Dynamic Topology Selection for High Performance MPI in the Grid environments

Kyung-Lang Park[1], Hwang-Jik Lee[1], Kwang-Won Koh[1], Oh-Young Kwon[2], Sung-Yong Park[3], Hyung-Woo Park[4], and Shin-Dug Kim[1]

[1] Dept. of Computer Science, Yonsei University
134 Shinchon-Dong, Seodaemun-Gu, Seoul 120-749, Korea
{lanx, bear22, sugare, sdkim}@parallel.yonsei.ac.kr
[2] Dept. of Computer Engineering, Korea University of Technology and Education
P.O. BOX 55, Chonan, 330-600, Korea
oykwon@kut.ac.kr
[3] Dept. of Computer Science, Sogang University
1 Shinsoo-Dong, Mapo-Ku, Seoul 121-742, Korea
parksy@ccs.sogang.ac.kr
[4] Korea Institute of Science and Technology Information,
P.O. BOX 122, Yusong, Taejun, 305-806, Korea
hwpark@hpcnet.ne.kr

Abstract. MPI (Message Passing Interface) is getting more popular and important even in the Grid, but its performance still remains a problem, which is caused by the communication bottleneck on wide area links. To overcome such performance wall problem, we propose a dynamic topology selection which is a kind of resource selection method. It provides an effective resource selection service based on four principles which are derived from wide area message passing paradigm. Consequently, the proposed method can improve overall application performance by reducing the communication load. To demonstrate the proposed method, we executed parallel benchmark programs and measured each execution time in five geometrically distributed clusters by changing resource selection method. When using topology selection method, experimental results show that performance gain can be achieved by up to 200%.

1 Introduction

The Grid [1] is an emerging technology considered as a next generation computing infra structure. In the Grid environment, users can use practically unlimited resources as a single entity. However, there is a significant difference between the ideal Grid and current implementations. Especially, it is difficult to find useful Grid applications. In this situation, MPI (Message Passing Interface) [2] is the most significant technology to make Grid applications. It is a parallel programming library which was widely used for supercomputers and massive parallel processors. As the computational Grid emerges, MPI starts to be used for the Grid environments.

The Grid-enabled MPI library represented as MPICH-G2 [3] has a lot of new features derived from characteristics of the Grid. For example, it supports the multi-requests on distributed machine environments and provides security. But, the purpose of the MPI is high-performance computing, so that the most important requirement must be performance rather than other functionalities. Nevertheless, conventional Grid-enabled MPI Implementations do not show any reasonable performance advantage for existing commercial network environments. Therefore, many researchers have attempted to improve performance of MPI applications.

Performance gain can be achieved by applying an efficient communication algorithm or by enhancing the physical network features. As a different point of view, another good approach is to select an optimized set of resources before running an MPI application. In Grid environment, resource selection can affect performance significantly because the Grid implicitly includes the bottleneck caused by wide area communication and the bottleneck can be reduced by the efficient resource selection.

The proposed topology selection is a kind of resource selection method which is based on network status and application characteristics. Additionally, it considers overall process topology rather than each node's status. Thus it provides a group of efficient nodes for a given MPI application and improves overall performance by reducing communication delay. To demonstrate the proposed method, we implement a simple resource selector which is based on the proposed topology selection. When measuring application execution time in five distributed clusters, topology selection method can provide 200% performance gain compared with general sequential resource selection method.

In Section 2, basic concept is introduced. In Section 3, the topology selection mechanisms are described in detail. Section 4 shows several results of the proposed implementation to demonstrate its effectiveness. Finally, we present the conclusions of the research in Section 5

2 Background

Resource selection method has aroused many researchers' interest after Grid and distributed computing technologies were appeared. Especially in the field of parallel computing with MPI programming environment, it must be a crucial issue in the aspect of high performance. There are several previous approaches on the resource selection [4, 5, 6, 7], which can provide resource selection services according to their own scheduling policies. When using such resource selection services, application performance can be improved significantly, and also significant performance can be achieved even in MPI applications.

