Pruning Cache: A Dynamic Directory Generation Scheme For Distributed Shared Memory

KATSUNOBU NISHIMURA † TOMOHIRO KUDOH ‡ HIROAKI NISHI † HIDEHARU AMANO †

† Department of Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, 223-8522 JAPAN

‡ Real World Computing Partnership
1-6-1 Takezono, Tsukuba, Ibaraki, 305 JAPAN

Abstract — The pruning cache directory method is proposed for dynamic hierarchical directory management on large scale CC-NUMA systems in which a node aggressively shares data with other nodes. By a combination with Reduced Hierarchical Bitmap Directory (RHBD) method, dynamic directory is quickly formed and managed with a small (about 5%) additional hardware.

The result of a trace driven simulation of a system with 64 processors shows that more than 90% hit ratio can be obtained with a small 2-way associative pruning cache with 32 entries. From the probabilistic simulation of a large system, it appears that the combination of the pruning cache and the RHBD achieves better latency and bandwidth than those of traditional directory management methods based on 1-to-1 message passing.

Key words — DSM, CC-NUMA, Cache directory

1 Introduction

Cache Coherent Non Uniform Memory Access model (CC-NUMA) machines have been developed by many companies and universities[2, 4, 8].

In these systems, an entry of cache directory is associated with each cache line, and invalidation type protocols are commonly used. According to various simulations and experiments using developed machines[4], it becomes clear that such schemes can achieve high performance for machines with tens to hundreds of processors.

An entry of the directory specifies processors which share the data, and thus coherence maintenance messages are needed to be sent. The simplest method is a full-map directory which uses a bit-map with the same length as the number of processors. When the number of processors in the system is large, use of full-map is too expensive because of the large amount of required memory.

In most cases, the number of processors which actually share the data is one or two when an entry is associated to a line and invalidation type protocol is used[1]. According to this characteristic, directory schemes such as limited pointers[1, 3] and chained directory [5, 12] have been used for large scale systems.

However, if an entry of the directory is associated with each line, the required amount of memory for directory becomes large as the size of the system increases. To reduce the required amount of memory, some systems such as Tempest[11] and a massively parallel machine JUMP-1[16] associate an entry of the directory to each page (i.e. relatively large memory block such as 4k bytes). For each line, few flags are provided to indicate the sharing state. In this scheme, an entry of the directory should specify all the processors which share at least one of the lines of the page. Thus, the number of the processors specified by an entry tends to be large.

On the other hand, when multiple processors participate in frequent data exchange using the same address of the shared memory, update type protocols are suitable. When an update type protocol is used, the number of processors which share a data also tends to be large[10].

In these cases, the limited pointers scheme will require frequent use of exceptional operations such as software emulation, and the chained directory scheme will form an extensively long list structure.

In this paper, a dynamic directory generation scheme called the pruning cache is proposed, and combination with the reduced hierarchical bitmap directory scheme(RHBD)[15] is discussed.

2 Pruning Cache

2.1 The concept of the pruning cache

When an entry of the directory is associated with each page and/or update type protocol is used, the set of processors which share a block (page or line) does not change frequently. It changes only when a new processor shares the data or a replacement occurs due to capacity miss. In this case, it is easy for a processor to judge whether it shares a block or not.

In the pruning cache scheme, the coherence maintenance message is first sent to a set of processors which includes all processors which actually share the block, and then the recipient processors notify the sender whether they actually share the block or not by returning acknowledge or not-acknowledge messages.

The sender can dynamically construct a full-map cache entry by setting "1" to the bits corresponding to processors which returned acknowledge and setting "0" to the bits corresponding to processors which returned not-acknowledge. Subsequent coherence maintenance messages can be just sent to sharing processors according to this temporal formed full-map entry. If the first coherence maintenance message is sent to the all processors of the system, the message transfer is broadcast(Figure 1).

