Cost Adaptive Workflow Scheduling in Cloud Computing

Dong-Ki Kang
Seong-Hwan Kim
KAIST
Daejeon
Korea
+82-42-350-5495
{dkkang, s.h_kim@kaist.ac.kr}

Chan-Hyun Youn
KAIST
Daejeon
Korea
+82-42-350-3495
chyoun@kaist.ac.kr

Min Chen
Huazhong University of Science and Technology
Wuhan
China
+86-133-4999-9659
Minchen2012@mail.hust.edu.cn

ABSTRACT

In cloud computing, it remains a challenge to allocate virtualized resource with financial cost minimization and acceptable Quality of Service assurance. In general, the VM instance is allocated to cloud service users based on not actual job processing time but the fixed resource allocation time predetermined by cloud pricing policy in contrast to grid environment. In this case, the unnecessary cost dissipation is occurred by the wasted partial instance hours of allocated resource. To address this problem, we propose the heuristic based workflow scheduling scheme considering cloud-pricing model in this paper. Our scheme is composed of two phases : VM packing and MRSR (Multi Requests to Single Resource) phases. In VM-packing phase, pre-assigned multi tasks are aggregated into the common VM instance sequentially, and these tasks are merged in parallel by MRSR phase. By using our proposed schemes, we are able to reduce the number of required VM instances and achieve the significant cost saving while we guarantee the user's SLA (Service Level Agreement) in terms of workflow deadline. Our proposed schemes cannot only reduce the cost by 30% compared to traditional workflow scheduling schemes but also assure user's SLA.

Categories and Subject Descriptors

C.2.4: Distributed Systems - Distributed applications

General Terms

Management, Economics, Theory

Keywords

Cloud Resource Management, Virtual Machine Allocation, Workflow Scheduling

1. INTRODUCTION

Cloud computing has been proposed as a new paradigm which can offer a feasible solution to solve the limitation of restricted amount of resources and reduce the cost for purchasing, maintaining and managing the physical resources [1]. In cloud computing environment, all the resources are represented as VM (Virtual Machine) instances and allocated to cloud service users based on the pay-as-you-go manner. In order to achieve success of cloud adoption, there are two main goals in terms of resource management: the cost-efficient VM allocation and user's SLA (Service Level Agreement) assurance. Obviously, the more expensive VM instance provides better computing performance to the cloud service users. There is a trade-off between the performance of resource and the resource purchasing cost. Therefore, many previous researches have been focused on finding the optimal point between the performance of VM instance and the allocated resource cost in accordance with the cloud service users' requests [2,3,4].

Generally, many cloud service providers such as Amazon, Google and GoGrid adapt the resource pricing policy which is not based on the actual job processing time but the fixed resource allocation time predetermined by cloud service providers [5,6,7]. In their policies, the VM allocation time is provided to cloud service users as not fine-grained time unit but coarse-grained time unit by the hour, month and year. Once the VM instance is allocated to the cloud service user, the user should pay the fixed price predetermined by cloud service providers regardless of the actual job processing time. That is, the user has to occupy the VM instance until the end of predetermined resource allocation period although no job to process remains more. In this case, the unnecessary cost dissipation is occurred by the wasted partial instance hours of allocated VM instance. Moreover, if the number of job is large and the required capacity for each job is trivial, the cost dissipation is more exacerbated since the idle capacity of each instance is increased. However, existing resource management schemes focused on traditional grid environments are disable to address this problem since they do not consider cloud-pricing model [8, 9, 10].

To address this problem, we propose the heuristic based workflow scheduling scheme with consideration for resource allocation period based cloud-pricing model in this paper. Our scheme is composed of two phases: VM packing and MRSR (Multi Requests to Single Resource) phases. Firstly, in VM packing phase, each sub task already assigned with resource flavor type is aggregated to the generated VM instance sequentially. That is, the fragmented partial instance hours is minimized since sub tasks that require same resource flavor type is aggregated to the common VM instance in VM packing phase.

Secondly, in MRSR phase, each sub task that aggregated by VM packing is merged in parallel. That is, tasks allocated in different VM instances are merged to the single VM instance and processed concurrently. In general, the single running job cannot utilize whole capacity of resource elements such as CPU, memory, storage, etc, simultaneously. The benefit of MRSR is based on the

ICUIMC(IMCOM)'14, January 9–11, 2013, Siem Reap, Cambodia.

