Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1

Tomohiro Kudoh† Hideharu Amano‡ Takashi Matsumoto* Kei Hiraki*
Yulu Yang ‡ Katsunobu Nishimura‡ Koichi Yoshimura† Yasuhito Fukushima†

†Tokyo Engineering University
‡Keio University
*The University of Tokyo

Abstract

JUMP-1 is currently under development by seven Japanese universities to establish techniques of an efficient distributed shared memory on a massively parallel processor. It provides a memory coherency control scheme called the *hierarchical bit-map directory* to achieve cost effective and high performance management of the cache memory. Messages for maintaining cache coherency are transferred through a fat tree on the RDT(Recursive Diagonal Torus) interconnection network. In this report, we discuss on the scheme and examine its performance. The configuration of the RDT router chip is also discussed.

1 Introduction

JUMP-1 is a massively parallel processor prototype developed by collaboration between 7 Japanese universities[4]. The major goal of this project is to establish techniques for building an efficient distributed shared memory on a massively parallel processor. For this purpose, a sophisticated methodology called Strategic Memory System (SMS) is proposed [11][4].

The cache directory and management scheme is one of the most important issues in the SMS. In traditional distributed memory systems, a directory entry is associated with every cache line. However, this requires a large amount of memory for a massively parallel processor which provides both a large number of processors and a large amount of memory space. In the SMS, each node processor shares a global virtual address space with two-stage TLB implementation, and the directory is attached not to every cache line but to every page, while the data transfer is performed by a cache line.

For the directory entry associated with each page, a hierarchical bit-map directory scheme was introduced[10][11]. In addition, to reduce the size of directory, a directory reduction scheme has been proposed. However, the original scheme [1] is built based on a simple n-ary tree network, and some extensions are required for the use in JUMP-1.

[1]The original scheme[10] is called the pseudo-fullmap directory scheme consisting of the hierarchical multicast(directory) scheme described here, local, and 1-to-1 communications.

In this paper, we extend the original hierarchical bit-map directory scheme to apply on an interconnection network RDT(Recursive Diagonal Torus) which is proposed for JUMP-1. In addition to the original directory reduction scheme, two other schemes are also proposed. In Section 2, JUMP-1 and its interconnection network RDT are briefly introduced. In Section 3, the hierarchical bit-map directory scheme and the directory reduction schemes are introduced. In Section 4, the implementation of the hierarchical bit-map directory schemes on the RDT is described. These schemes are compared with other directory schemes and the number of redundant packets are evaluated in Section 5. Finally, the implementation of the RDT router chip which enables hierarchical bit-map directory schemes are described.

2 JUMP-1 and the RDT

2.1 Structure of JUMP-1

As shown in Figure 1[4], JUMP-1 consists of clusters connected with an interconnection network RDT[14]. Each cluster also provides a high speed point to point I/O network connected with disks and high-definition video devices.

Each cluster is a bus-connected multiprocessor (Figure 2[4]) including 4 coarse-grained processors(CPU), 2 fine-grained processors (Memory Based Processor or MBP) each of which is directly connected to a main

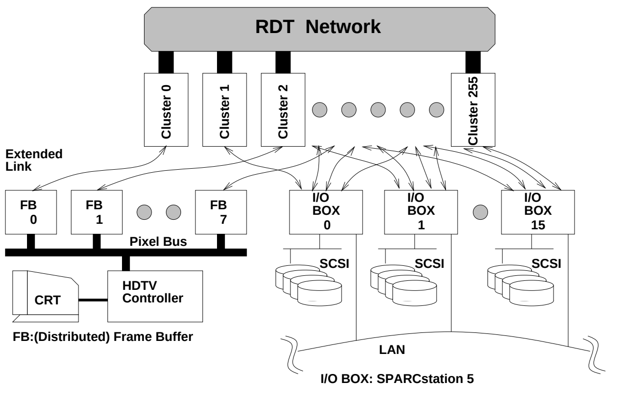

Figure 1: Structure of JUMP-1

memory and the RDT router chip. A CPU is an off the shelf RISC processor (SUN Super-Sparc+) which performs the main calculation of the program. The MBP, the heart of JUMP-1, is a custom designed fine-grained processor which manages the distributed shared memory, synchronization, and packet handling. The first prototype of JUMP-1 which is scheduled to be available in this winter provides 256 clusters, thus, 1024 processors.

