International Journal of Grid and Distributed Computing
Vol. 2, No. 4, December, 2009

An Ontology-based Resource Selection Service on Science Cloud

Hyunjeong Yoo, Cinyoung Hur, Seoyoung Kim, and Yoonhee Kim*

Dept. of Computer Science and Engineering
Sookmyung Women's Univ., Seoul Korea
http://dslab.sookmyung.ac.kr/
{warmy79, hurcy, sssyyy77, yulan}@sookmyung.ac.kr

Abstract

Cloud computing requires scalable and cooperative sharing the resources in various organizations by dynamic configuring a virtual organization according to user's requirements. Ontology-based representation of Cloud computing environment would be able to conceptualize common attributes among Cloud resources and to describe relations among them semantically. However, mutual compatibility among organizations is limited because a method applying ontology to Cloud is not established yet.

We propose to introduce a resource virtualization method using ontology. A new Virtual Ontology (VOn) is configured dynamically based on requirement of users, and the VOn is mapped to Cloud resources. Our service uses a Map/Reduce model for rapid and efficient merging a number of ontology. The execution environment is composed of selected resources on basis of a VOn, which is generated by Ontology Merge engine.

Keywords: *Ontology, Resource Selection, Semantic, Science Cloud, Cloud computing*

1. Introduction

A virtual organization in Cloud computing provides a uniform views by virtualizing various resources [1,2,3]. Cloud services are defined hierarchically by using Cloud computing ontology [7]. The layered approach represents inter-dependency and composability between the different services in the Clouds. This paper defines concepts, a method of applying ontology to Cloud computing is not tangible. Ontology descriptions and specific policies in resource management are not in the scope of this paper.

However, it is difficult to provide perfect resources from various organizations because management policies and descriptions about various resources are different in each organization. These differences cause a problem about providing a uniformed view from various resources.

Ontology-based resource description is proposed to solve these problems [4,5]. These researches are noticed as a novel method of a Grid resource description due to virtualizing common properties of resource based on ontology and representing relation among resources semantically. Moreover, there is research that produces a global ontology by merging each ontology existed in resource groups [6]. Currently, this research is at an early stage and is hard to provide interoperability among organizations because of merging resources from only static concepts in existed researches. In the paper [10], they propose a Semantically-Enhanced Resource Allocator (SERA) which is a scheduling system using customer requests and provides the ability of re-scheduling requests based on their priorities and considering advanced reservations. Nevertheless, the experiment is too limited to prove the proposed

* Corresponding Author

This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST)(No.2009-0084669)

concept. The characteristics of target applications are not considered.

In this paper, we propose an Ontology-based Resource Selection Service (OReSS) which provides a method of resource virtualization using merged ontologies which is interoperable among virtual organizations and then selects resources based on the VOn. OReSS on Cloud computing, especially, selects ontology candidates by calculating a degree of similarity function based on user's requirements. Selected ontologies are merged and provided to a user as a new VOn. In Cloud environment, following considerations are needed: first, searching resources by using Ontology is available, second, various ontologies in Cloud environment, which consists of diverse and heterogeneous virtual organizations, can be managed, and, finally, inference of similar resources with given information of a resource by Ontology can be possible.

The rest of the paper is organized as follows. Section 2 gives an overview of OReSS. In section 3, we describe an execution scenario of scientific applications in OReSS. Section 4 describes experiments and Section 5 discusses summary and future work.

2. Ontology-based Resource Selection Service (OReSS)

We propose an ontology-based resource virtualization about various Cloud resources. This section describes the architecture of OReSS and each function in the OReSS.

