2010 10th Annual International Symposium on Applications and the Internet

A History-Based Job Scheduling Mechanism for the Vector Computing Cloud

Yoshitomo MURATA
Cyberscіecnce Center
Tohoku University
Sendai, Japan, 980-8578
murata@sc.isc.tohoku.ac.jp

Ryusuke EGAWA
Cyberscіecnce Center
Tohoku University
Sendai, Japan, 980-8578
egawa@isc.tohoku.ac.jp

Manabu HIGASHIDA
Cybermedia Center
Osaka University
Osaka, Japan, 567-0047
manabu@cmc.osaka.ac.jp

Hiroaki KOBAYASHI
Cyberscіecnce Center
Tohoku University
Sendai, Japan, 980-8578
koba@isc.tohoku.ac.jp

***Abstract*—The wide-area vector meta computing infrastructure named a vector computing cloud has been proposed as a next generation high-performance computing infrastructure. However, in the vector computing cloud, the difference in site policies between organizations causes inefficient usage of vector computing resources. To achieve fairness and efficient job scheduling on the vector computing cloud, this paper presents a history-based job scheduling mechanism for a queue system. The proposed mechanism estimates the time to start the job execution in a queue system from the history of job-execution on vector supercomputers. Based on the estimation, the job scheduling mechanism automatically allocates the job to an appropriate site, which can execute the job earlier.**

The simulation results show that the proposed job scheduling mechanism improves the utilization efficiency of vector computing resources, compared to the conventional round-robin scheduling mechanism. In addition, the experiment using a prototype of the vector computing cloud indicates that the proposed job scheduling mechanism has enough potential for transparently executing jobs between the two SX-9 systems.

***Keywords*-Job Scheduling, Grid Computing, Cloud, HPC**

I. Introduction

The current high performance computing (HPC) applications require ever-increasing computational power. To execute such an HPC application on existing computing resources, one way can conceive to make co-operation among distributed supercomputers by GRID technologies. However, a grid environment still forces HPC users to consider available computing resources to running huge and massively parallel HPC applications. In the conventional distributed HPC systems, users should select an appropriate site to execute their codes and submit HPC jobs directly by themselves [1].

Recently, the cloud computing attracts a great deal of interest as a new generation IT infrastructure. Although there are many definitions of the cloud computing [2], [3], the cloud computing is the internet-based computing in the broad sense. In the cloud computing, the resources, infrastructures and software are provided as a cloud service through the internet. Here, these cloud services are virtualized, and the cloud users need not attend to the actual resources and infrastructures. So, an HPC service provided by the cloud computing, which is called a HPC cloud, is seen as a high possibility to execute massive parallel applications [4], [5]. In the HPC cloud, users would not have to care where their jobs should be operated since the supercomputers are virtualized as a huge single system and jobs are automatically assigned to appropriate sites.

In [6], the wide-area vector supercomputing environment named a vector computing cloud has been proposed as one of HPC cloud systems. The vector computing cloud realizes a single-sign-on for multiple vector computing systems by virtualizing vector computing resources. However, the vector supercomputers which make up the vector computing cloud employ different job execution policies. The difference in the policies makes job scheduling on the vector computing cloud complicated, and it causes inefficient usage of vector computing resources. To overcome this problem, this paper proposes a history-based job scheduling mechanism for the vector computing cloud.

The organization of this paper is as follows: Section II describes the vector computing cloud and its job scheduling problems. Section III introduces a job scheduling mechanism of the vector computing cloud. In Section IV, the performance of the proposed job scheduling mechanism is discussed. Section V concludes this paper.

II. the Vector Computing Cloud

A. System Overview

Aiming at realizing the HPC cloud with vector supercomputers, the wide-area vector meta computing infrastructure named a vector computing cloud has been proposed. A prototype system of the vector computing cloud consists of two nodes of SX9 vector supercomputers. Each node is located at Tohoku University and Osaka University with a distance of 800km [6].

The vector computing cloud has been designed based on the NAREGI Grid Middleware [7]. The vector computing cloud consists of multiple sites with vector supercomputers, each of which has several components of NARGI Grid Middleware: Portal, User and Virtual Organization Membership Service (UMS/VOMS), Information service (IS), Super Scheduler (SS), and GridVM for SX. Portal provides a web interface of the virtualized system, and UMS/VOMS authenticates the users and servers. IS manages the resource information of each site with gathering utilization statuses. IS also communicates with IS's in other sites to share utilization statuses. SS searches computing resources for user requests, and schedules jobs based on the information from

978-0-7695-4107-5/10 $26.00 © 2010 IEEE
DOI 10.1109/SAINT.2010.43

IS. Reservation Cache Service (RCS) performs as a global scheduler among each site. RCS aggregates and controls the requests from SS's to find out an appropriate site to execute a job. GridVM for SX is a virtual machine for a SX vector supercomputer, which performs synchronization control of vector computing resources in the site, and provides mete-computing environments based on high affinity with a local job scheduler [8]. GridVM for SX keeps high compatibility with the local job scheduler NQS II [9] on the SX-9.

