Mobile Netw Appl (2015) 20:328–336
DOI 10.1007/s11036-015-0593-4

Cost Adaptive VM Management for Scientific Workflow Application in Mobile Cloud

Woo-Joong Kim[1] · Dong-Ki Kang[1] · Seong-Hwan Kim[1] · Chan-Hyun Youn[1]

Published online: 17 April 2015

Abstract In this paper, to guarantee Service Level Agreement composed of the deadline and budget given by users for workflow application services in mobile cloud, we propose the two-phases algorithm with a cost adaptive VM management. Firstly, the greedy based workflow co-scheduling phase schedules a workflow by using a resource consolidation in a parallel manner to decrease a cost with the deadline assurance. Secondly, the resource profiling based placement phase locates a VM to a certain physical host in the multi-cloud using the profile on the property of clouds in order to comply with the budget while maximizing the service quality. We implement mobile cloud brokering system with the two-phases algorithm and demonstrate that our proposed system outperforms traditional cloud systems through several experimental results.

Keywords Mobile cloud computing · Mobile cloud brokering system · Cost adaptive resource management

✉ Chan-Hyun Youn
chyoun@kaist.ac.kr

Woo-Joong Kim
w.j.kim@kaist.ac.kr

Dong-Ki Kang
dkkang@kaist.ac.kr

Seong-Hwan Kim
s.h_kim@kaist.ac.kr

[1] Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea

1 Introduction

There have been many attempts to apply cloud computing to a mobile environment for various mobile services called as mobile cloud computing [1]. The main issue of these attempts is to build a middleware system to provide specific services using cloud resources to mobile users. In this system, the mobile service users can receive the service results on Virtual Machine (VM) instance through the middleware system in cloud. There are several works for this system to provide various application services such as mobile business, mobile commerce, mobile learning and mobile healthcare [2–5]. The one of main concerns in the mobile cloud service is the workflow application service which has data collection, transportation and processing such as the big data application service [6]. The several researches have been proposed to provide the workflow application service to users in mobile cloud [7, 8]. With these solutions, scientists can easily access to scientific workflow application services such as the generation of science-grade mosaics of the sky in astronomy and the underpinnings of complex diseases in bioinformatics, in cloud through a mobile devices and increase the efficiency of their researches [9, 10]. The workflow processing in cloud through mobile devices requires two main procedures in this wise, the workflow scheduling and the VM placement. The most important objective of those procedures is an assurance of Service Level Agreement (SLA) composed of the deadline and budget given by mobile service users [11]. However, the studies above mentioned [7, 8] only consider the mobile cloud framework and its functionalities for workflow application services, not the workflow processing. Therefore, the mobile cloud system which has the VM management module for workflow processing is required to assure the users SLA. To do this, there are several related

works to guarantee the users SLA based on the heuristic approach for workflow scheduling and VM placement in the conventional heterogeneous environment without mobile device for reference. Firstly, for workflow scheduling, a famous heuristic algorithm called Heterogeneous Earliest-Finish-Time (HEFT) is introduced for deadline assurance [12]. In this algorithm, the rank value is allocated to each task of the workflow to schedule them based on their priority. In [13, 14], an iterative resource rescheduling algorithm by using GAIN/LOSS weight is introduced to get the cost and makespan efficient scheduling. Even though this algorithm is not able to assure the optimal performance, it is valuable because of its low time complexity and simplicity.

Secondly, for VM placement, the data-aware VM placement algorithm is introduced to reduce the data transfer delay by the optimization for VM placement and their allocated data rates in [15]. In [16], the network-aware VM placement in a heuristic way is introduced to guarantee the network performance between VMs specified by user in distributed clouds. In this paper, we propose the two-phases algorithm for workflow processing in mobile cloud : the greedy based workflow co-scheduling phase and the resource profiling based placement phase. The greedy based workflow co-scheduling phase consolidates the scheduled VM instances in a parallel manner to decrease the resource usage cost within the deadline assurance. The resource profiling based placement phase locates a VM instance to a certain physical host in the multi cloud using the profile on the property of clouds to comply with the budget while maximizing the service quality. Finally, we implement the mobile cloud system called Mobile Cloud Broker(MCB) with cost adaptive VM management based on the two-phases algorithm for providing the workflow application service and demonstrate that our MCB outperforms the cloud systems with traditional VM management through several experimental results.

2 Brokering service model in mobile cloud

MCB is shown in Fig. 1. The connector(i.e. a type of mobile application) provides a remote control interface to use the virtual device of MCB. The virtual device provides augmented resources virtually to mobile service users with various services (e.g. scientific applications). Especially, the MCB provides the workflow designer through the virtual device so users can specify their request description and submit the workflow application graphically toxl the workflow manager of MCB. The workflow manager parses the submitted workflow and schedules VM flavor type and instance to each sub-task within the workflow according to the given SLA with the greedy based workflow co-scheduling. The resource manager decides the certain physical host within multi-cloud for the VM creation request of the specific VM type from workflow manager and provisions the VM instance with the resource profiling based placement. The resource profiler collects and maintains the meta-data of each cloud provider such as available cloud services, price and the performance related information to consider the heterogeneity of cloud resource.

