Proc. Intl. Symp. on High Performance Computer Architecture, Feb. 1997.

A Performance Comparison of Hierarchical Ring- and Mesh-connected Multiprocessor Networks

Govindan Ravindran and Michael Stumm
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada M5S 3G4
Email: gravin@eecg.toronto.edu

Abstract

This paper compares the performance of hierarchical ring- and mesh-connected wormhole routed shared memory multiprocessor networks in a simulation study. Hierarchical rings are interesting alternatives to meshes since i) they can be clocked at faster rates, ii) they can have wider data paths and hence shorter message sizes, iii) they allow addition and removal of processing nodes at arbitrary locations, iv) their topology allows natural exploitation in the spatial locality of application memory access patterns, and v) their topology allows efficient implementation of broadcasts.

Our study shows that for workloads with little locality, meshes scale better than ring networks because ring-based systems have limited bisection bandwidth. However, for workloads with some memory access locality, hierarchical rings outperform meshes by 20-40% for system sizes of up to 128 processors. Even with poor access locality, hierarchical rings will outperform meshes for these system sizes if the mesh router buffers are only 1-flit large, and they will outperform meshes in systems with less than 36 processors regardless of mesh router buffer size.

1 Introduction

Rings and meshes are currently popular choices for interconnection backplanes of large-scale shared memory multiprocessors. A number of commercial products use (or will use) these classes of networks [7, 9, 10], as do a number of experimental and research systems [3, 6, 17, 18, 26]. At this time, meshes seem to be the more popular choice, perhaps because it is relatively easy to build systems using off-the-shelf routers and processors and perhaps because of their scalability characteristics. Nevertheless, unidirectional ring-based multiprocessors are interesting alternatives for a number of reasons. Their simple node to ring interfaces allow rings to be clocked at faster rates. Under identical pin constraints, rings can have wider data paths and therefore shorter message sizes. Moreover, rings allow easy addition and removal of nodes at arbitrary locations. In many ways, a unidirectional ring can be considered the simplest way to connect multiple processing modules using point-to-point interconnection.

Multiprocessors based on a single ring are limited to a relatively small number of processors due to the fixed bandwidth of rings independent of ring size. To accommodate additional processors, it is necessary to interconnect multiple rings. Here, we only consider hierarchies of uni-directional rings. This topology can exploit the spatial locality of memory accesses often exhibited in parallel programs, which is critical to size scalability. Moreover, it allows efficient implementation of broadcasts, useful for implementing cache coherence [13] and multicast protocols [6]. However, unlike mesh networks, hierarchical ring networks have a constant bisection bandwidth regardless of system size. This limits the scalability of hierarchical ring-based systems.

In this paper, we analyse and compare the performance of hierarchical ring and mesh connected multiprocessor networks, using detailed flit-level simulations. Our results show that for workloads with good memory access locality, hierarchical rings outperform meshes for system sizes of up to 128 processors by 20-40%. For workloads with little or no locality, hierarchical rings also outperform meshes for these system sizes if the buffers in the mesh routers are only 1-flit large. If the size of mesh buffers are 4-flits or larger, then hierarchical rings perform better than meshes only for small system sizes (by 10-30%); for larger systems the performance of hierarchical rings is severely constrained due to bisection bandwidth limitations and meshes perform significantly better.

Although some previous work on the performance of hierarchical ring networks [4, 13, 16, 20, 21] and on mesh networks [1, 2, 8, 12, 23] has been published, we are aware of only one study that compares the performance of both types of networks [15]. That study uses analytical models to conclude that three-level hierarchical systems perform somewhat better than mesh systems.

The rest of the paper is organized as follows. Section 2 describes our simulated system, including details of the network, the simulator and our benchmark workloads. Hierarchical rings are evaluated in Section 3. It is shown that the performance of large scale hierarchical ring systems are bisection bandwidth limited. Section 4 evaluates the performance of meshes, which is strongly affected by the size of the router

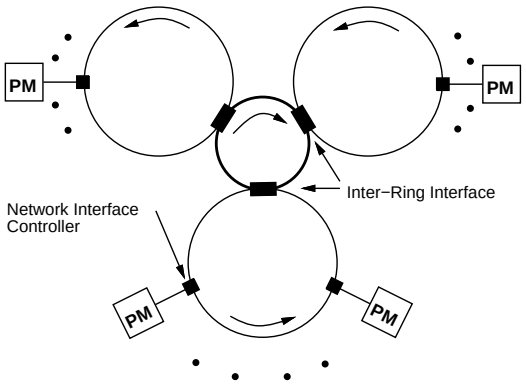


Figure 1: A hierarchical ring system with two levels.

