The Effectiveness of Combining in Reducing Hot-Spot Contention in Hypercube Multicomputers

Sivarama P. Dandamudi
School of Computer Science
Carleton University
Ottawa, Ontario K1S 5B6
Canada

Derek L. Eager
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N 0W0
Canada

Abstract
Hot-spot contention has been studied previously in the context of shared-memory multiprocessor systems. Two techniques have been proposed for shared-memory multiprocessor systems to increase the system capacity in handling hot-spot references: *hardware combining* and *software combining*. The effectiveness of the two corresponding approaches for multicomputer systems is studied here, for two hypercube-based static interconnection networks, using a combination of simulation and analytical models. Our results suggest that software combining provides greater performance improvements under the realistic constraints of limited buffer storage.

1. Introduction

In large parallel processing systems there is a possibility of many processors requesting access to the same data item, or "service" by the same processor, at the same time. Such hot-spot contention creates congestion in the system. In shared-memory multiprocessor systems, the memory module containing the "hot-spot" data item may become saturated. In distributed-memory multicomputer systems, communication links leading to the "hot-spot" processor may become saturated.

Hot-spot contention has been studied previously in the context of multiprocessor systems that utilize buffered multistage interconnection networks to connect processors to the shared memory modules [5, 7, 8]. It has been observed that the access times for all memory references may be severely degraded, not just the references to a hot-spot location, owing to a phenomenon termed *tree saturation* [7]. In tree saturation, all switches within a "tree" rooted at the hot memory and extending to the switches connected to the processors become "saturated" in that buffer queues in these switches fill to capacity. This degrades overall performance, particularly in large systems.

Two schemes have been previously proposed to increase the capacity of a system in handling concentrated references to a hot-spot -- *hardware combining* and *software combining*. Both these schemes are based on essentially the same underlying principle of combining several hot-spot requests into a single request and forwarding this request to the hot-spot memory. These two schemes, however, differ in the way this combining is accomplished.

This paper is concerned with hot-spot contention in distributed-memory multicomputer systems. Processors in such systems communicate by explicit message passing. It is easy to see that hot-spots can also exist in multicomputer systems. For example, consider database operations in a multicomputer system. In a shared-nothing system each node consists of a processor, memory and a disk drive. To exploit the parallel I/O capability in such a system, relations are horizontally partitioned into disjoint subsets across all disk drives in the system. Data from parallel paths need to be combined in order to compute the final result for each of the relational and scalar aggregation operations (e.g., max, min, sum etc.) [1]. In this situation, the processor coordinating the activity may become a "hot-spot" node. The impact of hot-spot contention in multicomputer systems that use binary hypercube-based static interconnection networks has been studied in [4]. This paper studies the effectiveness of hardware combining and software combining schemes in reducing hot-spot contention in such systems, using both simulation and analytical models.

Two types of hypercube-based static interconnection networks are considered; the binary hypercube, or BH and a type of binary hypercube-based "hierarchical interconnection network" (HIN), BH/BH. HINs can be informally described as follows [3]. Let N denote the total number of nodes in the network. These N nodes are grouped into K_1 clusters of $n_1 = N/K_1$ nodes each. Each cluster of n_1 nodes is linked together by a level 1 interconnection network. One node from each cluster is selected to act as an interface node, and these K_1 interface nodes are again grouped into K_2 clusters of $n_2 = K_1/K_2$ nodes each. A level 2 interconnection network is used to link each of these K_2 clusters of n_2 level 1 interface nodes. If desired, one node from each level 2 cluster could be selected as a level 2 interface node, and these nodes grouped into clusters, etc. Figure 1 shows an example of the type of HIN considered here: a two level "BH/BH" network in which both levels use a binary hypercube (BH) network.

The remainder of this paper is organized as follows. Section 2 gives details on the hot-spot and locality models used in this study. Sections 3 and 4 present results for networks with unbounded and bounded buffering capacity, respectively. Section 5 concludes the paper by summarizing the results.

