Thread Placement on the Intel Paragon: Modeling and Experimentation

E. Smirni, C.A. Childers, E. Rosti*, L.W. Dowdy
Department of Computer Science
Vanderbilt University, Nashville, TN 37235
[esmirni, cac, erosti, dowdy]@vuse.vanderbilt.edu

Abstract

In multicomputer architectures where communication latency is distance independent, thread placement is expected to have a limited impact on an application's performance. In this paper, the impact of thread placement on application performance is demonstrated on a wormhole routed multicomputer, the Intel Paragon. A communication intensive synthetic workload is used to "stress test" the effects of contention on communication latency induced by thread placement. It is shown by means of experimentation and modeling that appropriate thread placement patterns minimizing contention in the system's interconnection network improve performance. The analytic model and the experimental observations are in good agreement.

Keywords: *thread placement, network contention, communication latency, performance analysis*

1 Introduction

Interconnection networks with simple topologies and nearly distance independent communication latency are a key feature of recent architectures. Wormhole routing [3] is a popular distance independent routing scheme adopted in second generation multicomputers such as the Symult 2010, the Intel/DARPA Touchstone Delta, the nCUBE–2, and the Intel Paragon. Experimental and modeling studies confirm the negligible impact of path length on network latency under wormhole routing [1, 2]. However, in large scale wormhole routed multicomputers, shared resources such as network channels can become the bottleneck that degrades performance. In many such situations, strategic thread placements are superior to random placements [1].

In this paper, the problem of network contention induced by thread placement in medium scale multicomputers is addressed by means of modeling and experimentation. Analytical models based on Generalized Stochastic Petri Nets (GSPN) [5] of applications executing on a wormhole routed system are constructed. These models are used to investigate the impact of thread placement on channel contention under light to heavy communication traffic. Synthetic workloads are used to stress test the network and to provide a sensitivity analysis under various workload patterns. As expected, performance is insensitive to the relative distance of the communicating threads when messages do not compete for channels. However, when messages compete for channels forming a hot path (i.e., a path between two nodes which is simultaneously needed by several messages), the distance between threads impacts their communication time.

A series of experiments confirms that application response time depends on the placement of the application threads. The experiments are conducted on the Intel Paragon. The Paragon is an MIMD machine with distributed memory. It is organized in a 2–dimensional, not necessarily square, grid topology. Each node of the grid is composed of an iMRC mesh router chip, two i860 XP processors each with a 16 KB cache, and a 16 MB main memory. One of the two processors, the application processor, is dedicated to computational work. The second i860 is the communication processor, which provides communication services only. The iMRC mesh router chip is responsible for routing messages between processors. The router chip has 5 ports, one to the local communication processor and the other 4 to the iMRCs of the adjacent nodes. The iMRC may route up to 5 message packets at once with no contention, offering an effective bandwidth of 200 MB/sec per port, for a total

*While on leave from the Dipartimento di Scienze dell'Informazione, Università di Milano, Italy.
This work was partially supported by sub-contract 19X-SL131V from the Oak Ridge National Laboratory managed by Martin Marietta Energy Systems, Inc. for the U.S. Department of Energy under contract no. DE-AC05-84OR21400 and by Italian CNR "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo – Sottoprogetto 3" by grant N. 93.01636.PF69, and by M.U.R.S.T. 40% and 60% projects.

of 1 GB/sec. The system is organized in three partitions: the compute partition is dedicated to parallel programs, the service partition supports interactive use (e.g., login shells), and the I/O partition provides I/O services to peripheral devices such as RAID disks.

The switching technique used on the Paragon is wormhole routing. Messages are divided into small pieces of a few bytes called flits (flow control digits). The header flit (or flits) reserves the channels as it moves from source to destination, and the rest of the flits follow in a pipelined fashion. If the next required channel is busy, then the header flit is blocked until the channel becomes available. When the header flit blocks, the remaining flits are blocked at the intermediate nodes along the path. At each node there is a small number of flit buffers. If virtual channels are not built on top of the physical channels for message multiplexing, as with the Paragon, then no other message can use the channels reserved by the header flit. A channel is released only when the tail flit has traversed it. Deadlock is avoided at the routing level [3]. Depending upon the way the header flit establishes the path, routing schemes are classified as adaptive or deterministic [3]. With adaptive routing schemes, the header flit dynamically adjusts the message route according to network congestion. With deterministic routing, the route is fixed and independent of the network load. The deterministic routing algorithm adopted on the Paragon is XY routing, where flits are sent first along the X dimension and then along the Y dimension. In the absence of network traffic, the path length from source to receiver node has a negligible effect on communication latency. However, when channel contention increases, so does communication latency due to message blocking. Deterministic routing accentuates this effect since it does not adapt to changes in the network load.

