논문 2010-47CI-2-1

비규칙 토폴로지 스위치 기반 클러스터 시스템을 위한 메쉬 프로세스의 인접 기반 매핑

(Adjacency-Based Mapping of Mesh Processes for Switch-Based Cluster Systems of Irregular Topology)

모 상 만*

(Sangman Moh)

요 약

가상의 프로세스 토폴로지를 물리적인 프로세서 토폴로지로 매핑하는 문제는 병렬 프로그래밍에서 가장 중요한 이슈 중의 하나이다. 그러나 이 매핑은 토폴로지 비규칙성 및 라우팅 복잡성으로 인해 어려운 문제로 여겨지고 있다. 본 논문에서는 프로세스간 통신 패턴으로 2차원 메쉬 프로세스 토폴로지를 가정하여, 비규칙 클러스터 시스템을 위한 새로운 프로세스 매핑 기법인 인접 기반 매핑(AM)을 제안한다. 클러스터 시스템은 전통적인 규칙성 네트워크에서는 달성하기 어려운 상호연결 유연성과 시스템 확장성을 제공하기 때문에 여러 해 동안 활발히 연구 개발되어 오고 있다. 제안한 AM은 가상 프로세스 토폴로지에서 이웃하는 프로세스를 물리적인 프로세서 토폴로지의 인접한 프로세서에게 매핑시킨다. 시뮬레이션 결과에 따르면, 제안한 AM은 기존 방법에 비하여 매핑 품질이 우수하고 프로세스간 지연시간이 감소하는 결과를 나타낸다.

Abstract

Mapping virtual process topology to physical processor topology is one of the most important design issues in parallel programming. However, the mapping problem is complicated due to the topology irregularity and routing complexity. This paper proposes a new process mapping scheme called adjacency-based mapping (AM) for irregular cluster systems assuming that the two-dimensional mesh process topology is specified as an interprocess communication pattern. The cluster systems have been studied and developed for many years since they provide high interconnection flexibility, scalability, and expandability which are not attainable in traditional regular networks. The proposed AM tries to map neighboring processes in virtual process topology to adjacent processors in physical processor topology. Simulation study shows that the proposed AM results in better mapping quality and shorter interprocess latency compared to the conventional approaches.

Keywords : Cluster system, process mapping, process topology, irregular network, MPI.

* 평생회원, 조선대학교 컴퓨터공학부
(School of Computer Eng., Chosun University)

※ A preliminary version of this work was published at the Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications, June 2009[26]. This research was supported in part by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1021-0013).

접수일자: 2009년11월10일, 수정완료일: 2010년3월10일

I. Introduction

In parallel programming, domain decomposition has been one of the most important issues. A problem domain is divided into almost equal-sized partitions and they constitute a computational graph. It is called a process topology where each vertex represents the required computation or the corresponding process, and an edge represents the communication between

two processes. For executing a parallel program, processes in process topology should be mapped to processors in physical processor topology. The mapping is crucial for application performance[1] but it is relatively easy when processor topology is regular. In fact, the process topology is typically regular as many well-known computation-intensive problems are parallelized or decomposed into regular structures such as meshes or rings. The processor topology is also regular in traditional parallel machines which usually employ regular networks.

This paper discusses the process mapping problem in the cluster systems based on the switch-based irregular network[2~6], where the processor topology is irregular. Usually, switches are interconnected in an arbitrary manner and they form an irregular topology. The corresponding advantages of the irregularity are higher flexibility, greater scalability, and incremental system expandability which are not attainable in traditional regular networks. On the other hand, one important disadvantage is routing complexity. Here the irregular topology makes it difficult to avoid deadlocks among multiple packets traveling simultaneously[7~11]. One possible solution is to prevent deadlock by restricting the sequences of turns in the routing paths, which has been originally suggested for a regular mesh topology[12].

In this paper, we propose an efficient process mapping scheme for irregular cluster systems assuming that the two-dimensional (2D) mesh process topology is specified as an interprocess communication pattern. One such example is MPI_Cart_create() in MPI standard[13]. See Section II.2 for details. Due to the network irregularity and routing complexity, the mapping problem in such cluster systems becomes very complicated. The key idea is to map the neighboring processes in the process topology to physically adjacent processors. The proposed mapping scheme is called adjacency-based mapping (AM). Its time complexity is $O(n^{3/2})$, where n is the number of processes to be mapped. Simulation study shows that the proposed AM results in better mapping quality and shorter interprocess latency compared to the conventional mapping schemes. The proposed AM is a more advanced mapping algorithm compared to switch-based mapping (SM) and binary mapping (BM) studied in [14]. The metric introduced in [14], cardinality, is also used in this paper to represent the interprocess communication performance. Communication pattern of parallel applications are generally more complicated compared to regular networks. Cardinality is an application-oriented performance measure which accounts for the bandwidth along all network links specified in the communication pattern.

