 2 RC per second	
	Medium	Spec: 2 VCPU, 4 GB MEM, 80GB disk Unit time cost: 4 RC per second	
	Large	Spec: 4 VCPU, 8 GB MEM, 80GB disk Unit time cost: 8 RC per second	
	c4.small	Spec: 4 VCPU, 1 GB MEM, 80GB disk Unit time cost: 4 RC per second	
	m8.small	Spec: 1 VCPU, 8 GB MEM, 80GB disk Unit time cost: 4 RC per second	

The contracts can be made from the each resource contract on the workflow scheduling, also from the imposing penalty on service-level violation and even from the long-term VM reservation. Unlike the billing contract with hourly policy on real cloud service domain, we assign unit time as a second for the minute examination on the schemes. When the specification of resource can be simplified as a tuple $[r_c^i, r_m^i]$ of ith resource contract (where r_c^i is the number of CPU cores and r_m^i is the numeric value of RAM size in GB) and the weight vector $\vec{w} = [w_c, w_m]$ to apply the effectiveness of each element on tuple is presented, the unit time cost on VM allocation c_{uv}^i is calculated as follows:

$$c_{uv}^i = w_c \cdot r_c^i + w_m \cdot r_m^i. \tag{12}$$

In case the weight vector is assigned as [0, 0.5], the unit costs for each of VM instance types are calculated in Table IV. Then, relative cost on VM allocation can be represented in (13), where τ_r^i is unit time consumption on ith resource contract

$$c_v^i = c_{uv}^i \cdot \tau_r^i. \tag{13}$$

In addition, we define the relative cost on *j* th service-level violation as shown in (14) by using the penalty cost model in (9) where *j* is the sequence number of the workflow, c_{sv}^j is unit time cost on service-level violation, τ_{sv}^j is degree of service-level violation, and α is constant relative cost imposed per violation

$$c_{sv}^j = \begin{cases} c_{usv}^j \cdot \left(\tau_s^j - \tau_{sl}^j\right) + \alpha, & \tau_s^j - \tau_{sl}^j > 0 \\ 0, & otherwise. \end{cases} \tag{14}$$

6) VM Allocation Cost and Penalty Cost: The performance metrics "VM allocation cost" and "penalty cost" are defined in relative cost referred to [19]. Therefore, operation cost on a system can be compared within same performance metric.

7) Average Service-Level Violation: Finally, the performance metric "average service-level violation" stands for the degree of violation only for encountered failures that is unable to keep the service level.

We built the two cloud platforms with five computing nodes for the OpenStack and four computing nodes for the CloudStack, respectively. Due to the heterogeneity in the multiple cloud services (e.g., Amazon EC2, GoGrid, Windows Azure), various VM operation policies are implemented, while each platform is constructed with nodes of different hardware and software as illustrated in Fig. 6 and Table IV [21], [22]. Cloud OS systems (i.e., OpenStack and CloudStack) provide the VM lifecycle management (i.e., create, terminate, pause, reboot, snapshot) through the network service, volume service, scheduling service, image service, and virtualization.

TABLE V
MACHINE AND TASK HETEROGENEITY ON TESTBED [24], [25]

	Machine heterogeneity
OPENSTACK	14324.42835
CLOUDSTACK	35042.29707
TOTAL	23949.1633

	Task heterogeneity
NGS	85910.10043

Under the experimental environment, we orchestrated high performance scientific application as NGS with burrows-wheeler aligner (BWA) software package, as illustrated in Fig. 3(b). We orchestrated NGS with the BWA as a scientific workflow. NGS is used to determine the order of the nucleotide bases in DNA molecules. In addition, it is used to analyze the resulting sequences to understand biological phenomena and the relation between genotype and phenotype. In particular, BWA is a pipelined set of tasks for analyzing the genome by using the Burrows-Wheeler transform compression techniques. Similar to a typical scientific application, BWA is also computing/data intensive, time consuming, and divisible to some degree [14]–[16].