3 Two-phases algorithm for cost adaptive VM management

3.1 Phase 1: Greedy based workflow co-scheduling

The workflow is represented as a directed acyclic graph $G = \{T, E\}$, where T is a set of structure-dependent tasks t and E is a set of edges e between tasks. By the initial VM scheduling process in the workflow manager, VM instances are scheduled to each task in the light of the deadline dl^G. Obviously, as the performance of the VM instance is improved, its cost is also increased proportionally. The resource usage cost for processing of a workflow G in cloud service is given by

$$Cost(G) = \sum_{i=1}^{m} c^p_{vm_i} \left[ft^p \left(t^{vm_i}_{last} \right) - st^p \left(t^{vm_i}_{first} \right) \right] \quad (1)$$

where m is the number of allocated VM instances, vm_i is a i th allocated VM instance and $c^p_{vm_i}$ is an unit cost for the VM instance vm_i with billing time unit p such as hour, month and year. ft and st are the finish time and the start time of the task processing, respectively. $t^{vm_i}_{last}$ and $t^{vm_i}_{first}$ are the last and first task processed on vm_i. The total processing time of workflow G, tpt^G is given as follows

$$tpt^G = tpt(t_n) \quad (2)$$

$$tpt(t_i) = et^p(t_i) + \max_{\forall t_j \in T_{pred(i)}} \left\{ tpt(t_j) + tt^p(t_j, t_i) \right\} \quad (3)$$

$$0 \leq tpt^G \leq dl^G \quad (4)$$

where n is the number of tasks of workflow G, t_n is an end task of the workflow G, $N_{pred(i)}$ is a set of predecessor tasks of task t_i and $tt^p\left(t_j, t_i\right)$ is a transmission time of the data from task t_j to t_i. The total processing time of a task t_i, $tpt(t_i)$ means a critical path length from the start task t_1 to t_i. If the task t_i is the end task (i.e. $i = n$), then $tpt\ (t_i)$ equals a whole critical path length of the workflow G, tpt^G. It should be assured that tpt^G does not exceed the deadline dl^G of the workflow G as represented in above inequality Eq. 4. In our proposed algorithm, two arbitrary tasks are co-scheduled for a single VM instance instead of that they are scheduled for distinct VM instances. That is, two integrated tasks are processed on the common VM instance (or called target VM instance in this paper) simultaneously in parallel.

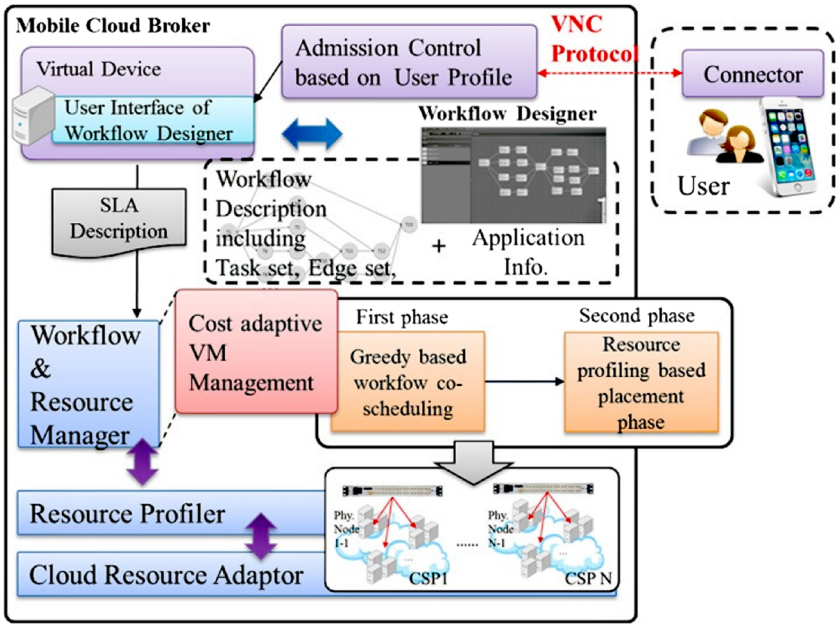

Fig. 1 Architecture of mobile cloud broker

Figure 2 shows an example of the proposed co-scheduling algorithm. Task 1 and 6 are co-scheduled and allocated in VM instance 2; task 2 and 5 are co-scheduled and allocated in VM instance 3. In Fig. 2a, each required resource allocation time (time unit is p) is 3, 2 and 2 for VM 1, 2 and 3, respectively. However, after task co-scheduling in Fig. 2b, the each required resource allocation time is decreased to 2, 2 and 1 for VM 1, 2 and 3, respectively. Consequently, the resource usage cost can be decreased by using proposed task co-scheduling approach. Obviously, while the resource usage cost is decreased by task co-scheduling, the undesirable computing overhead (e.g. context switching) may be occurred by the parallel execution of integrated tasks. The striped rectangles next to task 1, 2, 5 and 6 represent the delay caused by task co-scheduling. The each increased execution time for task n_i and n_j on target VM instance vm_{tar} through task co-scheduling is $et^{co-sched}_{(t_i,t_j)}(t_i, vm_{tar})$ and $et^{co-sched}_{(t_i,t_j)}(t_j, vm_{tar})$,respectively, and it is recorded in the co-scheduling time table.

It is assumed that we are able to establish the co-scheduling time table before the actual workflow processing by several performance prediction approaches referring to existing works [17]. In spite of the delay occurrence of each sub-task caused by task co-scheduling, the total processing time of workflow should satisfy the users deadline. Therefore a co-scheduled task pair which is composed of the inserted task (i.e. re-scheduled task to target VM instance) and the native task (i.e. originally scheduled task on target VM instance) should satisfy the constraints as follows

(1) Constraint 1 for task co-scheduling
When the arbitrary task t_i is re-scheduled to the target VM instance vm_{tar}, it is not allowable for VM instance vm_{tar} to have three or more overlapped running tasks simultaneously since it is difficult to predict its increased execution time by task co-scheduling. That is, the upper bounding number of processing tasks simultaneously on target VM instance is two.