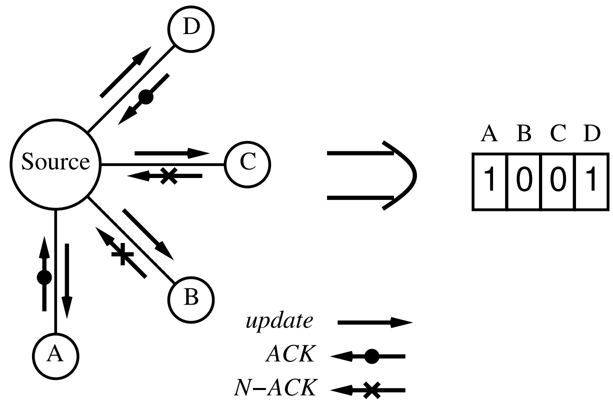


Figure 1: Pruning Cache

If there is a temporary locality of references to the blocks, limited number of pruning cache entries will be sufficient to achieve enough performance. To replace pruning cache entries, conventional method such as LRU can be used. If an access to the replaced block occurs, the consistency maintenance message is sent to the larger set again, and the full-map entry is reconstructed.

If a processor withdraws from sharing after a corresponding pruning cache entry has been generated, not-acknowledge is returned when the next coherence maintenance message is sent, and then the pruning cache entry is updated. When a new processor participates in sharing, corresponding pruning cache entry is invalidated and constrated again.

2.2 Application of pruning cache to the hierarchical bit-map directory

If there are considerable number of destinations, it will take a long time to send a message one after another(sequentially).

To reduce it, a hierarchical bit-map directory scheme is introduced. In this schemed, messages are transferred to different destinations simultaneously (i.e. multicast) using a tree structured multicasting path (multicasting tree). To specify the pattern of multicast at each node, directories are provided at each node of the tree structured path[15].

Figure 2 illustrates the concept of hierarchical bitmap directory scheme. In this case, the message is sent using 3-ary tree to the processors marked with "●". Leaves of the tree correspond to the processors. A message is first sent from the root of the tree. Each node can multicast a message to its branches at a time. If each node sends the message only to the branches which include sharing processors, the message is sent exactly to the sharing processors.

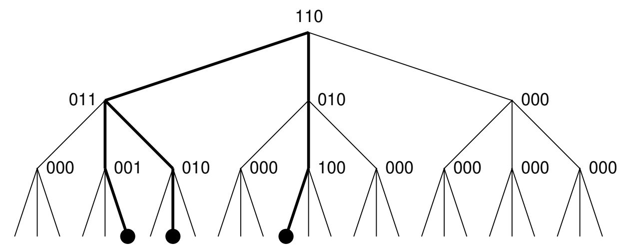


Figure 2: Hierarchical bit-map directory scheme

By providing three bit bit-map entries to all nodes, destinations can be exactly specified. One of the advantages of the hierarchical bit-map directory scheme is that there is no multiple transfers of the same message through the same path. Thus, multicasting is effectively performed.

By combining the pruning cache scheme with the hierarchical bit-map directory scheme, elements of the pruning cache can be distributed to the intermediate nodes of the tree. Each pruning cache element can be constructed with small amount of hardware, since each entry's bit length is equal to the number of branches of the node.

2.3 Combination with the RHBD

In the pruning cache scheme, a coherence maintenance message is first sent to a set of processors which includes all sharing processors. If the set includes all processors, the message transfer is broadcast. This broad area transfer is performed only when pruning cache mis-hit occurs, and thus, such transfers do not occur frequently. However, for a large scale systems, relatively small number of broadcast can degrade the performance.

To address this problem, the *Reduced Hierarchical Bit-map Directory*(RHBD) used for a massively parallel processor JUMP-1[7] is introduced.

2.3.1 RHBD

In the RHBD, following two methods are applied to the hierarchical bit-map directory in order to reduce the required amount of memory for the directory.

- The same bit-map is used for the node at the same level.
- Broadcast is used at some nodes.

Since just one bit-map is used at each level of the tree, $m \times n$ bits memory is required for a block of m level n-ary tree.

One of the significant advantage of the RHBD is that the small sized directory entry can be carried in the coherence maintenance message itself. Thus, the message can multicast at the intermediate nodes just by referring directory in the message.

There are three sub-schemes of the RHBD scheme[15]: the LPRA (Local Precise Remote Approximate) scheme, the LARP (Local Approximate Remote Precise) scheme and the SM (Single Map) scheme.

Due to limitations of space, here we explain the SM scheme only. In the SM scheme, all nodes at a level use a unique bit-map generated by bit-wise 'OR' of each bit-map used in the level as shown in Figure 3, and thus, no broadcast is made (unless a bit-map is all-1).

