

Heuristic Scheduling and Process Migration on the Grid

Yusuke Inoue[†], Takahiro Koita^{††}, Akira Fukuda[†] and Kenya Sato^{††}

[†] : Kyushu University, Kasugakoen 6-1, Kasuga, Fukuoka, Japan

{inoue, fukuda}@f.csce.kyushu-u.ac.jp, Tel:+81-92-583-7625, Fax:+81-92-583-7625

^{††} : Doshisha University, Tataramiyakodani 1-3, Kyotanabe, Kyoto, Japan

{tkoita, ksato}@mail.doshisha.ac.jp, Tel:+81-774-65-6261, Fax:+81-774-65-6261

Abstract—Reducing the response time of genomic application programs is important for correctly analyzing genomic information in practical periods. The Grid can provide high processing power to reduce the response time of such application programs. To reduce response time and efficiently use computational resources on the Grid, scheduling and process migration are important. InterProScan is a quite useful genomic application program widely used by genomic researchers. However, it is designed and implemented for one computer or a small PC cluster system, and its scheduling scheme is quite simple, allocating one scanning tool to a fixed computer configured at installation time. Furthermore, InterProScan does not have any process migration scheme, and process migration has to be considered to provide flexible execution on the Grid.

This paper presents heuristic scheduling and process migration schemes for InterProScan on the Grid. Several scheduling schemes are implemented and evaluated on OBIGrid, an experimental Grid for bioinformatics being developed by Japan Committees. This paper also discusses the problems of applying process migration to the scheduling scheme we implemented.

KEY WORDS

Heuristic Scheduling, Process Migration, InterProScan, Genomic Application, Grid Environments

I. INTRODUCTION

InterProScan is one genomic application program developed by EBI [1] that consists of multiple databases and scanning tools. Since InterProScan executes multiple scanning tools in databases and provides more information than single scanning tools with a single database, it is a quite useful program widely used by genomic researchers. However,

executing multiple scanning tools requires too much time. For example, when only ten protein sequences are used as input data, it takes more than 30 minutes on one new computer. New database construction by InterProScan takes more than several weeks to complete. Furthermore, since genomic researchers simultaneously execute many InterProScans, higher processing power than a PC cluster is needed. To make InterProScan more useful, mean response time must be reduced.

The Grid [2] has become a promising development in high-performance computing fields over the past few years. The Grid can provide higher processing power than a PC cluster and can reduce the mean response times of genomic applications. Many universities, research institutes, and commercial companies have developed related technologies. Among them, scheduling and process migration are important to utilize computer resources on the Grid. A scheduling scheme decides the initial allocation of processes before their execution, and a process migration scheme decides the reallocation of processes at execution time. Both schemes allow the Grid to provide effective use of computer resources.

On the other hand, InterProScan's scheduling scheme is designed and implemented for one computer or a small PC cluster system, and its scheduling scheme is quite simple, allocating one scanning tool to a fixed computer configured at installation time. The scheduling scheme that decides which scanning tools should be allocated to which computer is important for efficient InterProScan execution. To execute InterProScan on the Grid and

reduce mean response time, the scheduling scheme must be designed to consider the Grid and InterProScan characteristics.

In addition to the scheduling scheme, the Grid involves large system dynamics where the ability to migrate executing processes onto different sets of resources assumes great importance. The main motivations for migrating processes on the Grid are to provide fault tolerance and to adapt to load changes. Migrating processes for adaptation to load changes are needed to reduce the mean response time of InterProScan. Specifically, when several different scheduling schemes are used, their allocation might conflict with one other due to scheduling policies. If several different scheduling schemes are used, a scheme scheduling might try to allocate a process to a computer that has already been allocated a process by another scheduling scheme. Such allocation, which causes unnecessary overload on computer resources, can be avoided by migrating processes. Thus, on the Grid using several scheduling schemes, a process migration scheme that can coordinate those scheduling schemes has to be considered.

However, no studies have focused on the scheduling and process migration schemes of InterProScan on the Grid; no scheduling schemes have been implemented and evaluated. In this paper, heuristic scheduling schemes for InterProScan are implemented and evaluated on Open Bioinformatics Grid (OBIGrid) [3], an experimental Grid for bioinformatics being developed by Japan Committees. We also discuss the problems of applying process migration schemes to our implemented scheduling scheme.