(2) Constraint 2 for task co-scheduling
The deadline of workflow should not be violated by the delay caused by task co-scheduling. Suppose that co-scheduled tasks t_i and t_j on VM instance vm_{tar} comply with above Constraint 1. When the original execution time of the task t_i on vm_{tar} is $et(t_i, vm_{tar})$, then the increased execution time of task t_i, $et'(t_i, vm_{tar})$ is calculated as follows

$$et'(t_i, vm_{tar}) = et(t_i, vm_{tar}) + \left(et^{co-sched}_{(t_i,t_j)}(t_i, vm_{tar}) - et(t_i, vm_{tar})\right) \cdot \frac{ot_G(t_i, t_j, vm_{tar})}{ot_{co-timeTable}(t_i, t_j, vm_{tar})} \quad (5)$$

where $ot_{co-timetable}(t_i, t_j, vm_{tar})$ is an overlapped length of original execution time of task t_i and t_j on VM instance vm_{tar} in the co-scheduling time table and, $ot_G(t_i, t_j, vm_{tar})$ is that in the workflow G. After task co-scheduling of task t_i and t_j on VM instance vm_{tar}, the original execution time $et(t_i)$ of task t_i and $et(t_j)$ of t_j are replaced with increased execution time $et'(t_i)$ of task t_i and $et'(t_j)$ of t_j, respectively. The start time and finish time of the whole tasks associated with task t_i and t_j are also updated. Finally, the total processing time of workflow G, tpt^G is updated. The updated tpt^G should satisfy the deadline dl^G under Eq. 4.

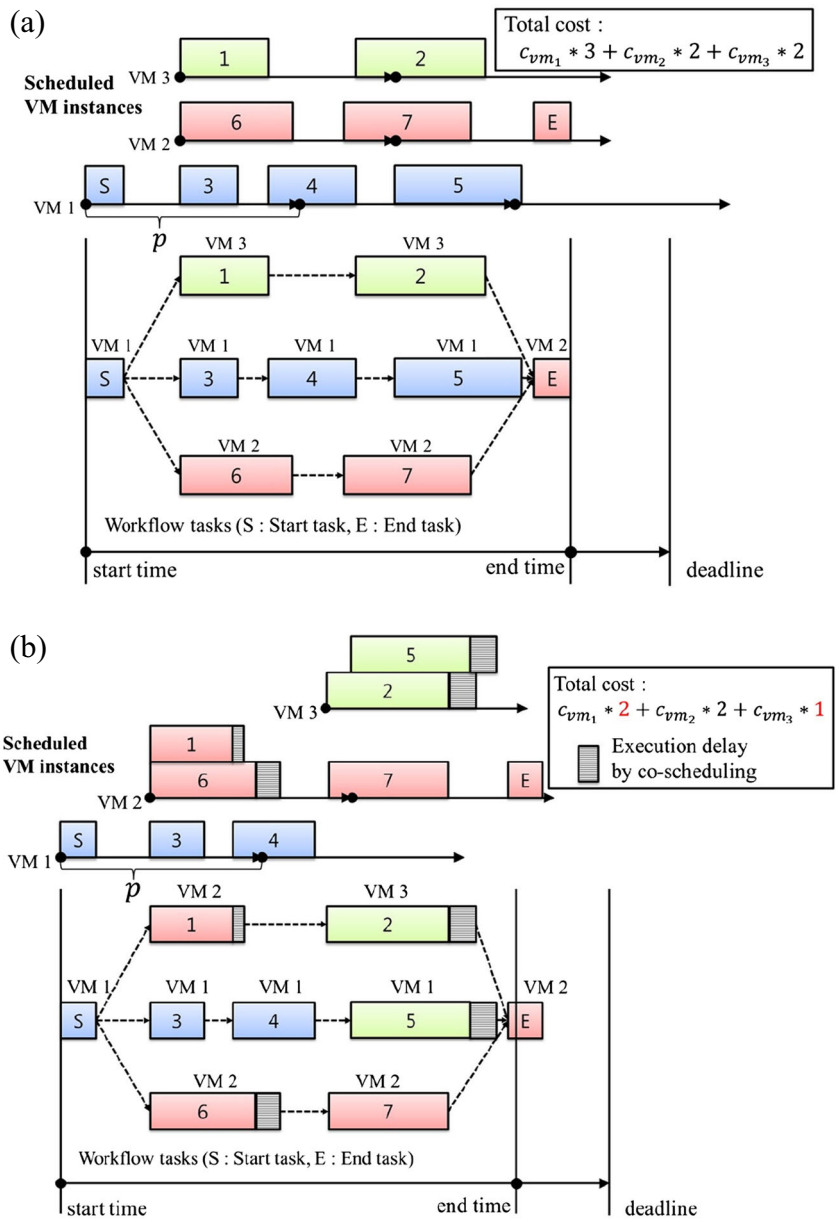

Fig. 2 An example of workflow co-scheduling process for resource usage cost saving. (**a**) before task co-scheduling of workflow, (**b**) after task pair (1, 6) and (2, 5) co-scheduling for VM 2 and 3, respectively

