Network-Aware VM Placement for Scientific Workflow Application

Woojoong Kim and Chan-Hyun Youn

Department of Electrical Engineering
KAIST, Daejeon, Korea
{w.j.kim, chyoun}@kaist.ac.kr

Abstract. In the case of research institute that maintains their own private cloud and it's metadata such as network topology and hardware specification, VM placement which decides a certain physical node (i.e. a blade server in a rack) within the cloud for creating virtual machine (VM) instance is important in order to guarantee the performance of communication between VMs and reduce the occurred data transmission delay between VMs for workflow processing. However, the conventional algorithms are concentrated on the placement for a VM creation requested at a certain time so cannot guarantees the network performance for the successive VM creation requests in time line from scientific workflow application. In this paper, to resolve this problem, we propose the new network-aware VM placement algorithm for scientific workflow applications. In addition, we consider the network interference occurred between the consolidated VMs on the same physical node in cloud environment.

Keywords: Cloud Computing, Scientific workflow application, VM placement.

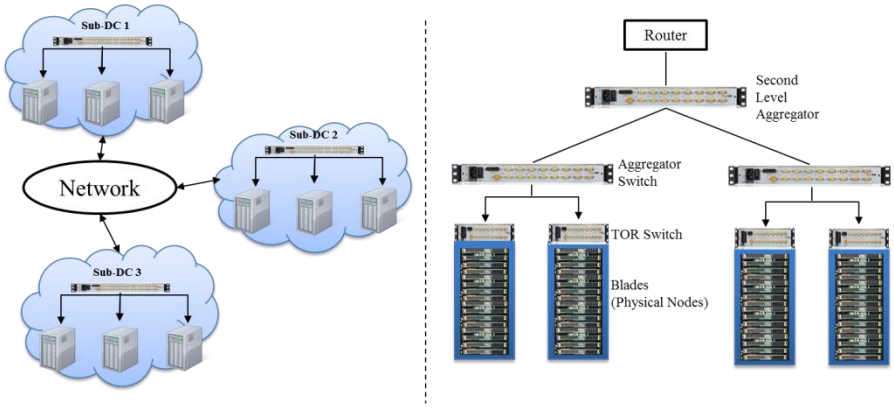
1 Introduction

Cloud Computing technology has been developed for years and considered as a new computing model, in which computing or storage resources are provided through internet in an on-demand fashion so, the resources can be leased or released when user want to use or not and they pay as they use [1]. According to Armbrust M et al. [2], cloud computing is defined as 'both the applications delivered as services over the internet and the hardware and systems software in the data centers that provide those services'. Due to the characteristics of cloud computing such as scalability and pay-for-use, many computer based field pay attention to cloud computing such as chemistry, biology and physics. The one of main concerns is a scientific workflow application service for the generation of science-grade mosaics of the sky in astronomy and the underpinnings of complex diseases in bioinformatics in cloud environment. The previous works for workflow management is mainly concentrated on the workflow scheduling which maps VM flavor type and instance to each sub-task within the workflow[3,4,5,6]. However, in the case of research institute that maintains their own private cloud and it's metadata such as network topology and hardware specification, VM placement which decides a certain physical node (i.e. a

blade server in a rack) within the cloud for creating virtual machine (VM) instance is also important in order to guarantee the performance of communication between VMs and reduce the occurred data transmission delay between VMs for workflow processing. Especially, for scientific workflow applications which make data in the unit of GB on processing, data transmission delay can be the considerable performance degradation for each request on a scientific workflow application. There are several previous works for VM placement. Zamanifar et al. [7] introduced the data-aware VM placement algorithm to reduce the data transfer delay by the optimization for VM placement and their allocated data rates. Alicherry et al. [8] introduced the network-aware VM placement in a heuristic way to guarantee the network performance between VMs specified by user in distributed clouds. However, these conventional algorithms are concentrated on the placement for a VM creation requested at a certain time so cannot guarantees the network performance for the successive VM creation requests in time line from scientific workflow application. In this paper, to resolve this problem, we propose the new network-aware VM placement algorithm for scientific workflow applications. In addition, we consider the network interference occurred between the consolidated VMs on the same physical node in cloud environment [9].