B. Job Scheduling Problem on The Vector Computing Cloud

In the vector computing cloud, each supercomputer has their own job execution policy. Then, the execution of a job submitted to the vector computing cloud also complies with the policy of allocated site. The difference in the policies makes job scheduling on the vector computing cloud complicated. In the prototype system of the vector computing cloud, SX-9 nodes at Osaka University employ a reservation system for job executions, and the execution time of jobs is limited. This reservation-based operation guarantees the time when a job execution starts (*job-start time*). Thus, the job-start time can easily be obtained by checking a reservation map of the system. On the other hand, SX-9 nodes at Tohoku University employ a queuing system for job execution in a FIFO manner and the system does not limit the execution time of jobs. The queuing-based operation allows running large scale jobs with no time limitation and provides the high-utilization of computing resources without reserving and allocating excess resources for small-size jobs. However, this queuing-based system cannot guarantee job-start time, because the execution time of jobs in a queue is undetermined.

From the viewpoints of users, the vector computing cloud should execute a job as soon as possible from the time that user submitted a job. However, if a scheduler in the vector computing cloud that consists of computing resources with different job operating policies could not understand job-start times of both job execution systems, the scheduler can not allocate a submitted job to a computation resource, which can execute the job earlier. Therefore, to achieve such a job scheduling on the vector computing cloud, estimating the job-start time in the queuing-based system plays important role.

III. Job Scheduling for the Vector Computing Cloud

To overcome the job scheduling problem mentioned in the previous section, this paper proposes a job scheduling mechanism which estimates the job-start time in a queuing-based system and allocates a job to an appropriate site in the vector cloud computing environment. Figure 1 shows the overview of the proposed job scheduling mechanism. The job scheduler consists of original NAREGI modules; SS and Reservation Map, and newly added SS's sub-module named *resource select module*. The *resource select module* is invoked by SS, and estimates a job-start time from the history of job-execution on vector supercomputers.

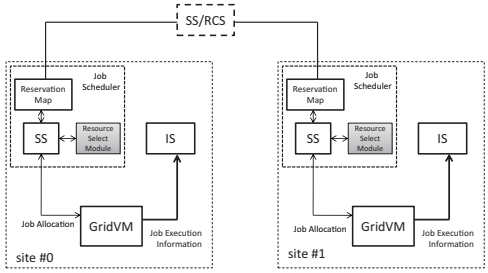

Figure 1. Overview of the Job Scheduler

A. Estimation of Job-start Time for a Queuing-based System

To estimate the job-start time for a queuing-based system, we focus on the high software reusability in HPC. For example, a parameter sweep experiment, which is one of the famous HPC jobs, executes one program with many different paramerters. Then, it is easy to estimate the execution time of the program by using the previous execution result.

We use a *job-execution time* of a job in a queue for a job scheduling in the vector computing cloud. The job-execution time indicates a required period to process the job. If we can obtain the job-execution time, we can also estimate job-start time by summing up job-execution time in the queuing-based system. By comparing the job-start time in a queuing-based system and that in a reservation-based system, the scheduler can assign the job to an appropriate site, which can execute the job early.

The scheduler records and archives job execution information to estimate the job-start time of the following jobs. When a job execution is completed, Grid VM sends the job execution information to IS. IS stores the information in a database, and provides an databese access interface to SS. Then, SS can obtain the job-execution time based on the job name, job type, and the user account information. The *resource select module* obtains the job-execution time in a queue by using the job information accumulated in the database of IS.

The process to obtain the job-execution time in a queue by the scheduler is described as follow. First, the *resource select module* accesses Grid VM, and obtains a list of queued jobs. Note that these jobs are not executed, and the job-execution time has not been decided yet. Next, the *resource select module* retrieves the execution time of all jobs in the queue from IS. In this process, the *resource select module* uses the command name as the search key, and retrieves the execution time of the corresponding job which has the same command name. If the *resource select module* cannot obtain the job-execution time from the database of IS, the *resource select module* uses the average job execution time accumulated in IS. Finally, by adding all job-execution time in the queue, the *resource select module* estimates the job-start time for all jobs in the queuing-based system.