2.1. Architecture of OReSS

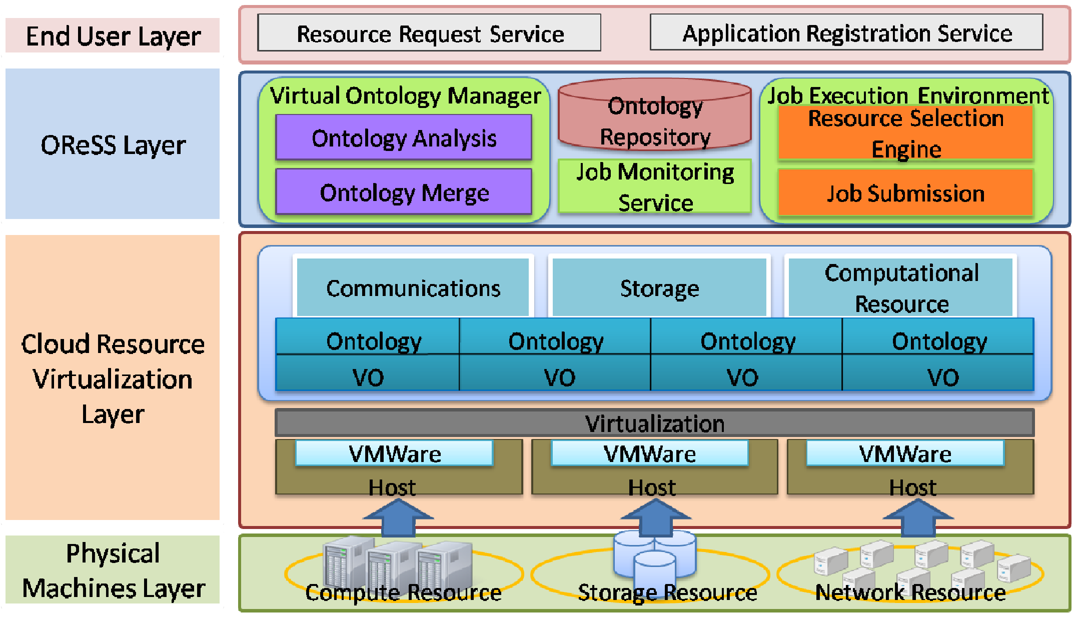

Figure 1. Architecture of OReSS

The architecture of OReSS (Figure 1) consists of four layers: Physical Machines (PM) layer, Cloud Resource Virtualization (CRV) layer, OReSS layer, and End User (EU) layer. The PM layer represents distributed and heterogeneous physical resource environments. The CRV layer supports resources which adapted to Cloud by using virtualization and Ontology for generating VOns. The OReSS layer creates a new resource among these VOns according to the user's requirements from the EU layer for providing adapting environment to user's program. The EU layer receives requirements about a resource from a user and supports an application execution service in the selected proper environment from the OReSS layer.

User's requests are information of resource components which users want to use, such as CPUs, memory sizes, network bandwidth, or IP numbers, and a Degree of Similarity (DS) as a reference value to be used in comparing resources with user's requirements and to choose proper resources which are closely related user's requirements. The OReSS layer can be categorized into: Virtual Ontology Manager, Ontology Repository, Job Monitoring Service, and Job Execution Environment. In detail, Ontology Analysis engine analyze ontologies and Ontology Merge engine creates a VOn from a pool of selected ontologies, and these engines consists Virtual Ontology Manager. The ontology is maintained by Ontology repository which is a storage space of resource ontologies. Job Execution Environment is made of Resource Selection engine, for selecting ontologies, and Job Submission, for connecting to the EU layer to receive user's requirements and to submit the new VOn.

2.2. Resource Selection Mechanism in OReSS

Job Submission receives user's requirements and DS from the EU layer. DS is used to merge associated ontologies which are resembled to required resource specification. Based on these requirements, Ontology Analysis engine calculates a similarity to each candidate resource in Ontology Repository. Then, this engine ranks candidate resources, according to the result of similarity calculation. During similarity calculation, we use a Similarity Computation Algorithm, ontology descriptions for finding ontology's concepts, and synonyms of reference [8].

After getting the result from Ontology Analysis engine, Resource Selection engine selects proper resources which have higher similarity value than DS. Then, Ontology Merge engine creates a new VOn by combining selected ontologies by Resource Selection engine. For merging ontologies, we use a method of the reference [6].

When resources are scheduled for executing jobs, following three factors are considered: the new VOn from Ontology Merge engine, status information of resources stored in VOns' resource pool, and resource requirements from a user. Finally, jobs run on the selected resource created by Ontology Merge engine.