2 Model

We focus on the distributed cloud environment which consists of many small-scale data centers distributed over a geo-graphic area [10]. This environment can reduce the access latency for customer requests by covering wide area locations compared to the traditional centralized cloud environment which large-scale datacenters are placed at a few locations. The network architecture in this environment is typically organized in a hierarchical manner [8]. Fig. 1 shows the distributed cloud environment with this network architecture.

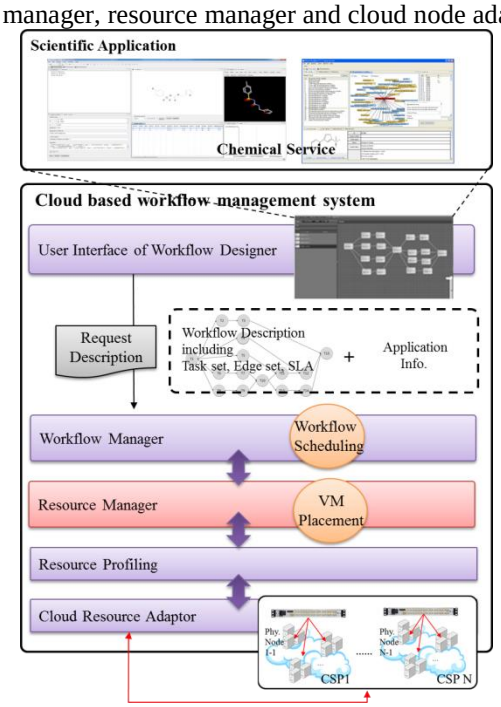


(a) A distributed cloud architecture **(b) A typical datacenter in the distributed cloud**

<Fig. 1. A distributed cloud environment **>**

Each rack in the distributed cloud environment shown in fig. 1 is communicated using aggregator switches. Several blade servers are included in each rack and communicated using a top-of-the-rack (TOR) in each rack. The VMs in the same blade server can communicate directly each other without going through any external switch. Each VM for a customer request is created on a blade server. As the distance between the created VMs increases, the network latency between VMs increases. Therefore, the network performance of the VMs for customer's application depends on the blade servers which they are allocated to. A cloud environment typically provides VM in VM flavor type which is the virtual hardware templates defined by sizes for RAM, disk, number of cores, and so on. (e.g. small, medium and large type provided by a cloud service provider such as GoGrid[11])

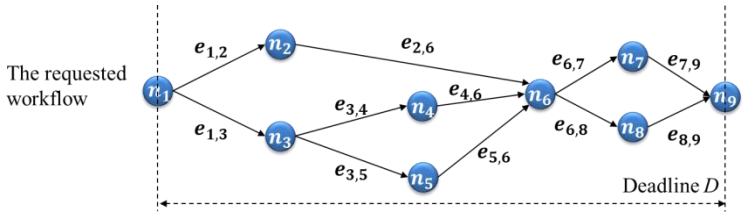
On this distributed cloud environment, we implement the cloud based workflow management system to provide a scientific workflow application shown in fig 2. This cloud based workflow management system consists of four main modules – workflow designer, workflow manager, resource manager and cloud node adaptor.



<Fig. 2. Cloud based workflow management system **>**

The workflow designer provides the GUI user interface for a scientific workflow application to users so they can specify their request description and submit the scientific workflow application graphically to the workflow manager. The request description is composed of user id *uid* and the workflow description *G*. Scientific workflow applications considered in this paper are represented as directed acyclic graph (DAG) as [12]. To represent the workflow *G* of DAG, n_i denotes the *i* th node of workflow *G* (node is identical to sub-task) in our denotation. The connection or dependency from node *i* to *j* in the workflow is represented as edge $e_{i,j}$ and the set of

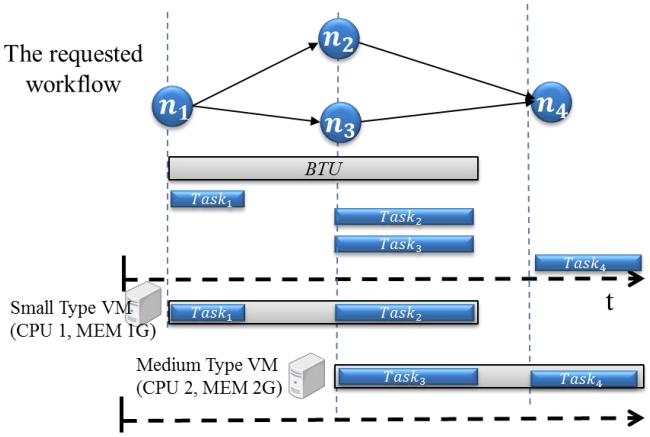
edge is represented as *E*. In addition, the QoS constraints *Q*, e.g. the execution time constraints required by user for the requested workflow, is also included in the workflow description. The workflow description is represented as {N = $\{n_1, n_2, \dots, n_k\}$, E, Q}. Fig. 3 shows the example of the requested workflow.