This paper is organized as follows: Section 2 presents a model describing the behavior of the system at the wormhole routing level. Experimental observations are compared against the model predictions in Section 3. Section 4 concludes the paper and summarizes the findings.

2 The GSPN Model

In this section, an analytic GSPN model at the message passing level is presented. Simplifying assumptions of synchronous communication primitives and uniquely paired sender and receiver threads are made. Each application thread is assumed to iterate between computation and communication sequences. Such a structure is characteristic of Single Program Multiple Data (SPMD) applications comprising much of the workload of parallel systems. The model reflects the deadlock free routing protocol and the lack of virtual channels. It captures the implicit priority that a message originating close to a link has over a message originating further away from that same link.

The general GSPN model is constructed by a subnet for every sender/receiver pair and one place for every utilized network link. Figure 1 illustrates the detailed model of a generic sender/receiver i subnet and the link connections utilized by the sender's messages. Place $P_{i:comp}$ represents sender i's computation

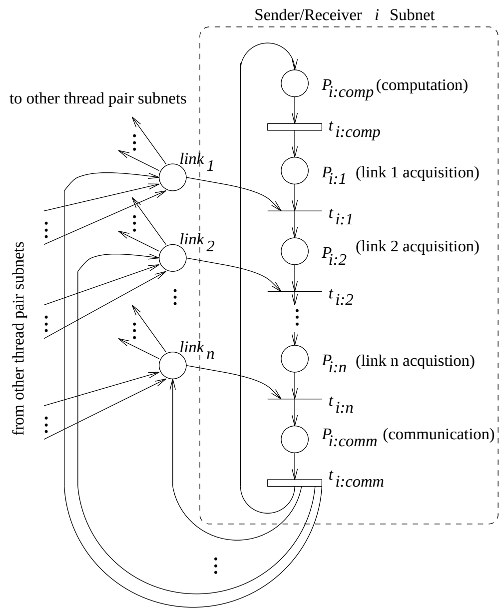

Figure 1: A generic sender/receiver thread pair subnet and link connections.

between consecutive communications. The rate of the corresponding timed transition $t_{i:comp}$ is $\mu_{i:comp}$ and represents the sender's computation rate. The initial marking has one token in each place $P_{i:comp}$ for all sender/receiver pair subnets. One token is assigned to each place $link_j$ to model the initial availability of all links. The acquisition of link j by sender i is modeled by the firing of the associated immediate transition $t_{i:j}$. If a token is present in place $link_j$, then the link is available. The unique token in each link place indicates that the link can be used exclusively by one thread at a time. The conflict between two or more transitions over a token in place $link_j$ models contention for link j. A token reaching place $P_{i:comm}$ from $P_{i:1}$ models the successful progression of the header flit through the path where message transmission takes place. The rate $\mu_{i:comm}$ associated with the timed transition $t_{i:comm}$ is a function of the message size and the network bandwidth. The firing of transition $t_{i:comm}$ returns tokens to places $link_j$ via their output arcs modeling link release at the end of communication. A token is returned to place $P_{i:comp}$ as well, indicating that the sender is computing again.

All sender/receiver subnets are connected via arcs representing common links in their communication paths. Message contention for a link along hot paths is modeled. The model is used in the next section to investigate the effects of thread placement on application performance under various contention conditions.

3 Performance Analysis

In this section, the results of the experimental and modeling performance study of the impact of thread placement in a wormhole routed multiprocessor are presented. The experiments are conducted on a 512 node Paragon under the Sandia–University of New Mexico Operating System (SUNMOS) [4]. The performance study initially focuses on the effects of various thread placements designed specifically to stress test the interconnection network. A sensitivity analysis of various computation/communication ratios follows.