2.3. Analyzing Ontology Similarity using Map/Reduce

Map/Reduce [9] computation is applied for rapid rank's calculation and merging the selected resources. Figure 2 shows the process of Map/Reduce computation for this paper. Ontology resources are split into M, which is the number of workers assigned to execute a map function. The workers for a Map function calculate a similarity value of resource ontologies based on user's requirements and resources' information in Ontology Repository. In this case, these workers, which are assigned a Map function, create <key, value> pairs as the intermediate value. In Reduce phase, the intermediate values are sorted based on similarity values in order of higher values. Then, a Reduce function collects sorted <key, value> pairs into an output called similarity rank. When any of the worker crashes, the other worker which have the same input files will run because of replica made by workers.

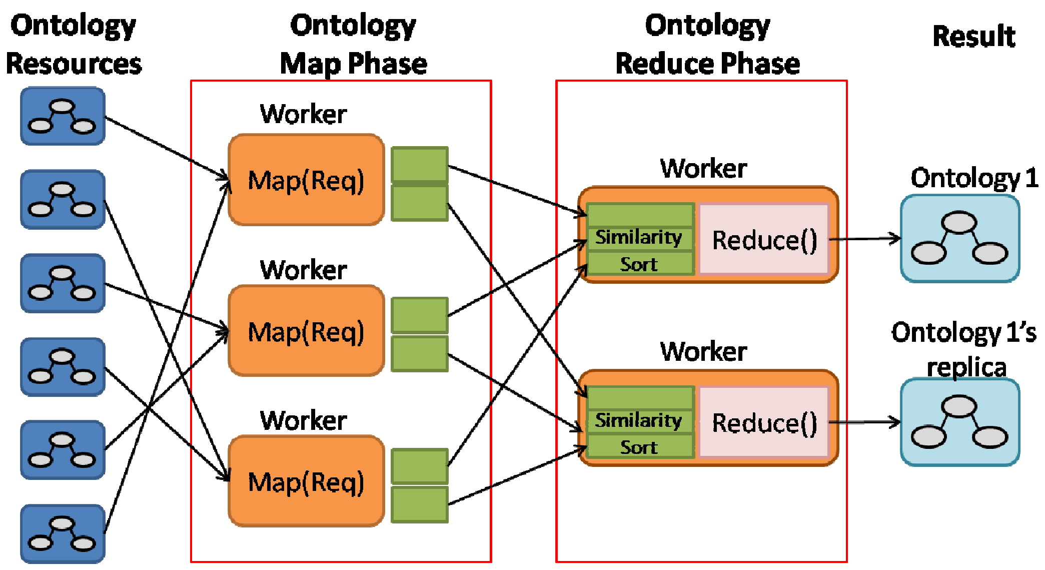

Figure 2. Map/Reduce computation

3. Scientific Applications Execution Scenario on Science Cloud

Understanding properties of a scientific application is important. For example, a Computational Fluid Dynamics (CFD) application (Figure 3) has multiple experiments with various parameter sets. Each experiment needs highly efficient and enormous computational resources concurrrently because these experiments are computational intensive and can be executed independently. With these requirements, generating a VOn in advance helps reduce resource selection time for locating appropriate resources.