B. Job Allocation Mechanism on The Vector Computing Cloud

The job submitted by the user is allocated to a computation resource by SS. This subsection describes the processes

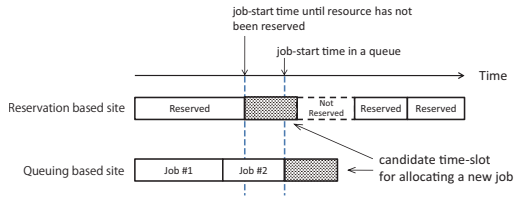

Figure 2. Job Allocation among a Reservation-based Site and a Queuing-based Site

of job allocation on the vector computing cloud.

First, SS obtains a list of computation resources, which satisfy the requirement of the job such as the number of processers, memory capacities and so on. Next, to select an appropriate resource from the list of computation resources, SS calls the *resource select module*.

Figure 2 shows a situation for selecting an appropriate one from the queuing-based site and the reservation-based site to allocate a new job. The *resource select module* obtains the job-start time of a queue in the queuing-based site and the reservation-based site. The job-start time of the reservation-based site can be obtained from the reservation map. Then, the *resource select module* compares the job-start time of the queuing-based site with that of reservation-based site, and allocates the new job to the computing resources which can execute the job earlier. In Figure 2, the *resource select module* allocates the new job to the reservation-based site because the reservation-based site becomes ready to execute the new job earlier than the queuing-based site.

IV. Performance Evaluations

A. Simulation Analysis

To evaluate the performance of the proposed job scheduling mechanism, the vector computing cloud environment composed of two queuing-based sites is modeled and simulated.

This evaluation uses six kinds of jobs, and the job-exetution times of all jobs are generated in the gamma distribution with the averages (μ) and the standard deviation (σ) shown in Table I. The six kinds of jobs are generated with zipf's law [10]. By using the zipf's law, many small-jobs and few large-jobs are generated in the simulation. The zipf's law can be applied to many natural and social phenomena, and this evalueation assumes the zipf's law as a real HPC user's workload. In the initial phase of the simulation, thirty jobs are sequentially submitted to the vector computing cloud. Then, thirty jobs always exist in the vector computing cloud while simulation is executed. To evaluate this situation, whenever one job execution has been finished, a new job is generated and submitted. The proposed job scheduling mechanism and the *round-robin* scheduling mechanism used in the NAREGI Grid Middleware are evaluated. These simulations are carried out until 1,000,000 simulation seconds, and the period from the submission time to the start time of a job(*waiting-time*), and the number of jobs which is completed are measured. The evaluated results are obtained by taking the average of twenty simulations.

Table I
Simulation Parameters

Parameter Name	Value
average execution time of each job (μ) [sec]	100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200
standard deviation of each job (σ)	11

Table II
Summary of simulation results

	Proposal	Round-Robin
number of executed jobs	3,914	3,458
average of waiting-time [sec]	7,131	8,085
maximum of waiting-time [sec]	81,464	94,113
standard deviation of waiting-time	6,477	10,368

Table II summarizes the simulation results. From this result, the proposed job scheduling mechanism improves the number of executed jobs and reduces the average and maximum waiting-time. The standard deviation of waiting-time shows that the waiting-time of jobs scheduled by the proposal concentrates on the average, but the waiting-time of jobs scheduled by round-robin is distributed over the wide region.

Figure 3 shows the histograms of the waiting-time. The horizontal axis indicates the waiting-time of jobs, and the vertical axis is the number of jobs that are within the waiting-time. Figure 3 shows that some jobs scheduled by the round-robin scheduler have been executed as soon as it is submitted. Because of the simulation condition that thirty jobs are always allocated to two sites, a case of the waiting-time being zero indicates the situation that all jobs are allocated to the one site, and no job is allocated to the another site. The round-robin scheduling makes the load-imbalance among two sites, and only a part of the entire computing power in the vector computing cloud is utilized. This load-imbalance by the roud-robin scheduling causes the low number of executed jobs in Table II. On the other hand, the proposed job scheduling mechanism can provide the fair waiting-time for all jobs and eliminate the load-imbalance among two sites. Then, the proposed job scheduling mechanism improves the utilization efficiency of the computing resources in the vector computing cloud.

B. System Test

The effectiveness of the job scheduler is also evaluated by using a prototype system of the vector computing cloud. Jobs with different job-execution time are successively submitted to the prototype system in keeping with the "submission order" in Table III. The sequence of jobs consists of three kinds of applications, and their execution times are set to 30 (small), 40 (middle) and 200 (large) seconds, respectively.