The GSPN model is solved via simulation. The computation rate of each thread, $\mu_{i:comp}$, is derived from measurements. The communication rate, $\mu_{i:comm}$, is the maximum measured transmission rate for a link. Both computation and communication rates are assumed to be constant.

3.1 The Synthetic Workload

To investigate under which circumstances communication can become the bottleneck and the resulting effects of various thread placement strategies, a synthetic workload is constructed. Each experiment consists of two phases, the initialization phase and the measurement phase. During the initialization phase, an equal number of sender and receiver threads are spawned, and each is bound to a specific processor. A global synchronization operation is performed such that all sender threads are simultaneously ready to send their messages and all participating receiver threads have posted their receive. Senders do not block because receivers are not ready.

Each sender and receiver pair iterates between computation and communication sequences. The measurement phase consists of timing the first 200 iterations of each thread execution and calculating the average time per iteration for each thread. A constant amount of communication load is put on the network during the measurement phase by forcing all threads to continue executing. The average execution time per iteration is independent of the number of iterations over which timing is performed.

The purpose of the performance study is to determine how the response time of each thread is affected by different thread placement schemes. Different placement schemes induce different communication patterns, which affect network contention over the various links.

3.2 Stress Test Analysis

To provide a communication intensive workload, the amount of computation performed by each sender between successive messages is limited to 2μsec. The parameters to be varied in the performance analysis are:

1. the number of sender/receiver pairs,
2. the message size, and
3. the thread placement patterns of the sender/receiver pairs.

The number of thread pairs, varying from 1 to 6, affects the total communication load in terms of the amount of imposed network traffic. The message size, varying from 16 KB to 64 KB, affects the time that channels are held by a message, thus impacting the blocking time of contending messages. Analogous results are obtained for a variable range of message sizes [6]. Larger messages maximize channel contention. Four different thread placement patterns change the amount of traffic along the hot paths.

Figure 2 illustrates the first communication pattern examined, pattern 0, for $n = 6$ thread pairs. With a total of n thread pairs, sender S_i is paired with receiver R_i, $i = 1, \ldots, n$. Threads are mapped onto adjacent processors along a common routing path to maximize the utilization of the link between sender S_1 and receiver R_1. Link utilization decreases symmetrically moving outward from the thread pair (S_1, R_1).

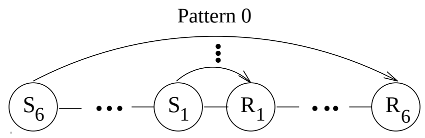

Figure 2: Thread placement pattern 0 with 6 pairs.

Figure 3 illustrates the measured and predicted response times of the outermost thread pair in pattern 0 as a function of the total number of thread pairs for different message sizes. The response time of the outermost thread pair increases exponentially with the total number of thread pairs. In the absence of network contention (i.e., with 1 thread pair), the effective measured bandwidth across the hot path is 110, 130, and 145 MB/sec for messages of size 16, 32, and 64 KB, respectively. The model predictions (dashed lines) are also graphed in Figure 3 and closely follow the experimental measurements.

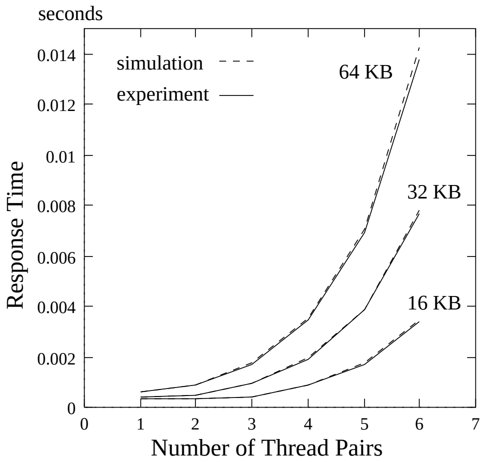

Figure 3: Measured and predicted response times of the outermost thread pair in pattern 0.