II. INTERPROSCAN

InterProScan is a genomic application program that combines different protein signature recognition methods native to the member databases into one resource that includes the look up of corresponding InterPro and GO annotation. InterProScan uses protein sequences as input data and executes multiple scanning tools with multiple databases against inputs. Those scanning tools are executed indepen-

dently. Table I shows the databases and scanning tools used by InterProScan.

TABLE I
INTERPROSCAN PROCESSES

process	Database	Scanning Tool
blprodom	PRODOM	BlastProDom
coil	Coiled-Coil	ncoils
fps	PRINTS	FingerPRINTSscan
hmmpfam	PFAM	HMMPfam
hmmpir	PIRSF	HMMPfam
hmmsmart	SMART	HMMPfam
prfs	PROSITE profile	Pfscan
scpr	PROSITE	ScanRegExp
seg	Seg	Seg
tigrfam	TIGFAMS	HMMPfam

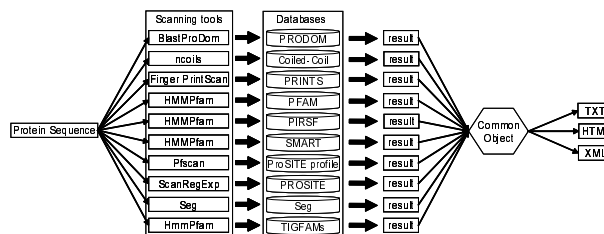


Fig. 1. InterProScan Overview

Figure 1 shows an overview of InterProScan. Scanning tools are executed independently with databases. The script program integrates the results obtained by those scanning tools, and the integrated results can be used as HTML/XML/TEXT formats. The input sequences can also be divided and independently processed. Thus, InterProScan consists of multiple independent processes that can be easily divided for distributed processing; InterProScan is a suitable application program for such distributed environments as the Grid.

Even though InterProScan already has a simple scheduling scheme for distributed execution, its scheduling scheme allocates one fixed process to one fixed computer decided at installation time. Such fixed allocation causes an imbalance of system load and an inefficient use of computers. Furthermore, the scheduling scheme ignores application programs and system information. If there are many computers with high processing power, they are

never used because the scheduling scheme allocates a maximum of ten computers (see Table I). Even if other processes are running on one fixed computer whose CPU is completely used, the fixed process is reallocated to this computer.

Currently, scheduling scheme implementation can only execute InterProScan on small, distributed environments that cannot efficiently use the Grid environment. Its simple scheduling scheme can use a maximum of ten computers because one process is allocated to one fixed computer. To execute InterProScan on the Grid, a more flexible scheduling scheme is needed that allocates those processes using application and Grid information.

III. EXECUTION TIME OF EACH PROCESS

In this section, we examine the execution time of InterProScan on one computer to obtain the basic characteristics of each process in one InterProScan.

InterProScan executes multiple processes whose execution times differ. To reduce the total execution time, it is important to know how each process occupies the total execution time and which process is the heaviest. By using such process information for scheduling schemes, we can reduce execution time by allocating such heavy processes to a computer with more processing power.

We examined the execution time of each InterProScan process on one computer that has one CPU (Celeron 1.3 GHz), 1024 MB of memory, and 50 input sequences.

TABLE II
PROCESSES' EXECUTION TIME

process	execution time [sec.]	rate [%]
blprodom	3.18	0.45
coil	0.03	0.01
fps	10.05	1.45
hmmpfam	423.02	61.07
hmmpir	18.73	2.70
hmmsmart	23.11	3.34
prfs	21.48	3.10
scpr	7.94	1.15
seg	0.02	0.00
tigrfam	185.17	26.73

Table II shows the execution time of each scanning tool and its total execution time rate. These results show that the execution time of hmmpfam occupies more than 60% of the total execution time. The execution time of tigrfam is also high, more than 20%. Since those two heavy processes occupy nearly 90% of the total execution time, reducing their execution time would significantly reduce the total execution time.

IV. IMPLEMENTATION OF SCHEDULING SCHEMES

A Grid environment can provide high processing power to reduce the mean response time of InterProScan. However, a flexible scheduling scheme, which uses application and Grid information, is needed to efficiently execute InterProScan on the Grid. In this section, we describe four types of scheduling schemes we implemented to compare scheduling schemes for Grid environments: Fixed, Random, Distributed, and Sequential.