However, those are focused on the selection of individual computing nodes based on their status. As shown in Figure 1, MPI execution environments are being changed into multi-site environments that are based on heterogeneous networks. In this situation, the performance of MPI applications can be affected by not also the computing power of individual nodes but also the relationship between any two processes because each link between processes shows different execution characteristics. Namely, overall application performance can be dominated by entire process topology rather than by the status of each process. Therefore the algorithm reflecting the overall process topology will be needed.

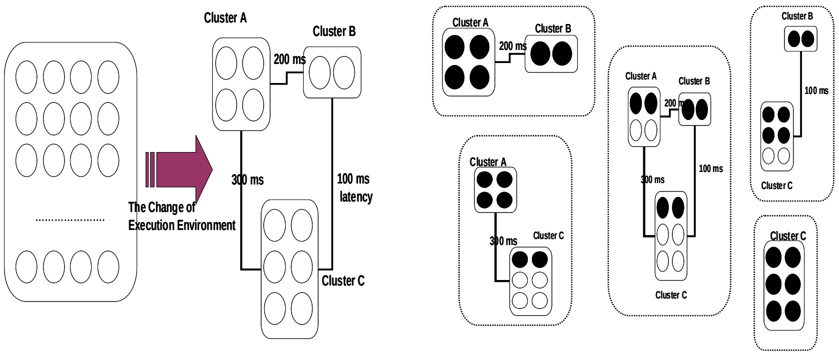


Fig. 1. Change of MPI execution environment (left). There are various cases of process topology and performance must be different each case (right). A black circle means that a process is allocated to the node.

3 Dynamic Topology Selection

3.1 Design Methodology

To design proposed topology selection method, we apply two basic principles specified from wide area message passing environments and consider two additional principles that are related to application characteristics. The basic concept reflected from the characteristics of the wide area message passing environment is that communication should be performed via wide area links as few as possible [8]. As mentioned before, the Grid environment is constructed as many wide area networks, so that communications through geometrically distributed sites can be significant performance bottleneck. This basic concept teaches us that it can reduce the amount of communication bottleneck to decrease the number of communications through wide area networks. To apply this concept more easily, we expanded it into two phases, the *use of minimum number of clusters* (A) and *giving priority to the cluster which has the lowest value of latencies over other clusters* (B). Principle A can decrease the frequency in use of wide area links. Also, it is more favorable to use a collection of clusters having better network performance according to Principle B.

Two additional principles are specified as HBPTS (High Burden Process based Topology Selection) and HBCTS (High Burden Channel based Topology Selection), which are both based on application characteristics. HBPTS is to *give higher priority to the high burden processes, which have more responsibility about the communication* (C). If previous two principles are used to determine which clusters should be used, Principle C is used to determine the order of processes in assigning resources with high performance. At last, HBCTS is *to allocate high burden channels into intra-cluster* (D) to minimize the communication latencies. It can improve performance for the applications that have heavy communications among specific processes. By applying Principle D, heavy communication channels are not used for wide area links.

Finally, our topology selection policy is constructed by combining above four principles. By using topology selection policy, the topology selection framework is implemented and explained next section.

3.2 Topology Selection Framework

In this section, we will explain the topology selection method and its detailed operational flow with a practical example. The architecture of our topology selection framework is shown in Figure 2.

