

Grid Middleware for Effectively Utilizing Computing Resources: CyberGRIP

● Akira Asato ● Yoshimasa Kadooka

(Manuscript received September 6, 2004)

Various research and development activities regarding Grid computing technology have recently been promoted. This technology integrates heterogeneous computing resources that are geographically dispersed and virtualizes them as a single computer system. However, there have been very few reports in the business world about Grid computing becoming successful or practical because most Grid computing research is carried out in government science projects. Fujitsu has developed CyberGRIP, which is a Grid middleware system for applying Grid computing to practical corporate simulations and verifying the effectiveness of Grid computing. This paper clarifies the needs and problems of these corporate simulations and describes how CyberGRIP is solving the problems. This paper also describes in-house verification using the CAD-Grid system.

1. Introduction

Based on the progress in computer performance and broadband network technology, various projects have recently started to research and develop Grid computing technology, which integrates geographically dispersed heterogeneous computing resources via a network and virtualizes them as a single computer system.¹⁾ However, most of these projects are national science projects and there have been few reports so far about practical and successful applications of Grid computing to business.

We have researched the practical application of Grid computing, focusing on the virtualization of computer power as the foundation of various Grid technologies, and developed a Grid middleware called Cyber Grid Innovation Platform (CyberGRIP). CyberGRIP is intended to enable advanced use of computers, mainly for simulations in enterprises.

This paper describes the need for simulation

in enterprises and related issues as the background of our research. Then, it outlines the configuration of CyberGRIP and the operation of its main components. Lastly, this paper describes the CAD-Grid system, which was developed to assist in the design of next-generation mobile communication systems, as an example application of CyberGRIP in business operations.

2. Present situation and problems with simulations

In this section, to clarify the issues to be solved by CyberGRIP, we describe the background of its development. Specifically, we describe the current needs for massive simulation in enterprises and the fact that these needs are not sufficiently satisfied.

As simulation technologies evolve, they are increasingly being used not only by universities and other research institutes but also by enterprises to support their business activities. For

example, in the processor development division of an enterprise, the engineers must execute massive simulations to check logic, optimize cell placement and wiring, and calculate delays and power consumptions. The need for efficient, high-speed simulation is steadily growing because the computations required for device development are becoming highly complex as devices advance. On the other hand, the market is demanding reductions in development terms and costs.

These circumstances in the manufacturing industry are also found in the finance industry, which requires massive simulations to speed up transaction settlements in order to improve customer service, appraise the current prices of various assets and futures products, and manage global risks.

Another example of an industry that requires simulations is the distribution industry, which needs a great deal of computation to plan marketing strategies by data mining. Simulations, therefore, are needed in many fields, and their use will obviously expand in the future.

As described above, many types of industries are performing simulations. However, an investigation we conducted showed that there are various inherent problems in the execution of simulations.

For example, in the manufacturing industry, computers are often used inefficiently in simulations because the availability of computers and the progress of entered simulation jobs are managed by hand. A problem arises when incorrect input data is entered for large simulations and the error is not found until the entire computation is completed. Even if the incorrect data is found in the middle of a computation, a problem occurs if the jobs related to the incorrect data cannot be easily identified. In such a case, data must be re-input to all parameters and the simulation must be run again. If errors still remain in subsequent processes (e.g., because there is insufficient time to remove them), it may be necessary to redesign a product (e.g., an LSI) or another large

loss may be incurred. Therefore, a major issue is how to efficiently perform massive simulations.

In the finance industry, it is already known that the latest financial engineering can increase the accuracy of simulations for risk management by one order of magnitude. However, it is sometimes difficult to obtain the required accuracy because the necessary computing resources cannot be allocated, which can result in missed business opportunities.

On the other hand, it has turned out that the computing resources in enterprises are not always used efficiently. One reason for this is that each division of an enterprise purchases and manages its own servers and, even when the machines in a division are idle, other divisions cannot easily use them. Another reason is that business-use personal computers (PCs), which have been upgraded year by year, are mainly used for making and reading documents and other purposes that do not require much CPU performance. Also, they are almost always idle at night.