While Figure 3 shows the response times for the outermost thread pair only, Figure 4 shows the response times for the intermediate thread pairs as a function of the total number of thread pairs. For instance, if the total number of thread pairs is 3, then three numbers are plotted, one for the outermost thread pair, one for the intermediate thread pair (outer-1), and one for the innermost thread pair (outer-2). Results are depicted for a message size of 32 KB and are representative of all message sizes. Regardless of the total number of thread pairs, the outermost pair and the pair next to it (outer-1) have identical response times. The two additional links traversed by a message coming from the outermost sender to reach the outermost receiver (e.g., those between senders S_6 and S_5 and receivers R_5 and R_6 in Figure 2) are contention–free and do not affect communication latency. The innermost thread pair is always significantly faster than the rest of the thread pairs. The greater the distance of the sender from the most utilized link in the hot path, the greater the communication latency of the communicating threads. Thread pairs that are placed closest to the most contended link are favored over more distant pairs. At each node, an incoming message and a message generated locally are equally likely to acquire the link. The longer the distance traveled by a message, the less likely it is to successfully capture all required links, and the smaller the fraction of link bandwidth it uses. Effectively, the amount of available bandwidth halves at each additional link for which the message must compete. As an illustration, the fraction of bandwidth of the link between sender S_1 and receiver R_1 in Figure 2 that sender S_6 can use is only 1/32 of the total available link bandwidth. In general, for a total number of n pairs in pattern 0, the network bandwidth at the innermost link is split among the participating thread pairs in the following manner: 1/2 to the innermost pair, $(1/2)^2$ to the second innermost pair (the closest to the innermost), $(1/2)^i$ to the i-th innermost, and $(1/2)^{n-1}$ each to the outermost and the next to the outermost pair. The model prediction curves correctly capture this behavior.

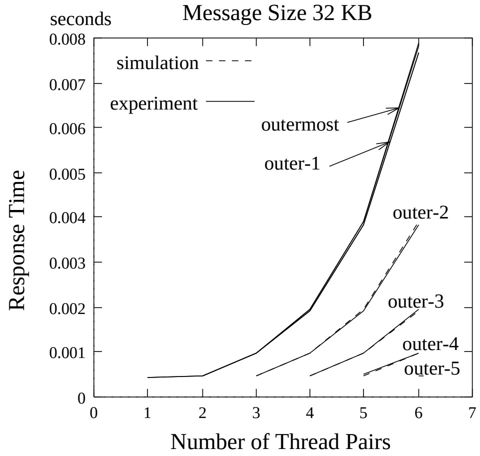

Figure 4: Measured and predicted response times of all thread pairs for pattern 0.

The effect of splitting the link bandwidth among competing messages for pattern 0 with 6 pairs is illustrated by the Gantt chart in Figure 5. This figure graphs the instants when messages are received along the execution time in a pattern 0 placement with 6 thread pairs for 32 KB messages. Each receiver is uniquely associated with a horizontal bar, which plots message reception events as a function of the execution time. The time for each receiving thread to receive the first 200 messages is indicated on the x–axis. Each thread's execution continues, keeping the link contention level constant even after the plotted 200 iterations. As expected, the time for each receiver to complete its first 200 iterations doubles for every additional contended link that its messages require. The response time of receiver R_2 is twice as long as that of receiver R_1, the response time of receiver R_3 is twice as long as that of receiver R_2, and so on. Receivers R_5 and R_6 have identical response times because they compete for the same set of links. Their bars are longer and less dense because link contention on the inner links reduces their message reception rate. Receiver R_6 actually uses two more links than receiver R_5, but since there is no contention for these two links, they do not impact performance.

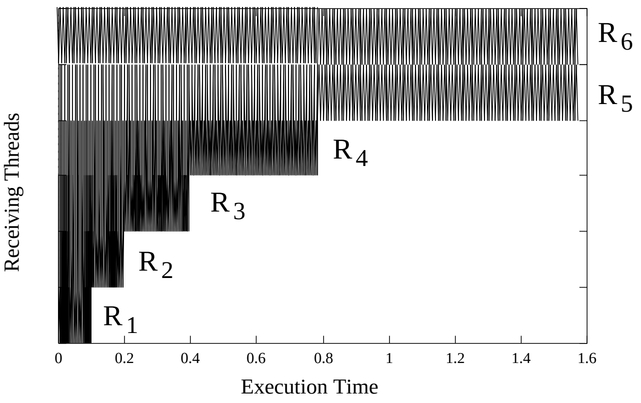

Figure 5: Instants when messages are received as a function of the execution time for 200 iterations.