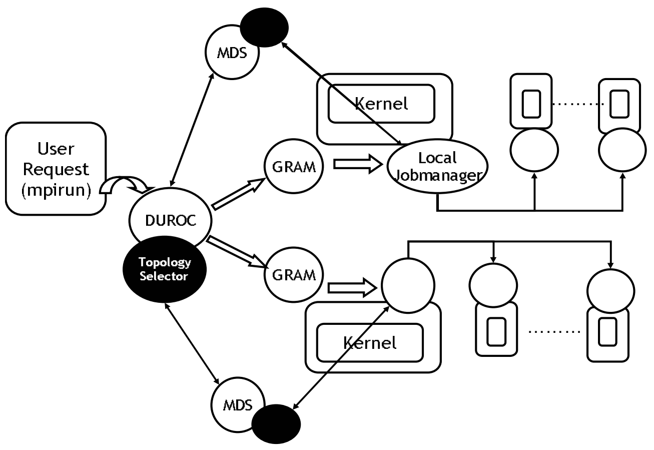
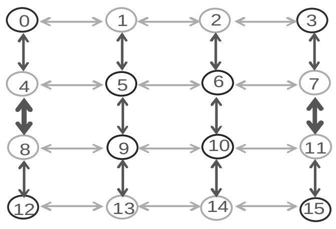


Fig. 2. Architecture of topology selection framework.

To make the topology selection framework simple, we leverage conventional Grid components in Globus Toolkit [9] and just add a topology selector drawn as black circle as in Figure 2. Modified MDS (Monitoring and Discovery Service) [10] provides necessary information to the topology selector, constructed as local queue information and dynamic network status. NWS (Network Weather Service) [11] is coupled with MDS to provide accurate network information. The topology selector makes a process list and a node list by using given information from MDS and/or NWS, and decides a collection of nodes, where MPI processes will be executed.

The flow of the topology selection comprises of 6 steps. The first step is to make a node list by using the information from MDS. Secondary, the topology selector sorts the node list by the order of communication performance. Those two steps are derived from Principle A and B. In the third step, the topology selector makes a process list by using user’s requests. In step 4, the topology selector tries to find high burden channels. If a high burden channel is discovered, it groups a pair of processes which are connected to the high burden channel. In step 5, topology selector sorts the process and group list. Finally, in step 6, the selector simply allocates sorted processes into the sorted nodes one by one.

A following example shows clear sequence of the topology selection steps. Assume that the workload is the LU solver included in NAS Parallel Benchmark [12] and the execution environment consists of five distributed clusters which have four nodes in each cluster. When using 16 processes, the LU solver has the communication pattern as shown in Figure 3. Process 5, 6, 9, and 10 are high burden processes and the channels between process 4 to 8 and process 7 to11 are high burden channels. No high burden channels exist in original LU solver, but we insert redundant communication code for easy-of-understanding. When we execute such a workload, topology selection step will be performed as in Figure 4. As shown in the figure, high burden processes (5, 6, 9, and 10) are allocated in cluster C which has the best network performance. Also, high burden channels (4 to 8 and 7 to 11) are allocated in the same clusters. The example introduced in this section was experimented in the real testbed, and the result will be shown in Section 4.



0 : 1, 4	1 : 0, 2, 5
2 : 1, 3, 6	3 : 2, 7
4 : 0, 5, 8	5 : 1, 4, 6, 9
6 : 2, 5 ,7, 10	7 : 3, 6, 11
8 : 4, 9, 12	9 : 5, 8, 10, 13
10 : 6, 9, 11, 14	11 : 7, 10, 15
12 : 8, 13	13 : 9, 12, 14
14 : 10, 13, 15	15 : 11, 14

Fig. 3. Communication patterns in LU solver. Process 5,6,9, and 10 have more responsibility for communication (high burden processes), and two channels between process 4-8 and 7-11 have more communication (high burden channel).