(3) Constraint 3 for task co-scheduling

The task co-scheduling procedure should derive benefit of resource usage cost since it bears the cost of the increased workflow processing time. Therefore, the resource usage cost has to be decreased after the task co-scheduling as follows, Eq. 6.

$$\begin{aligned} & c_{vm_{or}} \cdot \left[ft\left(t_{last}^{vm_{or}}\right) - st\left(t_{fisrt}^{vm_{or}}\right)\right] \\ & + c_{vm_{tar}} \cdot \left[ft\left(t_{last}^{vm_{tar}}\right) - st\left(t_{first}^{vm_{tar}}\right)\right] > \\ & c_{vm_{or}} \cdot \left[ft\left(t_{last}^{vm_{or}/\{t_i\}}\right) - st\left(t_{fisrt}^{vm_{or}/\{t_i\}}\right)\right] \\ & + c_{vm_{tar}} \cdot \left[ft\left(t_{last}^{vm_{tar}\cup\{t_i\}}\right) - st\left(t_{first}^{vm_{tar}\cup\{t_i\}}\right)\right] \end{aligned} \tag{6}$$

where $t_{first}^{vm_{or}}$ and $t_{last}^{vm_{or}}$ are the first and the last scheduled task on the VM instance vm_{or}, respectively. The left side of Eq. 6 represents the original resource usage cost before task co-scheduling and the right side represents the updated resource usage cost after task co-scheduling in which task t_i is re-scheduled from vm_{or} to the target VM instance vm_t. Equation 6 describes that task co-scheduling is available only if the resource usage cost reduction can be achieved by the task co-scheduling. The arbitrary co-scheduling pair composed of re-scheduled task t_i and its overlapped task t_j on VM instance vm_{tar}, which satisfies all the constraints which described above can be included in a co-scheduling candidate list. After all the possible co-scheduling pairs are found, we select a co-scheduling pair which achieves

the maximum resource usage cost saving among the co-scheduling candidate list. Therefore, we obtain Eq. 7 by using Eq. 6 as follows

$$co\text{-}SchedPair^G = \underset{(t_i,t_j)}{\operatorname{argmin}} \left\{ c_{vm_{or}} \cdot \left[ft\left(t_{last}^{vm_{or}/\{t_i\}}\right) - st\left(t_{first}^{vm_{or}/\{t_i\}}\right)\right] + c_{vm_{tar}} \cdot \left[ft\left(t_{last}^{vm_{tar}\cup\{t_i\}}\right) - st\left(t_{first}^{vm_{tar}\cup\{t_i\}}\right)\right]\right\}, \forall t_i, t_j \in T^G, \forall t_i, t_j \notin T^G_{co-sched} \quad (7)$$

After the task t_i and t_j are co-scheduled on VM instance vm_{tar}, they are released from the co-scheduling candidate list and inserted into the co-scheduling completed list $T^G_{co-sched}$. Consecutively, the schedule of all the sub-tasks of the workflow G is updated, and the co-scheduling candidate list is evacuated. The procedure for generation of new possible co-scheduling pairs is repeated for remain tasks are not yet co-scheduled in the workflow G. Finally, the task co-scheduling procedure is definitely finished when there are no possible co-scheduling pair in the workflow any more. Algorithm 1 shows the task co-scheduling procedure in detail. The notation $OC(t, vm)$ and $OT(t, vm)$ represent the count of overlapped tasks of task t and its overlapped tasks on VM instance vm, respectively. The temporary duplication G' of the input workflow G initially scheduled by [13, 18] is generated to enable the resume procedure in line 13.

Algorithm 1 Greedy based task co-scheduling of workflow.

Input: workflow G initially by using [13, 18]
Output: workflow G after greedy based task co-scheduling.

```
1: while true do
2:     Temp workflow G' ← workflow G
3:     for each task t ∈ T^{G'} ∩ t ∉ T^G_{co-sched} do
4:         for each vm ∈ scheduled Virtual Machines do
5:             if st(vm) ≤ st(t) & OC(t, vm)
                   < 2 & OT(t, vm) ∉ T^G_{co-sched} then
6:                 task ot ← OT(vm, t)
7:                 calculate et'(t, vm), et'(ot, vm)
                     by Eq. 5
8:                 et(t) ← et'(t, vm), et(ot) ← et'(ot, vm)
                     and apply to G'
9:                 update st,ft of each sub-task of
                     G', tpt^{G'}, Cost(G')
10:                if tpt^{G'} ≤ dl^G & Cost(G')
                       < Cost(G) then
11:                    co-SchedCandidateList ← insert (t, ot)
12:                end if
13:                G' ← resume G
14:            end if
15:        end for
16:    end for
17:    if co-SchedCandidateList = 0 then
18:        T^G_{co-sched} ← insert a pair co − SchedPair^G of
             co-SchedCandidateList satisfying Eq. 7 and
             update workflow G ← G'
19:        co-SchedCandidateList ← 0
20:    end if
21: end while
```

3.2 Phase 2: Resource profiling based placement

The resource profiling based placement algorithm locates a VM instance to a certain physical host in the multi-cloud using the profile on the property of clouds in order to comply with the budget while maximizing the service quality such as the data transmission delay, the heterogeneity [19] of cloud resources on a workflow application service. There are also two policies compute-aware and network-aware placement policy. The compute-aware placement policy provides the cloud resource having better computing capability in the given cost, considering the heterogeneity [19] of cloud resource. The better computing capacity means the ability to finish a certain task faster. To achieve this object, we use CPU model rank table provided from [20]. The available physical nodes *availablePN* in multi-cloud are sorted based on the rank in the CPU model rank table and the high-ranked physical node is provisioned with high priority. If there is no available resource capacity in the high-ranked physical node for the requested flavor type f, the another physical node ranked higher while having enough resource capacity is decided. Algorithm 2 shows the compute-aware placement policy.