In summary, the key issue to be solved is how to perform massive simulations efficiently when the computing resources in enterprises are not fully used. If the free computing resources can be used effectively, it might be possible to greatly improve the efficiency of simulations and solve most of the current problems in simulations. We started the development of CyberGRIP based on this idea.

3. Development of CyberGRIP

In this section, we describe the configuration of CyberGRIP and the behavior of its main components.

As described above, efficient execution of a massive simulation requires 1) mechanisms for efficiently submitting many jobs without needing to monitor the status of computing resources or the jobs being executed and 2) mechanisms for using as many in-house computing resources as possible to the fullest extent. To meet these requirements, we developed the CyberGRIP Grid

middleware. Next, we describe CyberGRIP.

3.1 Overall architecture

Figure 1 shows the overall architecture of CyberGRIP.

CyberGRIP operates on a central UNIX or Linux server and consists of three components: Organic Job Controller (OJC),²⁾ Grid Resource Manager (GRM), and Site Resource Manager (SRM). OJC controls how the jobs submitted by the user are executed. GRM determines the optimum computing resources for the jobs transferred from OJC. SRM monitors the status of the computing resources and manages the communication between them. There is one SRM for each computing resource, and the SRM for a Windows PC is called the “Grid Mediator for Windows (GMW) manager.”

Each computing resource must have middleware to communicate with its SRM and execute jobs. When the computing resource is a Solaris or Linux machine, a general batch system, for

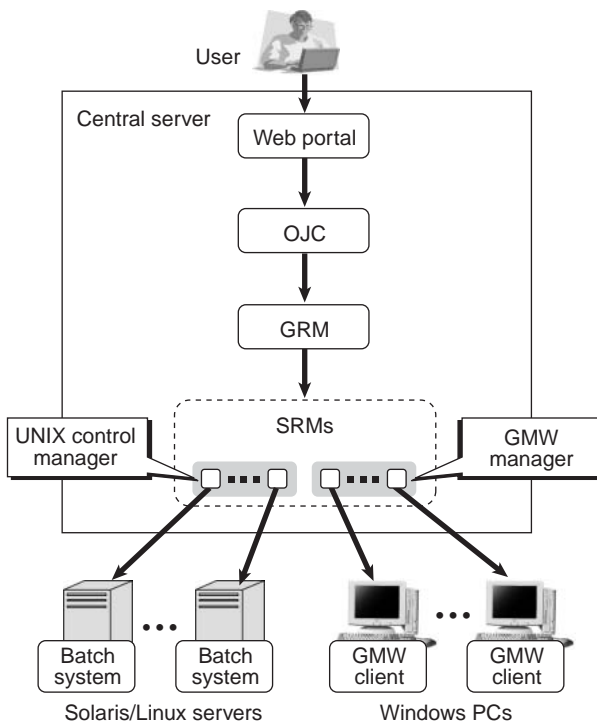


Figure 1
Architecture of CyberGRIP.

example, Condor,³⁾ can be used as the middleware. When the computing resource is a Windows machine, the GMW client corresponding to the GMW manager must be installed in the machine.

CyberGRIP can realize an environment in which the user can submit jobs to virtualized computing resources consisting of not only Solaris and Linux machines but also Windows machines for office use via the Web portal of a central server. The user can do this without being aware of the performance and other characteristics of the individual computers.

3.2 Job execution control by OJC

When many jobs are executed at a time, very often there are dependencies between jobs (e.g., the results of one job are reflected in the parameters of a subsequent job.) In these cases, the total efficiency of job execution can be increased by scheduling job execution according to the dependencies. The effect of such scheduling is expected to be especially large in a Grid computing environment, in which individual jobs often terminate asynchronously on independent resources. As described in the previous section, input parameter errors inevitably occur in a massive simulation. However, even in a massive simulation, if the inter-job dependencies are known, only the jobs that depend on jobs with parameter errors need to be re-executed. Therefore, the number of jobs to be re-executed can be minimized.