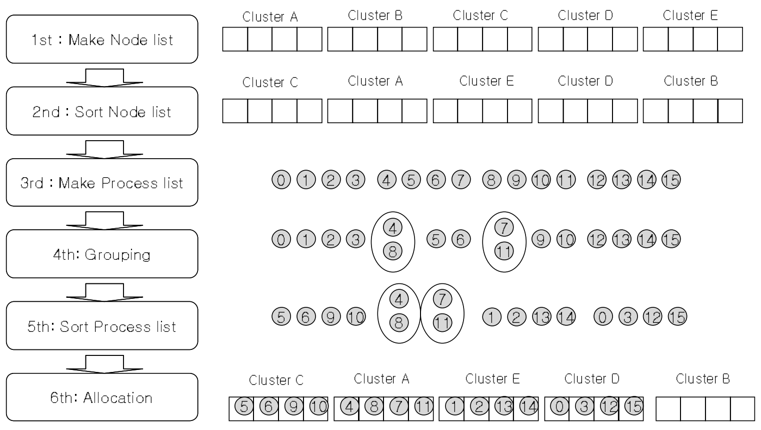


Fig. 4. 6-level Framework for dynamic topology selection (Assume that communication performance of clusters: C > A > E > D > B).

4 Experimental Result

To evaluate the impact of our topology selection, we will show the experimental results in a practical Grid environment. Table 1 shows our MPI testbed which consists of four clusters of Pentium machines connected by general purpose Internet in five universities located at different cities.

Table 1. MPI Testbed Status

Name	Type	Nodes	Job -Manager	Globus	CPU	Location
Y	Cluster	4	PBS	2.0	P4/1G	S
A	Cluster	4	PBS	2.0	P4/1.6G	I
Ki	Cluster	4	PBS	2.0	P4/2G	D
Ku	Cluster	4	PBS	2.0	P3/850	C
P	Cluster	4	PBS	2.0	P4/1.7G	P

In this environment, we executed four benchmark programs and compared execution time by changing scheduling policy and problem size. To experiment more precisely, we collect sufficient samples in 95% confidence interval.

Figure 5 shows the results of four benchmark programs. Each program has its own unique communication pattern [13]. In this figure, the sorted node list means that only two steps are performed in resource selection. Surely, the full topology selection means that all six steps are performed and no policy means that there is no resource selection service. In the case of LU benchmark as explained in the previous section, the topology selection method improves the performance by up to 200% compared with no policy. IS (Integer Sort) comprises of a number of broadcast and reduce functions. Thus the root process can be a high burden process. In this case, there is a little difference between the full topology selection and the sorted list because root process can be considered as a high burden process automatically. MG (Multi Grid) has similar communication pattern with LU. It shows the 4D hypercube topology in 16 processes and process 4, 5, 6, 7, 12, 13, 14, and 15 has more communication rate than others. In MG benchmark, performance improvement can be achieved by 180%. Differently, the processes merely communicate each other in EP benchmark. Topology selection cannot help such application because topology selection is based on communication pattern and network status.

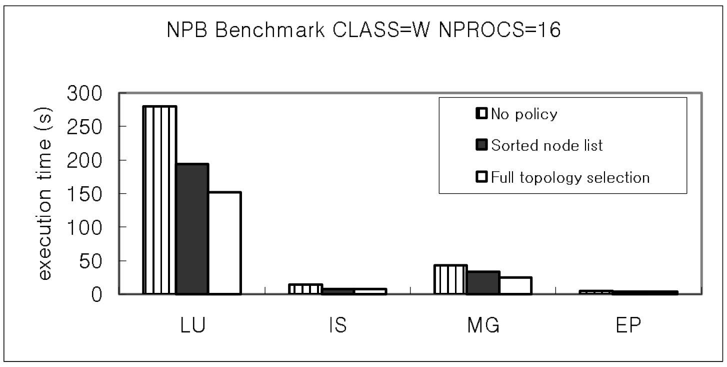


Fig. 4. Execution time of four types parallel benchmark programs with 16 processes and W class.

Also we experimented by changing problem size. NPB benchmark suite provides 5 classes of problem size, which are S, W, A, B and C, where S is the smallest and C is the largest. As shown in the figure, for the class C which is the largest, performance gain can be achieved by 200% compare with no policy and by 140% compared with sorted node list. We also can see that the performance gain become bigger if the problem size is larger because larger problem size causes more communication overhead.