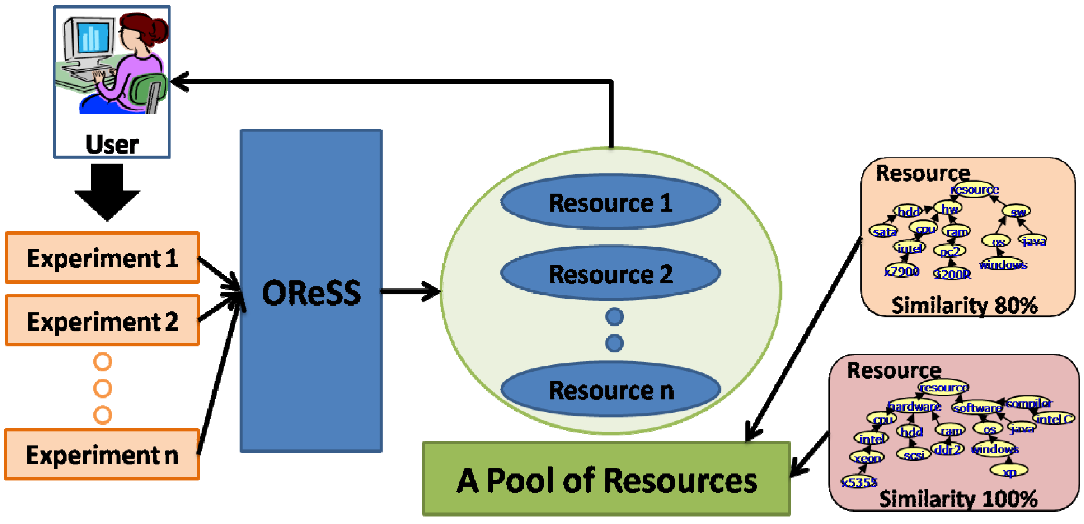

Figure 3. CFD Simulation on Cloud

In many times, total execution time of CFD applications depends on the number of assigned resources. To maximize performance of application execution, selecting appropriate resources for an application is important. When a user provides resource requirements for an application, many times we confront lack of resources which meet to the user's requirements. Small pool of resources causes longer job queuing time for an application to wait for its turn to use the resource. To reduce total execution time, decreasing job queuing time by expanding a number of available resources based on user's requirements is more efficient than diminishing job execution time. For instance, when the degree of similarity sets above 80 percent, throughput increases because many computational jobs are executed concurrently on selected ontologies. However, providing resources with the degree of similarity, 100 percent, results in increase of total execution time due to large queuing time because of limited resources.

4. Experiments

A target application in our experiment is a batch application composed of several jobs requiring large scale computation with small size of input data. Typically, a scientific computational application requires high throughput computing and has no dependency among jobs. According to this feature, the application depends more significantly on CPU performance, the number of resources, and memory size. However, network bandwidth or IO load to the application is not a major factor for performance. Therefore, in this experiment, we consider three factors for computing a similarity among clusters; CPU architecture, the number of CPU per a cluster, and MIPS based CPU performance.

To show the proof concept, we experiment an ontology created by the several VOns in PRAGMA environment with information of resource's components, and set 90% as DS. PRAGMA is Pacific Rim Application and Grid Middleware Assembly which has 35 institutions for establishing sustained collaborations and advancing the use of grid technologies. Difference of Cloud computing and Grid computing is booting overhead based on virtual machine images. However, this overhead is not significant, especially, when application is executed in a long time. That is, the time which is needed to boot a virtual machine has fewer effects on the overhead in longer execution time. So, in the paper, our experiment is applicable to PRAGMA.

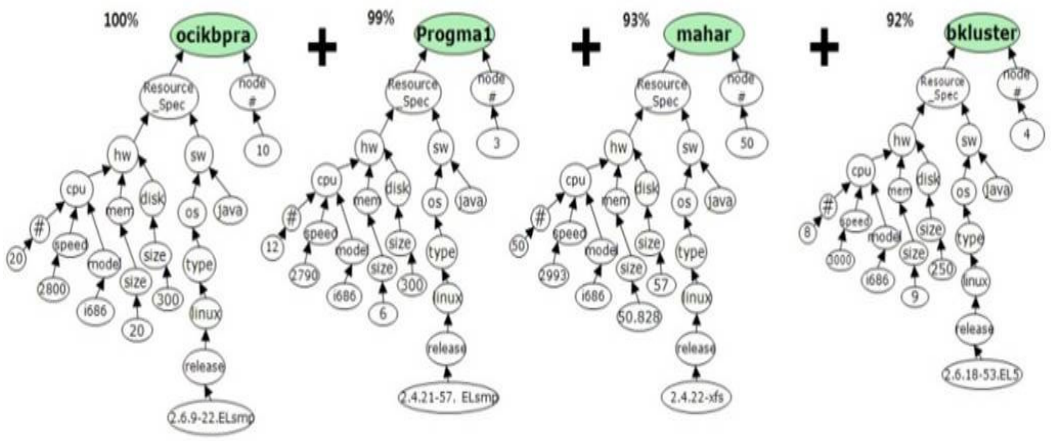

Figure 4. The target pool for the application execution

After calculating a similarity with the specific information of resource's components and merging the VOns whose calculated similarity is above 90%, respectively the target pool for the application execution is shown Figure 4.