From the application profiling on BWA applications [23], data collected on each service are expressed as expected execution time matrix T. The matrix gathers the expected execution time of each task on heterogeneous VM types for partial tasks under the testbed environment. The matrix analysis shows performance diversity over the distributed cloud resources [24], [25].

In the conventional heterogeneity computing researches [24], [25], they defined degree of expressing heterogeneity of machines and tasks to show the effectiveness of schemes in various environment. The machine heterogeneity is degree of the machine execution times varying for a given task and can be calculated as averaging variations of each row on the matrix T. Similarly, the task heterogeneity is the degree of the task execution times varying for a given machine and can be calculated as averaging variations of each column on the matrix T. As shown in Table V, the testbed environment shows high machine heterogeneity and high task heterogeneity.

B. Experimental Results and Discussion

In the experiments, we evaluate proposing framework in two criteria.

1) We focused on evaluating increase in task affinity for efficient cost-performance resource utilization in heterogeneous cloud while guaranteeing service level represented in Section IV. The task division policy in Fig. 5 enlarges the solution set of workflow scheduling with task parallelization with profit model [refer to (8)]. In addition, the division policy simplifies constraint workflow scheduling problem into consideration of single metric,

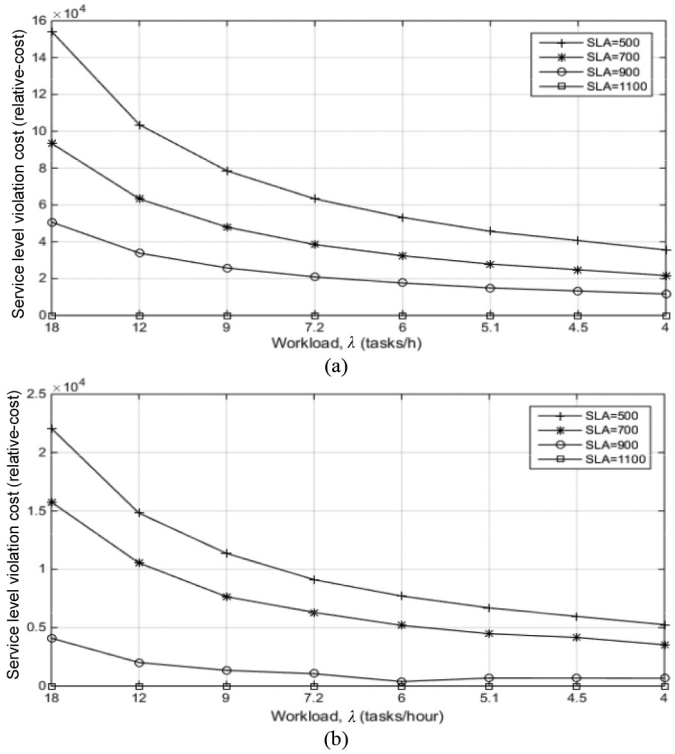

Fig. 7. Test 1—comparison of service level violation cost w.r.t. different deadline requirements as 500, 700, 900, and 1100 s over various workload. (a) Without task division policy. (b) With task division policy.