The rest of this paper is organized as follows: The mapping problem for irregular processor topology is discussed along with some related issues in the following section. In Section III, the proposed process mapping scheme is presented and its complexity is analyzed in terms of cost-quality tradeoff. The proposed AM is evaluated using simulation study in Section IV. The conclusions and future works are discussed in Section V.

II. Process Mapping in Cluster Systems

This section covers the mapping problem for cluster systems based on a switch-based irregular network. Throughout this paper, we use process topology and virtual topology interchangeably. The latter is used to contrast with the physical processor topology. The concept of routing topology is also described.

1. Mapping Problem

Cooperating processes running on multiple processors exchange messages through the underlying interconnection network and, in most cases, the corresponding communication pattern is logically configured in a regular topological structure such as a mesh or a simple graph. There usually exist more communications between neighboring

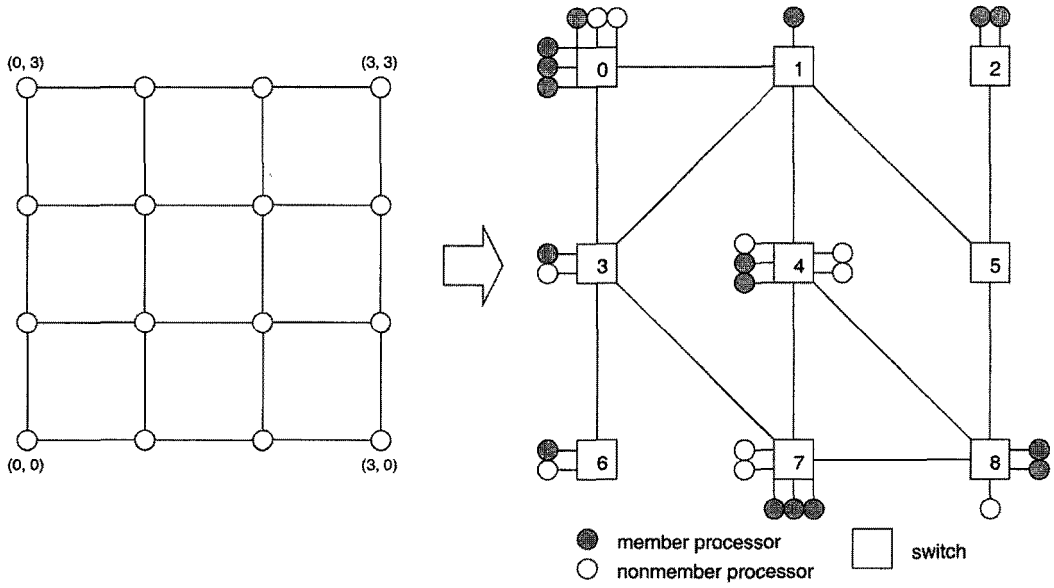


그림 1. 가상 토폴로지 (a) 및 프로세서 토폴로지 (b)
Fig. 1. Virtual topology (a) and processor topology (b).

processes than between distant processes. It is, thus, reasonable to map the neighboring processes to the physically neighboring processors in order to reduce the communication overhead. General formulation of this problem is known to be computationally equivalent to the graph isomorphism problem, and has been shown to be NP-complete[15~16].

A processor topology shows how the processors are interconnected through the switch-based irregular network as shown in Fig. 1. Each switch has a set of ports and each port is connected to a computational node or another switch. Some ports may be left open and they can be used for further system expansion. The system in Fig. 1 consists of nine 8-port switches, 26 processors, and 12 inter-switch links. In this example, the mapping problem is to map a 4 × 4 mesh type of virtual topology onto the 16 processors (the dark circles in Fig. 1(b)) selected for running a parallel application.

However, the routing topology will be used instead of the processor topology because the latter does not reflect the routing complexity and deadlock prevention. A routing topology shows how the communication occurs between the switches as shown later in this section. Here we assume the routing topology based on up/down routing[7] which is a variant of the turn model[12]. Upon startup of a cluster system, a processor in each switch starts the configuration algorithm to figure out the overall topology. A breadth-first spanning (BFS) tree is computed and all switches eventually agree on a unique spanning tree. Deadlock-free routing is based on loop-free assignment of direction to the operational links and the following up/down rule: a legal route must traverse zero or more links in the up direction followed by zero or more links in the down direction. Routing algorithms for Myrinet switch from Myricom[17] and Berkeley[8] are similar to the up/down routing. Basic up/down routing algorithms have been extended to allow adaptivity[9~11] and multicast has also been studied in switch-based arbitrary networks[18~21]. Detailed explanation of the BFS and the up/down routing can be found in [7, 21].