2. The Workload and System Models

The two characteristics of communication in a distributed-memory multicomputer that are of interest in this study are the *communication locality* and the *hot-spot proportion* (the proportion of traffic that is directed towards a hot-spot). Each is modelled by a single parameter: α (defined below) is used to model locality, and a parameter h (defined below) is used to model the hot-spot proportion.

In our model of communication locality, system nodes (of total number $N = 2^D$) are conceptually divided into equal-sized clusters of size 2^d. (These conceptual clusters become physical clusters in the BH/BH network.) Locality is measured by the probability α (cluster-size dependent) that a message is a nonhot-spot message destined for a node within the same cluster as the source node. It is assumed that:

- Intra-cluster (except hot-spot) communication is uniformly random (i.e., an intra-cluster message is destined to each node within the cluster with equal probability).
- Inter-cluster (except hot-spot) communication is uniformly random (i.e., an inter-cluster message is destined to each cluster other than the source cluster with equal probability, and to each node within the destination cluster with equal probability).

Hot-spots can be modelled in several ways [8]. In this study, as in [4], we assume that there is only a single hot-spot and use the hot-spot model of Pfister and Norton [7]. In this model, each message has a probability h of being a *hot-spot* message (a combinable message destined for the hot-spot node) and a probability $(1-h)$ of being a *regular* message (assumed here to be a non-combinable). Here, the regular messages are further divided into *intra-cluster* and *inter-cluster* messages. When a message reaches its destination node, it is assumed that a reply is generated. Thus, there are hot-spot replies, intra-cluster replies, and inter-cluster replies. When each nonhot-spot node generates messages at rate λ, it generates hot-spot messages at rate $h\lambda$, intra-cluster messages at rate $\alpha\lambda$, inter-cluster messages at rate $(1-\alpha-h)\lambda$, intra-cluster replies at rate $\alpha\lambda$, and inter-cluster replies at rate $(1-\alpha-h)\lambda$. The hot-spot node generates messages and replies at the same rates, except that it also generates hot-spot replies at rate $Nh\lambda$. Note that h must be less than or equal to $(1-\alpha)$.

Links are classified according to whether they join nodes in the same cluster ("cluster links") or in different clusters ("non-cluster links"). The following assumptions are made about the network and its workload:

(*i*) The time between successive message generations at each node is exponentially distributed with mean $1/\lambda$.

(*ii*) The node message generation processes are independent of each other.
(*iii*) Message service times are exponentially distributed; each cluster link processes messages at rate μ_{CL} while each non-cluster link processes messages at rate μ_{NCL}.
(*iv*) Packet-switching is used for message transmission.

3. Performance with Unbounded Buffering Capacity

In this section it is assumed that each node has unbounded message buffering capacity. This assumption is relaxed in Section 4, which considers the case of finite buffering capacity. The impact of hot-spot contention on the average message delivery time or "delay" R_{avg} is derived analytically in [2]. Figure 2 gives a representative sample of the results showing the impact of hot-spot contention in both the BH and BH/BH networks. Figure 2*b* gives two sets of plots corresponding to μ_{NCL} =1.4 and μ_{NCL} =2.8. The latter message processing rate (i.e., μ_{NCL} =2.8) can be achieved, for example, by mapping each logical link to two physical links [3]. Even though this increases link cost, this increase is more than compensated for by the decreases achieved in message delivery times and increases in throughput capacity [3]. In the remainder of the paper, we use μ_{NCL} =2.8 for the BH/BH network. Sections 3.1 and 3.2 study the performance improvements that may be possible with hardware combining and software combining schemes, respectively. These two sections assume a routing algorithm that routes messages by selecting randomly among shortest paths.