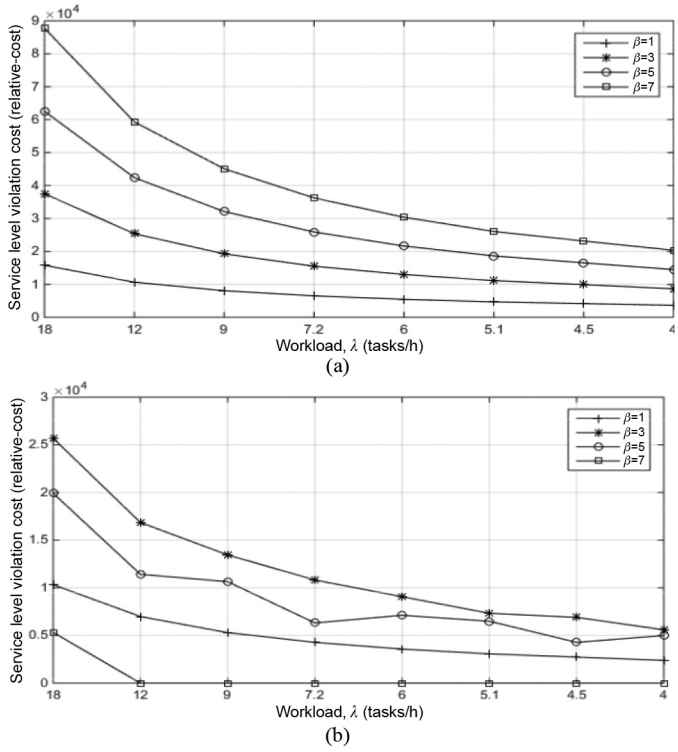

Fig. 8. Test 2—comparison on service level violation cost w.r.t. different parameters on the penalty cost function as $\beta = 1,\ 3,\ 5,\ 7$ [refer to (9)]. (a) Without task division policy. (b) With task division policy.

concluded in finding optimal resource planning to execute workflow with high cost-performance efficiency upon the constraint. The evaluations focused on tolerance of the division policy over tight constraint and system failure.

2) The evaluation focused on the VM allocation cost through RVM management with VMP management scheme in Section III (Fig. 6), cost-adaptive resource manager.

In test 1, with different workload of contracts on workflow execution in an exponential distribution from 18 to 4 tasks/h, we measured the relative cost of service-level violation according to the different SLA requirements of schemes "without division policy" in Fig. 7(a) and "with division policy" in Fig. 7(b). Because there is no dependency between the workflow scheduling due to avoidance of competition for resources through the provision of abundant resources from the multiple cloud services over worldwide providers, we can obviously figure out the decline of relative cost over the longer interarrival time accordingly by considering a small workload when experiment times are the same. When we compare Fig. 7(a) and (b), the scheduling with the division policy shows a lower cost demand for the same workload, and we found that there is about an 85% decrement of service-level violation cost on average. As service-level requirements become tight, penalty cost c_{sv} of (9) is increased with SLA Violation $SV$of (10). When the SV is higher than the certain value, it makes more profit to divide and parallelize a task with additional resource, which means the increase in the VM leasing cost c_{v}, in order to reduce SV and c_{sv} and guarantee the SLA requirements. Because the decrease of c_{sv} is higher than the increase of c_v in that case, the scheduling with the division policy can perceive that case with comparison to the cost with division or no and do the task parallelization adaptively in order to maximize the profit of (8) and guarantee the SLA requirements. However, the scheduling without division policy cannot provide this mechanism, so the SV in the tight SLA requirements keeps increasing. As a result, our proposed scheme shows lower values on the service-level violation cost as shown in Fig. 7.

In test 2, with a different workload of contract on workflow execution in exponential distribution from 18 to 4 tasks/h, we measured the relative cost of service-level violation according to the different parameters of the penalty cost function for both the schemes "without division policy" in Fig. 8(a) and "with division policy" in Fig. 8(b). When a user imposes excessive penalty cost on service-level violation, c_{sv} is rapidly increased depending on SV according to (9), so even the small increase of SV makes high cost. In this case, the division policy might try to avoid the violation by an intensive rate of division to reduce the c_{sv} and guarantee the SLA Requirement. Although the additional VM leasing cost is required in this way, the decrease of c_{sv} by violating SLA Requirement is higher than the increase of c_v by task division; therefore, the scheduling with division policy decides to divide and parallelize a task to maximize its profit (8) and guarantee the SLA Requirement. On the other hand, when an insignificant penalty cost function is imposed, the division policy might not divide and parallelize the task with the comparison of penalty cost and the leasing cost in the profit model described in (8). The reason is that the increase of c_{sv} by violating SLA Requirement is higher than the decrease