A. Fixed

A Fixed scheme, which uses the default scheduling scheme of InterProScan, is a fixed process allocated to one fixed computer decided at installation time. Thus, a maximum of ten computers can be used because InterProScan consists of only ten processes. Even if there are many idle computers on the Grid environment, only predecided computers can be used. For a Fixed scheme, if InterProScan is executed infrequently and the number of computers on the Grid is quite small, mean response time will not be so large. However, the Fixed scheme is not suitable for the Grid and the multiple execution of InterProScan.

B. Random

The Random scheme randomly allocates one process to one computer. In this case, each process can be widely executed on the Grid, and the number of computers used for allocation is larger than in the Fixed scheme. However, since the Random scheme does not use any information, it is the simplest scheduling scheme to compare.

C. Distributed

The Distributed scheme uses two types of computer information on the Grid environment for allocation. When a new process arrives at the Grid environment, its information is used for its allocation. Static and dynamic information of the computers are used. The former is the execution time of InterProScan on one computer obtained previously. Execution time is obtained by executing one InterProScan on one computer with no other running process, which indicates execution time when the processor can be fully used for only one InterProScan. Dynamic information is the processor utilization rate that indicates how much processing power can be used for a new process. By using this information, the scheduling scheme estimates the execution time of the new arrival process. Estimated time for new processes can be represented by $T_{exec}/(1 - Load_{cpu})$. T_{exec} and $Load_{cpu}$ denote the execution time of InterProScan on one computer and the current processor utilization rate, respectively. For example, if T_{exec} is 1000 (seconds) and $Load_{cpu}$ is 0.2, the estimated execution time is $5000=1000/(1-0.2)$ (seconds). In this case, if all new processes of one InterProScan arrive at a computer, the execution time on this computer is the estimated time. By using this estimated time, the Load scheme allocates a computer having a minimum estimated time to the new process. For the Distributed scheme, one InterProScan is divided into multiple processes against one input sequence. One InterProScan is distributively executed on several computers.

D. Sequential

In contrast to the Distributed scheme, in the Sequential scheme one InterProScan is allocated to one computer, and one InterProScan is sequentially executed on one computer. However, input sequences are divided, and one divided sequence is allocated to one computer as one allocation unit.

The Sequential scheme also uses static and dynamic information as well as the Distributed scheme. This scheme allocates a computer having a minimum estimated time to the new process. Only

the dividing process differs from the Distributed scheme.

In the Sequential scheme, if there are many idle computers and the number of input sequences is less than the number of computers, execution time is not reduced more than the Distributed scheme because the Sequential scheme can only use computers having the same number of input sequences. However, if there are more input sequences than computers, this scheme can reduce allocation overhead because it does not divide InterProScan.

V. EVALUATION

We used ten computers at two sites on OBIGrid to evaluate the scheduling schemes described in the previous section. Since OBIGrid is not dedicated to this evaluation and is sometimes used for other purposes, it can be described as an actual Grid environment.

We compared the mean response time of the scheduling schemes. Load level, calculated from arriving intervals, is an index value used to estimate system load. If a process creation interval is used as an X axis, identifying whether system load is low or high is difficult. A load level of 100% means that the system is congested with no marginal processing power to process the InterProScan. More details can be seen in [4], [5]. For evaluation, we assume that multiple InterProScans arrive randomly at the Grid. We use script that randomly generates the InterProScan process, and the generated InterProScan arrives at a scheduler. The scheduler allocates the processes to a computer decided by the scheduling scheme. The arriving interval time is given by Poisson distribution.

Figure 2 shows the mean response time of each scheduling scheme versus load level. When load level is low, the mean response times of those schemes are relatively small except in the Sequential scheme. As load level increases, response times also increase. In contrast to low level, the mean response time of the Sequential scheme is lowest at high load levels.

For the Fixed scheme, mean response time greatly increases at low load levels because it only allo-

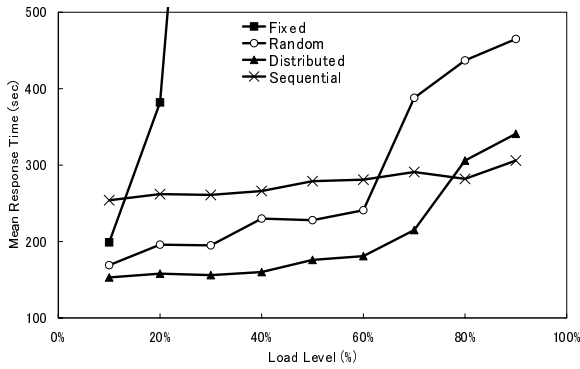


Fig. 2. Scheduling Scheme's Mean Response Time

cates one process to one computer. Even if other computers are not used, allocation never changes. Thus, since the Fixed scheme does not efficiently use computers, mean response time is high even when load level is low.