A user inputs requirements of resources' components, the requirements' priority information which is used in a similarity computation phase, and DS. Then, in similarity computation, Ontology Analysis engine calculates similarity among the requirements and resources based on the priority information. Then, resources are selected by the similarity, and these resources are merged by Ontology Merge engine for executing jobs. 100 percent is based on similarity computation, and clusters which are greater than user's requirements are re-calculated based on 100 percent because selecting and supporting similar clusters with user's requirements are economical to users in the cost sides.

We assume that the cluster which matches 100 percent to user's requirements is ocikbpra and DS is 90 percent. Moreover, we set the priority in order of CPU architecture, MIPS based CPU performance, and the number of CPU per a cluster. The number of assigned jobs in each experiment is 100. When a job is assigned to these resources, Random scheduling is used. According to these conditions, selected resources are Pragma1 (99%), bkluster (93%), and mahar (94%). Table 1 shows properties of these experiment clusters.

Host Name	Node number	CPU number	Memory	Disk	CPU architecture	CPU clock	Similarity	Final Similarity
bkluster	4	8	9	250	i686	3000	107.14286	**92.85714**
mahar	50	50	50.828	57	i686	2993	106.89286	**93.10714**
Pragma1	3	12	6	472	i686	2790	99.642857	**99.64286**
ocikbpra	10	20	20	300	i686	2800	100	**100**

Table 1. Properties of selected clusters and a standard cluster

In Figure 5, we measure queuing time and total execution time in ocikbpra, which is the 100 percent matched cluster to user's requirements, with 100 jobs. In this result, there is no queuing time before 40th node, and, after 41th node, queuing time increases by around 200000. Moreover, after executing 81th node, queuing time and total time increase once more.