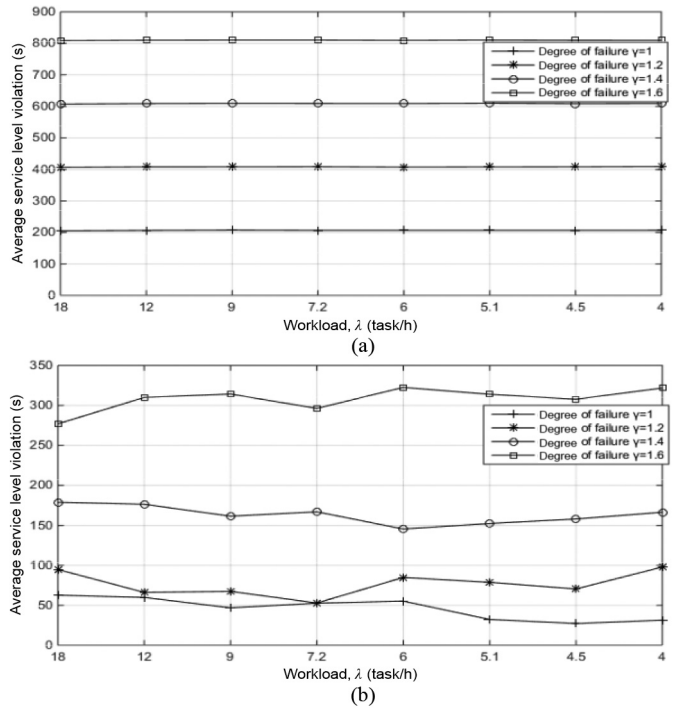

Fig. 9. Test 3—average service level violation w.r.t. degree of failure γ as 1, 1.2, 1.4, and 1.6 (i.e., workflow with failure is obtained by multiplying γ on execution time of each task in the workflow instance). (a) Without task division policy. (b) With task division policy.

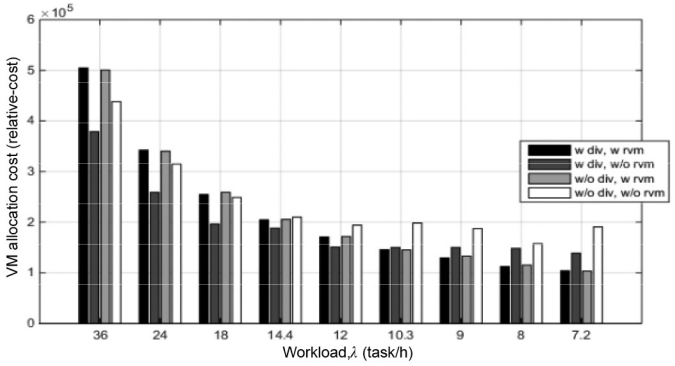

Fig. 10. Test 4—relative cost on VM allocation cost with different scheduling policies (i.e., with/without task division policy, with/without RVM management).

of c_v by task division in this case. The beta value indicates the imposed relative cost per second for service-level violation. Unlike the increment of the violation cost in proportion to beta in Fig. 8(a), the division policy in Fig. 8(b) selects the best solution of division with the profit-cost model. As the beta value becomes higher, the certain value of SV which triggers the division in our proposed policy becomes lower; hence, c_{sv} also becomes lower. Furthermore, in the extreme case that beta is 7, our proposed policy does not allow SLA Requirement to violate by radical division because of the excessive penalty cost as shown in Fig. 8(b). However, the scheduling without division policy cannot treat it adaptively on the beta value. As a result, our proposed scheme shows about 50% improvement of cost efficiency for service-level violation on average compared to workflow scheduling without division.

In test 3, with different workload of contract on workflow execution in exponential distribution from 18 to 4 tasks/h, we measured average service-level violation according to the difference degree of failure γ on each task for both "without task division policy" in Fig. 9(a) and "with task division policy" in Fig. 9(b). To figure out the influence of unexpected failures or delays in distributed cloud computing, we simulated artificial delay with imposing arbitrary delay on workflow by multiplying degree of failure γ on execution time of each task in the workflow instance. Then, we can figure out that the case without division cannot handle the additional burden from delay and show result of excessive service-level violation. On the other hand, the proposed scheme shows its capability to handle unexpected failures in some degree. Because scheduling with division policy decides whether the division policy is applied or not before executing each task on a workflow request to maximize its profit (8) and guarantee the SLA requirement, it can treat the unexpected failures or delays adaptively. We expect that if there are more divisible tasks in workflow instance, system will have more capacity to cover failures.