<Fig. 3. The example of the requested workflow>

Workflow manager decides and allocate VM flavor type to each sub-task within the requested workflow from each user, in other words, workflow manager does workflow scheduling. Resource manager locates and creates the requested VM of the decided VM type from workflow manager for the requested workflow by each user, in other words, resource manager does VM placement. The cloud node adaptor provides the integrated interface to access, create and delete VM instances for different cloud environments through RESTful API interface [13] so integrates each cloud environment into the single huge cloud environment virtually.

When a request for a scientific workflow application is submitted represented by the request description through the workflow designer, the workflow manager parses the workflow description and does workflow scheduling to satisfy the requested QoS constraints such as deadline while minimizing the cost for leasing VMs. To do this, we use the conventional workflow scheduling algorithms - VM Packing[14] and Loss/Gain[5]. Fig 4 shows the example procedure of the conventional workflow scheduling.



<Fig. 4. The example procedure of the conventional workflow scheduling [14,5]>

After completing the workflow scheduling process, the workflow manager requests a VM instance to the resource manager for each sub-task within the requested workflow based on the scheduled information mapping between each sub-task and the VM instance and dispatches the sub-task to the created VM by the resource manager. In this process, the resource manager decides a certain physical node within the cloud environment for creating the requested VM instance based on our proposed network aware VM placement algorithm. In next section, we describe the proposed algorithm in detail and show that it can reduce the occurred data transmission delay between VMs and improve the performance (i.e. total processing time) for workflow processing efficiently.

3 Network aware VM placement algorithm

This algorithm guarantees the network performance for the successive VM creation requests in time line from scientific workflow application considering the network interference occurred between co-location VMs, so that reduce the occurred data transmission delay between VMs for workflow processing efficiently. To do this, the last used resource table *LastUsedResourceTable* is maintained in order to store the physical node used lastly to create the VM for each user. Since there is no available last used resource information when a user requests a VM first, the VM is created to a physical node *maxCapacityPN* having maximum remaining resource capacity within distributed clouds in order to increase the probability that the physical node keep a certain amount of resource capacity for the next VM of the user. After creating the VM, the physical node of the created VM are recorded to the last used resource table with the user id *uid*. Since the last used resource information is available in the last used resource table when the user requests again, a VM is created in the same physical node with the last used resource if possible in order to retain the maximum bandwidth by communicating directly. If the resource capacity of this physical node is not available for the requested flavor type *f* from the user, available physical nodes *availablePNs* 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* and having the smallest network traffic decided to minimize the network interference from other VMs on the physical node. The network traffic of each physical node is monitored in real-time. After finding the new physical node and creating the VM, the resource information on the new physical node is updated to the last used resource table for the corresponding user. Algorithm 1 shows the procedure of the proposed network aware VM placement algorithm.

Algorithm 1. Network-aware VM placement algorithm

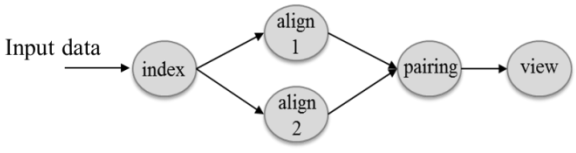
Input *: uid^k, f (uid^k : request id, f : flavor type)*