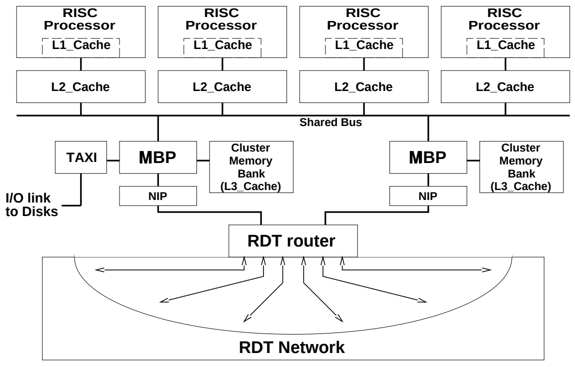

Figure 2: Structure of a JUMP-1 cluster

2.2 Interconnection network RDT

The RDT is a network consists of recursively formed two-dimensional square diagonal toruses. In order to reduce the diameter, preparing bypass links for the diagonal direction is the best way for the torus network. Assume that four links are added between a node (x, y) and nodes $(x \pm n, y \pm n)$. Here, n is called the *cardinal number.* Then, the additional links form a new torus-like network. The direction of the new torus-like network is at an angle of 45 degrees to the original torus, and the grid size is $\sqrt{2}$n times of the original torus. Here, we call the torus-like network the rank-1 torus. On the rank-1 torus, we can make another torus-like network (rank-2 torus) by providing four links in the same manner. Figure 3 shows rank-1 and rank-2 toruses when n is set to be 2. The RDT consists of such recursively formed toruses.

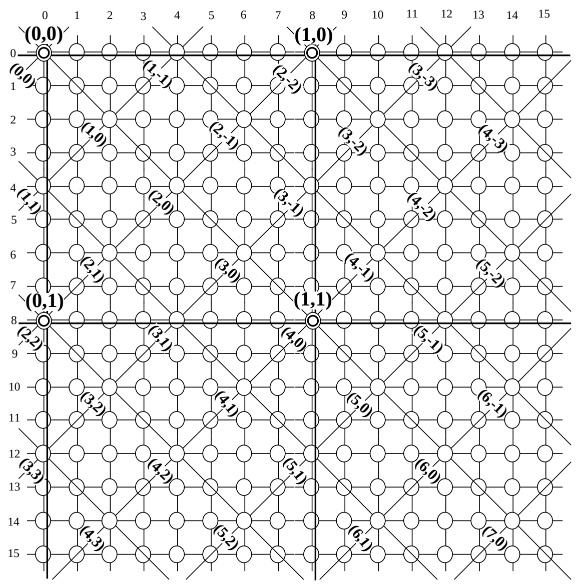

Figure 3: Upper rank toruses

Recursive Diagonal Torus RDT(n,R,m) can be defined as a class of networks in which each node has links to form base (rank-0) torus and m upper toruses (the maximum rank is R) with the cardinal number n. Note that, each node can select different rank of upper toruses from others.

The RDT in which every node has links to form all possible upper toruses is called the perfect RDT (PRDT(n,R)) where n is the cardinal number (usually, 2) and R is the maximum rank. Although PRDT is unrealistic because of its large degree (4(R+1)), it is important as a basis for establishing routing algorithm, broadcasting/multicasting, and other message transfer algorithms.

The JUMP1 must be scalable to the system with ten thousand nodes. In this case, m is set to be 1 (degree = 8). For this number of nodes, the maximum rank of upper toruses is 4. Thus, the RDT(2,4,1) is treated here.

In the RDT, each node can select different rank toruses from others. Thus, the structure of the RDT(2,4,1) also varies with the rank of toruses which are assigned to each node. This assignment is called the *torus assignment.* Various torus assignment strategies can be selected considering the traffic of the network. If the local traffic is large, the number of nodes which have low ranks should be increased. However, complicated torus assignment introduces difficulty to the message routing algorithm and implementation. For the JUMP-1, we selected a relatively simple torus assignment shown in Figure 4.

In this assignment, a node has eight links, four for the base (rank-0) torus and four for rank (1-4) torus (Most of links for upper rank toruses are omitted in Figure 4). Note that a node with a rank torus has neighboring

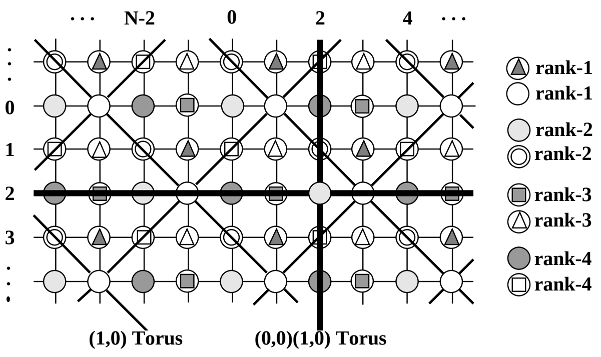

Figure 4: Torus assignment used in the JUMP-1

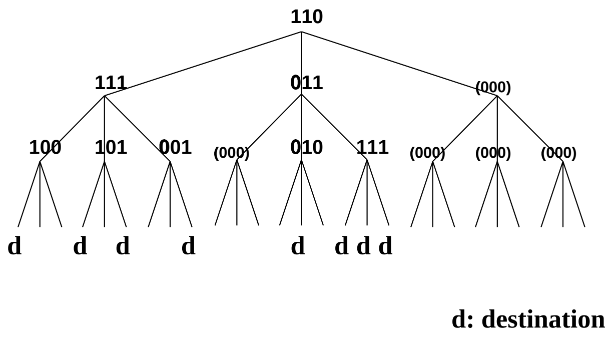

Figure 5: Hierarchical bit-map directory scheme

nodes with toruses of other three ranks. Therefore, any rank torus can be used with a single message transfer between neighboring nodes. This property reduces the diameter and average distance between nodes.