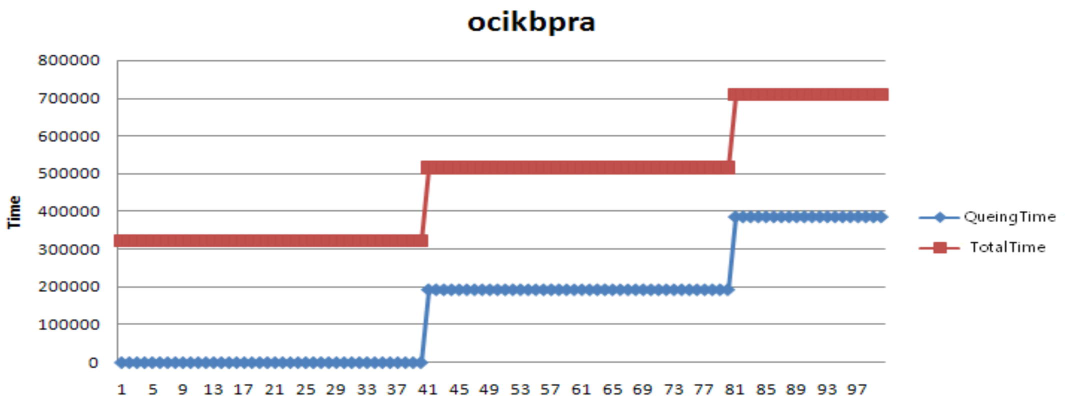

Figure 1. Queuing time and total time in ocikbpra

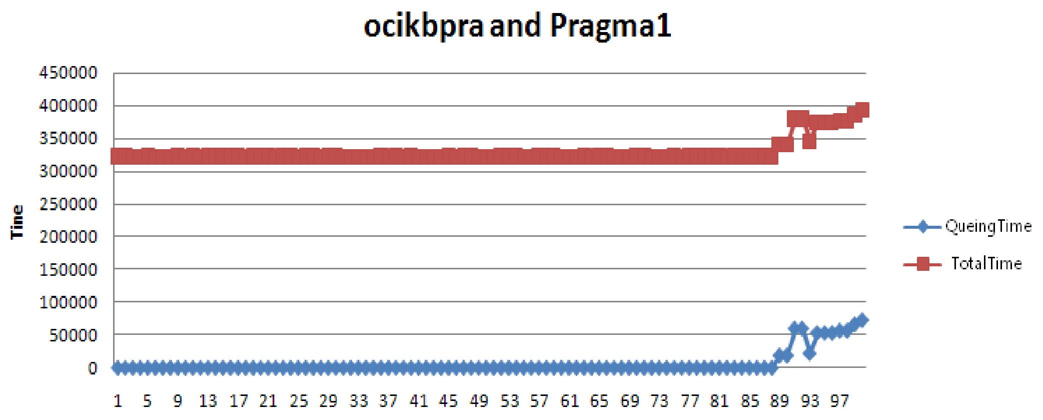

Figure 6. Queuing time and total time in ocikbpra and Pragma1

As can be seen the Figure 6, the merged cluster with ocikbpra and Pragma1, which is allowed to 99 percent similarity, indicates better performance than Figure 5. This cluster does not have queuing time before 88th node. Although, after 89th node, queuing time occurs, queuing time decreases because jobs are assigned to nodes with good performance in the expanded cluster. Average total execution time also decreases approximately by half compared with Figure 5 because the same reason.

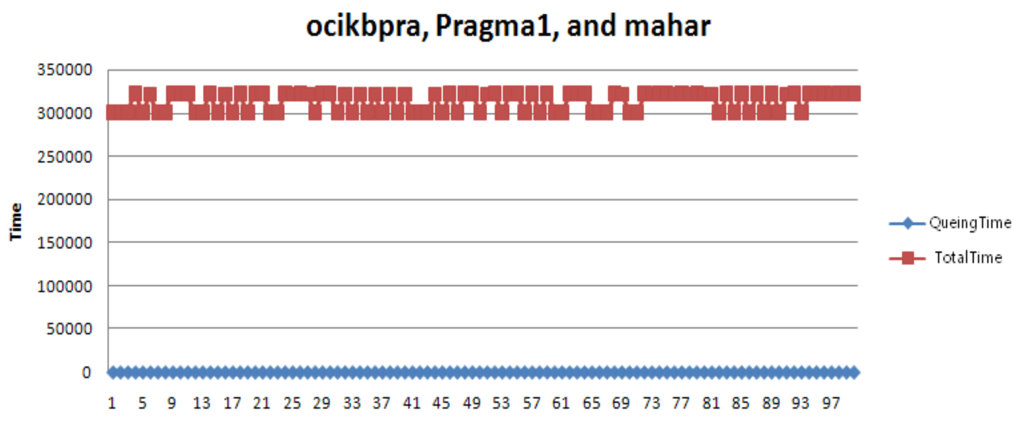

Figure 2. Queuing time and total time in ocikbpra, Pragma1, and mahar

Figure 7 shows the result of job execution in unified environment which consists of ocikbpra, Pragma1, and mahar. These resources have above 94 percent similarity. Because of adding resources with better performance, there is no queuing time during job execution, and assigned jobs are totally completed between 300000 and 350000. The range of total execution time is made by difference in execution time according to various CPU's performance among resources.