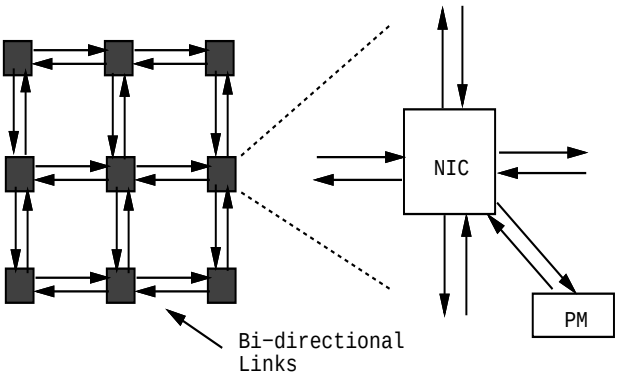


Figure 2: A 2D mesh system with 9 processors.

buffers. In Section 5, the performance of hierarchical ring networks is compared with that of 2-dimensional bi-directional mesh networks for different system sizes. In Section 6, we consider a minor modification to the hierarchical ring structure whereby the highest level ring is clocked at twice the speed of other rings. This type of modification is feasible, as it only affects a small portion of the network, allowing the use of more expensive technology in a critical part of the system without affecting the overall system cost by much. This modification allows us to build larger systems with higher throughput. These systems generally give better performance than equivalent mesh networks for systems as large as 128 processors.

2 Simulated system

Figures 1 and 2 show shared memory multiprocessor systems containing P processing modules, in the former case connected by a 2-level hierarchy of unidirectional rings, and in the latter case connected by a square 2D bi-directional mesh. Each processing module (PM) contains a processor, a local cache and a portion of the main memory. In the case of hierarchical rings, all processing modules are connected to lowest level rings, which we also refer to as *local rings*. A *global ring* connects several of these local rings. The channel width (data path) of the ring is assumed to be 128 bits wide, based on the NUMAchine (a CC-NUMA machine) design [6]. For a mesh connected system, a number of variations on the basic topology shown are possible. In our study, the connection between each pair of adjacent nodes in a mesh is bi-directional (implemented as two 32-bit wide uni-directional channels) and there are no *end-around* connections. We choose this topology because of its simple *e-cube* deterministic deadlock free routing algorithm that does not require virtual channels.

Both systems provide a flat, global (physical) address space, and each PM is assigned a unique contiguous portion of that address space, determined by its location. All processors can transparently access all memory locations in the system. The target memory is determined by the address of the memory being accessed. Local memory accesses do not involve the network. Remote memory accesses require a request packet to be sent to the target memory, followed by a response packet from the target memory to the requesting processor.

Packets sent are of variable size[1] and are transferred in flits,[2] bit-parallel, along a unique path through the network. The switching technique used determines how packets are forwarded through the network. We assume wormhole switching in our study, where a packet is sent as a contiguous sequence of flits with the header flit containing the routing and sequencing information [11]. The head flit of a packet acquires network resources (links and buffer slots) as it proceeds through the network, while the tail flit then frees them. Since only the head flit of a packet contains routing information, it is essential that the flits of a packet not be interleaved with the flits of other packets. When a packet cannot move forward because the next link is busy, it is blocked in place and a flow-control signal is back propagated to the previous node to prevent further transmission over the incoming link. Thus, wormhole switching may stall packets across multiple links if the buffers in the network nodes are smaller than the size of a packet.

2.1 Hierarchical ring system description

For a hierarchical ring, there are two types of network nodes: *Network Interface Controllers* (NIC) connect processing modules (PM) to local rings, and *Inter-Ring Interfaces* (IRI) connect two rings of adjacent levels. The NIC examines the header of a packet and switches 1) incoming packets from the ring to a PM, 2) outgoing packets from the PM to the ring, and 3) continuing packets from the input link to the output link. The IRI controls the traffic between two rings and is modelled as a 2×2 crossbar switch.

Possible implementations of the network nodes are depicted in Figures 3 and 4. The NIC has a FIFO *ring buffer* to temporarily store transit packets arriving from the network not destined to the local PM

[1] Four main packet types are simulated, namely read request, read response, write request and write response.

[2] No distinction is made between a phit (physical transfer unit) and a flit in our study.