Based on the above idea, we developed OJC as a mechanism to properly and flexibly handle the inter-job dependencies. **Figure 2** shows the basic structure of OJC. The user of OJC only needs to write an OJC script and input it to the OJC interpreter. (The OJC script represents inter-job dependencies simply by using the syntax shown in Figure 2.) The OJC interpreter interprets the OJC script and defines the tree structure representing the dependencies. During job execution, the jobs are synchronized and queued according to the tree structure in OJC and executable jobs are transferred to the job execu-

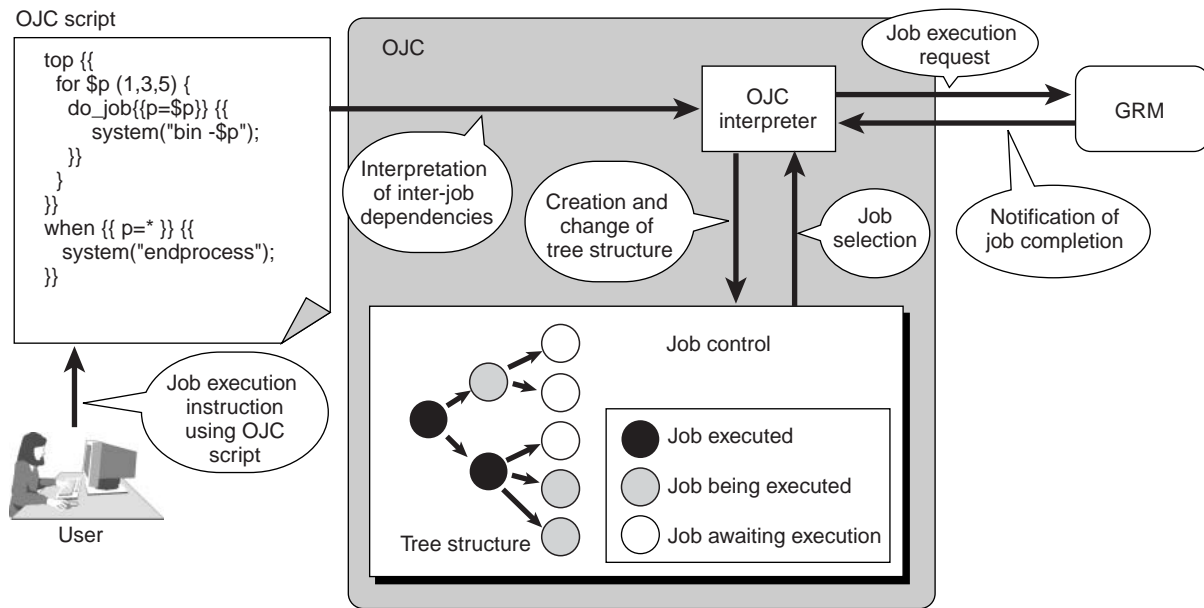


Figure 2 Basic structure of Organic Job Controller (OJC).

tion component (GRM). This mechanism enables optimum control of job execution even in a Grid system, in which jobs terminate asynchronously.

OJC also has an interface that can be used to re-execute jobs when an input parameter error occurs. When the user requests, via the interface, re-execution of a job that contains a parameter error, every job that depends on that job is automatically re-executed according to the tree structure. Therefore, the user need not specify re-execution of individual jobs. It is obvious that these functions of OJC can solve or alleviate most of the problems described in the previous section that occur in massive simulations.

OJC also has a dynamic job control function (**Figure 3**). This function works when the number of jobs to be executed cannot be decided statically at initial job entry. This occurs, for example, when an input file of undefined size must be processed in sections or parameters must be optimized. In these cases, the number of jobs to be executed depends on the file size or the optimization status.

Conventionally, because it was difficult to automatically perform dynamic job control,

operators usually took one of two approaches: 1) execute all the jobs that have been submitted without considering which ones need to be executed or 2) individually decide whether to execute each job based on the execution results of the previously executed job. Both these approaches are very inefficient. By automating these decisions, therefore, OJC significantly increases the efficiency of job execution.