3.1. Hardware Combining

In hardware combining, the queues associated with links can be classified as either *forward* queues or *return* queues. A queue associated with a link l that carries messages from node a to node b is termed a forward queue if node b is closer to the hot-spot node than node a, and a return queue if node a is closer to the hot-spot node than node b. A "wait buffer" must be associated with each node. When several hot-spot messages are queued in a forward queue, they are combined into a single hot-spot message, called a *combined message*, which is forwarded toward the hot-spot node. A record of all messages combined is kept in the wait buffer. When the reply from the hot-spot node returns (routing must be such that a reply follows the same path, except in reverse, as the corresponding message), the node, using the information in the wait buffer, generates replies to all combined messages and places them in return queues. Note that a message may participate in many combinings before reaching the hot-spot node. In this section, unbounded buffering space for forward queues, return queues, and wait buffers is assumed, and no restrictions are placed on the number of messages that may be combined at a single node.

The results of simulation experiments are shown in Figure 3. It is obvious from these graphs that hardware combining is effective in reducing the hot-spot contention. Interestingly, delays associated with regular and hot-spot messages reduce with an increasing hot-spot proportion h. This is because with higher h larger numbers of hot-spot messages get combined, effectively reducing the load and thus the contention within the network. The impact of increasing h is more pronounced in the BH/BH network because this network forces all hot-spot messages (from the same cluster) to go through the corresponding interface node, leading to a greater chance of participating in a combining operation.

In practice, hardware combining may not perform as well as Figure 3 would suggest because of finite queue sizes and wait buffers, and limitations on the number of hot-spot messages that may be combined at a time. Finite queues are considered in Section 4. Unbounded queues, however, provide an upper bound on the achievable performance improvements.

3.2. Software Combining

Software combining has been studied in detail by Yew *et al.* [8] in the context of shared-memory multiprocessors. In a distributed-memory multicomputer, software combining can be applied, for example, to the problem of recognizing the end of a computation by logically constructing a tree of nodes that exchange messages so as to finally result in a "root" node being notified of the termination. This root node may then broadcast the fact that the computation has terminated back down the tree.

There are many ways a software combining tree can be constructed on a BH network. A simple mapping scheme that is applicable to both BH and BH/BH networks is considered here. The structure of the software combining tree is shown in Figure 4. This structure is a two-level, unbalanced tree. Nodes within a cluster are linked by a level 1 tree (shown in dashed lines). The root nodes of each of these subtrees (i.e., nodes 0, 8, 16, and 24 in Figure 4) are linked by a level 2 tree (shown in solid lines). At each level, the mapping is done as follows. A node is selected as the root node (for example, node 8 in Figure 4). (This root node selection may be made randomly for the networks considered here, except in the selection of the cluster root nodes in a BH/BH network for which the interface nodes should be chosen.) The children of this root node are the nodes that are one link away from the root node (e.g., nodes 9, 10, and 12), the grandchildren of this root node are all those nodes that are two links away (e.g., nodes 11, 14, and 13), and so on. In any part of the tree, the addresses of a child node and that of its parent differ in only one coordinate. Therefore, there is a link that directly connects each parent-child pair. The members of a set of grandchildren nodes (e.g., 11, 14, 13) are distributed over a set of child nodes (e.g., 9, 10, 12) as uniformly as possible. For example, the grandchildren nodes 11, 14, and 13 are attached one each to the child nodes 9, 10 and 12. Further, if there is a choice in regard to which node a node can be attached in building the software combining tree, a node is randomly selected. For example, node 15 can be attached to either node 13 or node 14. It should be noted that, irrespective of the degree of a node, all links in the tree process hot-spot messages at rate $h\lambda$.

The analysis of delay with a software combining tree is given in [2]. The assumptions stated in Section 2 allow each link to be modelled as a queueing center. Because of space limitations, the analysis is not presented here; only the results are discussed.

The impact of utilizing a software combining tree is presented in Figure 5*a* for a 256-node BH network. All parameters remain the same as those used in Figure 3. The figure shows that software combining is effective in reducing the adverse effect of hot-spot contention. Note that for all values of h, the message generation rate at which the network saturates, λ_{sat} remains constant. This is because the effective message arrival rate seen by a cluster link participating in the combining tree (a tree cluster link) is independent of h and these tree cluster links saturate first, for the parameter values considered here. In general, it can be shown that λ_{sat} will never decrease as h increases, when the proposed software combining tree is utilized [2].