Figure 6 illustrates three different thread placement patterns for 9 sender/receiver pairs. The measured and predicted response times for the receiving threads for each pattern are indicated in the graph. The impact of contention on the links is demonstrated. Compared with pattern 0 (see Figure 2), these patterns gradually reduce contention on the links. Pattern 1 is similar to pattern 0, consisting of two independent pattern 0–like placements, hence 2 independent hot paths. Sender S_9's communication path is the concatenation of the two hot paths. Threads closer to sender S_9 suffer from greater link contention than those closer to receiver R_9. If sender S_9 has acquired all the links that senders S_1, S_2, S_3, and S_4 require for communication and is blocked by sender S_5, S_6, S_7, or S_8, then the response time of receivers R_1, R_2, R_3, and R_4 increases. Receivers R_9 and R_4 have identical response times.

Pattern 2 combines four independent 0–like placement patterns of 2 pairs each. An additional pair, (S_9,R_9), competes for all links. Compared to pattern 1, pattern 2 exhibits a higher locality of contention. Due to the shorter hot paths, the response time of the receivers of the 4 components mimics pattern 1's components behavior on a different scale. The further away the receivers are placed from sender S_9, the better their response times. As with pattern 1, the last receiver of the first pattern 0–like placement, receiver R_2, and receiver R_9, have equal response times and are the slowest.

Pattern 3 represents the ideal thread placement, the one that provides nearest neighbor communication. As expected, the response time for all threads is identical due to the absence of network contention. Since latency is nearly distance independent with wormhole routing, any communication pattern that does not result in link contention is equivalent to nearest neighbor communication.

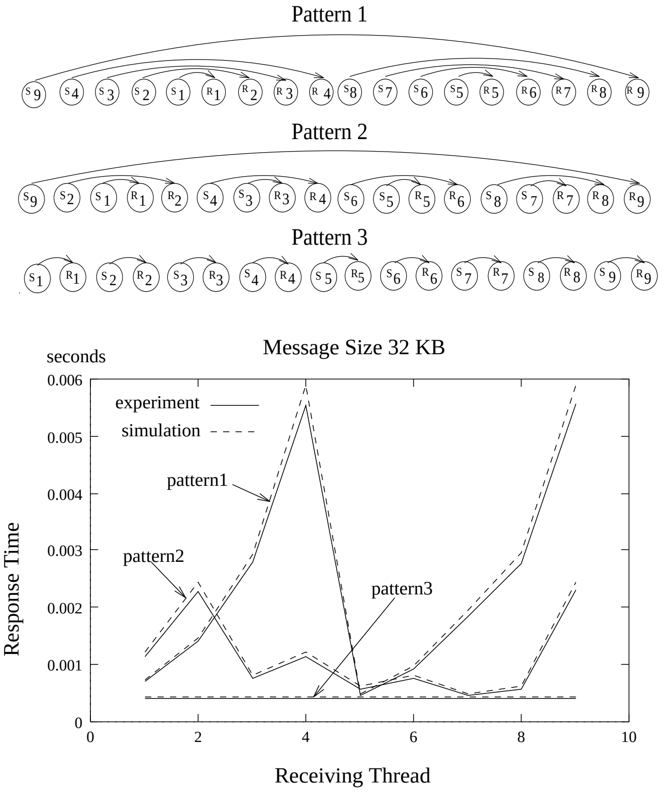

Figure 6: Measured and predicted response time of all thread pairs for patterns 1, 2, and 3.

3.3 Sensitivity Analysis

The experiments described above are communication intensive cases designed to stress test the system and validate the model. A sensitivity analysis of the same workload is used to investigate the effects of thread placement under more realistic assumptions. In this section, the computation/communication ratio is varied. The thread placement pattern considered is pattern 0. The amount of computation determines the rate at which senders transmit messages, thus affecting the load on the network. The computation times of each sender are varied from 2×10^{-6} to 39.6×10^{-4} seconds. The message size is fixed at 32 KB. Analogous results are obtained for different message sizes but are not presented here for the sake of brevity.