For the processor topology of Fig. 1, the corresponding routing topology is shown in Fig. 2. Note that switches 4 and 7 are close neighbors in the processor topology as shown in Fig. 1 but they are

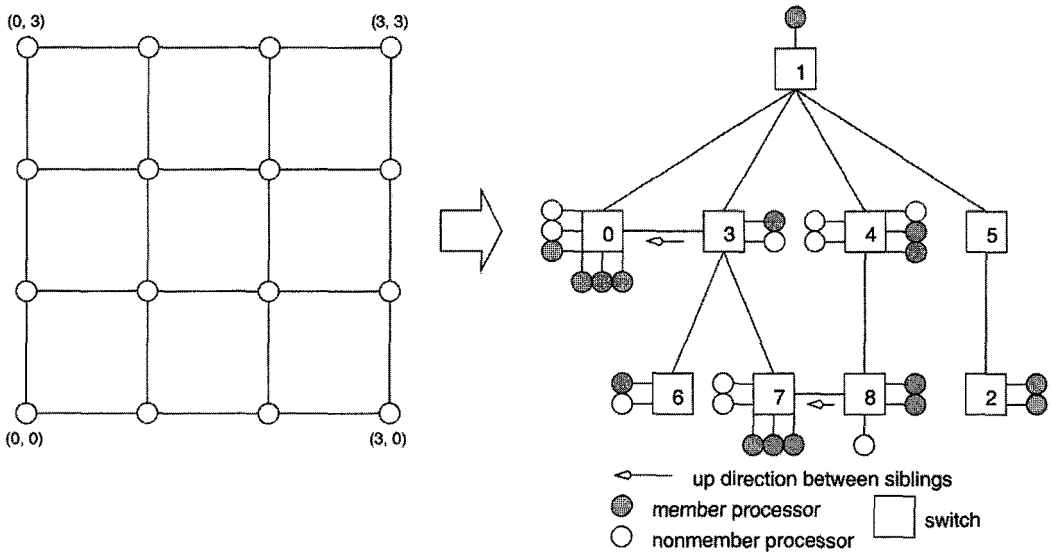


그림 2. 가상 토폴로지 (a) 및 라우팅 토폴로지 (b)
Fig. 2. Virtual topology (a) and routing topology (b).

three links (hops) away in the routing topology as shown in Fig. 2. This is due to the routing algorithm and it implies that some communication links in the processor topology may not actually be utilized. Communication cost between two processors of switches 4 and 7, respectively, is high even though it is not evident in the processor topology. Thus, we use the routing topology instead of the processor topology to map processes to physical processors, as shown in Fig. 2.

2. MPI Process Topology Model

There have been many researches and actual implementations of message passing libraries for distributed memory multicomputers. Among them, MPI[13] is gaining wide acceptance in industry mainly due to its simple interface. When multiple processes are created by MPI_Init() routine, they form a communicator among the process group to exchange messages. Depending on the communication pattern of an application, a node may exchange more messages with one node than the others. In order to explore the application-specific characteristics, there must be a way to inform the communication pattern of a parallel application. As other message-passing libraries, MPI provides the concept of virtual topology, on which applications can be implemented [16]. In PARMACS[22] and p4[23] libraries developed at the GMD (German National Research Center for Computer Science) and ANL (Argonne National Laboratory), respectively, the virtual topology can either be a Cartesian structure with up to three dimensions or a general graph[22]. Using MPI_Cart_create() and MPI_Graph_create() APIs, a virtual topology can be specified.

In many parallel programming interfaces, little consideration is paid to the mapping of the processes to the processors. Usually the mapping of the processes is either random or left to the user to do it manually. In case of MPI, the mapping between the process topology and the processor topology is implementation-dependent and beyond the scope of MPI even though any MPI implementation would try to have a sensible mapping of processes onto processors. Our main effort in this paper is, thus, to propose efficient mapping algorithms which place the adjacent processes on adjacent processors as much as possible.