Figure 5*b* shows results for the BH/BH network. The network parameters are the same as those used in Figure 3. Again, as for the BH network, software combining works well. Note that for the BH/BH network, as with hardware combining, λ_{sat} increases with an increasing proportion of hot-spot traffic h.

Comparing Figures 3 and 5, we note that hardware combining and software combining provide performance that is comparable to each other in terms of λ_{sat} values. Limited buffer space, however, may reduce the effectiveness of hardware combining because of the more limited opportunities for combining messages. This effect is demonstrated in the next section, which discusses the effectiveness of both these schemes with bounded buffering capacity.

4. Performance with Bounded Buffering Capacity

This section presents simulation results for the case of bounded buffering capacity. When the number of buffers is finite, there is a possibility of store-and-forward deadlock [6]. Store-and-forward deadlock refers to the situation in which there is a set of buffers, all of which hold messages waiting to be forwarded, and in which these messages can be forwarded only to other buffers of the set. The result is a deadlock.

The simulation experiments implemented a scheme for deadlock avoidance proposed by Merlin and Schweitzer [6]. Let M be the maximum number of hops a message can make in the network. At each node, suppose that there are M+1 buffers, say [B_0, B_1, ..., B_M]. When a message is generated, it is placed in buffer B_0 at the source node. For a given message, let t denote the total number of hops from source to destination. When a message has made $s<t$ hops, it can be placed in any buffer B_g at the next node on the message route such that

$$0 \leq g \leq M-(t-s)+1.$$

It can be shown that if this rule is followed, there will not be store-and-forward deadlocks. For the BH network, $M=D$ and each node should have at least (D+1) buffers. For the BH/BH network, $M=D+d$ and each node should have a minimum of ($D+d$+1) buffers. If the actual number of buffers C is greater than M, the message can be placed in any buffer B_g such that

$$0 \leq g \leq C-(t-s)+1.$$

The simulation experiments were conducted on a 256-node network with a cluster size of 8 (d=3). Thus, each node should have a minimum of 12 buffers.

The simulation experiments and the analysis described in Sections 3.1 and 3.2 used a routing algorithm in which each of the shortest paths between the source and destination nodes is randomly selected with equal probability. However, the links that are used by hot-spot messages tend to saturate much earlier than the other links. Thus, if the routing algorithm can be improved such that it chooses that shortest path with the least traffic, the adverse impact of hot-spot contention on system performance can be reduced. Previous studies have demonstrated the suitability of such improved algorithms [3]. This section uses an improved routing algorithm that works as follows. At each node, if a message can take one of l links (on shortest path), it is routed over the link that has the shortest queue of messages waiting to be forwarded.

The impact of hardware combining on both the average delays of hot-spot (R_{hot}) and non-hot-spot (R_{reg}) messages and replies is shown in Figures 6 and 7. Note that these graphs use throughput rather than the message generation rate λ on the x-axis, and that the throughput may be less than the message generation rate because of buffer limitations. These graphs show that hardware combining is still effective in reducing the adverse effects of hot-spot contention. For the BH network, using hardware combining practically eliminates the interference of hot-spot messages on regular message transmission. As shown in Figure 6a, the impact of varying the hot-spot proportion h on regular message delay is negligible. However, with bounded buffering capacity, hot-spot messages still experience considerable delay as shown in Figure 6b. This is not the case with infinite buffering capacity as shown in Figure 3a. For hot-spot messages, hardware combining provides only marginal improvement in λ_{sat} values. Thus we may conclude that with finite buffer space, hardware combining eliminates completely the adverse impact of hot-spot contention on regular messages, but hot-spot messages themselves still experience a considerable degradation in performance.