Figure 7 graphs the experimentally measured outermost thread pair response time for various amounts of computation performed between two consecutive send operations. Since an increase in the amount of computation decreases the message transmission rate, the number of thread pairs needed to saturate the hot path increases. For the highest amount of computation shown, 39.6×10^{-4} seconds, response times are constant for all thread pairs since the time spent communicating is negligible compared to the computation time. As the amount of computation decreases (i.e., the computation/communication ratio decreases), fewer thread pairs can bottleneck the links. For 82.7×10^{-6} seconds of computation, response times remain constant for up to 5 thread pairs. After this point, communication becomes the bottleneck and adversely affects the response time of the outermost thread pair. For computation times of 43.5×10^{-6} and 2×10^{-6} seconds, communication becomes the bottleneck at 3 and 2 thread pairs, respectively. As the computation time decreases, the knee of the curve moves further to the left where response time begins to increase exponentially.

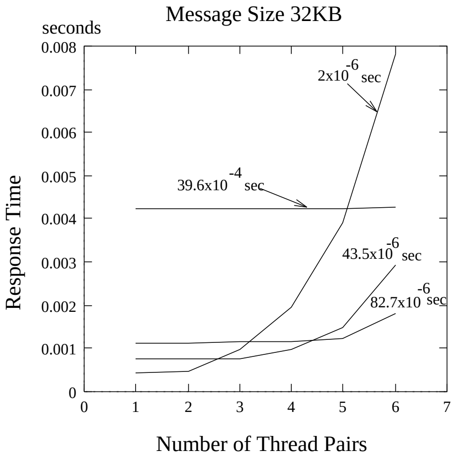

Figure 7: Measured response times for the outermost thread pair with various computation times.

As part of the sensitivity analysis under less extreme conditions, the model was used to predict the speedup of an actual application. An implementation of matrix transposition was studied for three thread placement patterns that induce various levels of network contention. Various matrix sizes and numbers of threads were considered [6]. The application's performance reflects the results obtained by the stress test analysis. Placement patterns that limit network contention and smaller message sizes favorably impact performance.

4 Conclusions

In this paper, the performance implications of strategic thread placement on the Paragon, a wormhole routed multicomputer with a deterministic routing scheme, are presented. A Generalized Stochastic Petri Nets model is built and used for performance prediction. It accurately captures the effects of the switching technique on the communication times of threads physically closer to the communication bottleneck. Synthetic workloads are used to stress test the interconnection network. Although communication latency is nearly distance independent in a lightly loaded network, it is not distance independent when communication patterns cause contention for network links. Experimental results are reported for systems with a limited number of processors but analogous results have been obtained for larger systems.

Future work includes the modeling and experimentation of alternative thread placement strategies over a variety of actual application codes. Guidelines for thread placement and better application design in parallel wormhole routed multicomputers are expected.

References

[1] S. Chittor, R. Enbody, "Performance degradation in large wormhole-routed interprocessor communication networks," *Proc. of the 1990 Int'l Conf. on Parallel Processing*, vol. 1, pp. 424-428, 1990.

[2] J. Kim, C.R. Das, "Modeling wormhole routing in a hypercube," *Proc. of the 11th Int'l Conf. on Distributed Computing Systems*, pp. 386-393, 1991.

[3] L.M. Ni, P.K. Mckinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, N. 2, pp. 62-76, February 1993.

[4] B. Maccabe, K.S. McCurley, R. Riesen, "SUNMOS for the Intel Paragon: A Brief User's Guide," Sandia National Laboratories Tech. Report, 1993.

[5] M.A. Marsan, G. Conte, G. Balbo, "A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems*, vol. 2, N. 2, pp. 93-122, May 1984.

[6] E. Smirni, C.A. Childers, E. Rosti, L.W. Dowdy, "Thread placement on the Intel Paragon: modeling and experimentation", Tech. Report 94-05, Vanderbilt University, Computer Science Department.