III. Adjacency-Based Mapping

Usually, the communication between neighboring processes is dominant compared to that between distant processes. So, it is important that the adjacency of processes is kept as much as possible. Fig. 3 shows the proposed adjacency-based mapping algorithm. We first map the center process of the mesh to one of the nodes connected to the root switch (or a switch closest to the root switch), and the rest of the mesh is partitioned into four disjoint quadrants. Then, the routing topology R is partitioned and assigned to four quadrants as described in Step 5 to 7. Such a mapping procedure is repeated in a spanning sequence of the routing tree until there is no remaining process in the virtual topology. As a result, the root node of a routing tree is located at the center of the virtual topology, while leaf nodes are assignedto the processes at the boundaries. Note that the algorithm in this section searches a switch in the longest distance from the root as shown in Step 7 for partitioning R into four sets i.e., the farthest switches are taken first. Finally, the same mapping algorithm is called recursively for every quadrant as shown in Step 8 to 11.

Given a base process, $b(x_b, y_b)$, and a set of vertices (nodes or processes) A in a virtual topology, four disjoint quadrants of a 2D mesh $Q_{+X}(b)$, $Q_{-X}(b)$, $Q_{+Y}(b)$, and $Q_{-Y}(b)$ are defined as follows:

Adjacency_Mapping(R, A)
/*
Let A be the virtual topology with a set of processes, and R be the routing topology with a set of nodes. Let b be the base process of A (b is the center process at the top-level partitioning or the closest process to the upper-level base process, otherwise).
*/
1: If there are no more processes to be mapped in A, return.
2: Map the base process b in A to a processor closest to the root switch (or a switch closest to the root switch) in R.
3: Partition A into four quadrants Q_{+X}, Q_{-X}, Q_{+Y} and Q_{-Y}.
/*
R is partitioned into four sets R_{+X}, R_{-X}, R_{+Y} and R_{-Y}, and four process quadrants Q_{+X}, Q_{-X}, Q_{+Y} and Q_{-Y} are mapped to the four sets, respectively, with the following steps.
*/
4: Initialize the average mapping distance (AMD) of R_{+X}, R_{-X}, R_{+Y} and R_{-Y} to ∞.
5: Select the next quadrant Q_i to be mapped (in the order of counter-clockwise direction from $+X$).
6: Select a switch with the longest distance from the root in R, and assign the member nodes of the switch to the corresponding R_i as far as the unmapped processes in Q_i remain.
7: If AMD of R_i is larger than that of the previous R_i, then go to Step 6. Else, if there are more switches or processors to be assigned in R, go to Step 5.
/*
In case there is no unassigned processor left in R, do the followings.
*/
8: call Adjacency_Mapping(R_{+X}, A_{+X})
9: call Adjacency_Mapping(R_{-X}, A_{-X})
10: call Adjacency_Mapping(R_{+Y}, A_{+Y})
11: call Adjacency_Mapping(R_{-Y}, A_{-Y})

그림 3. 인접 기반 매핑 알고리즘
Fig. 3. Adjacency-based mapping algorithm.