characteristic of multi-programming in which, the processing time for tasks

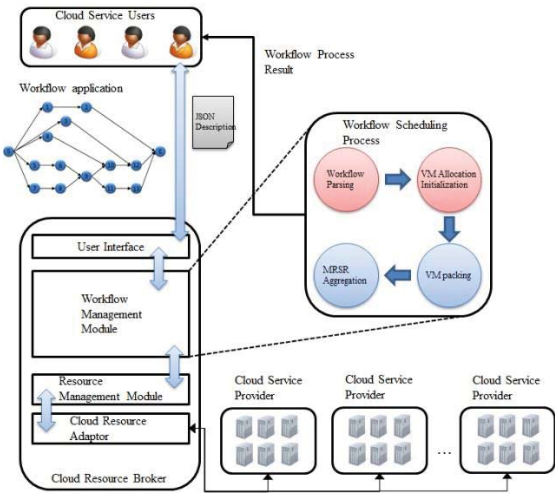


Figure 1. Cloud based Workflow Management System

running on single machine in parallel is shorter than in sequential manner if the thrashing is not occurred [11]. By using our proposed VM packing and MRSR schemes, we are able to reduce the number of required VM instances and achieve the significant saving of resource purchasing cost while we guarantee the cloud service user's SLA in terms of workflow deadline. In order to evaluate and demonstrate the performance of our proposed schemes, we establish the open-source cloud platform Openstack based testbed [12] and perform the Montage application in terms of workflow example [13, 14, 15].

2. CLOUD BASED WORKFLOW MANAGEMENT SYSTEM

In the cloud environment, the pricing model is based on not the actual job processing time but the fixed resource allocation time predetermined by cloud service provider in contrast to the grid environment. Table 1 shows the VM instance flavor types and its associated cost according to the allocation period in cloud service from GoGrid [7]. Therefore, the unnecessary resource dissipation cost might be occurred when the grid based workflow scheduling approaches are applied to the cloud environment. Once a VM instance is generated, it should be allocated for a period of certain time based on the resource allocation time predetermined by cloud service providers. That is, if the required processing time for the application is shorter than the predetermined instance hours of the allocated resource, then the resource dissipation is occurred insomuch as the remaining allocation time of the resource. The resource dissipation cost is given as shown as follows,

$$Cost_{waste}(R,n) = Cost(R,p) \cdot (\lceil t_{n,R}^{p} \rceil - t_{n,R}^{p}) \qquad (1)$$

where $t_{n,R}^{p}$ is the processing time of the task n on the VM instance flavor type R and $Cost_{waste}(R,n)$ is the wasted cost of the generated VM instance with flavor type R processing the task n. $Cost(R,p)$ is the cost of the VM instance flavor type R per the allocation period unit p. For example, the p is hourly and the processing time of task on VM instance R is 1 and half hour, then $t_{n,R}^{p}$ = 1.5. By Equation (1), we regard the ceiling function of $t_{n,R}^{p}$ as a real resource allocation duration. For example, if the actual processing time of task is 1.2 then the VM allocation duration is 2

Table 1. Virtual Machine Instance Pricing by Flavor Type

VM	RAM	Cores	Storage	Hourly	Monthly	Annual
Small	1GB	1	50GB	$0.08	$36.25	$362.50
Medium	2GB	2	100GB	$0.16	$72.50	$725.00
Large	4GB	4	200GB	$0.32	$145.00	$1,450.00
X-Large	8GB	8	400GB	$0.64	$290.00	$2,900.00

hours. In this case, 0.8 hour duration of VM instance is dissipated. We focus on this surplus capacity of VM instance and maximize the utilization of allocated resource by using VM packing and MRSR aggregation. By using our proposed scheme, we can reduce the dissipation of the allocated resource and furthermore, minimize the additional delay caused by the VM instance startup.

Figure 1 shows our proposed cloud based workflow management system. Workflow applications with required SLA description derived from cloud service users are submitted to the Cloud Resource Broker. The Cloud Resource Broker associates the multi cloud resource providers in order to allocate the tasks to the suitable virtualized resources according to the performance and cost of each provider. The workflow applications are represented as JSON (JavaScript Object Notation) description to be transferred from cloud service users to the User Interface of Cloud Resource Broker. The Workflow Management Module is a main component in the Cloud Resource Broker. There are four steps to process the workflow application in the Workflow Management Module. Firstly, the whole workflow are parsed and divided into individualized sub-tasks. By using traditional workflow scheduling algorithms [18], the VM instance flavor types are allocated to each sub tasks initially. Secondly, Tasks having same allocated VM instance flavor type are packed sequentially into the common VM instance in VM packing phase. Finally, the packed tasks are merged in parallel with the deadline assurance in the MRSR phase.

We describe the process of workflow management scheduling in more detail. As shown in Figure 1, the process of MRSR scheme is as follows in detail,

1) When the workflow description represented in JSON from the cloud service user is arrived at User Interface, the VM allocation for sub-tasks constituting workflow is performed by Workflow Management Module.

2) The sub-tasks of workflow are parsed and the appropriate VM flavor types are allocated to each task. In VM allocation initialization phase, the VM flavor types for each task are determined by traditional GAIN/LOSS approaches [18].

3) In VM packing phase, the sequential merging of tasks with same flavor type is performed heuristically. In contrast to traditional allocation scheme based on grid environment, the already assigned VM instance which has slack capacity caused by fixed allocation period is reallocated to new task in order to reduce the resource dissipation cost.