Similar conclusions can be drawn for the BH/BH network as well. As shown in Figure 7a, increasing the hot-spot proportion h reduces the delay associated with the regular messages. (This is not the case in Figure 2b because those results were obtained with a random routing algorithm while the results in Figure 7 were obtained by using the improved routing algorithm.) As far as hot-spot messages are concerned, these still experience considerable delays. For smaller h values (1-4%) λ_{sat} values are not affected (see Figures 7a and 2b). For higher h values (8% and 16%), λ_{sat} decreases with increasing h, but the actual λ_{sat} values remain largely unaffected whether or not hardware combining is used. Again, as with the BH network, primarily the regular messages benefit from hardware combining when buffer space is limited.

Figures 8 and 9 show the influence of software combining on the average delays of regular and hot-spot messages for the BH and BH/BH networks. The results show that software combining achieves message combining more effectively than hardware combining in this case. These results further demonstrate that the simple mapping scheme used here for the software combining tree is adequate, even with finite buffer space. Regular messages experience similar delays with both hardware combining and software combining schemes (compare Figures 8a and 9a with Figures 6a and 7a). However, for both networks, the hot-spot message delays with software combining are substantially less compared to the delays with hardware combining.

5. Summary

This paper has studied the impact of hardware combining and software combining on hot-spot contention in two types of static interconnection networks that are appropriate for use in distributed memory multicomputer systems. One was the standard binary hypercube (BH) network and the other was a hierarchical interconnection network -- the BH/BH network.

With unbounded buffering capacity at each node, both schemes tend to provide similar performance improvements and completely eliminate the adverse effects of hot-spot contention on both the regular (non-hot-spot) messages and hot-spot messages.

With bounded buffering capacity, hardware combining could eliminate only the influence of hot-spot contention on regular messages; the hot-spot messages experience degraded performance whether or not hardware combining is used. With bounded buffering capacity, software combining is more effective than hardware combining. Although regular messages experience similar delays with both hardware combining and software combining, only the software combining scheme provides substantial improvements in the throughput capacity and in the delays associated with hot-spot messages. However, as the number of buffers is increased, hardware combining tends to improve its performance for hot-spot messages as well. But this may require an infeasibly large number of buffers.

It should be noted that the results for the finite buffer case (in particular, the ability to eliminate the interference of hot-spot contention on regular messages) are also dependent on using a good (adaptive) routing algorithm.

References

[1] C. K. Baru and O. Frieder, "Database Operations in a Cube-Connected Multicomputer System," *IEEE Trans. Computers*, Vol. 38, No. 6, June 1989, pp. 920-927.

[2] S. P. Dandamudi, *Hierarchical Interconnection Networks for Multicomputer Systems*, Ph.D. thesis, Department of Computational Science, University of Saskatchewan, Saskatoon, Canada 1988 (Available as Tech. Rep. 88-18).

[3] S. P. Dandamudi and D. L. Eager, "Hierarchical Interconnection Networks for Multicomputer Systems," *IEEE Trans. Computers*, Vol. C-39, No. 6, June 1990.

[4] S. P. Dandamudi and D. L. Eager, "Hot-Spot Contention in Binary Hypercube Networks," to appear in *IEEE Trans. Computers*.

[5] G. Lee, C. Kruskal, and D. J. Kuck, "The Effectiveness of Combining in Shared Memory Parallel Computers in the Presence of 'Hot Spots'," *Proc. 1986 Int. Conf. Parallel Processing*, 1986, pp. 35-41.

[6] P. M. Merlin and P. J. Schweitzer, "Deadlock Avoidance in Store-and-Forward Networks -- I: Store-and-Forward Deadlock," *IEEE Trans. Communications*, Vol. COM-28, No. 3, March 1980, pp. 345-354.

[7] G. F. Pfister and V. A. Norton, " 'Hot Spot' Contention and Combining in Multistage Interconnection Networks," *IEEE Trans. Computers*, Vol. C-34, No. 10, October 1985, pp. 943-948.