$$Q_{+X}(b) = \{(x,y) | (x,y) \in A, x > x_b, \text{ and } y \geq y_b\},$$

$$Q_{-X}(b) = \{(x,y) | (x,y) \in A, x < x_b, \text{ and } y \leq y_b\},$$

$$Q_{+Y}(b) = \{(x,y) | (x,y) \in A, x \leq x_b, \text{ and } y > y_b\}$$

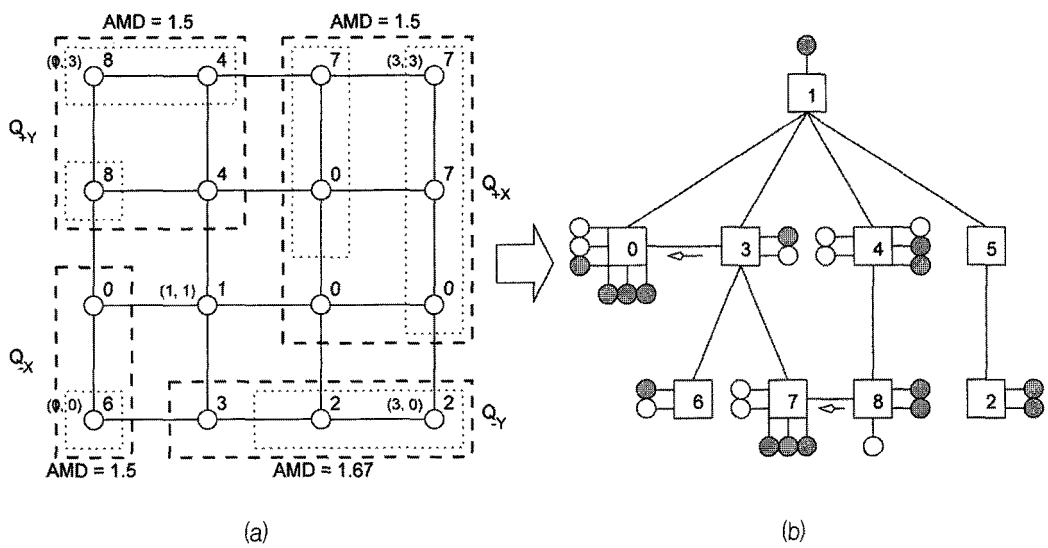
and

$$Q_{-Y}(b) = \{(x,y) | (x,y) \in A, x \geq x_b, \text{ and } y < y_b\}.$$

Note from Fig. 4, with the process at (1, 1) (labeled 1) as a base process, the four quadrants are shown in dark dotted lines. In each quadrant, the process closest to the base process becomes the next-level base process, and the rest of them are partitioned into four sub-quadrants recursively. Except the top-level partitioning, each partitioning results in one or two partitions out of four possible partitions since the base process is located at a corner of the quadrant as can be seen in Fig. 4.

Here, we use average mapping distance (AMD) of a partition as the average distance between the root and all the member nodes with the partition. The basic step of the adjacency-based mapping algorithm is to map the nodes such that the AMD values of four quadrants are as close as possible at each recursive step. Note that since AMD is the distance measure from the root, the following adjacency-based mapping algorithm considers the adjacency properties from the root process only. It is a reasonable simplification since we wanted to alleviate the communication problem centered at the root node, which is unavoidable with the BFS-like routing scheme as explained in Section II.1 and Fig. 2.

In the example shown in Fig. 4, the first, top-level, partitioning of a virtual topology produces four quadrants Q_{+X}, Q_{-X}, Q_{+Y} , and Q_{-Y} around the root at (1, 1). In order to partition the routing topology, the algorithm starts from the processors of the distant switch from the root since they affect AMD more than those of nearby switches. In this example, switches with the longest distance from the root are 6, 7, 8, and 2 as shown in Fig. 4. We randomly select a switch among those with the longest distance, say switch 2, and map the two processors of the switch to a quadrant Q_{-Y} (AMD = (2+2)/2 = 2). Now, we select others 7, 8 and 6 in that order, and map the corresponding processors to the quadrant Q_{+X} (AMD = (2+2+2)/3 = 2), Q_{+Y} (AMD = (2+2)/2 = 2) and Q_{-X} (AMD = 2/1 = 2). We repeat the same procedure for switches with the next longest distance from the root, 0, 3, 4 and 5 in this example. Such partitioning is recursively continued until there is only one process remained. Finally, four top-level quadrants



(a) (b)

그림 4. 인접 기반 매핑의 예

Fig. 4. An example of the adjacency-based mapping.

have AMD of 1.5, 1.5, 1.5, and 1.67 which are almost the same.

The adjacency-based mapping has a time complexity of $O(n^{3/2})$. Here $O(n)$ is due to the repetitive operations of switch sorting and node assignment (partitioning) at each recursive step, and $O(n^{1/2})$ is due to the number of worst-case recursive steps. With a virtual topology of a $k \times k$ mesh and n member nodes to be mapped, the dimension size k is $\sqrt{n}$ and the number of worst-case recursive steps in top-level quadrants is at most $\sqrt{n}/2 + \lceil \sqrt{n}/2 \rceil$, resulting in $O(n^{1/2})$.

IV. Performance Evaluation

In this section, the performance of the proposed mapping strategies is evaluated using extensive simulation. The performance metrics used for evaluating the mapping quality are established first, along with the experiment environment. The performances of the three proposed algorithms are then evaluated and compared.

1. Quality of Mapping

One way to measure the quality of a mapping is to compute cardinality which is similarly introduced in [24]. For each edge in a virtual topology, the links required to be traversed in the corresponding routing topology is counted. For example, for the edge between two processes on the left of Fig. 4, which are the nodes of switches 8 and 0, the cardinality is 3 from Fig. 4. Cardinality is defined as the ratio of sum of these values obtained for all the edges in the virtual topology to the total number of edges in the virtual topology. That is, the cardinality is equivalent to the average number of physical links corresponding to an edge of the virtual topology.

For the mapping of Fig. 4, cardinality is 33/24 = 1.375. Mapping with a cardinality of one, which is unrealizable in most cases, has minimum interprocessor communication overhead since all processes that need to communicate each other lie on the neighboring processors. To find the optimal value of cardinality, which is equal to or larger than one, we must try all possible $n!$ mapping functions, where n is the number of processors to be mapped[25]. Despite the ignorance of the optimal value, we can compute the cardinalities of mapping strategies to compare application-specific communication performance.

Communication latency is also an important performance metric which is measured by the interval from the time when a communication routine is invoked to the time when the communication is completed. Once a network is established and configured at a time, each node-to-node bandwidth is automatically determined for the corresponding route. Thus, the only communication latency is investigated in this paper.