4) In MRSR phase, the parallel merging of tasks is performed heuristically. The pair of tasks is chosen and merged continuously until the workflow deadline is violated. By using VM packing and MRSR scheme, the required number of VM instance and slack capacity is minimized.

In next chapter, we describe our proposed schemes in more detail and show that proposed scheme is also efficient to improve both the cost performance and SLA assurance of the workflow management system.

3. A COST ADAPTIVE WORKFLOW SCHEDULING

We are able to reduce the required resource cost for the workflow process by using MRSR scheme compared to the traditional workflow scheduling. To do this, we represent the workflow process as Directed Acyclic Graph (DAG) as [8]. In our notation for DAG of the workflow g, n denotes the node (this is identical to task) and i denotes the index of node, that is n_i^g denotes the i th node of the workflow g. The arbitrary predecessor node, namely the certain j th parent node of the node n_i^g is represented as $n_{j,par(i)}^g$ and the set of parent nodes of node n_i^g is represented as $N_{par(i)}^g$. On the other hand, the arbitrary successor node, namely the certain j th child node of the node n_i^g is represented as $n_{j,chi(i)}^g$ and the set of child nodes of node n_i^g is represented as $N_{chi(i)}^g$.

The directed connection from node i to j in the workflow g is represented as edge $e_{i,j}^g$ and the whole edge set is represented as E^g. In our workflow graph g, a node which does not have any parent node is called a start node (or entry node) s^g and a node which does not have any child node is called an end node (or exit node) e^g. In addition, the deadline d is included in the workflow g. The workflow g is represented as follows,

$$\text{Workflow } g = \{N = \{n_1, n_2, n_3, \dots, n_k\},\ E,\ d\},$$
$$k\ = \#\ of\ nodes\ in\ workflow\ g \tag{2}$$

By the VM initialization process in the Workflow Management Module, the the VM instances are allocated to each task considering the deadline d. As shown in Table 1, as the performance of the VM instance is improved, the price of the resource is also increased proportionally. When we process the workflow g by using cloud resource, the cost for the processing is calculated as follows,

$$\text{Cost}(g) = \sum_{i=0}^{k} c_{(af_i,p)} \cdot \left\lceil t_{i,af_i}^p \right\rceil \tag{3}$$

where af_i is the allocated VM flavor type for task i and $c_{(af_i,p)}$ is the cost of the VM instance flavor type allocated to the task i per unit period of allocation p similar to the Equation (1). If the excessive VM instance is allocated to the task, then the utilization of VM instance is low and the resource dissipation is occurred. Otherwise, the deficient VM instance is allocated to the task, the deadline violation might be occurred. Therefore, the efficient workflow scheduling is required to satisfy both the resource purchasing cost saving and the deadline assurance.

To do this, firstly we apply the traditional heuristic based workflow scheduling to our workflow process. Through the GAIN/LOSS workflow scheduling approaches, the VM flavor type is assigned to each task and corresponding start time, processing time, end time of all the tasks associated with allocated VM flavor type are calculated in the VM initialization allocation process. After initial VM allocation process, the derived workflow g has to satisfy following conditions [8, 9],

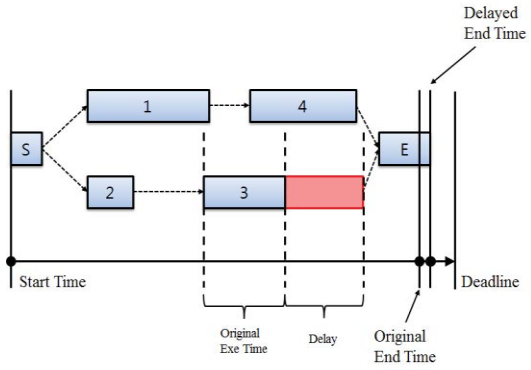


Figure 2. Workflow tasks based on time flow

$$0 \le tpt^g \le d^g \tag{4}$$
$$\max\{et\left(N_{par(i)}^g\right)\} + t_{alloc,af_i}^p = st(n_i^g), \forall n_i^g \in N^g \tag{5}$$

where t_{alloc,af_i}^p is the startup delay of VM instance with flavor type af_i for the task i. et and st are the end time and the start time of the task, respectively. If t_{alloc,af_i}^p is sufficiently small to neglect then it can be considered zero in our paper. Equations (6-8) calculate the total processing time of workflow g, tpt^g as follows,

$$tpt^g(i,i) = pt\left(n_i^g\right) \tag{6}$$
$$tpt^g(i,j) = pt\left(n_i^g\right) + \max\ \left(tpt^g\left(N_{chi(i)}^g, j\right)\right) \tag{7}$$
$$tpt^g\ = tpt^g(s,e) \tag{8}$$

where pt is the processing time of the task. We derive the $tpt^g(i,j)$ by Equation (7) since the processing time from task i to j depends on the critical path between task i and j. If the node i is a start node and the node j is an end node, then $tpt^g(i,j)$ equals to tpt^g. It is allowable to increase the total processing time of the workflow within the deadline as in Equation (4).