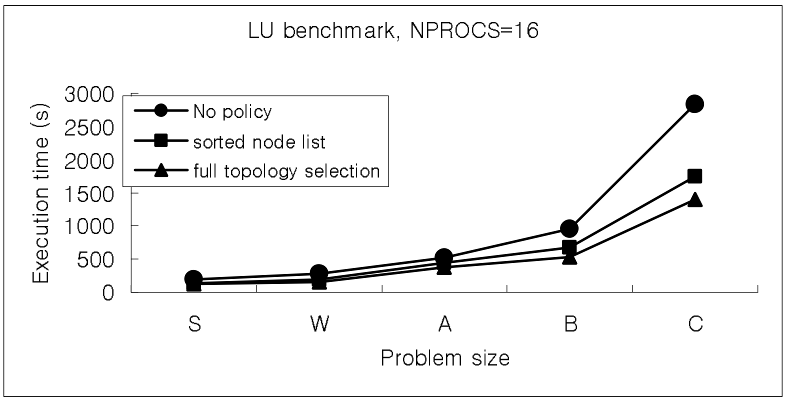


Fig. 5. Execution time according to the change of problem size.

5 Conclusion

Since the emergency of the Grid, many previous experiments show us that applications cannot be executed in Grid environments without tuning performance and functionalities of software or hardware components which comprise of the Grid. Proposed dynamic topology selection is one of the tuning methods for MPI applications. It provides an efficient resource selection service based on four principles which are derived from wide area message passing paradigm. Those principles are the use of minimum number of clusters, giving priority to the cluster which has the lowest value of latencies over other clusters, giving priority to the high burden processes, and allocating high burden channels into the intra-cluster. Consequently, the topology selection can improve the overall application performance by reducing the communication load. To demonstrate the proposed method, we measured application's execution time in four geometrically distributed clusters by changing resource selection method. When using topology selection method, performance gain can be achieved by 200% compared with no policy and by 140% compared with sorted list only policy in given execution environments.

References

1. I. Foster and C. Kesselman, eds. The GRID : Blueprint for a New Computing Infrastructure, Morgan Kaufmann, (1998).
2. Message Passing Interface Forum, MPI: A Message-Passing Interface standard, International Journal of Supercomputer Applications, 8(3/4), (1994), 165-414.
3. I. Foster and N. Karonis, A grid-enabled MPI: Message passing in heterogeneous distributed computing systems, In Proc. Supercomputing '98, 11 (1998).
4. F. Berman, and R. Wolski, The AppLes project: A Status Report, in Proc. 8th NEC Research Symposium, Berlin, Germany, (1997)
5. J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, Condor-G: A computation Managerment Agent for Multi-Institional Grids, Cluster Computing, 5(3), (2002), 237-246
6. D. Angulo, I. Foster, C. Liu, and L. Yang, Design and Evaluation of resource selection framework for grid applications, In Proc. International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, 7 (2002).
7. D. Abramson, J. Giddy, and L.Kotler, High Performance Modeling with Nimrod/G: Killer Application for the Global Grid, In Proc. International Parallel and Distributed Processing Symposium, (2000)
8. T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang, MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems, In Proc. Symposium on Principles and Practice of Parallel Programming , Atlanta, GA, 5 (1999), 131-140.
9. I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit", International Journal of Supercomputer Applications, 11(2), (1997), 115-128.
10. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, In Proc. International Sysmposium on High Performance Distributed Computing (HPDC-10), 8 (2001)
11. R. Wolski, N. Spring, and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Journal of Future Generation Computing Systems, 15(5-6), (1999), 757-768
12. D. Bailey, T. Harris, W. Saphir, R. Wijngaart, A. Woo, and M. Yarrow, NAS Parallel benchmark 2.0, Technical Report 95-020, NASA Ames Research Center, 12 (1995).
13. J. Subhlok, S. Venkataramaiah, and A. Singh, Characterizing NAS Benchmark Performance on Shared Heterogeneous Networks, In Proc. International Parallel and Distributed Processing Symposium, (2002)