The mean response time of the Random scheme is shorter than the Fixed Scheme because the Random scheme uses more computers than the Fixed scheme. However, allocation is decided randomly, which causes allocation mismatches. If one computer is already being used, this scheme might reallocate the process to the computer, increasing its mean response time.

For the Distributed scheme, mean response time is lower than the other schemes except when load level is lower than 80%. With Adaptive schemes, process allocation is decided by application and system information. Thus, mismatch allocation does not occur, and processes are executed efficiently with adequate computers. When load level is high, mean response time increases, caused by the allocation overhead of the Distributed scheme.

When load level is higher than 80%, the mean response time of the Sequential scheme is the lowest because it has the lowest allocation overhead, which can effectively execute InterProScan when load level is high. At high load levels, the number of sequences is larger than the number of computers. Thus, most computers are used with lower allocation overheads than the Distributed scheme. In low load levels, the number of sequences is smaller than

the number of computers because there are many idle computers in the OBIGrid, and so the mean response time cannot be reduced.

VI. ADAPTIVE SCHEDULING SCHEME

The above results show that the Distributed scheme reduces the mean response time at low load levels and that the Sequential scheme achieves the lowest mean response time at high load levels. Based on the results, we try to improve the scheduling scheme by focusing on load level and implementing an Adaptive scheduling scheme that changes the scheduling scheme according to the load.

However, since it was difficult to obtain accurate load levels of the Grid, we employ an index value to identify load level. We use the number of idle computers, defined as a computer whose CPU load was lower than 50%. This means that idle computers were not used for computation because they indicate current load level. When load level is low, there are many idle computers on the Grid. When load level is high, there are fewer idle computers. We also employ a threshold to identify load level and select a scheduling scheme. If the number of idle computers is less than or equal to the threshold, the Distributed scheme is used as the scheduling scheme. If the number of idle computers is more than the threshold, the Sequential scheme is used as the scheduling scheme. By adaptively selecting a scheduling scheme according to the number of idle computers, an adequate scheduling scheme can be selected whose mean response time is the shortest for all load levels. We call this scheduling scheme the Adaptive scheme.

We compare the Adaptive scheme and the Distributed and Sequential schemes. In this evaluation, 2 is the threshold to identify load level. Figure 3 shows the mean response time of the Adaptive Scheme versus load levels.

When load level is low, the mean response time of the Adaptive scheme is close to the Distributed scheme. When load level is higher than 80%, the mean response time of the Adaptive scheme is

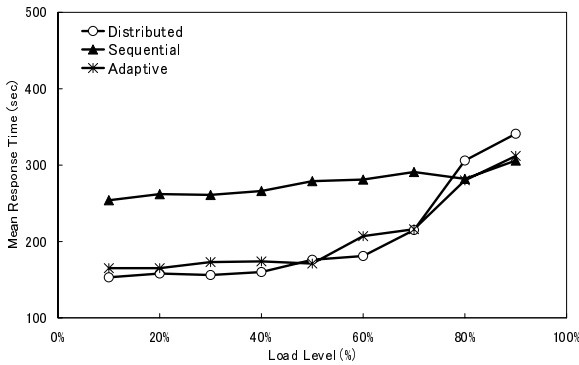


Fig. 3. Adaptive Scheme's Mean Response Time

also close to the Sequential scheme. By using the number of idle computers as an index value of load level, we can select adequate scheduling schemes for all load levels. In this study, we manually decided the threshold, an important value for selecting scheduling schemes. Further evaluation and automatic threshold adjustment are needed to apply such scheduling to general Grid environments.

VII. DISCUSSION OF PROCESS MIGRATION

In this section, we discuss the problems of applying process migration to InterProScan.

Process migration is the act of transferring a process between two machines. It enables dynamic load distribution, fault resilience, eased system administration, and data access locality. Problems exist when using existing migration schemes ([6], [7], [8], [9], [10]) for improving the response times of executing InterProScan.