Figure 2 shows the workflow tasks based on the time flow. There are three cases in which the task delay is occurred in the workflow process. Firstly, the delay occurred by the specified task does not influence the total processing time of the workflow. In this case, the processing time length of the branch set in which the task delay is occurred is overwhelmed by the critical path of the workflow. Second, the delay occurred in the specific task influences the total processing time of the workflow but does not violate the deadline. In this case, the branch set in which the task delay is occurred becomes the new critical path in the workflow. Finally, the delay occurred in the specific task influences the total processing time of the workflow and the deadline is violated.

We can apply our proposed MRSR scheme to the workflow process in Figure 2. In order to deploy the MRSR scheme, MRSR candidate list should be generated before merging the tasks. In the merge plan of MRSR scheme, all the merged tasks should assure the workflow deadline and achieve the resource purchasing cost saving. If we represent the workflow $g(n)$ as the workflow with performed MRSR n times, then the constraints for the generation of MRSR candidate list are as follows,

Constraint 1 in MRSR scheme

The arbitrary task i and j to be merged together should have time overlapped section. That is, the condition $st(n_i^g) \leq st(n_j^g) \leq et(n_i^g)$ has to be satisfied.

Constraint 2 in MRSR scheme

The deadline of the workflow should not be violated by the additional delay caused by the merge the arbitrary two tasks.

That is, the condition $tpt^{g(n)} = tpt^{g(n-1)} + t^p_{(ij),af_i} - (t^p_{i,af_i} + t^p_{j,af_j}) \leq d^g$ has to be satisfied where $t^p_{(ij),af_i}$ is the processing time of task i and j running concurrently on the VM instance allocated to the task i. We assume that we are able to pre-establish the concurrent processing time table for MRSR integration.

Constraint 3 in MRSR scheme

The resource usage cost should be decreased after the merge of two tasks i and j.

That is, the condition $c_{(af_i,p)} \cdot \left\lceil t^p_{i,af_i} \right\rceil + c_{(af_j,p)} \cdot \left\lceil t^p_{j,af_j} \right\rceil < c_{(af_i,p)} \cdot \left\lceil t^p_{(ij),af_i} \right\rceil$ has to be satisfied.

As described above, if all the constraints are satisfied, then a pair of task i and j of the workflow $g(n)$ can be added to MRSR candidate list $N^{g(n)}_{MRSR\_C}$. The construction of MRSR candidate list continues until there is no pair of tasks which satisfy above constraints 1-3 more.

After establishment of MRSR candidate list $N^{g(n)}_{MRSR\_C}$ is completed, then we select a pair of tasks to be merged from $N^{g(n)}_{MRSR\_C}$ in which the performance of resource cost saving is best. After MRSR process, the chosen pair of tasks is added to the MRSR merged list $N^{g}_{MRSR\_M}$. The tasks in the $N^{g}_{MRSR\_M}$ are not considered as target for the construction process of next MRSR candidate list $N^{g(n+1)}_{MRSR\_C}$. The criterion to select a pair of tasks which achieve the best cost saving in $N^{g(n)}_{MRSR\_C}$ is described in Equation (11),

$$mergePair^{g(n)} = \underset{(n_i,n_j)}{argmax}(c_{(af_i,p)} \cdot \left\lceil t^p_{i,af_i} \right\rceil + c_{(af_j,p)} \cdot \left\lceil t^p_{j,af_j} \right\rceil - c_{(af_i,p)} \cdot \left\lceil t^p_{(ij),af_i} \right\rceil), \quad \forall n_i, n_j \in N^{g(n)}_{MRSR\_C},\ \forall n_i, n_j \notin N^{g}_{MRSR\_M} \quad (11)$$

The MRSR process is performed repetitively until it is impossible to find any pair of tasks which satisfy the MRSR constraints 1-3 (i.e. the MRSR candidate list is empty).

Algorithm 1 shows the process of our proposed VM packing scheme in detail. Firstly, the heuristic based scheduling of the input workflow g is performed as shown in line 1. From line 8 to 14, the already running VM instance is reallocated to the new task if it has idle capacity. If there is no idle VM instance to pack the new task, then the new VM instance is generated and allocated to the task.

When the re-allocation for all the tasks is completed, the VM packing is finished.

Algorithm 1. VM packing for workflow scheduling

Input : Workflow Directed Acyclic Graph $g(N, E, d)$

Output : Scheduled Workflow g with VM packing

```
01: Workflow Scheduling for g by using GAIN/LOSS [18]
02:
03: taskList N, N' <- extract tasks from workflow g
04:
05: while(true)
06:   taskIndex i = earliest start task(N)
07:   requiredFlavor f = getFlavor(n_i in N)
08:   for(each VM j ∈ allocated VMs)
09:     if(getFlavor(j) = f & state(j, (st(n_i), et(n_i)= idle)
10:       N' <- allocateVM(n_i, j)
11:       packedFlag = true
12:       break
13:     End if
14:   End for
15:   if(packedFlag = false)
16:     VM k <- generateVM(f)
17:     N' <- allocateVM(n_i, k)
18:     allocated VMs <- insert VM k
19:   End if
20:   packedFlag = false
21:   remove n_i from N
22:  if(N = empty) break
23: end while
24:
25: workflow g <- apply N'
```

Algorithm 2 shows the MRSR scheme in detail. Firstly, the scheduled workflow g by the VM packing is inputted to the Cloud Resource Broker. We set the MRSRCandidateList $MRSR\_C$ in order to collect the available task pairs for MRSR process. Tasks that are done with MRSR process are added to the MRSR merged list $MRSR\_M$ and they are not considered as targets for MRSR aggregation any more.