3.3 Resource and job management by GRM, SRM, and GMW

GRM selects the computing resource that is most suitable for executing the job requested from OJC and submits the job. The information transferred from OJC to GRM includes job attributes as well as the relevant program name and input parameters. The job attributes include whether the platform (e.g., Pentium or SPARC) and operating system (e.g., Windows or Linux) are specified, the estimated execution time, and the resource selection policies. GRM continuously monitors the operation status of the computing resources under GRM's control via SRM and compares the monitoring results with transferred

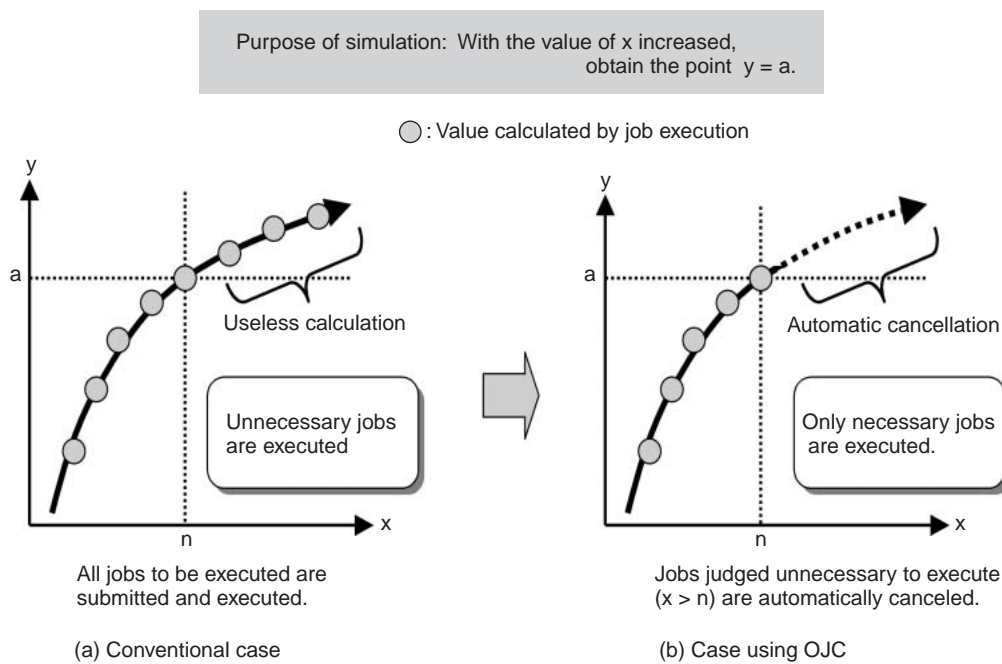


Figure 3
Dynamic job control of OJC.

job attributes to allocate the optimum resources for each job. The resource policies supported by the current version of GRM are only simple ones, including the selection of the fastest machine; however, the selection policies will be extended in the future.

The binary program data of a job is delivered from the central server after a computing resource has been allocated to execute the job. Therefore, if the same program is executed on computer resources that use different platforms or operating systems, multiple types of binary data must be prepared beforehand on the central server.

SRM is the middleware that manages job execution by the computing resources, and there is one SRM for each computing resource. The following explanation is for when the computing resource is a Windows PC (hereinafter, simply called the PC). **Figure 4** shows the sequence of job execution on the PC. The SRM associated with the PC consists of the GMW manager running on the server and the GMW client running on the

PC (Figure 1). When GRM allocates the PC to execute a job, GRM immediately notifies the GMW manager about the allocation. Then, the GMW manager transfers the relevant program and input file to the GMW client and instructs it to start the program. Upon reception of the instruction, the GMW client immediately starts executing the job as a background job. When the job is completed, the GMW client transfers the job results as an output file to the GMW manager. While the job is being executed, the GMW manager periodically sends the GMW client a message to check the progress of job execution. If the GMW manager does not receive a response to one of these messages, the GMW manager assumes that job execution has been stopped because of an error in the PC. In these cases, the GMW manager selects and allocates another PC to the job and attempts to re-execute it.

When a job is submitted on a PC under control of CyberGRIP, the job is given the lowest execution priority to avoid adverse influences on other operations being made by the PC owner.