Algorithm 2 Compute-aware placement policy.

Input: $uid^k, f\ (uid^k : request\ id, f : flavor\ type)$
Output: createdVM

```
1: get the sorted available physical nodes list using CPU
   model rank table
2: for r^i ∈ availablePhysicalNodes do
3:     if r^i is available for flavor type f then
4:         createdVM = createNewVM(r^i, f)
5:         return createdVM
6:     end if
7: end for
```

The network-aware placement policy provides the cloud resource having better network performance on the past created VM for each user in order to guarantee the data transmission time between VMs. To achieve this object, the table *LastUsedResourceTable* which stores the last used resource information such as cloud, physical node for each user is maintained and the VM instance requested in the future is provided in the same physical node with the last used resource if possible for each user. If the resource capacity of this physical node is not available for the requested flavor type f, available physical nodes *availablePN* are sorted in the closest order from the physical node of the last used resource. The new physical node as close as possible to the physical node of the last used resource while having enough resource capacity for flavor type f is decided. After providing the VM instance in the new physical node, the new physical node is updated to the last used resource table for the corresponding user. Algorithm 3 shows the network-aware placement policy.

Algorithm 3 Network-aware placement policy.

Input: $uid^k, f\ (uid^k : request\ id, f : flavor\ type)$
Output: createdVM

```
if LastUsedResource of uid^k is available then
    cloud c^k, physical node p^k ← get LastUsedResource
      of uid^k from LastUsedResourceTable
    if physical node p^k in cloud c^k is available
      for falvor type f then
        createdVM = createNewVM(p^k, f)
        return createdVM
    else
        sort availablePNs in closest order from p^k
        for p^i ∈ availablePNs do
            if p^i is available for falvor type f then
                createdVM = createNewVM (p^i, f)
                record {uid^k, c^k, p^k}
                    into LastUsedResourceTable
                return createdVM
            end if
        end for
        return null
    end if
else
    maxCapacityPN ← max (remainCapacity (p^j))
      ... (p^j ∈ PNSet of c^i, c^i ∈ cloudSet)
    if maxCapacityPN is available for flavor type f then
        createdVM = createNewVM(maxCapacityPN, f)
        record {uid^k, c^k, p^k} into LastUsedResourceTable
        return createdVM
    else
        return null
    end if
end if
```

4 Experimental environment and performance evaluation

In this section, we demonstrate the superiority of MCB by evaluating the performance of cost adaptive VM management. We use the Openstack cloud environment and the available VM types in Openstack cloud environment are small type(1 CPU, 2GB MEM, 10GB Disk), medium type(2 CPU, 4GB MEM, 10GB Disk) and large type(4 CPU, 8GB MEM, 10GB Disk). We build the cloud testbed using 6 nodes with the deployment of openstack cloud environment as shown in Fig. 3. [21]. Node 1,5 are nova controller and node 2,3,4,6 are nova compute nodes. Node 1-4 have the hardware with Intel, Xeon E5620 2.4G, Core 16, MEM 16G, HDD 1T and the software with Ubuntu 12.04 OS. Node 5,6 have the hardware with Intel, Xeon W3520 2.67G, Core 8, MEM 16G and the software with Ubuntu 12.04 OS Fig. 3.

4.1 Performance evaluation for phase 1

The greedy based workflow co-scheduling algorithm is evaluated against conventional workflow scheduling approaches such as MDP based partitioning and Gain/Loss algorithms [13, 22]. We adopt the Montage project as a practical workflow example that is a famous open-source based scientific application [9].The Montage project has been invoked by the NASA/IPAC Infrared Science Archive as a toolkit for assembling Flexible Image Transport System (FITS) images into custom mosaics. We design the Montage workflow example as shown in Fig. 4. The deadline dl^G with deadline factor α $(0 \leq \alpha \leq 1)$ is established as follows

$$dl^G = \min\left(tpt^G\right) + \alpha\left(\max\left(tpt^G\right) - \min\left(tpt^G\right)\right) \quad (8)$$

The minimum processing time of the workflow is achieved when all the sub-tasks of workflow are processed with fastest VM flavor type allocation. Otherwise, the maximum processing time of the workflow is derived when all the sub-tasks of workflow are allocated on cheapest VM flavor type instance.