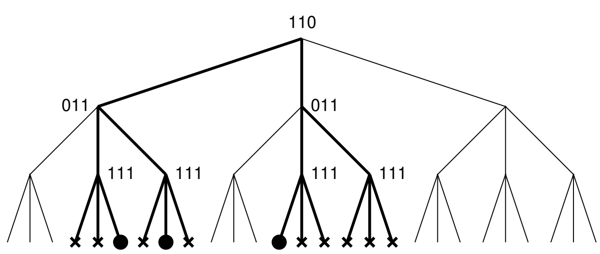


Figure 3: RHBD (SM) scheme

In this example, bit-maps (`110`,`011`,`111`) are used of each level of the hierarchy respectively. Note that the same bit-map is used for all nodes at same level. In the original hierarchical directory scheme, bit-maps `001`,`010` and `100` are used in the lowest level. Therefore, `111` which is the logical OR of the bit-maps is used for the nodes of the level in the SM scheme.

2.3.2 The combination of the pruning cache

Here, the combination of pruning cache and the SM scheme is discussed as an example. First, an invalidation/update message is sent according to the SM scheme as shown in Figure 3. At that time, acknowledge/not-acknowledge message which indicates whether the message is required or not is returned. These messages are collected in the upper hierarchy, and the bit-map is registered to the pruning cache as shown in Figure 4. The disadvantage of the pruning cache is that a large number of unnecessary packets are required when mis-hit occurs and the entry of the pruning cache is structured. Using the RHBD,

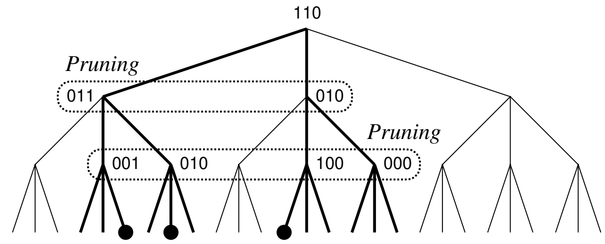


Figure 4: Combination of the pruning cache and the RHBD

the number of unnecessary packets are drastically reduced compared to broadcasting when mis-hit occurs.

On the other hand, the problem of the RHBD is that a large number of unnecessary packets is always sent if the node which share the block is not well located. If the hit ratio of the Pruning Cache is high enough, most of unnecessary packets can be reduced.

Therefore, with the combination of two methods, the problem of each method can be solved.

3 Evaluation

3.1 Evaluation with trace data

Like common cache memory, the effect of pruning cache completely depends on the possibility that the bit-map in the cache is used for pruning. Thus, the hit ratio of the pruning cache is evaluated first through the trace driven simulation.

Four benchmark programs (FFT, LU, Ocean and Radix) in SPLASH2[9] are simulated with an instruction level simulator ISIS[14] for obtaining the address trace. Then, pruning cache based on a simple 4-ary tree is simulated. The number of processors is set to be 64 from the limitation of the huge trace data.

The page size and line size are set to be 4K bytes and 32 bytes, respectively. The direct map and 2-way/4-way associative caches with 16, 32 and 64 entries are evaluated. The LRU replacement algorithm is used in this simulation.

3.1.1 Properties of generating messages

First, the number of messages, the number of destinations, and the number of messages which must be transferred using the RHBD scheme without pruning cache are analyzed(Table 1).

Table 1: Properties of messages

	Messages	Destinations	Packets
FFT	4028482	110154548	144791521
LU	4304719	66863862	134106632
Ocean	53399253	957170367	2696172050
Radix	296035	11500186	14738705

This table shows that from 50% to 200% unnecessary messages must be transferred using the RHBD without pruning cache.

3.1.2 The hit ratio

In order to avoid the large latency for accessing memory outside the chip, pruning cache is assumed to be implemented inside the router chip. In this case, it is difficult to provides the cache with a large number of entry. Here, the hit ratio of the pruning cache with 16, 32 and entries are analyzed(Table 2). The number to way is set to be 1 (direct map), 2 and 4.

Although the hit ratio is only about 60% with 16 entries, it is much improved when the number of entries exceeds 32. With 64 entries, more than 90% hit ratio is achieved in most cases. While the difference between direct map and 2 set associative is not negligible, 4 way set associative does not improve the performance so much especially with 32 or 64 entries. Considering the requirement of hardware, 2 way set associative cache with 32 entries is efficient structure of the pruning cache.