[8] P. -C. Yew, N. -F. Tzeng, and D. H. Lawrie, "Distributing Hot-Spot Addressing in Large-Scale Multiprocessors," *IEEE Trans. Computers*, Vol. C-36, No. 4, April 1987, pp. 388-395.

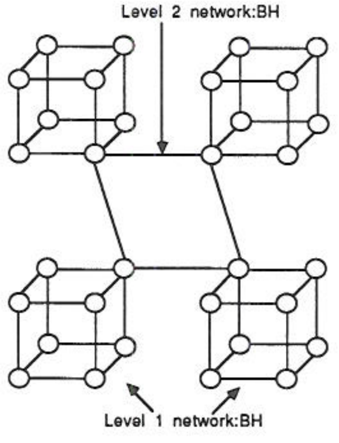

Figure 1 A BH/BH hierarchical interconnection network

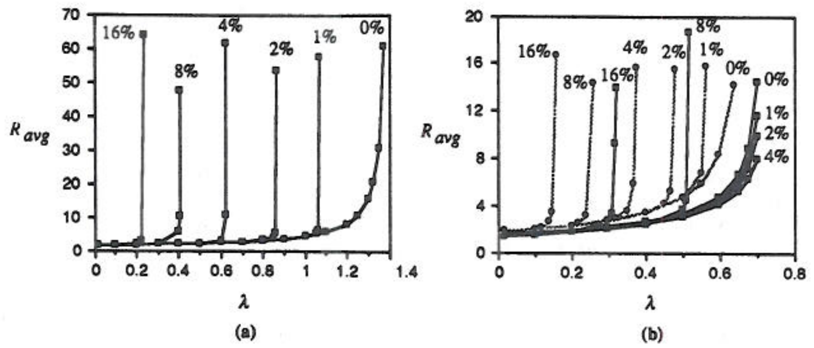

Figure 2 Impact of hot-spot contention on the BH and BH/BH networks (h is varied from 0% to 16%) (a) BH network (d = 3, D = 8, $\mu_{CL} = \mu_{NCL} = 1.4$, $\alpha = 0.75$) (b) BH/BH network (d = 3, D = 8, $\mu_{CL} = 1.4$, $\alpha = 0.75$): $\mu_{NCL} = 2.8$ for solid lines; $\mu_{NCL} = 1.4$ for dotted lines. (from [4])

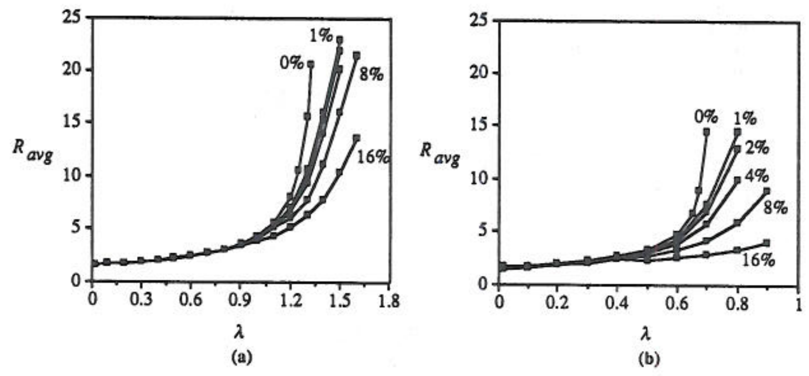

Figure 3 Impact of hardware combining on hot-spot contention in the BH and BH/BH networks (h is varied from 0% to 16%) (a) BH network (d = 3, D = 8, $\mu_{CL} = \mu_{NCL} = 1.4$, $\alpha = 0.75$) (b) BH/BH network (d = 3, D = 8, $\mu_{CL} = 1.4$, $\mu_{NCL} = 2.8$, $\alpha = 0.75$)