2. Simulation Environment

In this paper, we consider three cases of interprocess communication patterns for evaluating the performance of the proposed mapping strategies. They are (i) each process communicates only with its neighboring processes, (ii) each process communicates with both its neighboring processes and 10% of random distant processes in the process group, and (iii) each process communicates with both its neighboring processes and 20% of random distant processes in the group. For each case, any two communicating processes send and receive a message with each other, resulting in two messages transferred between a pair of processes. It is also assumed that point-to-point communication is used.

Based on the performance values of contemporary silicon technology, we assume the following default performance parameters: communication startup time (t_s) of 2 *μsec*, link propagation delay (t_p) of 20 *nsec*, and switch (router) delay (t_r) of 300 *nsec*. We also assume that the network interface delay is almost the same as the switch delay for our evaluation. Consequently, the communication latency of a message transfer can be approximated to $t_s + d \times t_p$

+ $(d + 1) \times t_r$, where d is the distance between the source and destination nodes in a communication.

For all experiments, unless otherwise stated, we assume that the cluster system consists of 256 nodes (processors) and 75 switches and the network is interconnected with 8-port switches having 75% connectivity. As in [18], we define the connectivity, or connection ratio, f of k-port switches as the ratio of the average number of connected ports over k. Hence, $f \times k$ is the average number of ports in a switch, which are connected to either other switches or computational nodes. If we have s k-port switches, there exist at least $(s - 1)$ links connecting the switches. Consequently, $2(s - 1)$ ports are used for connecting the switches, and $ks - 2(s - 1) = (k - 2)s + 2$ ports remain. The remaining ports are either used for connecting nodes or left open. The 75% connectivity ($f = 0.75$) means that 6 out of 8 ports in a switch are connected to either other switches or computational nodes on the average.

3. Simulation Results and Discussion

The cardinality data is shown in Fig. 5 for four different mapping strategies, where RM, SM, BM and AM represent random mapping, switch-based mapping, binary mapping and adjacency-based mapping, respectively. Notice from the figure that the proposed AM shows improvement over the conventional schemes, while the improvement gets larger as the size of mesh increases. This is intuitively acceptable since the effect of mapping gets more significant as the size of process topology increases.

The communication latency is evaluated for the three different communication patterns in terms of average latency between communicating nodes. Fig. 6 shows the simulation results of the case that each

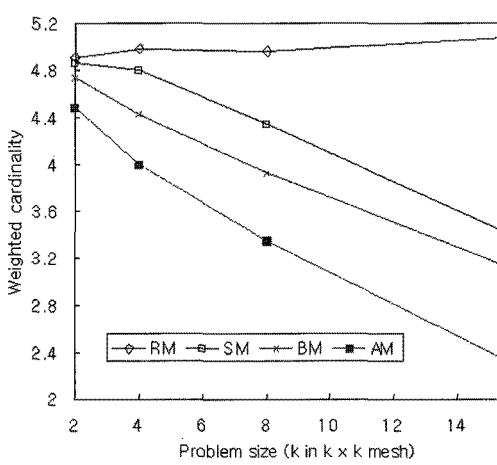


그림 5. 256개의 노드와 75개의 스위치로 75% 연결성을 제공하는 네트워크에서의 카디널리티

Fig. 5. Cardinality on the network with 256 nodes and 75 switches having 75% connectivity.

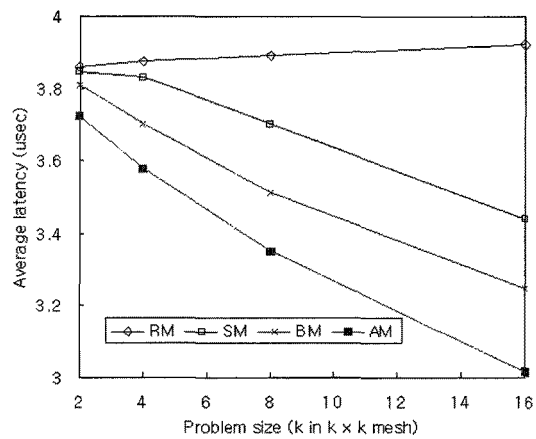


그림 6. 256개의 노드와 75개의 스위치로 75% 연결성을 제공하는 네트워크에서 경우 (i)의 통신 지연시간

Fig. 6. Communication latency of the case (i) on the network with 256 nodes and 75 switches having 75% connectivity.