The RDT provides various advantages as an interconnection network of the JUMP-1. It includes two dimensional mesh, and simple near-optimal routing algorithm enables smaller diameter than that of the hypercube (11 for 2^{16} nodes) with smaller degree (8 links per node). Moreover, it includes the fat-tree of mesh structure. Using this structure, hierarchical multicast can be efficiently supported. This feature is important to implement the hierarchical bit-map directory scheme treated here.

3 Hierarchical bit-map directory scheme

Most of conventional non-bus based shared memory multiprocessors equip a cache directory whose entries are associated with cache lines. However, in a massively parallel machine both with a large number of processors and a large address space, a large amount of memory required for the directory will be unacceptable. In JUMP-1, directory entries are associated with pages while the data are transferred by a cache line[11] in order to reduce the required amount of memory for the directory. Using this strategy, both the expansion of the directory memory and the congestion of the network caused by the large size message transfer can be avoided.

However, in this case, number of destinations of the coherence maintenance messages increases especially when an update type protocol is used. If there are considerable number of destinations, it will take a long time to send a message if they are sent one after another(sequentially).

To cope with this problem, the hierarchical bit-map directory scheme transfers messages for different destinations simultaneously (i.e. multicast) using a tree structured multicasting paths (multicasting tree).

Figure 5 illustrates the concept of hierarchical bit-map directory scheme. Leaves of the tree correspond to the clusters of JUMP-1 and a message is first sent from the root of the tree. Each node can multicast a message to its multiple branches at a time. To specify the destination leaves precisely, an n-bit bit-map is required for each node when an n-ary tree is used. Thus, for m height n-ary tree, total of $\sum_{k=1}^{m} n^k$bits are required for each entry. Although the required amount of memory is larger than that of the full-map directory scheme in which n^m bits are required (since there are n^m leaves), the message can be multicast using the tree structured path in the hierarchical bit-map directory scheme.

3.1 Reduction of the directory length

Since JUMP-1 is a massively parallel processor, the $\sum_{k=1}^{m} n^k$ bits directory entry (for m height n-ary tree) is not feasible. Two ways of reducing the size of the directory can be considered.

1. The bit-maps are stored and maintained at the node. When a message comes from the upper level, a bit-map stored at the node is accessed according to the address (tag) of the message.

2. The bit-maps for a message are attached to the message header and referred at the appropriate nodes of the tree.

While the former scheme effectively reduces the size of directories stored in the entire system, the overhead caused by the accesses of the bit-maps will prevent quick cache coherent management. Thus, the latter scheme is mainly adopted for JUMP-1. However, in this method, a large size of header is required if bit-maps for all level of hierarchy are attached.

In order to reduce the size of bit-maps, a bit-map is provided not for each node, but for each level of the tree. Each node multicasts packets to its children either according to the bit-map for the level or all children of the node (thus, broadcasting).

Here, the LPRA which is the original scheme proposed in [10][11], and two novel schemes, the SM and LARP are introduced.

LPRA scheme: When the multicast is started, the message is sent from the source cluster to the root