In test 4, with different workload of contract on workflow execution in exponential distribution from 36 to 7.2 tasks/h, we measured the VM allocation cost according to different policies to determine the effect on RVM management over various policies. From the cost efficiency of an RVM with a long-term leasing contract, we can operate a cost-adaptive resource management system with RVM management. However, RVM cannot be operated in an on-demand manner because it can waste utilization of resource when it is idle. Obviously, we can realize how the influence of cost-adaptive resource management on division policies varies with various interarrival times. Generally, the resource provisioning policy without division policy shows better synergy on the VM allocation cost than the others. From the detailed analysis of raw data, we found that the division policy burst the simultaneous VM requests and run out of the RVM in resource pool because the amount of operational RVMs determined as the average of arrival VM demand by resource provisioning policy is usually lower than the temporary burst demand. Therefore, this phenomenon induces creating more OVM for the additional VM used by task division. Eventually, the case with division has a poor performance on the VM allocation cost compared to the case without division which makes the best use of RVM, although the workloads are the same in both cases. However, when we consider the operation cost calculated by the addition of VM allocation cost and service-level violation cost, the resource provisioning policy with division policy always show better cost efficiency than the others. Also, the results of tests 4 and 5 demonstrate the relationship of the workload and number of leased RVMs.

Finally, we derived the requirements of the VM provisioning scheme as shown in Fig. 6 in terms of computational time complexity that stands for operational feasibility. We assumed that historical data are already stored by VM types and are also sorted in chronological order. When we denote number of VM types as K, number of historical data on each VM type as N, the

computational complexity can be derived as at most $O\left(KN^2\right)$ that is acceptable time consuming in implementation domain. Moreover, we explore the computation time complexity on the task division policy in workflow scheduling that is depicted in Fig. 5. The task division policy is comparison algorithm of the operational profit between task division case and nondivision case. The policy traverses a DAGs with the different topologies based on control logic represented in the Petrinet model. In our scheme, the traversal technique of the Petrinet topology can be identical to Breadth-First Search (BFS) algorithm when we cut out the edges that make visit same vertex twice. The computational time complexity of BFS algorithm is already derived as $O\left(|V|+|E|\right)$ where the variable $|V|$ denotes the number of vertex and the variable $|E|$ denotes the number of edge, respectively [26]. Therefore, we can conclude that the task division policy is also a practical solution.

VI. Conclusion

In this paper, to address the difficulties on the highly scalable and flexible system for processing various scientific applications, we designed the SGC platform that is able to process multiple scientific applications concurrently upon heterogeneous cloud environment while considering user-specified constraints. Furthermore, we adjust cloud broker model to the SGC, which mediate VM among multiple cloud service providers and users to provide better resource availability. In the cost-adaptive resource management scheme, we proposed a VM provisioning scheme that determines the amount of operational RVMs with respect to the distribution of arrival. In the experiments, the results showed that the proposed scheme can reduce the VM allocation cost by about 20% compared to the system without the scheme. However, it can be rather inefficient for wrong decision of overprovisioning in case of static resource pool management. Especially, the burst-typed pattern on the request showed cost performance degradation according to run out of the resource pool. Also, we proposed a workflow scheduling scheme with the division policy. To overcome the hardness on finding a resource schedule in heterogeneous cloud environment, we integrated multiple objectives into one performance metric. Our experiments showed that the workflow scheduling scheme with the division policy decreases the VM allocation cost while guaranteeing the service level required by the user. Even though there are unexpected failures caused by delay with resource unavailability, division policy operated with profit function and Petrinet control make the SGC tolerable. We expect that the proposal SGC might provide cloud ubiquitous processing by interworking between the cloud data center and mobile devices via simplified interface. The cloud users might be able to submit, monitor, and manage their scientific application processing on VM instance anytime, anywhere.