From line 7 to 18, the detailed process for generation of MRSR CandidateList is shown. All the possible task pairs except those in $MRSR\_C$ or $MRSR\_M$ are considered to be put into MRSR CandidateList. When the MRSR process is performed, the total processing time and resource purchasing cost of the workflow are recalculated to check constraints of MRSR scheme. If the new total processing time of workflow complies with the deadline and the resource purchasing cost is decreased, the pair of tasks is

added to the MRSRCandidateList. After the all the MRSR processes are performed, then we find the mergePair that minimizes the resource purchasing cost of the original scheduling of the workflow g in line 22. We apply that mergePair to the workflow g and repeat the MRSR process until the MRSRCandidateList is empty.

In conclusion, we can generate the final scheduled workflow g that decreases the resource purchasing cost compared to the traditional workflow scheduling approaches while guaranteeing its deadline by using MRSR scheme. In next chapter, we evaluate our proposed approach compared to the existing approaches in views of resource purchasing cost and processing performance by experiments based on the open source cloud platform.

Algorithm 2. Task aggregation of workflow by MRSR scheme

Input : Scheduled workflow g with VM packing

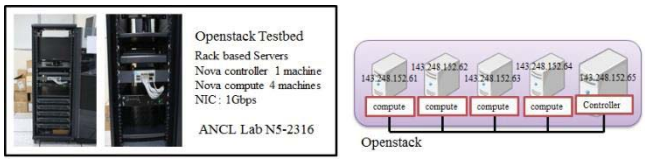
Output : Scheduled workflow g with MRSR scheme

```
01: tempWorkflow g' <- copy workflow g
02: MRSRCandidateList MRSR_C
03: mergedList MRSR_M
04:
05: while(true)
06:   cost c <- calculate the cost of g by using (1)
07:   For(each task i ∈ g & ∉ MRSR_C, MRSR_M)
08:     For(each task j ∈ g & ∉ MRSR_C, MRSR_M & ≠ i )
09:       if(st(i) ≤ st(j) ≤ et(i))
10:         workflow g' <- apply MRSR aggregation with (i, j) to workflow g
11:         calculate total processing time t of workflow g' by using (9), (10)
12:         calculate cost c' of workflow g' by using (1)
13:         if(t ≤ deadline(g') & c < c')
14:           MRSR_C <- insert task pair(i, j)
15:         end if
16:       end if
17:     end for
18:   end for
19:   if(MRSR_C = empty)
20:     break
21:   end if
22:     mergePair <- getTaskPair with minimum cost from MRSR_C by using (11)
23:   workflow g <- apply MRSR aggregation with mergePair
24:   MRSR_M <- mergePair
25:   MRSR_C <- empty
26: end while
```

4. EXPERIMENTAL ENVIRONMENT AND PERFORMANCE EVALUATION

In order to evaluate the performance of our proposed VM packing and MRSR scheme, we implement the Cloud Resource Brokering System based on the open-source cloud platform called OpenStack [12] that supports a variety of hypervisors such as XEN, KVM, etc. Especially, we deploy the component of the OpenStack called Nova that provides the virtualized computing services to the cloud service users and manages VM instance allocation, VM image registration and VM flavor type management. The Nova controller operates as a front-end machine in OpenStack and Nova compute nodes practically allocate VM instances to process the tasks or applications from the cloud service users. The detailed configuration of our experimental environment is shown below in Figure 3.

We have 5 HP Xeon (2.4Ghz) physical machines including 1 controller node and 4 computing nodes. Each node has two wired NICs which constitute public and private networks and we are able to support VM flavor types from gogrid.small to xlarge since all the nodes have 16 cpu-cores and 16GB memory. Our Openstack cloud platform is based on Ubuntu 12.04 and called Essex. Through the Cloud Resource Brokering System with cloud node adaptor based on RESTful webservice API, the VM instances, which are on the Nova compute nodes, are allocated to cloud service users in order to process their applications. All the tasks of the workflow process are submitted to the Workflow Management Module in the Cloud



	Node 1	Node 2	Node 3	Node 4	Node 5
Functions	Cloud Controller node Network, Volume, API, Scheduler and Image Services Nova compute	Nova Compute	Nova Compute	Nova Compute	Nova Compute
Specification	H/W : HP, Xeon E5620 2.4G, 8 Cores, Mem 16G, HDD 1T, Wired NIC 2EA OS : Ubuntu 12.04				
IP addresses	Eth0 (143.248.152.61)	Eth0 (143.248.152.62)	Eth0 (143.248.152.63)	Eth0 (143.248.152.64)	Eth0 (143.248.152.65)
	Eth1 (192.168.3.1)	Eth1 (192.168.3.2)	Eth1 (192.168.3.3)	Eth1 (192.168.3.4)	Eth1 (192.168.3.5)

Figure 3. Openstack based experimental system

Resource Brokering System and distributed to each VM instances according to the VM packing and MRSR scheme.