3.2 Evaluation on a practical implementation

Here, hardware requirement and network congestion of the pruning cache with the RHBD are evaluated on a practical implementation. Although the pruning cache with the RHBD can be implemented on any network with hierarchical structure, congestion around the root will bottleneck the system with the simple tree network. Thus, a hierarchical network with multiple roots like the fat-tree is suitable for practical implementation. Here, we introduce an implementation example of the pruning cache with the RHBD on a network called *Recursive Diagonal Torus*(RDT)[18, 17]. The RDT is used in the real massively parallel machine JUMP-1[7] and the router chip is already operational[6].

3.2.1 Interconnection network RDT

The RDT is a network consisting of recursively overlayed two-dimensional square diagonal tori. In order to reduce the diameter, bypass links are provided in the diagonal direction. When four links are added between node (x, y) and nodes $(x \pm n, y \pm n)$ (n: *cardinal number*) respectively, additional links result in a new torus-like network. New torus-like network is formed at an angle of 45 degrees to the original torus, and the grid size is $\sqrt{2}$n times that of the original torus(Figure 5).

We call this new torus-like network the rank-1 torus. On the rank-1 torus, we can form another torus-like network (rank-2 torus) by providing additional links

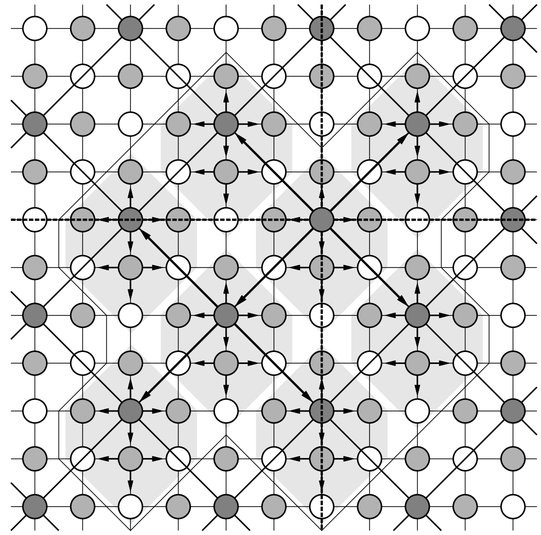


Figure 5: Recursive Diagonal Torus

in the same manner. The RDT consists of such recursively formed tori.

3.2.2 Octal tree on the RDT

Since the RDT consists of hierarchy of tori, the tree structure can easily be emulated. As shown in Figure 6, by multicasting twice from an node on the upper level (or coarse) torus, octal tree can be logically formed. The message is multicast to a hexagon-like area. We call this area *territory.*

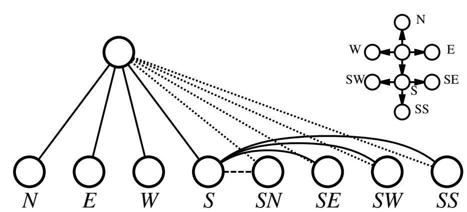


Figure 6: Octal tree on the RDT

The number of nodes in a territory depends on the level of the torus from which the multicast starts. The source node is always at the center of the territory. Figure 5 shows the case when multicast is started at rank-1 torus. That is, multicast from the second level of the octal tree. In this figure, the message is multicast to neighboring 64 nodes of the source node. Since RDT has multiple upper rank tori from which multicast can be started, it forms Fat-tree and thus no congestion at the root node occur.

3.2.3 Evaluation of latency

The problem of the pruning cache scheme is the possibility of network congestion with unnecessary message broadcast generated when the pruning cache does not hit. Here, the network congestion of a practical implementation with 256 nodes (16×16 nodes) as mentioned

Table 2: Hit ratio of Pruning Cache

	direct map			2 way set associative			4 way set associative		
	16	32	64	8×2	16×2	32×2	4×4	8×4	16×4
FFT	53.86%	84.12%	93.97%	57.74%	86.15%	96.38%	56.52%	88.30%	96.68%
LU	95.84%	99.75%	99.82%	96.64%	99.96%	99.99%	97.32%	99.97%	99.99%
Ocean	60.59%	94.96%	99.39%	66.35%	95.44%	99.72%	69.78%	99.39%	99.78%
Radix	60.24%	68.73%	74.47%	68.73%	75.61%	78.73%	73.76%	77.96%	79.56%

before is evaluated. Since the trace driven simulation is difficult for a large system with hundreds of nodes, the probabilistic model is used for data access while the behavior of the router and network congestion are simulated based on the design of the above described RDT router. In this router, a packet with various number of flits can be multicast in three clocks. Here, the length of packets are fixed to be 8 flits.