References

[1] M. Chen *et al.*, *Big Data: Related Technologies, Challenges and Future Prospects*. New York, NY, USA: Springer, 2014, ISBN: 978-3-319-06245-7.

[2] M. Chen, M. Shiwen, and L. Yunhao, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, Apr. 2014.

[3] Y. Ren, S.-H. Kim, D.-K. Kang, B.-S. Kim, and C.-H. Youn "A cost-efficient job scheduling algorithm in cloud resource broker with scalable VM allocation scheme," *KIPS Trans. Softw. Data Eng.*, vol. 1, no. 3, pp. 137–148, 2012.

[4] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Proc. 10th Int. Symp. Pervasive Syst. Algorithms Netw.*, 2009, pp. 4–16.

[5] B. Kim *et al.*, "An adaptive workflow scheduling scheme based on an estimated data processing rate for next generation sequencing in cloud computing," *J. Inf. Process. Syst.*, vol. 8, no. 4, pp. 555–566, 2012.

[6] Y. Han, C.-H. Youn, and D. K. Kang, "Adaptive management framework for scientific workflow applications," in *Proc. 6th Int. Conf. Ubiq. Inf. Manage. Commun. (ICUMIC'12)*, 2012, Art. ID 6-8.

[7] W.-J. Kim, D.-K. Kang, S.-H. Kim, and C.-H. Youn "Cost adaptive VM management for scientific workflow application in mobile cloud," *Mobile Netw. Appl.*, vol. 20, no. 3, pp. 328–336, 2015.

[8] M. Mao and M. Humphrey, "Scaling and scheduling to maximize application performance within budget constraints in cloud workflows," in *Proc. 27th Int. Symp. Parallel Distrib. Process. (IPDPS)*, 2013, pp. 67–78.

[9] Science Gateway [Online]. Available: http://www.sciencegateway.org/

[10] J. Yao *et al.*, "Facilitating bioinformatic research with mobile cloud," in *Proc. 2nd Int. Conf. Cloud Comput. GRIDs Virtual. (CLOUD COMPUTING'11)*, 2011, pp. 161–166.

[11] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in *Proc. IEEE 4th Int. Conf. eSci.*, 2008, pp. 222–229.

[12] D.-K. Kang *et al.*, "Enhancing a strategy of virtualized resource assignment in adaptive resource cloud framework," in *Proc. 7th ACM Int. Conf. Ubiq. Inf. Manage. Commun. (ICUMIC)*, 2013, Art. ID 9-5.

[13] S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Apr./Jul. 2012.

[14] C. Nantasenamat, C. Isarankura-Na-Ayudhya, T. Naenna, and V. Prachayasittikul, "A practical overview of quantitative structure-activity relationship," *Excli. J.*, vol. 8, pp. 74–88, 2009.

[15] BWA [Online]. Available: http://bio-bwa.sourceforge.net/

[16] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings Bioinf.*, vol. 11, no. 5, pp. 473–483, 2010.

[17] Z. Su, "Next-generation sequencing and its applications in molecular diagnostics," *Expert Rev. Mol. Diagn.*, vol. 11, no. 3, pp. 333–343, 2011.

[18] Z. Xiao and Z. Ming, "A method of workflow scheduling based ons colored Petri nets," *Data Knowl. Eng.*, vol. 70, no. 2, pp. 230–247, 2011.

[19] M. Malawski *et al.*, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IAAS clouds," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2012.

[20] L. Wu, S. K. Garg, and R. Buyya, "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in *Proc. 11th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput. (CCGrid)*, 2011, pp. 195–204, Art. ID 22.

[21] Openstack [Online]. Available: http://www.openstack.org/

[22] Cloudstack [Online]. Available: http://cloudstack.apache.org

[23] K. Hoste *et al.*, "Performance prediction based on inherent program similarity," in *Proc. 15th ACM Int. Conf. Parallel Archit. Compil. Techn.*, 2006, pp. 114–122.

[24] S. Ali *et al.*, "Task execution time modeling for heterogeneous computing systems," in *Proc. 9th Heterogen. Comput. Workshop*, 2000, pp. 185–199.