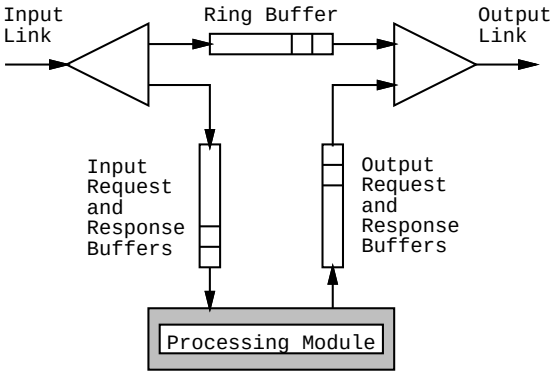


Figure 3: Ring Network Interface (NIC).

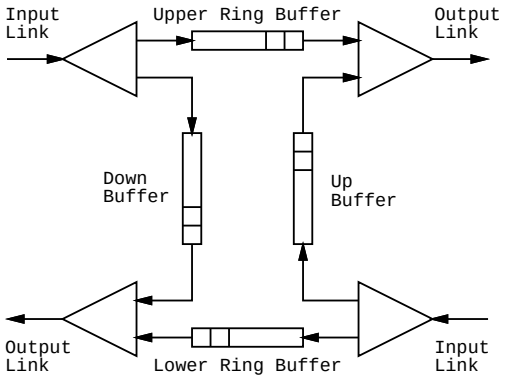


Figure 4: Inter-Ring Interface (IRI).

when the output link is currently transmitting another packet from the local PM. If the ring buffer is empty and no packet is currently being transmitted, then an incoming transit packet will be forwarded to the output link directly, bypassing the ring buffer. The NIC also has a FIFO *input buffer* for storing packets destined for the local PM and a FIFO *output buffer* for storing packets originating from the PM destined for nodes elsewhere in the network. Both of these are split into request and response queues. For best performance, priority for transmission to the next node is given to ring packets either waiting in the bypass buffer or having just arrived from the previous node. Otherwise, if there are packets in one of the output queues then priority is given to response packets over request packets.

The IRI has two ring buffers, one for the lower ring and one for the upper ring. It also has a *down buffer* and an *up buffer*. The down buffer stores packets arriving from the upper ring destined for the lower ring, while the up buffer stores packets arriving from the lower ring destined for the upper ring. The down and up buffers are also split into request and response queues. Switching takes place independently at the lower and upper ring sides. Priority is given to packets that do not change rings. Arriving transit packets block and are placed in the ring buffer if the output

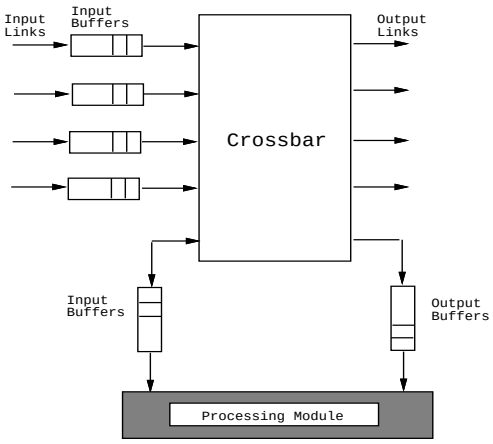


Figure 5: Mesh Network Interface Controller.

link is in the process of transmitting a packet from the up or down queue or when packets are already waiting in the ring buffer.

Both the NIC and IRI have flow control units (not shown in the figures) which are used to signal upstream neighbors when to stop sending packets. Each buffer in the NIC and IRI is large enough to accommodate exactly one packet containing a cache line. We assume all communication occurs synchronously; that is, within a clock cycle, each NIC can transfer one flit to the next adjacent node (if the link is not being blocked), and receive a flit from the previous node it connects to. An IRI, in one clock cycle, can transmit and receive a flit on each ring (if there is no blocking). The routing delay through the NIC and the IRI is assumed to be one network cycle, based again on the NUMAchine implementation [6].[3]

2.2 Mesh system description

In a mesh connected system, there is only one type of network node, namely the mesh *Network Interface Controller* (NIC), to connect processing modules to the mesh. A NIC for a bi-directional mesh is schematically shown in Figure 5. It provides the basic switching function from router inputs to outputs. The mesh NIC is modelled as a 5×5 crossbar with four input/output links from and to its four direct neighbors and one input/output link from and to the local PM. The input links have FIFO buffers to store flits that are blocked in the network.

The NIC performs basic wormhole routing and flow control functions. It examines the header flit of a packet to determine which output link the packet should be forwarded to. The NIC also does proper arbitration if there are competing requests for an output link. We assume that arbitration is round-robin. If a requested output link is not available, then the requesting flit is blocked and stored in the corresponding input buffer(s).