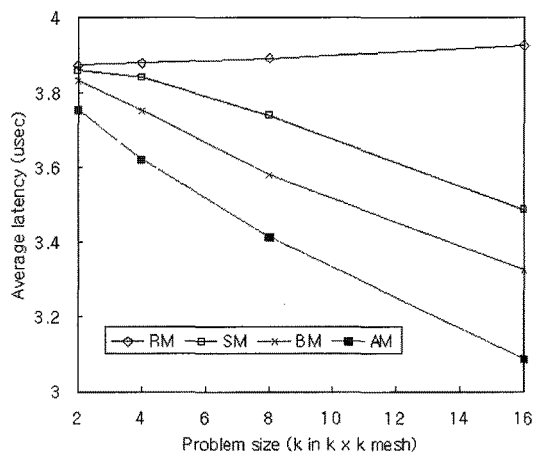


그림 7. 256개의 노드와 75개의 스위치로 75% 연결성을 제공하는 네트워크에서 경우 (ii)의 통신 지연시간

Fig. 7. Communication latency of the case (ii) on the network with 256 nodes and 75 switches having 75% connectivity.

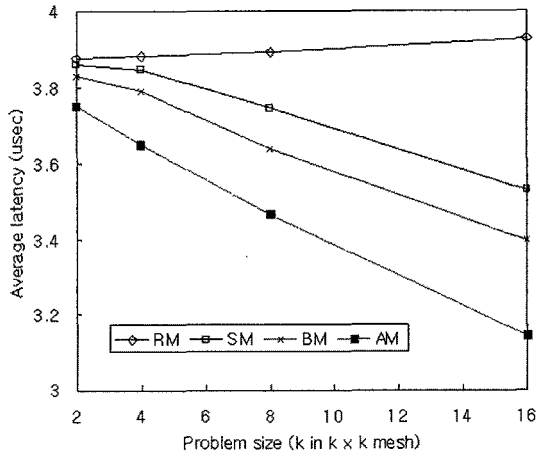


그림 8. 256개의 노드와 75개의 스위치로 75% 연결성을 제공하는 네트워크에서 경우 (iii)의 통신 지연시간

Fig. 8. Communication latency of the case (iii) on the network with 256 nodes and 75 switches having 75% connectivity.

node communicates only with its neighboring nodes. Fig. 7 and Fig. 8 are for the other two cases of more communication between distant nodes. In Fig. 6, it is shown that, for a process topology of a 16 × 16 mesh, AM outperforms RM, SM and BM in terms of interprocess communication latency. Interestingly, the three plots display almost the same trends as the plot of cardinality. This is because cardinality represents the degree of closeness between the routing topology and virtual topology after mapping. Small cardinality implies that the average routing distance between neighboring processes is small, while the communication latency between them is linearly proportional to this value.

V. Conclusion

Since the mapping problem is computationally NP-complete in nature, we have proposed a novel heuristic mapping algorithm called adjacency-based mapping (AM). The proposed AM then evaluated in terms of cardinality and interprocess communication latency for different types of network traffic. Simulation study shows that the proposed AM results better mapping quality and shorter interprocess communication latency compared to the conventional process mapping approaches. This is mainly because AM allows the overall structure of the routing topology to be kept as much as possible in the virtual topology after mapping. Also the efficiencies of the mapping schemes get more significant as the size of the virtual topology increases. In addition, if long-distance distributed setups are considered, this work has much more merit.

One of future works is to investigate the heuristic algorithms that tackle the mapping problem for different process and processor topologies. MPI implementation and measurement are also an interesting research topic.

References

[1] H. Chen, and P. Wyckoff, "Simulation Studies of Gigabit Ethernet versus Myrinet using Real Application Cores," Proc. of 4th Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing (CANPC), 2000.

[2] R. Buyya, High Performance Cluster Computing: Architectures and Systems, Prentice-Hall Inc., NJ, 1999.

[3] G. F. Pfister, In Search of Clusters, 2nd Ed., Chapter 5, Prentice-Hall, Inc., NJ, 1998.

[4] P. Czarnul, "Dynamic Process Partitioning and Migration for Irregular Applications," Proc. of Int. Conf .on Parallel Computing in Electrical Engineering, 2002.

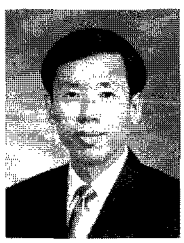
[5] J. M. Orduna, F. Silla, and J. Duoto, "On the Development of a Communication-Aware Task Mapping Technique," Journal of Systems Architecture, Vol. 50, No. 4, pp. 207-220, Mar. 2004.

[6] D. Avresky and N. Natchev, "Dynamic Reconfiguration in Computer Clusters with Irregular Topologies in the Presence of Multiple Node and Link Failures," IEEE Trans. on Computers, Vol. 54, No. 5, pp. 603-615, May 2005.