The hit ratio of the pruning cache is given as a parameter based on the above evaluation results with trace simulation. The source and the destination nodes of coherence maintenance messages are determined in random. However, to reflect the spatial locality of mapping, both X and Y projections of the distance between the source and a destination node are determined so as to follow normal curve of distribution of standard deviation 5. The number of destinations of a message is set to 6 according to analysis of the trace data[15].

Figure 7 show the average latency of a multicast message when the pruning cache is provided in the rank-0 torus. The number of destination (d) is set to be 4 and 6. For comparison, the latency when traditional directory management methods based on 1-to-1 message transfer ("1 to 1") is also shown.

The latency of the RHBD with pruning cache is from 30% to 80% better than that of the traditional method when interval of message is long. This shows the effect of multicast when the network is not congested. However, when the network is congested with frequent generated messages, the latency with the RHBD rapidly increases since the usable bandwidth of the network is degraded by unnecessary messages. On the contrary, the combination of pruning cache and the RHBD, the usable bandwidth is almost same as that of the traditional method. With the combination of both methods, the effect of multicasting overcomes the congestion caused by unnecessary packets.

By providing the pruining cache only in the lowest level, the hardware requirement can be reduced.

4 Related work

Similar concept is proposed by Scott and Goodman[13], applied to the k-ary n-cube, and evaluated theoretically. It is based on the simple broadcasting unlike our method based on the RHBD. Although there is no description on the detail of protocol management, Scott and Goodman's pruning cache is for a traditional CC-NUMA system with invalidation protocol and directory associated to the cache line. Thus, the background situation is also different.

Hanawa and his colleagues proposed similar mechanism for MIN structure. Their approach is only for a cache with invalidation protocol, and the entry of pruning cache is removed by every multicast of invalidation messages. Under this situation, the performance improvement with the pruning cache is severely limited.

5 Conclusion

The pruning cache directory method is proposed for dynamic hierarchical directory management on large scale CC-NUMA systems in which a node aggressively shares data with other nodes. By a combination with the RHBD method, dynamic directory is quickly formed and managed with a small additional hardware.

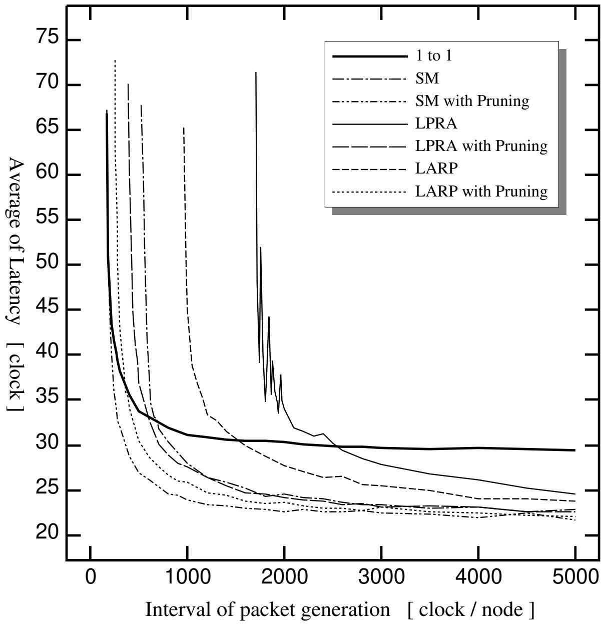
The result of a trace driven simulation of a system with 64 processors shows that more than 90% hit ratio can be obtained with a small 2-way associative pruning cache with 32 entries. From the probabilistic simulation of a large system, it appears that the combination of the pruning cache and the RHBD achieves better latency and bandwidth than those of traditional methods based on 1-to-1 message passing.

At the current stage of study, the evaluation of pruning cache is not enough since the precise simulation needs much time and computation power. In order to demonstrate the performance of the combination of the pruning cache and RHBD, extensive simulation studies are required as our future work for a system with larger number of processors using various applications.