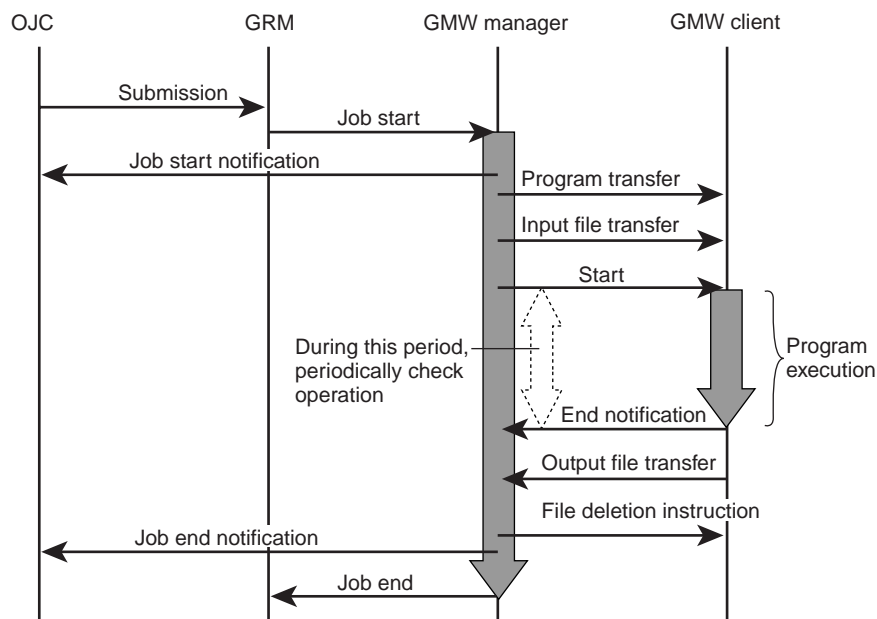


Figure 4
Job execution sequence (Windows PC).

Figure 5 shows the result of performance monitoring on a PC being used as a computing resource. The first peak indicates the momentary increase in CPU load caused when the program input from the GMW manager was started on the PC. Next, in Period A, the initiated job used almost 100 percent of the CPU capacity when the user was not using the PC for other jobs. Then, in Period B, the user started using a spreadsheet program (Excel) and the CPU capacity required for the operation was given over.

4. Application to CAD-Grid

In this section, we briefly introduce an example application of CyberGRIP for massive in-house simulation.

As mentioned at the beginning of this paper, we thought it was important to show some practical and successful applications of Grid computing to business. Therefore, as a first step we applied CyberGRIP to some in-house work and verified its effects. CAD-Grid was developed for this purpose.

CAD-Grid was designed to accommodate the PC clusters and UNIX servers deployed in the

company and the desktop PCs connected to in-house networks for office work as a virtualized single-computer system.

CAD-Grid was applied to simulations for designing a mobile communication system in the division responsible for developing base stations and mobile terminals and produced significant improvements over conventional methods. For example, the simulation period was reduced to about one-fourth and the person-hours for the simulations were reduced to about one-third. These successful results indicate that the approach with CyberGRIP will be useful for resolving the problems in massive simulations. Also, the aim of showing an example of a practical and successful application of Grid computing was achieved with better-than-expected results. The application of CAD-Grid is described in another paper in this special issue.⁴⁾

5. Conclusion

The initial phase of CyberGRIP provided better-than-expected results. On the other hand, the verification using CAD-Grid revealed a few points that need to be improved. For example,