The job allocation results are confirmed by a portal site as shown in Figure 4. In this screenshot, the first row indicates allocated jobs to Tohoku University's site and the second row indicates those of Osaka University's site. The horizontal

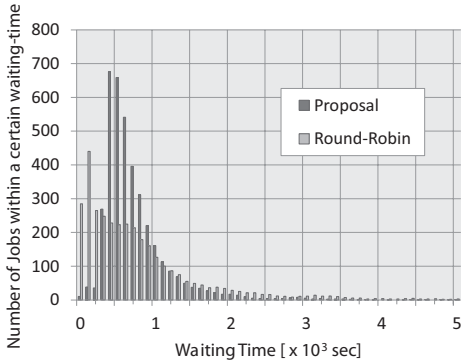

Figure 3. Histogram of Waiting-time (1,000 second intervals)

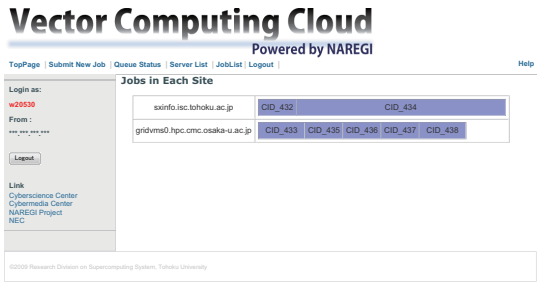

Figure 4. Portal site page : result of job scheduling

axis is the time sequence. In this result, after job **CID_434** which has the longest execution time is allocated to Tohoku University, the job-start time of Tohoku University becomes much longer than that to Osaka University. As a result, the jobs **CID_435** - **CID_438** are sequentially allocated to Osaka University to make quick response to users and reduce the imbalance of job-start time between both sites. From the results, the job scheduler can well estimate the execution time of each job, and select the appropriate site to execute the job as early as possible in the vector computing cloud that consists of computing resources with different job operating policies.

V. Conclusions

This paper has presented a job scheduling mechanism for a queuing-based system in the vector computing cloud. The proposed job scheduling mechanism obtains the job-start time in a queuing-based system from the history of the job-execution times, and automatically allocates a job to an appropriate site, which can execute the job earlier. The experiment results indicate that the proposed job scheduling mechanism has enough potential for transparently operating jobs between the two SX-9 systems with coexistence of conventional jobs and cloud jobs. As a result, the proposed job scheduling mechanism contributes to the achievement of the HPC cloud.

In our future work, we will evaluate and discuss the performance when the number of sites becomes three, four or more. Next, to execute the wide-area MPI application among the queuing-based system and the reservation-based system, we will implement a co-allocation mechanism based on the proposed job-start time estimation method. In addition, improving the estimation accuracy of the job-start time for the queuing-based system is needed to achieve more efficient job execution. For this end, we will analyze the logs of the job-execution at the supercomputer centers, and establish a more accurate estimation method.

Table III
List of Submitted Jobs

Submission Order	Job ID	Job Size
1	CID_432	small
2	CID_433	middle
3	CID_434	large
4	CID_435	middle
5	CID_436	small
6	CID_437	small
7	CID_438	small
8	CID_439	middle

VI. Acknowledgments

The author would like to thank the reviewers for their useful comments. This work is partially performed as the CSI program of National Institution of Informatics. The author would like to extend our thanks to the members of the Center of GRID Research and Development (NII), and Kenji Oizumi, Eiichi Ito of Cyberscience center Tohoku University, and Masa-aki Yamagata, Norio Kamiyama, Hironobu Konno of NEC Corp. for their help to implement the prototype system .

References

[1] The Globus Alliance. http://www.globus.org/.

[2] A. Weiss. Computing in the clouds. *netWorker*, 11(4):16–25, 2007.

[3] B. Hayes. Cloud computing. *Communications of the ACM*, 51(7):9–11, 2008.

[4] B. Sotomayor, K. Keahey, and I. Foster. Combining batch execution and leasing using virtual machines. In *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, pages 87–96, New York, NY, USA, 2008. ACM.

[5] B. Sotomayor, R. Montero, I. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13:14–22, 2009.

[6] R. Egawa, M. Higashida, Y. Murata, and H. Kobayashi. Prototyping of a vector meta-computing environment. In *International Symposium on Grid Computing 2010*, 2010.

[7] Center for Grid Research and Development. National research grid initiative pages. *http://www.naregi.org/index_e.html*.

[8] M. Higashida. *High Performance Computing on Vector Systems 2009*, chapter The Grid Middleware on SX and Its Operation for Nation-Wide Service, pages 109–119. Springer Berlin Heidelberg, 2009.

[9] Toshiyuki Kitagawa, Shoichi Hasegawa, and Akihiro Yamashita. Job scheduling function nqs ii for sx-6. *NEC technical journal*, 55(9):50–53, 2002/9.

[10] George K. Zipf. Human behavior and the principle of least effort. *Addison-Wesley*, 1949.