In order to evaluate the performance of our proposed scheme, we adopt the Montage project as a practical workflow example that is a famous open-source based scientific application [13]. Generally, the scientific workflow application has dozens or hundreds of tasks to derive the scientific results. 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.

The custom mosaics are built from the 2-Micron All Sky Survey (2MASS) Atlas images. In Tera Grid project, the Montage application is processed on the grid environment as a workflow [13]. Similarly, we design the Montage workflow example to evaluate the performance of our MRSR scheme. We schedule the tasks constituting Montage workflow example that is shown in table 2 onto the VM instances in order to evaluate the efficiency of our proposed MRSR algorithm. Our Montage workflow example consists of three processes for image M105, M106 and M108. The measure range vale of M105 is 1.5, M106 and M108 is 1.7. The scope of the range value is from 0.05 to 7.0 and as the size of the range value is increased, the processing time is also increased since the image volume increases in size.

In order to process the tasks of the workflow, all the images are submitted to the cloud broker as FITS typed raw data files and generated as visible JPEG files by processing sub tasks of the workflow such as mImgtbl, mMakeHdr, mProjExec, mAdd and mJPEG. However, mImgtbl and mMakeHdr among all the sub tasks constituting the Montage workflow example require

negligible processing time compared to other sub tasks, therefore we only focus on the mProjExec, mAdd and mJPEG to be processed by MRSR approach. Firstly, to perform MRSR algorithm to our workflow example, we need to check the processing time of each sub tasks in the workflow.

As in table 2, generally the sub task mProjExec requires the longest processing time regardless of the image type, therefore the whole processing time of the workflow depends on mProjExec. The deadline of the workflow is set by 170 minutes. In order to compare our prosed MRSR approach and other traditional researches, we measured resource operation cost, resource dissipation cost, workflow processing time and resource utilization ratio. We evaluate the traditional approaches called MDP based workflow partitioning and heuristic based workflow scheduling approaches [8, 18].

Table 2. The processing time of Montage workflow

Images	Tasks	small	medium	large	xlarge
M105 1.5	mProjExec	127	120	117	116
	mAdd	27	19	7	7
	mJPEG	7	7	6	6
M107 1.7	mProjExec	157	153	145	145
	mAdd	36	31	9	8
	mJPEG	8	7	7	7
M108 1.8	mProjExec	150	140	140	140
	mAdd	36	20	8	7
	mJPEG	9	8	8	8

In Figure 4, we show the resource operation cost graphs of the workflow partitioning, heuristic based GAIN/LOSS and our MRSR approach. The MDP based workflow partitioning and heuristic based GAIN/LOSS algorithm are optimized to the grid environment but they are not to the cloud environment which adapts the pricing policy based on the predefined resource allocation period, therefore unnecessary resource operation cost is imposed in the cloud environment. That is, in both two approaches, the VM instances are allocated to each task and its allocation duration is determined by cloud resource provider regardless of the practical task processing time, therefore resource dissipation might be occurred. However, we are able to reduce the number of allocated VM instances by using VM packing which packs multi tasks to the single VM instance.

We deduct several results in the Figure 4. Firstly, the cost reduction ratio of LOSS is bigger than GAIN, but this result might be different based on the task structure of the workflow. The important thing is that both in GAIN and LOSS algorithms, the resource operation cost can be decreased by using our VM packing approach. When the sequential task packing by VM packing is finished, then the parallel task packing is performed by using MRSR approach. That is, two chosen task pairs are reassigned to the single VM instance without any deadline violation, and the number of VM instance is decreased again by MRSR scheme. By using GAIN based MRSR and LOSS based MRSR, the cost reduction ratios are 24%, 31%, 24% and 34%, 40%, 34% respectively, compared to the MDP based workflow partitioning and GAIN/LOSS approaches.