In order to compare our proposed workflow co-scheduling algorithm with existing approaches, we measured resource usage cost and workflow processing time. The resource usage cost of each workflow scheduling algorithm is shown in Fig. 5. As the deadline factor α is

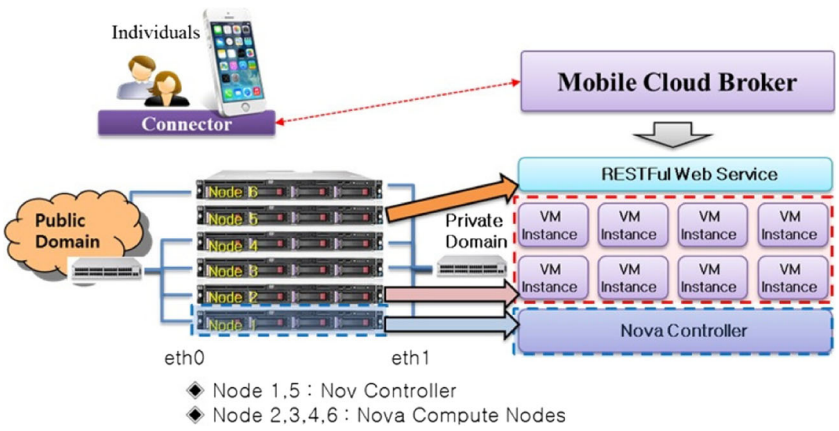

Fig. 3 Experiment environment of mobile cloud broker with Openstack

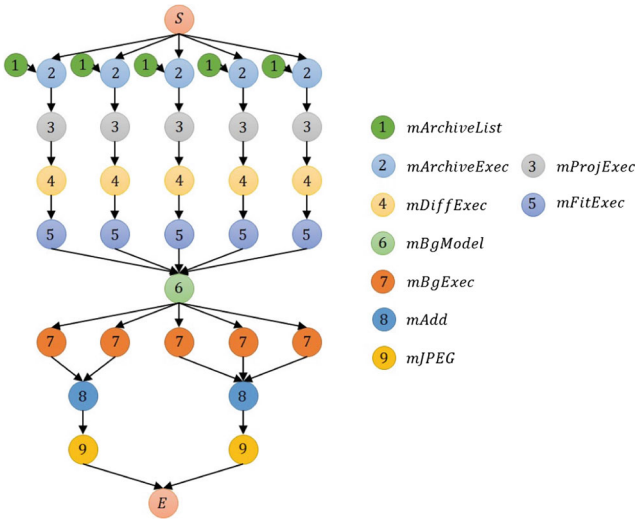

Fig. 4 Montage example for performance evaluation

increased, the whole resource usage cost is decreased since the necessity for the capacity of the allocated resource to be high is decreased because of the loose deadline of a workflow. Our proposed algorithm always achieves the higher cost saving performance than that of existing workflow scheduling algorithms regardless of the size of deadline factor. The resource usage cost curve of GAIN algorithm at $\alpha = 0$ shows a dramatic rise since it is not an optimized but a heuristic based approach so this is not able to assure the consistent cost saving performance in all the cases. However, our proposed algorithm achieves the consistent performance improvement even though it is also a heuristic based approach because of its greedy characteristic. As shown in the graph, the resource usage cost of the workflow co-scheduling is lower than the MDP based partitioning by 17 % and the GAIN/LOSS algorithm by 30 %. Figure 6 shows the total processing time of each workflow scheduling algorithm. By using Eq. 8, the each deadline is (0, 482), (0.2, 507), (0.4, 532), (0.6, 556), (0.8, 581) and (1, 605) according to each deadline factor α in Fig. 5. As

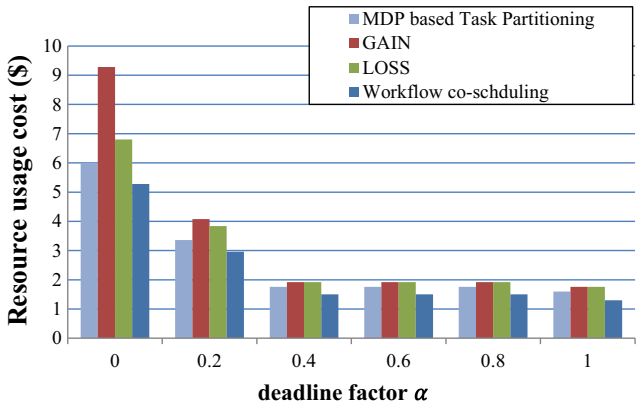

Fig. 5 Resource usage cost of the workflow co-scheduling algorithm and conventional approaches

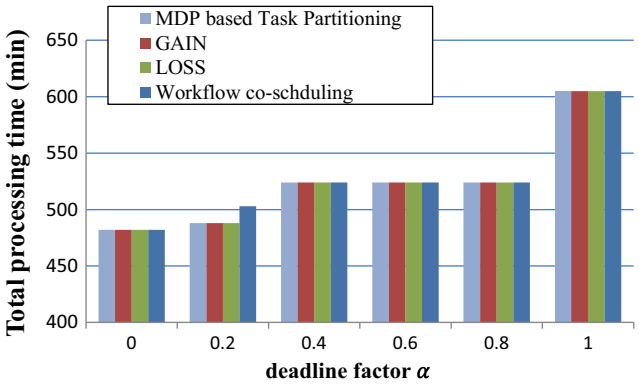

Fig. 6 Total processing time of the workflow co-scheduling algorithm and conventional approaches

shown in Fig. 6, we verify that the workflow co-scheduling algorithm assures the deadline compliance since it is allowable only for the task pair satisfying the Constraint 2 described earlier to be co-scheduled in our proposed algorithm.