First, due to the use of predefined conditions for suspension and migration and due to the lack of knowledge of the remaining execution time of the applications, applications can be suspended and migrated even when they are about to quickly finish execution. This is certainly less desirable in a performance oriented Grid where large load dynamics will lead to frequent satisfaction of predefined conditions and hence will lead to frequent invocation of suspension and migration decisions. For InterProScan, remaining execution time can be estimated if the size of input sequences is known

before its execution, as described in III. Thus, the remaining execution time can be used to avoid unnecessary migration and suspension even when the process is about to finish execution or when migration cost is high.

Secondly, due to the use of different scheduling schemes simultaneously on one Grid, those scheduling schemes allocate processes to computers to which processes have already been allocated. Some sophisticated scheduling schemes can avoid such allocation; however, some simple scheduling schemes ignore allocation by other scheduling schemes like the default scheduling scheme of InterProScan (identical to the Fixed scheme). Especially when the execution time of allocated processes is small and the processes are executed frequently, such allocation generates frequent load changes and also leads to frequent invocation of suspension and migration. For example, if some users execute InterProScan using the default scheduling scheme and most processes have small execution time, load on the allocated computers changes frequently. The existing migration schemes attempt migrating processes without disturbing the execution of other processes. However, when the predefined condition for suspension and migration is frequently satisfied due to frequent load change, migration is ineffective to reduce response time and might degrade execution time. Such frequent suspension and migration should be alleviated by monitoring the execution time of processes and predicting the execution time described in [11].

Thus, the process migration scheme for InterProScan should consider both load and application characteristics. The process migration scheme should be implemented so that executing applications are suspended and migrated only when better systems are found for application execution, thereby invoking migration decisions as infrequently as possible.

VIII. CONCLUSION

In this paper, we implemented several scheduling schemes and evaluated their mean response time on

OBIGrid. Our results showed that the Distributed scheme is effective at low load levels and the Sequential scheme is effective at high load levels. We also improved those scheduling schemes by employing the idle computers to identify load level and implementing the Adaptive scheme. By using the idle computers, the Adaptive scheme can identify load level and select an adequate scheduling scheme according to load level. We also discussed the problems of applying process migration schemes to the scheduling scheme we implemented. We are planning to further evaluate process migration schemes based on this discussion.

REFERENCES

- [1] EBI, <http://www.ebi.ac.uk/>.
- [2] I. Foster and C. K. eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, ISBN 1-55860-475-8, 1999.
- [3] OBIGrid, <http://www.obigrid.org/>.
- [4] Yusuke Inoue, Takahiro Koita, and Akira Fukuda, Performance Evaluation of Grid Scheduling for InterProScan, Proc. of the IASTED Int'l Conf. on Communications, Internet, and Information Technology (CIIT2003), pp. 699-703, 2003.
- [5] Takahiro Koita, Yusuke Inoue, and Akira Fukuda, Implementation and Evaluation of Resource Allocation for a Genomic Application Program on the Grid, Proc. of the IASTED Int'l Conf. on Parallel and Distributed Computing and Networks (PDCN2003), pp. 524-528, 2003.
- [6] Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam, The impact of migration on parallel job scheduling for distributed systems, Lecture Notes in Computer Science 1900, Volume 6th International Euro-Par Conference, pp. 242-251, 2000.
- [7] J. Casas, D. Clark, P. Galbiati, R. Konuru, S. Otto, R. Prouty, and J. Walpole, *MIST: PVM with Transparent Migration and Checkpointing*, 1995.
- [8] J. Gehring and A. Reinefeld, MARS - A Framework for Minimizing the Job Execution Time in a Metacomputing Environment, *Future Generation Computer Systems*, Vol.12, No.1, pp.87-99, 1996.
- [9] Kasidit Chanchio and Xian-He Sun, SNOW: Software Systems for Process Migration in High-Performance, Heterogeneous Distributed Environments, *ICPP Workshops*, pp. 589-596, 2002.
- [10] Tom Boyd and Partha Dasgupta, Process Migration: A Generalized Approach Using a Virtualizing Operating System, Proc. of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02), pp. 385-392, 2002.
- [11] Lingyun Yang, Jennifer M. Schopf, and Ian Foster, Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments, Proc. of the ACM/IEEE SC 2003 Conference (SC'03), pp. 31-47, 2003.