In Figure 5, the resource dissipation cost of allocated VM instances is shown as graph. The formula of VM dissipation cost is as follows,

$$\sum_{i=1}^{n}(allocation\ time_i - active\ time_i) \cdot cost_i \qquad (12)$$

That is, as the idle time of allocated VM instance is increased and the number of idle high-capacity flavor typed VM instances is increased, then the resource dissipation cost is also increased. Similar to the Figure 4, in the MDP based workflow partitioning and heuristic based GAIN/LOSS approaches in which VM instances are assigned to each task respectively, the idle period of resource allocation is large, therefore resource dissipation cost is also large as shown in Figure 5. However, in our VM packing and MRSR approaches, tasks are packed as many as possible to the allocated VM instance without any deadline violation so as to reduce the resource dissipation cost compared to the traditional approaches. By using GAIN based MRSR and LOSS based MRSR, the resource dissipation cost is decreased about 54%, 57%, 54%

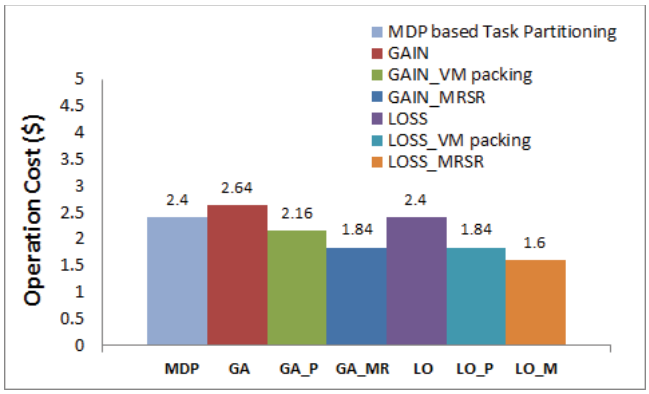


Figure 4. Resource operation cost of traditional and proposed approaches

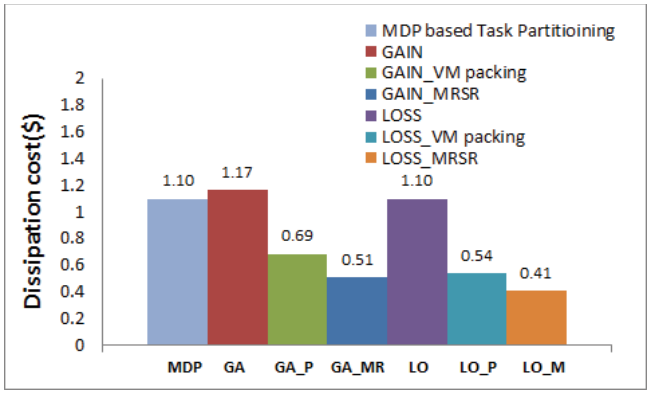


Figure 5. Resource dissipation cost of traditional and proposed approaches

and 63%, 65%, 63% compared to the MDP based workflow partitioning and GAIN/LOSS.

The resource utilization ratio is shown in Figure 6. In MDP based workflow task partitioning and GAIN/LOSS approach, the resource utilization ratio is about 50%, this means that the half of allocated resource is dissipated. Our MRSR approach improves the performance of resource utilization about 20% better than existing approaches.

The whole processing time of Montage workflow example is shown in Figure 7. The total processing time of workflow is determined by the critical path length of sub tasks constituting the workflow. This means that although the processing time of tasks that are not included in the critical path of the workflow is increased, if it is within the critical path length then the total processing time is not increased. That is, although the processing time is increased by task aggregation by using out VM packing and MRSR approach, the deadline of the workflow is not violated. Compared to the traditional GAIN/LOSS approaches, in our VM packing and MRSR approaches, the workflow processing time is not increased but rather is decreased about 1 minutes in GAIN based MRSR approach.

In the VM packing process, the task overlapping is not permitted, therefore the workflow processing time is not increased. Contrast to VM packing process, each task processing time might be increased by task overlapping in the MRSR approach. However the total processing time is not increased if the aggregated tasks are not

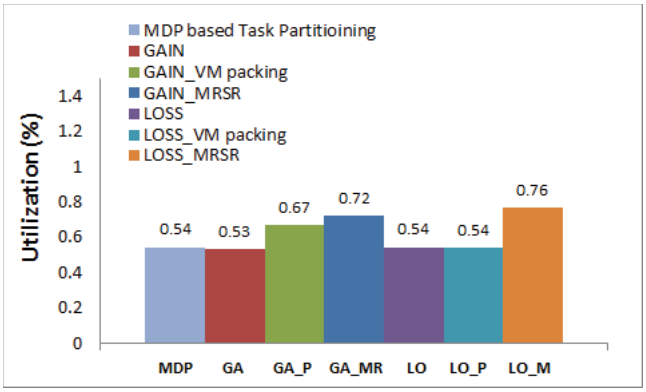


Figure 6. Resource utilization of traditional and proposed approaches

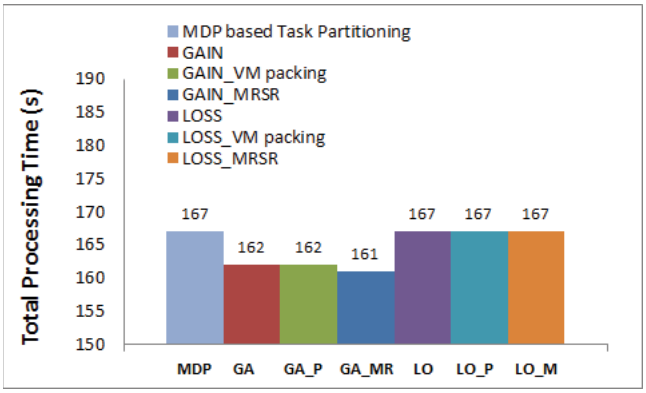


Figure 7. Processing time of traditional and proposed approaches

belong to the critical path or the reassigned VM flavor type has better capacity than original assigned flavor type. In latter case, the performance improvement by VM reassignment is larger than the performance degradation by task aggregation.