4.2 Performance evaluation for phase 2

We want to evaluate the performance of compute-aware placement policy compared to the conventional placement algorithm proposed by [23] first. We adopt the chemical application service treated by [23]. Specifically, we use a single compute-intensive task *QSAR Analysis* of MapChem application which is an integrated chemical application for collaborative pharmaceutical research and the available input data types are sdf30, sdf100, sdf200 (sdf100 means the input data which includes a hundred of chemical compounds information expressed by structure data format). In this experiment, to evaluate the ability which guarantees the SLA required by user while minimizing cost, we measure the SLA violation and the total cost when the requests are occurred in the fixed interval over a period of time on each algorithm. We repeat these experiments with different request interval times (4 sec, 3.6 sec, 3.2 sec, 2.8 sec, 2.4 sec, 2 sec, 1.6 sec, 1.2 sec, 0.8 sec) within 3min. The SLA of QSAR service is defined as deadline and it is set in random on each request. The input data type of the request is also randomly chosen from available input datas such as sdf30, sdf100, sdf200. The metric of cost is defined as *relative cost* which has a theoretical meaning for comparison between algorithms [23] and the metric of SLA violation is defined as the number of request which violates the deadline required by user.

In Fig. 7, we see that the total cost of the proposed algorithm [23] is lower than that of the conventional one by 58 % on average over all interval times of the request. The conventional algorithm prepares an extra VM in advance to reduce the delay from the VM initiation time so at least one

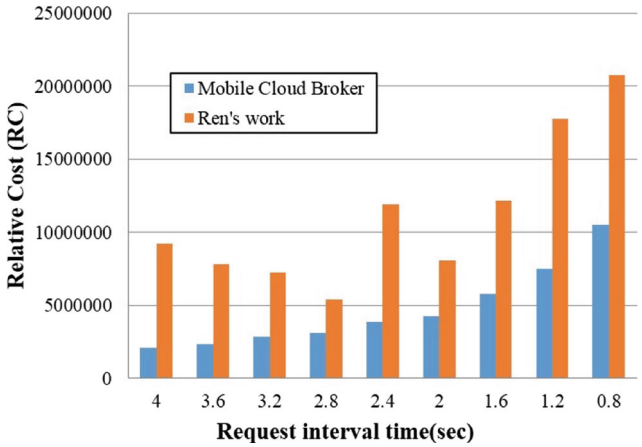

Fig. 7 Cost performance of proposed system and conventional system

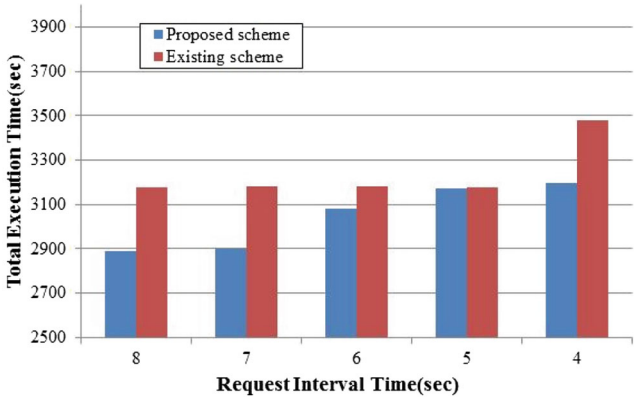

Fig. 9 Total execution time of proposed system and conventional scheme

empty VM is always kept. This mechanism makes the waste of resource and occurs the high cost. On the other hand, the proposed algorithm does not make the extra resource because we judge that the VM initiation time is not critical compared to the execution time of a request and provides the more minimal cost on the same VM specification compared to the conventional algorithm considering the heterogeneity of cloud resource.

Figure 8 shows that the proposed algorithm has the similar or smaller SLA violation over the entire request interval time compared to the conventional algorithm. It means that the proposed algorithm achieves to guarantee the SLA while minimizing the cost considering the heterogeneity of cloud resource. Secondly, to evaluate the performance of the network-aware placement policy, we adopt *Burrows-Wheeler Aligner (BWA)* which is the typical bio scientific application for aligning or mapping low-divergent sequences against a large reference genome, such as the human genome [24]. This application has various tasks (i.e. BWA index, alignment, pairing and view etc.) and has dependencies between the tasks so network-intensive

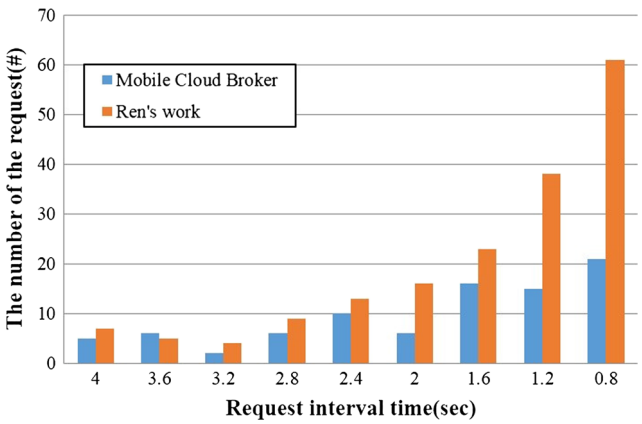

Fig. 8 SLA violation performance of proposed system and conventional system

because the data transmission is often occurred between tasks. We evaluate the performance of the network-aware placement policy compared to the typical network-aware placement algorithm proposed in [16]. In this experiment, to evaluate and compare the data transmission delay occurred while executing the BWA service, we measure the total execution time on each algorithm when the requests are occurred in the fixed interval over a period of time. We repeat the experiment with different interval times (8 sec, 7 sec, 6 sec, 5 sec, 4 sec) within 3 min. The request is comprised of the BWA tasks such as index, alignment, pairing and view and the input data to each task is same for each request. To be independent to the computing performance, we use the physical nodes in the same specification (Node1 - Node 4) and all tasks are allocated in the large VM type. Figure 9 shows that the proposed algorithm has the smaller total execution time by 6 % on average over the entire request interval time compared to the conventional algorithm [16]. It means that the data transmission delay occurred on the proposed scheme is smaller than the conventional algorithm. In other words, the proposed algorithm guarantees the network performance between VMs on the BWA service. The conventional algorithm is concentrated on the placement for a VM creation requested at a certain time so cannot guarantees the network performance for the successive VM creation request. As a result, the conventional algorithm shows the worse performance on the BWA service which needs the successive VM creation.