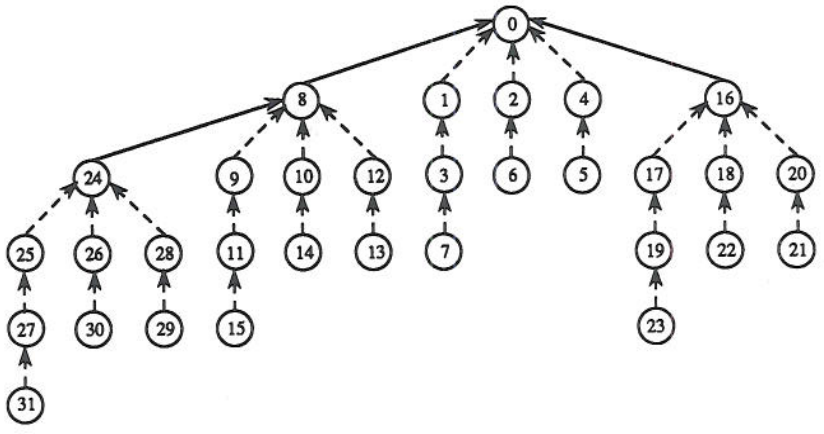

Figure 4 Software combining tree for the BH and BH/BH networks (D= 5 and d = 3)

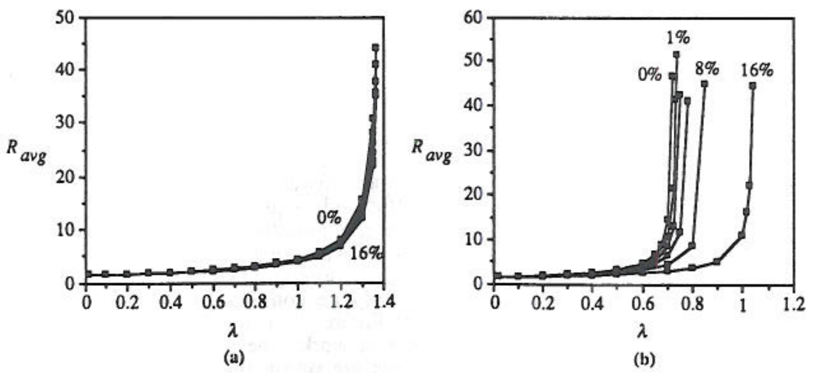

Figure 5 Impact of software combining on hot-spot contention in the BH and BH/BH networks (h is varied from 0% to 16%) (a) BH network (d = 3, D = 8, $\mu_{CL} = \mu_{NCL} = 1.4$, $\alpha = 0.75$) (b) BH/BH network (d = 3, D = 8, $\mu_{CL} = 1.4$, $\mu_{NCL} = 2.8$, $\alpha = 0.75$) (R_{avg} here represents the average of non-hot-spot message/reply delays and the delays of hot-spot replies.)

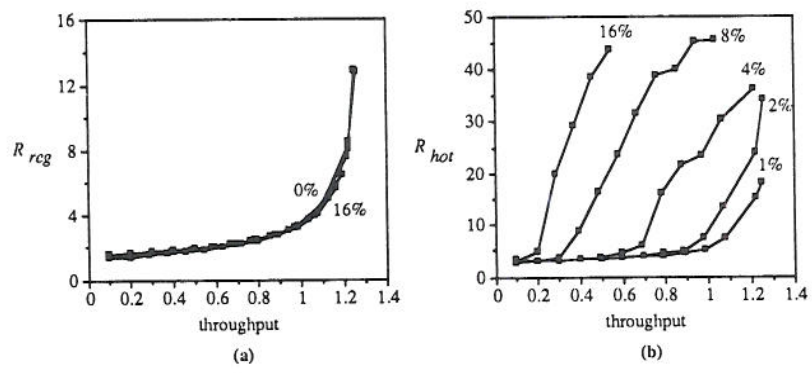

Figure 6 Impact of hardware combining on hot-spot contention in the BH network -- the case of finite buffering capacity (h is varied from 0% to 16%) ($d = 3, D = 8, \mu_{CL} = \mu_{NCL} = 1.4, \alpha = 0.75$, number of buffers/node = 32)