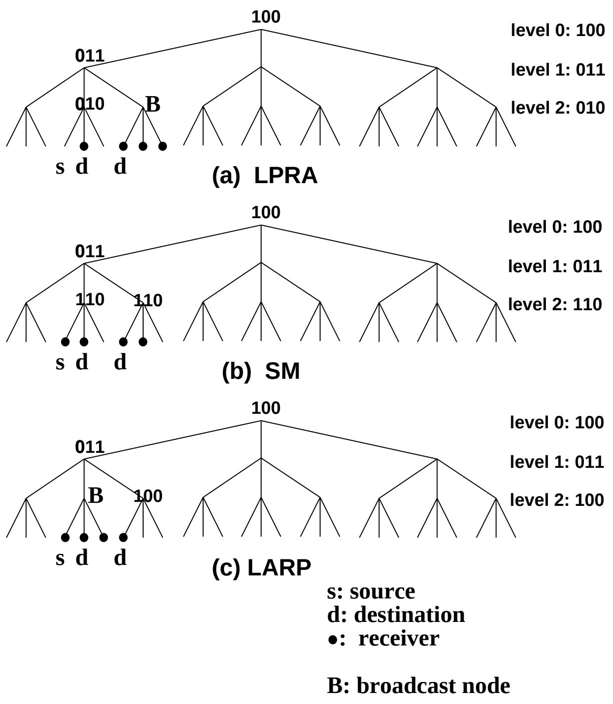

Figure 6: Hierarchical bit-map directory schemes

of the multicast tree. In the LPRA (Local Precise Remote Approximate) scheme, the bit-map is used only nodes which is the root of the subtree including the source node. For nodes in other subtrees, the message is broadcast to all children. Figure 6(a) shows an example of this scheme for the 3-ary tree. In this figure, 's' is the source cluster, and 'd' is the destination to which packet is sent. • indicates clusters which receive data. It is desirable if the number of clusters which have • but without 'd' is small as possible. Using this scheme, the bit-map is used for local nodes of the source node, while the message is broadcast in the remote subtree marked **B**. This scheme is advantageous when a node send a message to a remote node group.

SM scheme: In the SM(Single Map) scheme, all nodes at a level use a unique bit-map as shown in Figure 6(b), and thus, no broadcast is made (unless a bit-map is all-1). This scheme is advantageous when the number of destination is not so large.

LARP scheme: The LARP is a complimentary scheme of the LPRA. In this scheme, broadcast is done at nodes which are the root of the subtree including the source node while a unique bit-map is used in other subtrees like the SM scheme. Note that the broadcast is started from children nodes whose parent starts the multicasting. In the example shown in Figure 6(c), since the multicast starts at the level 1 (The top node (level 0) only sends a message to a child.), the broadcast (marked **B**) starts at the level 2 to the subtree which includes the source node. In this case, the broadcast is done for local nodes of the source node, while the bit-map is used for remote nodes. This scheme is advantageous when the mapping can make the best use of the locality of communication.

4 Fat tree on the RDT

In the original hierarchical bit-map directory scheme, messages are transferred on a tree structured communication paths. However, if a simple n-ary tree is used, following problems will bottleneck the performance.

- If multiple nodes multicast simultaneously, congestion may be caused around the root node of the tree.
- Depending on the location in the tree, some messages should have transferred through the very root of the tree just to move to neighboring node.
- It takes a large latency if a sender node at a leaf sends a message upstream to the root of tree.

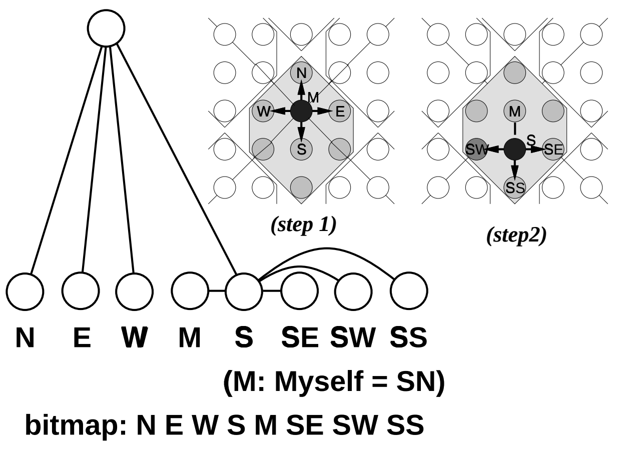

Figure 7: The 8-ary tree and bit-map pattern for multicast on the RDT

However, since the RDT involves a fat tree of toruses with multiple root nodes, these problems can be avoided. The pattern of message transfers for emulating a fat tree is shown in Figure 7. Two steps are required: (1) each node transfers a message to four neighbors, (2) a neighbor transfers the message to three neighbors. (South direction in this figure). As shown in Figure 7(b), the eight nodes which received the data does not duplicated with node which received other nodes (marked X in Figure 7(a)) on the same rank-i torus. Thus, if all nodes with rank-i toruses executes this pattern, the message is transferred to all nodes with rank-(i-1) toruses. By the iteration of this data transfer from the maximum rank to the rank-0, 8-ary tree in which is formed on the RDT. In this case, a rank in the RDT directly corresponds to the level of the tree. Moreover, in the RDT, the upper rank torus can be used within a step of message routing. Thus, the message can be directly transferred from the sender node to the root node without using the tree structure.

Figure 7 shows the 8-ary tree involved in the RDT, and bit-map pattern for the hierarchical bit-map directory scheme. When schemes described in the previous section are applied on this 8-ary tree on the RDT, followings are advantageous:

- On the RDT, there are a number of nodes with the maximum rank torus, and all of them are used as a root node. Thus, the message multicast is almost directly started from the root node without going upstream on the tree.

- Since there are many upper rank toruses in the RDT, the tree is a kind of "fat-tree" which provides many root nodes in each rank. Therefore, the congestion of root nodes is relaxed even if many source nodes multicast their data simultaneously and independently.