5 Conclusion

In this paper, we propose the two-phases algorithm and implement the mobile cloud broker for workflow application service in the mobile cloud with the cost adaptive VM management. In the experimental results, the greedy based workflow co-scheduling algorithm reduces the resource

usage cost about 17 % at least compared to the traditional workflow scheduling approaches while guaranteeing the deadline. The resource profiling based placement not only reduces the resource usage cost about 58 % on compute-intensive application but also improves the data transmission delay about 5 % on network-intensive application compared to existing placement algorithms. Finally, we demonstrated that our proposed system with this two-phases algorithm outperforms other traditional systems in terms of both cost and processing time.

Acknowledgments This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2012-0020522) and the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014 and the MSIP under the ITRC support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA.

References

1. Srirama SN, Paniagua C, Flores H (2011) Croudstag: social group formation with facial recognition and mobile cloud services. Procedia Comput Sci 5:633–640
2. Yang X, Pan T, Shen J (2010) On 3g mobile e-commerce platform based on cloud computing. In: 2010 3rd IEEE International conference on ubi-media computing (U-Media), pp 198–201. IEEE
3. Gao H-Q, Zhai Y-J (2010) System design of cloud computing based on mobile learning. In: 2010 3rd International symposium on knowledge acquisition and modeling (KAM), pp 239–242. IEEE
4. Doukas C, Pliakas T, Maglogiannis I (2010) Mobile healthcare information management utilizing cloud computing and android os. In: 2010 annual International conference of the IEEE engineering in medicine and biology society (EMBC), pp 1037–1040. IEEE
5. Chen M (2014) Ndnc-ban: supporting rich media healthcare services via named data networking in cloud-assisted wireless body area networks. Inf Sci 284:142–156
6. Chen M, Mao S, Zhang Y, Leung VCM (2014) Big data: related technologies, challenges and future prospects. In: Springer briefs series on wireless communications, 1st edn. Springer, New York
7. Yao J, Zhang J, Chen S, Wang C, Levy D (2011) Facilitating bioinformatic research with mobile cloud. In: The 2nd International conference on cloud computing, GRIDs, and virtualization, Cloud Computing 2011, pp 161–166
8. Flores H, Srirama SN (2014) Mobile cloud middleware. J Syst Softw 92:82–94
9. Montage http://montage.ipac.caltech.edu/
10. Oinn T, Li P, Kell DB, Goble C, Goderis A, Greenwood M, Hull D, Stevens R, Turi D, Zhao J (2007) Taverna/mygrid: aligning a workflow system with the life sciences community. In: Workflows for e-Science. Springer, Berlin Heidelberg New York, pp 300–319
11. Yu J, Buyya R (2006) Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Program 14(3):217–230
12. Topcuoglu H, Hariri S, Wu M-Y (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
13. Sakellariou R, Zhao H, Tsiakkouri E, Dikaiakos MD (2007) Scheduling workflows with budget constraints. In: Integrated research in GRID computing. Springer, Berlin Heidelberg New York, pp 189–202
14. Sakellariou R, Zhao H (2004) A hybrid heuristic for dag scheduling on heterogeneous systems. In: Proceedings of 18th International parallel and distributed processing symposium, 2004, p 111. IEEE
15. Zamanifar K, Nasri N, Nadimi-Shahraki M (2012) Data-aware virtual machine placement and rate allocation in cloud environment. In: 2012 2nd International conference on advanced computing & communication technologies (ACCT), pp 357–360. IEEE
16. Alicherry M, Lakshman TV (2012) Network aware resource allocation in distributed clouds. In: 2012 Proceedings of IEEE, INFOCOM, pp 963–971. IEEE
17. Singh K, İpek E, McKee SA, de Supinski BR, Schulz M, Caruana R (2007) Predicting parallel application performance via machine learning approaches. Concurrency and Computation: Practice and Experience 19(17):2219–2235
18. Kang D-K, Kim S-H, Youn C-H, Chen M (2014) Cost adaptive workflow scheduling in cloud computing. In: Proceedings of the 8th International conference on ubiquitous information management and communication, p 65. ACM
19. Farley B, Juels A, Varadarajan V, Ristenpart T, Bowers KD, Swift MM (2012) More for your money: exploiting performance heterogeneity in public clouds. In: Proceedings of the 3rd ACM symposium on cloud computing, p 20. ACM
20. Cpu model rank table http://www.cpubenchmark.net/
21. Openstack foundation http://www.openstack.org/
22. Yu J, Buyya R, Tham CK (2005) Cost-based scheduling of scientific workflow applications on utility grids. In: 2005 1st International conference on e-Science and grid computing, pp 140–147. IEEE
23. Ren Y (2012) A cloud collaboration system with active application control scheme and its experimental performance analysis. Master's thesis, Korea Advanced Institute of Science and Technology
24. Burrows-wheeler aligner (bwa) http://bio-bwa.sourceforge.net/