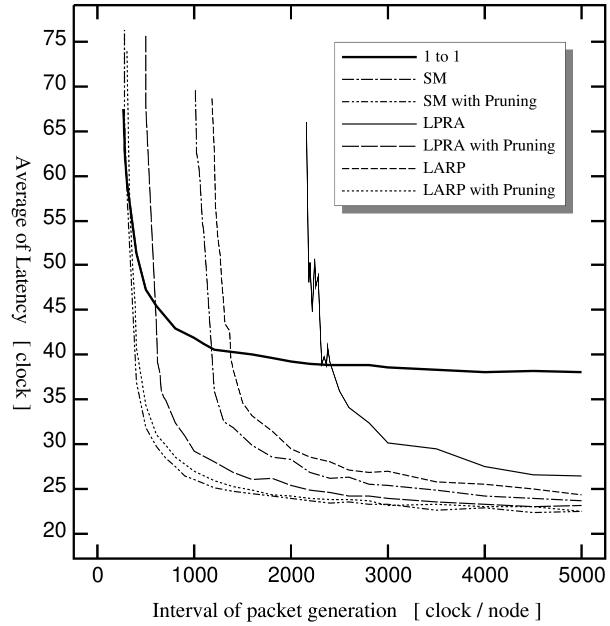
References

[1] A.Agarwal, et al. An Evaluation of Directory Schemes for Cache Coherence. In *Proc. 15th ISCA*, 1988, 280 – 289.

[2] D.Chaiken and A.Agarwal. Software-Extended Coherent Shared Memory: Performance and Cost. In *Proc. The 21st ISCA*, 1994, 314 – 324.



(Pruning cache is provided only in rank-0, d = 4)



(Pruning cache is provided only in rank-0, d = 6)

Figure 7: The latency vs. message generation frequency

[3] D.Chaiken, J.Kubiatowwicz, and A.Agarwal. LimitLESS Directories: A Scalable Cache Coherence Scheme. In *Proc. ASPLOS IV*, 1991, 224 – 234.

[4] D.Lenoski, et al. The Stanford DASH Multiprocessor. *IEEE Computer*, 1992, 25(3):63 – 79.

[5] D.V.James, et al. Distributed-Directory Scheme: Scalable Coherent Interface. *IEEE Computer*, 1990, 23(6):74 – 77.

[6] H.Nishi, et al. The JUMP-1 Router Chip: A versatile router for supporting a distributed shared memory. In *Proc. IPCCC '96*, 1996, 158 – 164.

[7] H.Tanaka, et al, editor. *The Massively Parallel Processing System JUMP-1.* Ohmu corp., 1996. ISBN4-274-90083-5.

[8] J.Kuskin, et al. The Stanford FLASH Multiprocessor. In *Proceedings. of The 21st International Symposium on Computer Architecture*, 1994, 302 – 313.

[9] J.P.Singh, et al. SPLASH: Stanford Parallel Applications for Shared-Memory. Tech. report, computer system laboratory, Stanford University, 1992.

[10] K.Nishimura, et al. Memory Coherency Contorl Schemes on the RDT Interconnection Network. *Trans. IPSJ*, 1996, 37(7):1367 – 1377. (in Japanese).

[11] M.D.Hill, J.R.Larus, and D.A.Wood. Tempest: A Substrrate for Portable Parallel Programs. In *Proc. COMPCON '95*, 1995, 327 – 332.

[12] M.Thapar and B.Delagi. Distributed-Directory Scheme: Stanford Distributed-Directory Protocol. *IEEE Computer*, 1990, 23(6):78 – 80.

[13] S.L.Scott and J.R.Goodman. Performance of Pruning-Cache Directories for Large-Scale Multiprocessors. *IEEE Trans. Parallel and Distributed Processing*, 1993, 4(5):520 – 534.

[14] T.Kisuki, et al. Shared vs. Snoop: Evaluation of Cache Structure for Single-chip Multiprocessors. In *Proc. Euro-Par'97 Parallel Processing*, 1997, 793 – 797.

[15] T.Kudoh, et al. Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1. In *Proc. 1995 ICPP*, 1995, I–186 – I–193.

[16] T.Matsumoto and K.Hiraki. Distributed shared-memory architecture using memory-based processors. In *Proc. JSPP'93*, 1993, 245 – 252. (in Japanese).

[17] Y.Yang and H.Amano. Message Transfer Algorithms on the Recursive Diagonal Tours. *IEICE Transaction on Information and Systems*, 1996, E79-D(2):107 – 116.

[18] Y.Yang, et al. Recursive diagonal torus: An interconnection network for massively parallel computers. In *Proc. 1993 SPDP*, 1993, 591 – 594.