Output *: created VM*

```
01: if LastUsedResource of uid^k is available
02:       physicalNode p^k ← get LastUsedResource of uid^k
03:       if physicalNode p^k is available for flavor type f
04:          createdVM = createNewVM(p^k, f )
05:          return createdVM
06:       else
07:          sort availablePNs in closest order from p^k
                and split availablePNs into sameLevelPNs (∈ availablePNs )
08:          for sameLevelPNs ∈ availablePNs
                sort sameLevelPNs in smallest traffic order
             for p^i ∈ sameLevelPNs
09:             if p^i is available for flavor type f
10:                createdVM = createNewVM(p^i, f )
11:                record {uid^k, p^k} into LastUsedResource
12:                return createdVM
13:             end if
             end for
14:          end for
15:          return null
16:       end if
17:    else
18:    maxCapacityPN ← max( remainCapacity(p^j) )      ···  ( p^j ∈ PNSet )
19:    if maxCapacityPN is available for flavor type f
20:       createdVM = createNewVM(maxCapacityPN, f )
21:       record {uid^k, p^k} into LastUsedResource
22:       return createdVM
23:    else
24:       return null
25:    end if
26: end if
```

4 Experiment

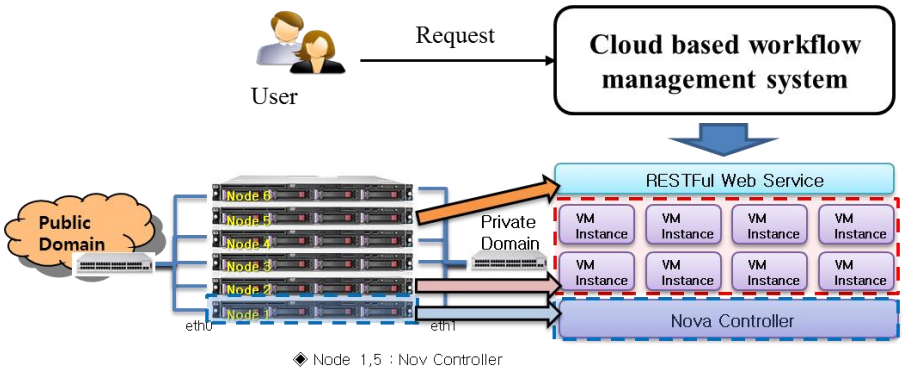
To evaluate the performance of the proposed network-aware VM placement algorithm, we evaluate the proposed scheme compared to the typical network-aware resource provisioning scheme proposed by Alicherry, M et al[8]. We use Burrows-Wheeler Aligner (BWA) which is the typical bio scientific application for analyzing the genome[15]. This application has various tasks (i.e. BWA index, alignment, pairing and view etc.). A request is comprised of the BWA tasks as shown in the fig. 5 and the input data is same for each request.



< **Fig. 5.** A request of BWA service for the experiment>

To be independent to the workflow scheduling, all tasks are allocated in large VM type. Finally, we make the requests in the different interval time (8 sec, 7 sec, 6 sec, 5 sec, 4 sec) within 3min and measure the occurred total data transmission delay in this experiment.

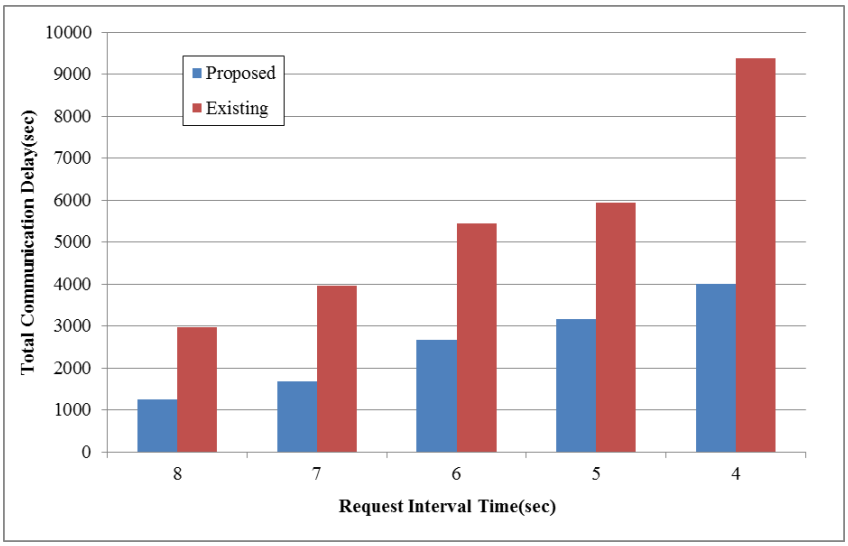
We use openstack cloud environment and the available resource policies in openstack cloud environment are small type(1 CPU, 2GB MEM, 10GB Disk), medium type(2 CPU, 4GB MEM, 10GB Disk) and large type(4 CPU, 8GB MEM, 10GB Disk). We build the cloud testbed using 4 nodes with the deployment of openstack cloud environment[16]**.** Node 1 is nova controller and node 2,3,4 are nova compute nodes. Each node has the hardware specification described in Table 1.