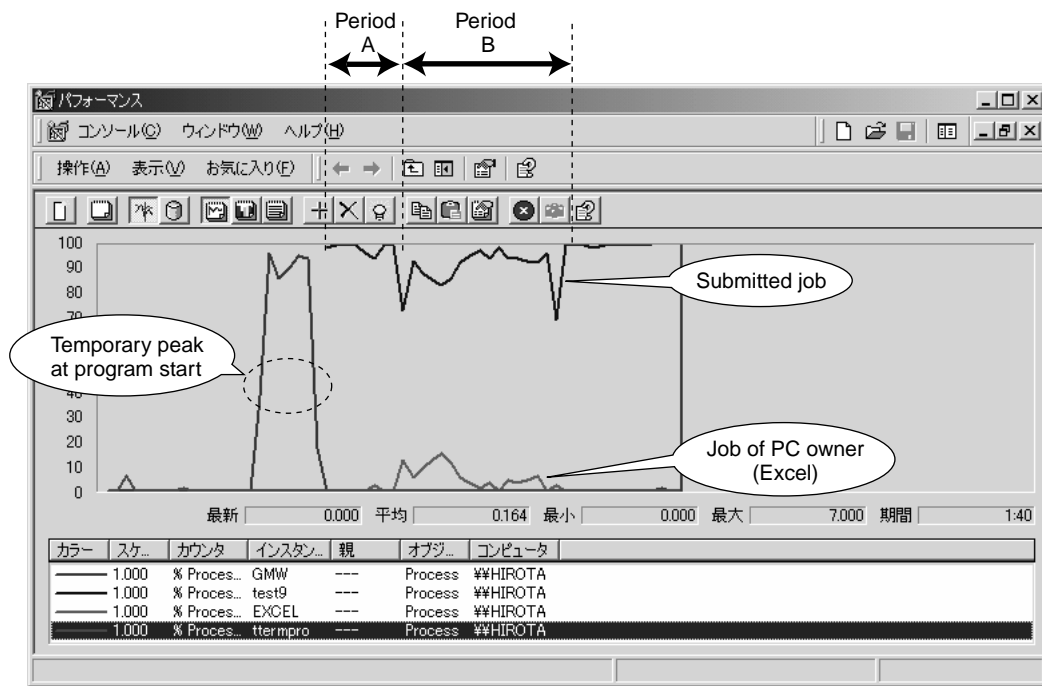


Figure 5
Results of performance monitoring.

the mechanism for allocating computing resources is too simple to achieve preferential execution of urgent jobs. Also, the overhead cannot be ignored when the number of jobs to be handled increases because sometimes not all the processes are optimized. In addition to these problems in function and performance, security enhancement remains an important issue to be solved.

Further efforts will be made to solve these problems and thereby upgrade the quality of CyberGRIP. In-house application of CyberGRIP will be expanded from the simulation of the mobile communication system to many other projects to further increase the efficiency of in-house work. Moreover, business applications of CyberGRIP will be promoted in cooperation with related divisions.

Acknowledgement

This research has been partially funded by the New Energy and Industrial Technology Development Organization (NEDO).

References

- 1) I. Foster and C. Kesselman: The GRID: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, SC-18, 1999.
- 2) I. Foster and C. Kesselman: The GRID2: Morgan Kaufmann, 2nd Edition, 2003.
- 3) Condor: <http://www.cs.wisc.edu/condor/>
- 4) T. Yamashita, T. Nakamura, and H. Noguchi: CAD-Grid System for Accelerating Product Development. *FUJITSU Sci. Tech. J.*, **40**, 2, p.224-231 (2004).



Akira Asato received the B.S. degree in Information Science from the University of Tokyo, Tokyo, Japan in 1983. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1983, where he has been engaged in research and development of computer architectures. He is a member of the Information Processing Society of Japan (IPSJ), a secretary of the SIG-Architecture Committee of the IPSJ since 2000, and a member

of the Editorial Board of IPSJ Transactions on Advanced Computing Systems (ACS) since 2001.

E-mail: asato@jp.fujitsu.com



Yoshimasa Kadooka received the M.Sc. degree in Mathematics from Kyusyu University, Fukuoka, Japan in 1982 and the Ph.D. degree in Science from Kanazawa University, Ishikawa, Japan in 2004. He joined Fujitsu Ltd., Kawasaki, Japan in 1982, where he was engaged in development of communication systems and multimedia systems. He moved to Fujitsu Laboratories Ltd., Kawasaki, Japan in 2002,

where he has been engaged in research of Grid computing technology. He is also a Professor at the Visiting Faculty of Kansai University, Osaka, Japan. He is a member of the Japan Society for Computational Engineering and Science and the Japanese Society of Computational Statistics. His research interests are Grid computing, problem solving environments, and fluid dynamics.

E-mail: kadooka@labs.fujitsu.com