[25] S. Ali *et al.*, "Representing task and machine heterogeneities for heterogeneous computing system," *Tamkang J. Sci. Eng.*, vol. 3, pp. 195–207, 2000.

[26] E. F. Moore, "The shortest path through a maze," in *Proc. Internat. Sympos. Switching Theory 1957, Part II.*, cambridge, MA, USA, 1959, pp. 285–292.

Seong-Hwan Kim received the B.S. degree in media and communications engineering from Hanyang University, Seoul, South Korea, in 2012. He is currently pursuing the Integrated Master's Ph.D. degree in electrical engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea.

He is a Member of Network and Computing Laboratory in KAIST. His research interests include mobile cloud computing and cloud collaboration.

Dong-Ki Kang received a M.S. degree in computer engineering from Chonbuk National University, Jeonju, South Korea, in 2011. He is currently pursuing the Ph.D. degree in electrical engineering at Korea Advanced Institute of Technology (KAIST), Daejeon, South Korea.

His research interests include virtualized resource management, and cloud computing.

Woo-Joong Kim received the M.S. degree in electrical engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2014. He is currently pursuing the Ph.D. degree in electrical engineering at Korea Advanced Institute of Technology (KAIST).

His research interests include cloud collaboration, mobile cloud computing, mobile traffic offloading, and social networks.

Min Chen, (SM'09) is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University (SNU), Seoul, South Korea, from September 2009 to February. 2012. He was R&D director at Confederal Network Inc. from 2008 to 2009. He worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of British Columbia (UBC), Vancouver, BC, Canada, for three years. Before joining UBC, he was a Postdoctoral Fellow with SNU for one and half years. He has more than 170 paper publications. He serves as an Editor or Associate Editor for *Information Sciences*, *Wireless Communications and Mobile Computing*, *IET Communications, IET Networks*, *Wiley I. J. of Security and Communication Networks*, *Journal of Internet Technology*,*KSII Transactions Internet and Information Systems*, *International Journal of Sensor Networks*. He is a Managing Editor for IJAACS and IJART. He is a Guest Editor for the IEEE NETWORK, IEEE WIRELESS COMMUNICATIONS MAGAZINE, etc. He is the Co-Chair of the IEEE ICC2012-Communications Theory Symposium, and the Co-Chair of the IEEE ICC 2013-Wireless Networks Symposium. He is the General Co-Chair for the 12th IEEE International Conference on Computer and Information Technology (IEEE CIT-2012).

Mr. Chen was the recipient of the Best Paper Award from IEEE ICC 2012, and the Best Paper Runner-up Award from QShine 2008.

Chan-Hyun Youn (S'84–M'87) received the B.Sc. and M.Sc. degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Sendai, Japan, in 1981, 1985, and 1994, respectively.

Before joining the University, from 1986 to 1997, he was the Leader of High-Speed Networking Team at KT Telecommunications Network Research Laboratories, where he had been involved in the research and developments of centralized switching maintenance system, maintenance and operation system for various ESS's system, high-speed networking, and ATM network testbed. Since 2009, he has been a Professor with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He also was the Dean of Office of Planning Affairs and the Director of Research and Industrial Cooperation Group, former Information and Communications University, in 2006 and 2007, respectively. He was a Visiting Professor at Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2003, and has been engaged in the development of physio-grid system with Prof. R. G. Marks Group in the Laboratory for Computational Physiology, MIT, since 2002. He is an Associate Vice-President of office of planning and budgets in KAIST. He is also a Director of Grid Middleware Research Center at KAIST. His research interests include mobile cloud, mobile collaboration system, Internet computing workflow management, distributed network architecture, communication middleware, advanced e-Healthcare system, high performance computing system and others.

Dr. Youn is currently serving as an Editor of Journal of Healthcare Engineering (U.K.), and served as an Editor-in-Chief in the Korea Information Processing Society, the Head of Korea Branch (computer section) of IEICE, Japan (2009, 2010).