< **Fig. 6.** The experiment environment with openstack[16]>

< **Table 1.** The specification of nodes in the openstack environment >

	Node1	**Node2**	**Node3**	**Node4**
Function	Nova Controller Node	Nova Compute Node	Nova Compute Node	Nova Compute Node
Specification	**H/W**: Intel , Xeon E5620 2.4G, Core 16, MEM 16G, HDD 1T **OS**: Ubuntu 12.04			
IP address	Eth0 (143.248.152.64)	Eth0 (143.248.152.61)	Eth0 (143.248.152.62)	Eth0 (143.248.152.63)



< **Fig. 7.** The total data transmission delay of proposed scheme and conventional scheme>

The proposed scheme has the smaller total data transmission delay over the entire request interval time compared to the conventional scheme as shown in fig. 7. The conventional scheme cannot guarantees the network performance for the successive VM creation request on BWA service compared to the proposed scheme because it is concentrated on the placement for a VM creation request and not considers the interference occurred by the traffic congestion. As a result, the conventional scheme shows the worse performance on the scientific workflow applications (including BWA service) which need the successive VM creation.

5 Conclusion

The conventional VM placement algorithms cannot guarantees the network performance for the scientific workflow applications (including BWA service) which need the successive VM creation in time line. In this paper, to resolve this problem, we propose the new network-aware VM placement algorithm for scientific workflow applications. In addition, we consider the network interference occurred between the consolidated VMs on the same physical node in cloud environment. Finally, we prove the proposed VM placement algorithm can reduce the data transmission delay for executing the scientific workflow application compared to the conventional algorithm.

Acknowledgments. This research was supported by Next-Generation Information Computing Development Program through the NRF funded by the Ministry of Education, Science and Technology (2010-002073) and the ICT R&D program of MSIP/IITP[10038768, The Development of Supercomputing System for the Genome Analysis]

References

1. Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the art and research challenges. J. Internet Services and Applications, 1(1), 2010.
2. Armbrust M et al (2009) Above the clouds: a Berkeley view of cloud computing. UC Berkeley Technical Report
3. Topcuoglu, Haluk, Salim Hariri, and Min-you Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing." Parallel and Distributed Systems, IEEE Transactions on 13.3 (2002): 260-274. ,
4. Sakellariou, Rizos, and Henan Zhao. "A hybrid heuristic for DAG scheduling on heterogeneous systems." Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. IEEE, 2004. ,
5. Sakellariou, Rizos, et al. "Scheduling workflows with budget constraints."Integrated Research in GRID Computing. Springer US, 2007. 189-202. ,
6. Yu, Jia, Rajkumar Buyya, and Chen Khong Tham. "Cost-based scheduling of scientific workflow applications on utility grids." e-Science and Grid Computing, 2005. First International Conference on. IEEE, 2005.
7. Zamanifar, Kamran, Nader Nasri, and M. Nadimi-Shahraki. "Data-aware virtual machine placement and rate allocation in cloud environment." Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on. IEEE, 2012.
8. Alicherry, Mansoor, and T. V. Lakshman. "Network aware resource allocation in distributed clouds." INFOCOM, 2012 Proceedings IEEE. IEEE, 2012.
9. Corradi, Antonio, Mario Fanelli, and Luca Foschini. "VM consolidation: a real case based on openstack cloud." Future Generation Computer Systems 32 (2014): 118-127.
10. SCOPE Alliance. Telecom grade cloud computing. www.scope-alliaince.org, 2011
11. GoGrid. [Online]. Available: http://www.gogrid.com/]
12. J. Yu, R. Buyya, “Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms Scientific Programming”, IOS Press (2006), pp. 14-217 -4
13. Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, 2000.
14. Kang, Dong-Ki, et al. "Cost adaptive workflow scheduling in cloud computing."Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication. ACM, 2014.
15. Burrows-wheeler aligner (bwa). http://bio-bwa.sourceforge.net/.
16. Openstack foundation. http://www.openstack.org/.