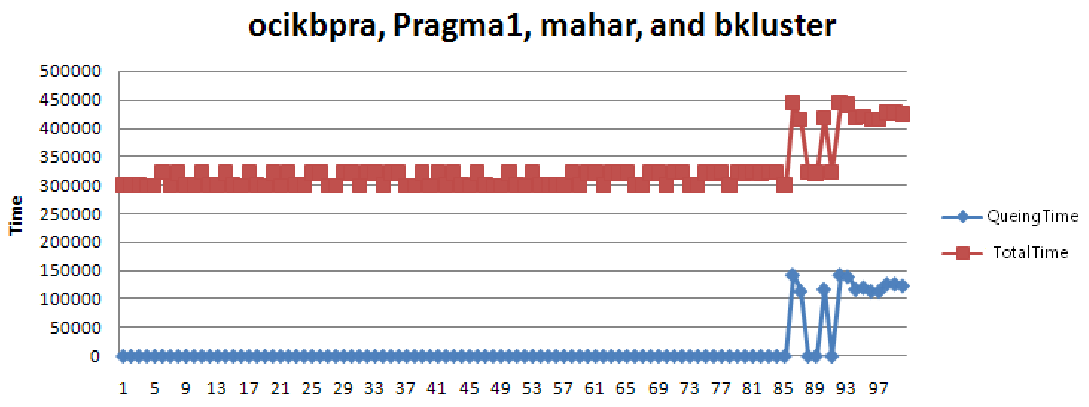

Figure 3. Queuing time and total time in ocikbpra, Pragma1, mahar, and bkluster

Figure 8 indicates the result of job execution in totally merged environment which consists of ocikbpra, Pragma1, mahar, and bkluster with higher similarity values than DS. In this experiment, before 85th node, there is no queuing time, and total execution time is similar to formal experiments. However, after 86th node, maximum total execution time sharply increases to 450000 because some jobs assigned to the last cluster, bkluster, are waiting to execution. This sharp increase is caused by the insufficient number of nodes in bkluster.

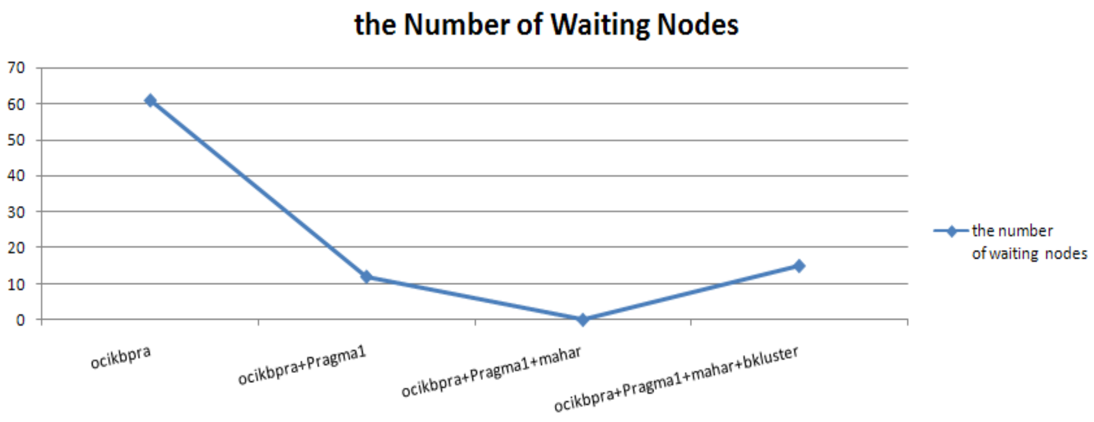

Figure 9. The number of queuing nodes in four types of clusters

Figure 9 shows the numbers of waiting nodes after 100 jobs are executed in the selected resources. While the number of waiting nodes is around 60 in ocikbpra, the number of waiting nodes in the merged resource with ocikbpra, Pragma1, and mahar, decreases by zero. Moreover, according to Figure 9, the best environment to execute user's jobs is the third merged resource.

As can be seen, executing jobs in the merged resource based on the calculated similarity value can be more efficient than running jobs on only one resource, even the 100 percent matched resource. Moreover, if DS is not 90 percent, but 94 percent, then the cluster which is created newly makes the best result. That is, selecting the proper DS is an important factor to make and guarantee the best result during job execution through assigning proper resources and the number of jobs. Using environment with several virtual resources, but not exactly same components with user's requirements, is more efficient than providing environment which consists of fewer resources with 100 matched percent, because of the amount of applicable resources. Then, this similarity value is recorded in a job profile and is able to utilize similar job execution later.