- In the RDT, nodes which receive the message through the tree whose root rank is 'i' are located around the source node. For larger 'i', the number of such nodes becomes large, thus the area which a message is multicast becomes wide. We call such an area "territory" of a multicast. Figure 8 shows territories of a multicast from rank-0 and rank-1. Since the territory of is always formed surrounding a source node, message multicast to local nodes are performed from a lower rank (thus, with only a small territory).

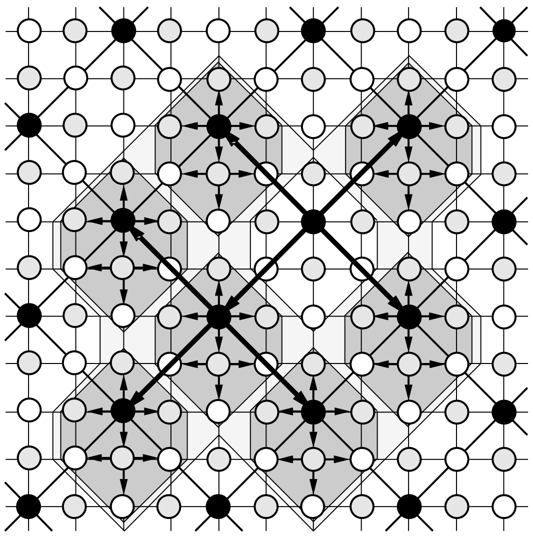

Figure 8: Territory of a multicast

5 Discussions

5.1 Comparison with other schemes

Three famous directory schemes for managing cache have been tried in multiprocessors with distributed shared memory. In the full map directory scheme which is used in Stanford DASH[2], each entry of the directory holds a bit map corresponding to each node of the multiprocessor. This scheme can be used only in a small system since the required bit map for each entry is equal to the number of processors.

In the limited pointer scheme[1] adopted in MIT Alewife[6], each entry holds a limited number of pointers each of which indicates the processor which has the copy of the line. Usually, the number of the pointers is limited in two or three. If the number of the copy is beyond to the number of pointers, the information is broadcast to every node (broadcast) or copies exceed the number of the pointers are invalidated (eviction).

In another scheme, the chained directory scheme[3][7] which will be used in Stanford FLASH[5], a chained list is used for holding node identifiers.

Both the limited pointer and chained directory will work efficiently even in massively parallel processors if the number of nodes which hold the copy of the same cache line is small. In the limited pointer, if the number of copies is large, it requires a large number of pointers, or unnecessary broadcast or invalidation is required. In the chained directory, it takes a long time to access the chained directory if the number of nodes which hold the copy is large. Through the simulation study, it appears that the number of nodes which receive the invalidate messages are one or two in most cases[12]. In this case, both schemes will work efficiently.

However, this simulation is done under the following conditions: (1) the entry of the directory is associated with the cache line, and (2) only invalidation type protocols are used. In JUMP-1, the entry of the directory is not associated with the cache line but with the page. This strategy much reduces the required memory for a directory of a large size of shared memory space that is essential for massively parallel processors. Moreover, in JUMP-1, update type protocols which are advantageous in most scientific calculations can be used with invalidation type protocols. In this case, as shown in our initial estimations, the number of copies is further increased.

Figure 9 shows the distribution of the number of destinations of invalidation/update messages when an invalidation type protocol and an update type protocol are used respectively. The bold line shows results when an entry is associated with a cache line, and the doted line shows ones with a page. Here, MP3D with 1024 molecules and water with 64 molecules from the SPLASH[8] parallel programs for shared memory is executed with 256 processors and 32 processors respectively. This result is generated from an address trace[9] when the size of a cache line is 32 byte and the size of a page is 4 Kbytes. The X axis corresponds to the number of destination processors in logarithmic scale, and the Y axis corresponds to the occurrence times.