[7] M. D. Schroeder, et.al., "Autonet: a High-speed, Self-configuring Local Area Network Using Point-to-point Links," SRC Research Report,

No.59, Digital Equipment Corporation, April 1990.
[8] A. M. Mainwaring, B. N. Chun, S. Schleimer, and D. S. Wilkerson, "System Area Network Mapping," Proc. of Annual Symp. on Parallel Algorithms and Architectures, 1997.
[9] W. Qiao and L. M. Ni, "Adaptive Routing in Irregular Networks Using Cut-Through Switches," Proc. of Int. Conf. on Parallel Processing, 1996.
[10] F. Silla and J. Duato, "On the Use of Virtual Channels in Networks of Workstations with Irregular Topology," Proc. of Workshop on Parallel Computer Routing and Communication, Also in Lecture Notes in Computer Science, Vol. 1417, pp. 202-216, 1997.
[11] F. Silla, J. Duato, A. Sivasubramaniam and C. R. Das, "Virtual Channel Multiplexing in Networks of Workstations with Irregular Topology," Proc. of Int. Conf. on High Performance Computing, pp. 147-154, Dec. 1998.
[12] C. Glass and L. Ni, "The Turn Model for Adaptive Routing," Journal of the ACM, Vol. 41 No. 4, Sep. 1994.
[13] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 2.1, June 23, 2008.
[14] S. Moh, C. Yu, H. Y. Youn, B. Lee, and D. Han, "Mapping Strategies for Switch-Based Cluster Systems of Irregular Topology," Proc. of 8th Int. Conf .on Parallel and Distributed Systems, pp. 733-740, Jun. 2001.
[15] O. KrÄamer and H. MÄuhlenbein, "Mapping Strategies in Message-Based Multiprocessor Systems," Parallel Computing, Vol. 9, pp. 213-225, 1989.
[16] T. Hatazaki, "Rank Reordering Strategies for MPI Topology Creation Functions," Lecture Notes in Computer Science, Vol. 1497, pp. 188-195, 1998.
[17] N. Boden, et.al., "Myrinet: A Gigabit-per-Second Local Area Network," IEEE Micro, Vol. 15 No. 1, pp. 29-36, Feb. 1995.
[18] R. Kesavan, K. Bondalapati, and D. K. Panda, "Multicast on Irregular Switch-Based Networks with Wormhole Routing," Proc. of Int. Symp. on High Performance Computer Architecture, 1997.
[19] R. Kesavan and D. K. Panda, "Multicasting on Switch-Based Irregular Networks using Multi-drop Path-Based Multidestination Worms," Proc. of Workshop on Parallel Computer Routing and Communication, Also in Lecture Notes in Computer Science, Vol. 1417, pp. 217-230, 1997.
[20] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Multicasting in Irregular Networks with Cut-Through Switches Using Tree-Based Multidestination Worms," Proc. of Workshop on Parallel Computer Routing and Communication, Also in Lecture Notes in Computer Science, Vol. 1417, pp. 39-52, 1997.
[21] C. B. Stunkel, R. Sivaram, and D. K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact," Proc. of Int. Symp. on Computer Architecture, 1997.
[22] R. Calkin, R. Hempel, H.-C. Hoppe, and P. Wypior, "Portable programming with the PARMACS message-passing library," Parallel Computing, Vol. 20, pp. 614-632, 1994.
[23] R. M. Butler and E. L. Lusk, "Monitors, Messages, and Clusters: the p4 Parallel Programming System," Parallel Computing, 1994.
[24] M. J. Berger and S. H. Bokhari, "A Partitioning Strategy for Nonuniform Problems on Multiprocessors," IEEE Trans. on Computers, Vol. C-36, No. 5, pp. 570-580, May 1987.
[25] S. H. Bokhari, "On the Mapping Problem," IEEE Trans. on Computers, Vol. C-30, No. 3, pp. 207-214, March 1981.
[26] S. Moh, "Adjacency-Based Mesh Process Mapping for Irregular Cluster Systems," Proc. of 11th IEEE Int. Conf. on High Performance Computing and Communications, pp. 500-505, Seoul, Korea, June 2009.

저 자 소 개



모 상 만(평생회원)
2002년 한국과학기술원(KAIST) 박사 졸업.
1991년~2002년 한국전자통신연구원(ETRI) 컴퓨터소프트웨어연구소 팀장.
2002년~현재 조선대학교 컴퓨터공학부 재직.
<주관심분야 : 모바일컴퓨팅, 애드혹네트워크, 센서네트워크, 병렬분산컴퓨팅, 컴퓨터구조 등>