5. Conclusion and Future work

Resource management in Cloud computing is not easy. Managing distributed heterogeneous resources causes some problems: difficulty of resource information management, no standard definitions of resource requirements, and difficulty of guaranteeing compatibility of resource allocation. To solve these problems, we propose a resource virtualization method using ontologies in Cloud. A VOn is created by an ontology merge method based on DS. From the experiment, expanding clusters with the proper DS shows guaranteeing the best execution time with minimize job queuing time.

In this paper, we experiment in Grid computing environment because the difference between Grid computing and Cloud computing is small, as previously mentioned. Then, we are expected to make the same experiment in Cloud computing environment. In addition, merging cluster with DS based on weighted factors according to application characteristics will be discussed near future.

References

[1] Bassem Nasser, Romain Laborde, Abdelmalek Benzekri, François Barrère, Michel Kamel, Dynamic Creation of Inter-Organizational Grid Virtual Organizations, e-Science 2005, pp.405-412, 2005.

[2] Cannon S., Chan S., Olson D., Tull C., Welch V., Pearlman L., Using CAS to manage Role based VO sub-groups, In CHEP 2003, 2003.

[3] B. Nasser, R. Laborde, F. Barrere, A. Benzekri, M. Kamel, Grid Virtual Organization: Access Control management, Gres05, Luchon,France, Mars 2005

[4] A. M. Pernas, M. A. R. Dantas., Using Ontology for Description of Grid Resources, 19th Int. Symposium on HPC Systems and Applications, Guelph, Canada, 2005, pages 223-229.

[5] W. Xing, M. D. Dikaiakos, and R. Sakellariou., A core grid ontology for the semantic grid, In CCGrid 2006, Singapore, May 2006, pages 178–184.

[6] J. G. R. C. Lopes, A. C. M. A. Melo, M. A. R. Dantas, and C. G. Ralha, A proposal and evaluation of a mechanism for grid ontology merge, 20th HPCS, 2006.

[7] L. Youseff, M. Butrico, and D. Da Silva, D, "Toward a Unified Ontology of Cloud computing", IEEE, 2008

[8] Kim. Jeu Young, "Ontology-based Resource Selection Methods for Grid Computing", Master thesis, Sookmyung women's university, 2008

[9] J. Dean and S. Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters", Proc. Of the 6th Symp. On Operating Systems Design & Implementation, 2004, pp.137-150

[10] Jorge Ejarque, Marc de Palol, Inigo Goiri, Ferran Julia, Jordi Guitart, Jordi Torres and Rosa M. Badia, "Using Semantics for Resource Allocation in Computing Service Providers," IEEE, 2008 MIPS based CPU performance

Authors

Hyunjeong Yoo.
e-mail: warmy79@sookmyung.ac.kr
2008 B.S. Dept. of Computer Science Sookmyung Women's University
2009 ~ present Master Candidate, Dept. of Computer Science and Engineering at Ohio State University
Research Interests : Ontology, Distributed Systems

Cinyoung Hur
e-mail: hurcy@sookmyung.ac.kr
2007 B.S. Dept. of Computer Science Sookmyung Women's University
2009 Dept. of Computer Science Sookmyung Women's University
2009 ~ present Graduate Assistant, Distributed Systems Laboratory, Sookmyung Women's University
Research Interests: Grid Computing, Meta-scheduling

Seoyoung Kim.
e-mail: sssyyy77@sookmyung.ac.kr
2009 ~ present Undergraduate Assistant, Dept. of Computer Science Sookmyung Women's University
Research Interests: Infrastructure of Cloud Computing

Yoonhee Kim.
e-mail: yulan@sookmyung.ac.kr
1991 B.S. Dept. of Computer Science Sookmyung Women's University
1996 M.S. Syracuse University, Computer and Information Science
2000 Ph.D. Syracuse University, Computer and Information Science
1991 ~ 1994 Research Staff Member, the ETRI(Electronics and Telecommunication Research Institute)
2001 Faculty, Computer Engineering dept., Rochester Institute of Technology
2000 ~ present Professor, Dept. of Computer Science at Sookmyung Women's University
Research Interests: Runtime support and management in distributed computing systems