While the bold line (entries are associated to cache

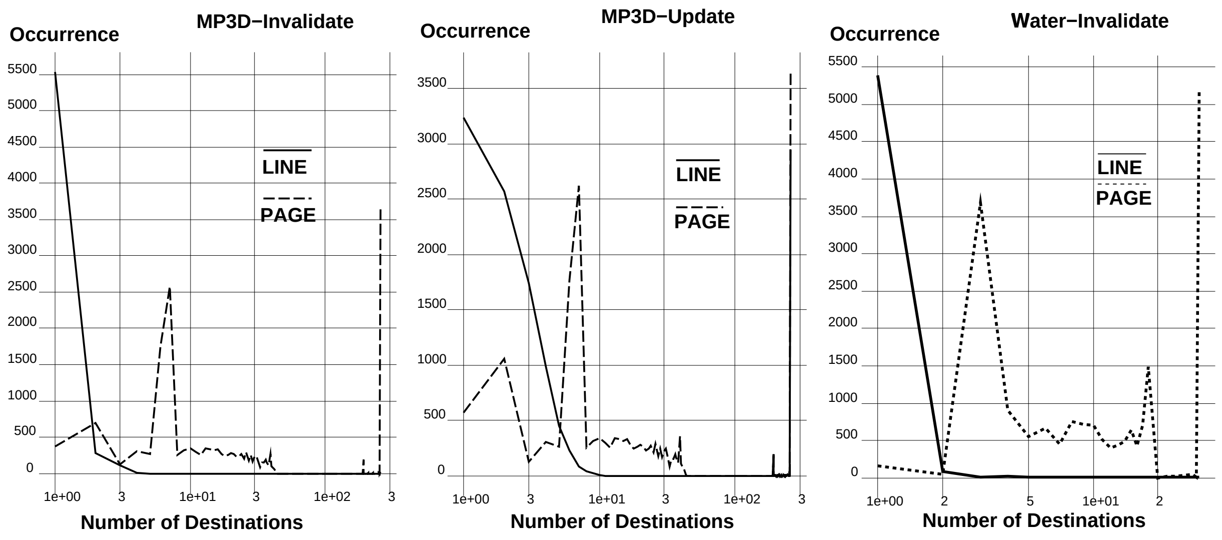

Figure 9: Distribution of number of destinations

lines) in the invalidation protocol demonstrates the small number of copies in the traditional conditions, other lines show that the number of copies exceeds three. When an entry is associated with the page, two peaks are shown. One is at 4-8 nodes and another is at all nodes in both protocols.

When 6 pointers are required in the limited pointers, the length of each directory entry becomes $6 \times 14 = 84bit$ when $2^{14} = 16K$ nodes are used. For the same size, the hierarchical directory schemes treated here require only $4(levels) \times 8(number\ of\ branch) = 32bits$. When the size of a cache line is 32 byte and the size of a page is 4 Kbyte, the total amount of required memory can be reduced to 1/128 at maximum. Although every cache of a page is not used in practice, usually the usage ratio is quite high because of the locality of reference. This demonstrates that the required memory is much reduced in the hierarchical directory schemes. Although the chained directory scheme also can support a small cost of memory, the access time for following the directory may degrade the performance.

5.2 The number of unnecessary messages

The major disadvantage of hierarchical directory schemes is that it requires unnecessary message multicast since it only use a single multicast or broadcasting in each level of the hierarchy. If destination nodes can be mapped inside the territory, the number of unnecessary messages can be drastically reduced. However, it depends not only on the mapping strategy but also on the characteristics of application programs. Here, we estimate the number of unnecessary messages with a simple probabilistic models as the first evaluation.

The destinations are determined by a random numbers which follow a given distribution. In the following graphs, average value calculated from 10,000 trials based on the random number generation library of Sun OS4.1.3 are shown.

Figure 10 shows number of receiving clusters when the destinations (D) follow a normal distribution of variance 1 (here, unit is a link on the base torus and the X and Y projections of the distance from the source cluster to the receiving cluster follow the distribution independently). This result represents the case which the application program has a strong locality of communication and well mapped. In this case, the number of receiving clusters is 80 to 180 even when the number of destinations are 32, thus unnecessary messages are not so many. The LARP is advantageous when the number of destinations is large.

Figure 11 shows when the distribution (D) is 5 and other parameters are unchanged. This case represents that there are a lot of destination nodes which are not local of the source node. The number of receiving clusters is considerably large, thus, many unnecessary messages are generated. The SM is advantageous when the number of receivers is small, and the LARP is advantageous when it is large. This tendency is also among other graphs, and the number of clusters when the performance of the LARP overcomes that of the SM is about 10. Although the LPRA cannot reduce the number of clusters compared with other schemes in these estimation, it will be useful when there are group of destination nodes distant from a source node.

As described earlier, when a directory entry is associated with a page, the number of destinations is likely to be at least 4. Figure 12 shows the number of average receivers when the number of destination is four and the variance is changed. the SM always yield least receivers, and the number is not more than around 120 even when the variance is large.