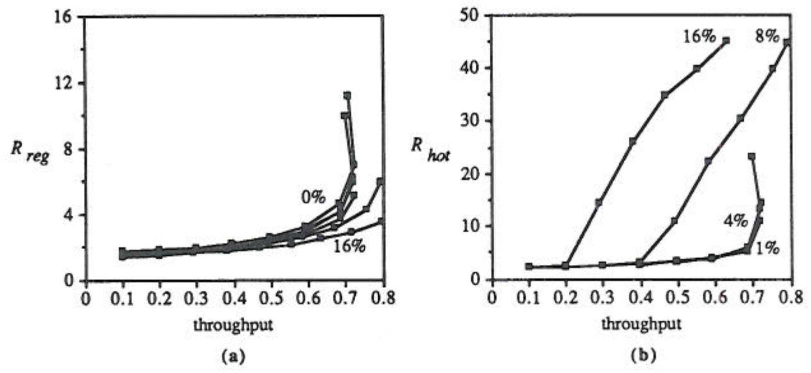

Figure 7 Impact of hardware combining on hot-spot contention in the BH/BH network -- the case of finite buffering capacity (h is varied from 0% to 16%) ($d = 3, D = 8$, $\mu_{CL} = 1.4, \mu_{NCL} = 2.8, \alpha = 0.75$, number of buffers/node = 32)

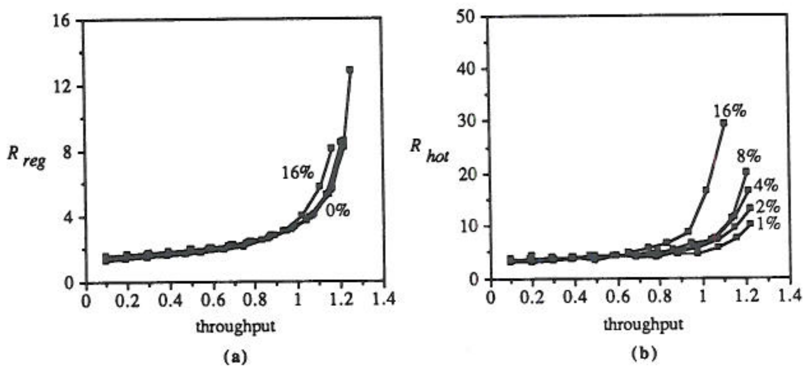

Figure 8 Impact of software combining on hot-spot contention in the BH network -- the case of finite buffering capacity (h is varied from 0% to 16%) ($d = 3, D = 8, \mu_{CL} = \mu_{NCL} = 1.4, \alpha = 0.75$, number of buffers/node = 32)

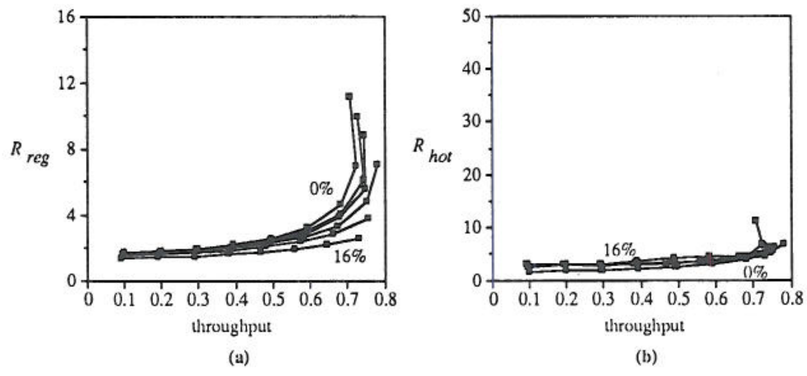

Figure 9 Impact of software combining on hot-spot contention in the BH/BH network -- the case of finite buffering capacity (h is varied from 0% to 16%) ($d = 3, D = 8$, $\mu_{CL} = 1.4, \mu_{NCL} = 2.8, \alpha = 0.75$, number of buffers/node = 32)