Obviously, this result cannot be generalized in all the cases, just in the specific workflow example cases, but we can conclude that we are able to have chance not only to reduce the resource operation cost but increase the performance of processing compared to traditional approaches beyond deadline assurance.

5. Conclusion

In this paper, we propose novel workflow scheduling approach that is optimized to the cloud environment, which has period based pricing policy in order to overcome the limitation of traditional workflow scheduling approaches those are optimized to the grid environment. Our scheduling approaches are composed of VM packing and Multi Requests to Single Resource (MRSR) scheme those are able to reduce the resource operation cost significantly with users SLA assurance by allocate multi tasks of the workflow to the single VM instance.

We evaluated our VM packing and MRSR approach on the Openstack based cloud platform and result in the performance improvement in terms of resource operation cost about 30% compared to MDP based workflow partitioning and heuristic based workflow scheduling algorithm without any deadline violation.

In future works, we consider not only processing cost but also communication cost of tasks between allocated VM instance in order to apply our proposal to the practical workflow management system.

6. 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 by the IT R&D program of MSIP/KEIT (10038768, The Development of Supercomputing System for the Genome Analysis) and by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-(H0301-13-4006)) supervised by the NIPA (National IT Industry Promotion Agency).

7. REFERENCES

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing : State-of-the-art and research challenges," J. Internet Services and Applications, vol. 1, issue 1, pp. 7-18, 2010.

[2] H. N. Van, and F. D. Tran, "Autonomic virtual resource management for service hosting platforms," Proc. Int'l Workshop. CLOUD, 2009.

[3] M. Mao, J. Li, M. Humphrey, "Cloud Auto-scaling with Deadline and Budget Constraints," Proc. Int'l Conf. IEEE/ACM Grid Computing, 2010.

[4] S. Son, and K. M. Sim "A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations," IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 42, no. 3, June 2012.

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

[6] https://cloud.google.com

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

[8] J. Yu, R. Buyya, and C. K. Tham, "Cost-based Scheduling of Scientific Workflow Applicationcs on Utility Grids," Proc. Int'l Conf. e-Science and Grid Computing, pp. 140-147, July 2005.

[9] J. Yu, R. Buyya, and C. K. Tham, "Qos-based Scheduling of Workflow Applications on Service Grids," Proc. Int'l Conf. e-Science and Grid Computing, pp. 140-147, July 2005.

[10] J. Yu, and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," J. Grid Computing, vol. 3, issue 3-4, pp. 171-200, 2005

[11] J. Tao, K. Furlinger, L. Wang, and H. Marten, "A Performance Study of Virtual Machines on Multicore Architectures," Proc. Int'l Euromicro Conf. Parallel, Distributed and Network-based Processing, 2012.

[12] Openstack (2013) http://www.openstack.org/

[13] http://montage.ipac.caltech.edu/

[14] G. B. Berriman, E. Deelman, J. Good, J. Jacob, D. S. Katz, C. Kesselman, A. Laity, T. A. Prince, G. Singh, and M. H. Su, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," Proc. SPIE5493, Optimizing Scientific Return Astronomy through Information Technologies, 2004.

[15] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the Montage example," Proc. Int'l Conf. ACM/IEEE supercomputing, 2008

[16] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274, Mar 2002.

[17] R. Sakellariou, and H. Zhao, "A hybrid heuristic for DAG scheduling on heterogeneous systems," Proc. Int'l Conf. Parallel and Distributed Processing Symposium, 2004.

[18] R. Sakellariou, and H. Zhao, "Scheduling workflows with budget constraints," Integrated Research in GRID Computing, CoreGRID Series, S. Gorlatch, and M. Danelutto, eds., pp. 189-202, Springer, 2007.

[19] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 8, pp. 1400-1414, Aug 2012.

[20] M. Mao, and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis (SC), 2011.

[21] M. Mao, and M. Humphrey, "Scaling and Scheduling to Maximize Application Performance within Budget Constraints in Cloud Workflows," Proc. Int'l Symp. Parallel and Distributed Processing, 2013.

[22] D. K. Kang, S. H. Kim, Y. Ren, B. S. Kim, W. J. Kim, Y. S. Kim, C. H. Youn, and C. S. Jeong, "Enhancing a Strategy of Virtualized Resource Assignment in Adaptive Resource Cloud Framework," Proc. Int'l Conf. ACM ICUIMC, 2013.