These results suggest that the hierarchical bit-map directory scheme requires a lot of unnecessary messages when the destination processes cannot be mapped into local processors of the source nodes. However, if processes which communicates each other can be locally

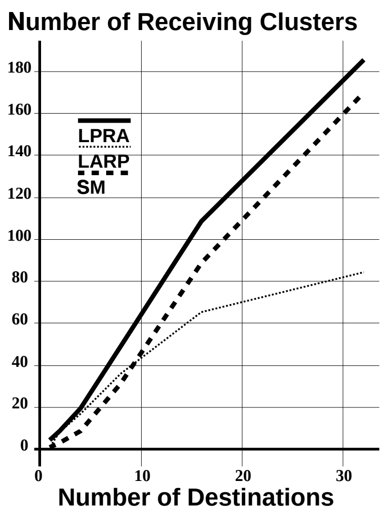

Figure 10: Number of receiving nodes vs. number of destinations (D =1)

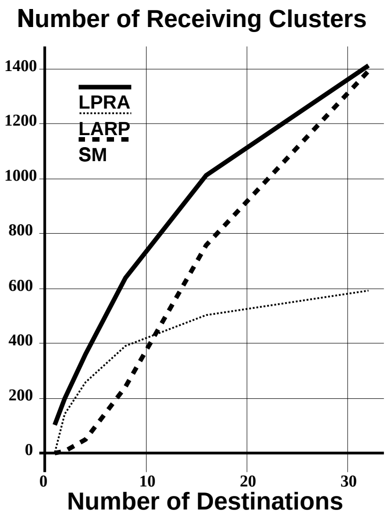

Figure 11: Number of receiving nodes vs. number of destinations (D =5)

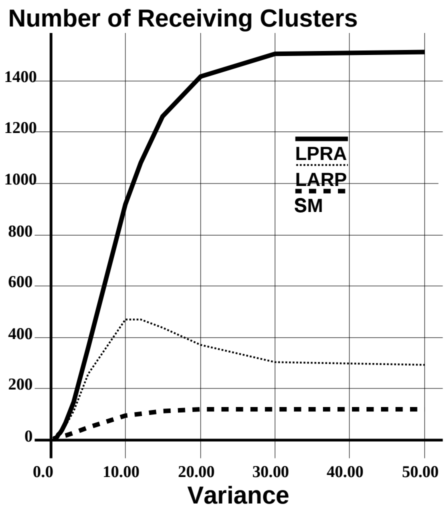

Figure 12: Number of receiving clusters vs Variance

mapped, the number of unnecessary messages can be not so large with using the SM, LPRA and the LARP appropriately.

6 The RDT router chip

An LSI router chip which supports all hierarchical bit-map directory schemes discussed here has been implemented for JUMP-1. The structure of this RDT router chip is shown in Figure 13. The core of the chip is a 10×11 crossbar which exchanges packets from/to 10 18-bits-width links, that is, four for the rank-0 torus, four for the upper rank torus, and two for the MBPs which manage the distributed shared memory of JUMP-1. In JUMP-1, two RDT router chips are used in the bit-sliced mode to form 36 bits width for each link.

All packets are transferred between router chips synchronized with a unique 60MHz clock. In order to maximize the utilization of a link, packets are bi-directionally transferred. Maximum packets length is 16flits (36 bits-width 16flits-length) so as to carry a line of the cache. 3 flits header which carries the bit-map of the hierarchical bit-map are attached to all packets, but the length of the body is variable.

The virtual cut-through flow control is adopted to cope with frequent multicasting. Two packet buffers each of which can hold the maximum size packet form two virtual channels. In the RDT, the deadlock is avoided with the modified e-cube routing method using two virtual channels [13]. Using this technique, every 1-to-1 packet transfer or multicast is performed without deadlock. Since the route of the multicast is fixed, the FIFO assumption is ensured.

Two bits in the packet header are used for selection of three hierarchical bit-map directory schemes, the LPRA, SM and LARP. Thus, three schemes are selectable for each packet, and the mixture of packet with different schemes are allowed. Moreover, the packet is forced to transfer to the MBP at any node of each hierarchy that multicast is started. In this case, the MBP re-makes the new bit-map for each hierarchy, and start multicast again. Although this scheme requires the latency to re-make the bit-map, unnecessary packets multicast can be reduced.

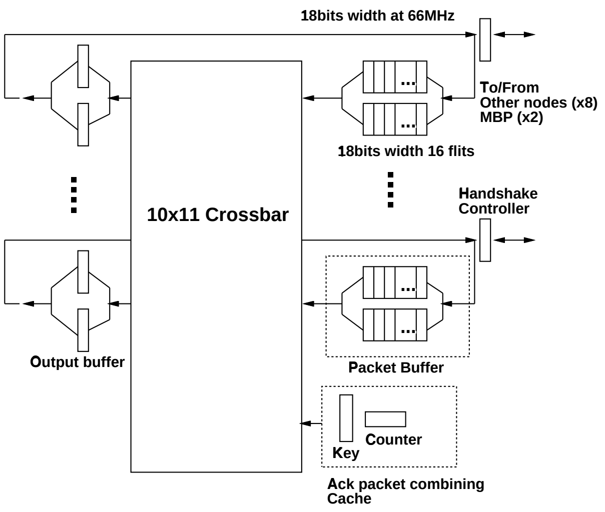

Figure 13: The structure of the RDT router

$0.5\mu m$ Hitachi BiCMOS SOG which provides 125K gates in maximum is utilized. Lines are directly driven with ECL interface of this chip. Using the dual port RAM, packet buffers allow to push and pull a flit of the packet simultaneously. The required number of gates are 90522 gates. Random logics require 50000 gates in total while areas corresponding to about 4000 gates are required for the dual-port RAM. The crossbar body and arbiter, which are simple but high performance is required, are designed in the logic level, while the complicated controllers are described in the VHDL.

7 Conclusion

Hierarchical bit-map directory schemes on the Recursive Diagonal Torus are proposed and discussed. Through a simple estimation, a small memory requirement of these schemes is demonstrated when a directory entry is associated with a page or update protocols are used on massively parallel processors. The problem of these schemes is congestion of the network with unnecessary multicast packets. Since the evaluation shown here is based on a simple probabilistic model, precise evaluations under practical conditions are required. However, it is difficult with simulations as a large number of processors must be simulated with practical application programs and mapping strategies.

A high speed RDT router which supports all schemes discussed here is available. Using these chips, JUMP-1 will start its operation on the next spring. A precise evaluation of these schemes will be done on this prototype.

Acknowledgment

The authors would like to express their sincere gratitude to the members of the Joint-University project for their valuable advice and discussion. The authors also express their thanks to Professor Takuya Terasawa, Mr. Osamu Gotoh, Mr. Kazuhito Kanda, and Mr. Hiromitsu Ueda of Tokyo Engineering University for supporting the environment of the multiprocessor instruction level simulator MILL.

A part of this research was supported by the Grant-in-Aid for Scientific Research on Priority Areas, #04235130, from the Ministry of Education, Science and Culture.

References

[1] Agarwal A., Simoni R., Hennessy J., and Horowitz M. An evaluation of directory schemes for cache coherence. In *Proc. of the 15th Annual International Symposium on Computer Architecture*, 1988.

[2] D.Lenoski, J.Laudon, K.Gharachorloo, W.-D.Weber, A.Gupta, J.Hennessy, M.Horowitz, and M.S.Lam. The stanford dash multiprocessor. *IEEE Computer*, 1992.

[3] James D.V., Laundrie A. T., Gjessing S., and Sohi G. S. Distributed-directory scheme: Scalable coherent interface. *IEEE Computer*, 1990.

[4] K. Hiraki, Hideharu Amano, Morihiro Kuga, Toshinori Sueyoshi, Tomohiro Kudoh, Hiroshi Nakashima, Hironori Nakajo, Hideo Matsuda, Takashi Matsumoto, and Shin ichiro Mori. Overview of the jump-1, an mpp prototype for general-purpose parallel computations. In *Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'94)*, 1994.

[5] J.Kuskin, D.Ofelt, M.Heinrich, J.Heinlein, R.Simoni, K.Gharachorloo, J.Chapin, D.Nakahira, J.Baxter, M.Horowitz, A.Gupta, M.Rosenblum, and J.Hennessy. The stanford flash multiprocessor. In *Proc. of the 21st Annual International Symposium on Computer Architecture*, 1994.

[6] K.Kurihara, D.Chaiken, and A.Agarwal. Latency tolerance through multithreading in large-scale multiprocessors. In *Proc. of International Symposium on Shared Memory Multiprocessing (ISSMM)*, 1991.

[7] Thapar M. and Delagi B. Distributed-directory scheme: Stanford distributed-directory protocol. *IEEE Computer*, 1990.

[8] J.P. Singh, W. Weber, and A. Gupta. Splash: Stanford parallel applications for shared-memory. In *Tech. Report, Computer System Laboratory, Stanford University*, 1992.

[9] T. Terasawa and H. Amano. Performance evaluation of the mixed-protocol caches with instruction level multiprocessor simulator. In *Proc. of IASTED International Conference of MODELLING AND SIMULATION*, 1994.

[10] T.Matsumoto and K.Hiraki. A shared-memory architecture for massively parallel computer systems. In *IEICE Japan SIG Reports, Vol. 92, No. 173, CPSY 92-26 (in Japanese)*, 1992.

[11] T.Matsumoto and K.Hiraki. Distributed shared-memory architecture using memory-based processors. In *Proc. of Joint Symp. on Parallel Processing'93 (in Japanese)*, 1993.

[12] W.D.Weber and A.Gupta. Analysis of cache invalidation patterns in microprocessors. In *Proc. of ASPLOS III*, 1989.

[13] Y. Yang and H. Amano. Message transfer algorithms on the recursive diagonal torus. In *Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'94)*, 1994.

[14] Y. Yang, H. Amano, H. Shibamura, and T.Sueyoshi. Recursive diagonal torus: An interconnection network for massively parallel computers. In *Proc. of 1993 IEEE Symposium on Parallel and Distributed Processing*, 1993.