[3] The NUMAchine system implements slotted ring switching and not wormhole switching.

	Channel width	Cache line size	NIC memory requirements		
			cl	4-flit	1-flit
Rings	128b	16B	32B	-	-
		32B	48B	-	-
		64B	80B	-	-
		128B	144B	-	-
Meshes	32b	16B	128B	64B	16B
		32B	192B	64B	16B
		64B	320B	64B	16B
		128B	576B	64B	16B

Table 1: A comparison of memory requirements for ring and mesh NIC buffers of different sizes.

Number of Processors	Cache Line Size			
	16B	32B	64B	128B
4	4	4	4	4
6	6	6	6	2:3
8	8	8	2:4	2:4
12	12	2:6	2:6	3:4
18	2:9	3:6	3:6	3:2:3
24	2:12	3:8	2:2:6	2:3:4
36	3:12	2:3:6	2:3:6	3:3:4
54	2:3:9	3:3:6	3:3:6	3:3:2:3
72	2:3:12	3:3:8	2:2:3:6	2:3:3:4
108	3:3:12	2:3:3:6	2:3:3:6	3:3:3:4

Table 2: Optimal hierarchical ring topology for given number of processors and cache line sizes for workloads with R=1.0 and C=0.04

Under the assumption of constant pin constraints, a 128-bit wide channel for rings with one input and one output connection per ring NIC translates approximately into a 32-bit wide channel for meshes with four input and four output connections per mesh NIC. It is assumed that our mesh NIC can connect all inputs to outputs in a single clock cycle. Once a switch connection between an input and output link is established, it is broken only after the last flit of a packet has been transferred.

We consider mesh NICs with buffers of size 1, 4, and *cl* flits, where *cl* is equal to the number required to hold a cache line with a header. In our study, *cl* will be either 8, 12, 20 or 36 flits to hold a cache line of size 16, 32, 64 or 128 bytes, respectively (assuming 4 flit headers). In contrast, for hierarchical rings, we always assume *cl*-sized ring buffers, where *cl* will be either 2, 3, 5 or 9 flits (assuming 1 flit headers) to hold a cache line of size 16, 32, 64 or 128 bytes, respectively. Since, there is only one ring buffer in a ring NIC as opposed to four of them mesh NICs, this assumption is justifiable when one considers the memory requirements shown in Table 1.

2.3 Simulator

The simulator we use reflects the behavior of the system at the register-transfer level on a cycle-by-cycle basis. It was implemented using the *smpl* simulation library [19]. The batch means method of output analysis was used, with the first batch discarded to account for initialization bias. A hierarchical ring base simulator was validated against measurements taken from the Hector prototype, a hierarchical slotted ring architecture [16, 26]. The base simulator was then extended to model other switching techniques, such as wormhole routing. For 2D meshes, the processor and memory modules are essentially the same as in the ring simulator with new NIC modules that incorporates switching, routing and flow control in meshes.

Our primary measure of performance is the average round-trip access latency, defined as the time from when a request is first issued until the corresponding response is received, measured in network clock cycles. We also consider network utilization measured in percent of maximum network utilization. We assume the network clock cycle is the same as the PM clock cycle,[4] with the exception of hierarchical rings where in some cases (that will be pointed out) the global ring is clocked at a higher speed than the PM clock.

2.4 Benchmark description

We use synthetic micro-benchmarks to drive our simulator in order to accurately evaluate the performance of both mesh and ring interconnection networks under controlled conditions. A series of memory references (i.e. cache misses) is generated at each processor by a Multiprocessor Memory Reference Pattern (M-MRP) address generator similar to the one developed by Saavedra to measure the performance of real systems [22]. More formally, an M-MRP is a set of P uniprocessor memory reference patterns, one for each processor, each accessing its own region of the memory address space. The access regions of the processors typically overlap.

An M-MRP in our simulation is characterized by the following three attributes: 1) the size of the memory region, R, accessed by each processor, 2) the cache miss rate, C, of each processor, and 3) the number of outstanding access transactions, T, a processor is allowed before it blocks. By varying each of these attributes, we can exercise the interconnect in a specific and predictable way and measure how the network responds under controlled conditions.

Parameter $R \in (0, 1)$, the size of the memory access region, allows us to control different degrees of locality and thus the sharing between processors. If P represents the number of processors in the system, then a processor accesses memory in the $\lceil R \cdot P - 1 \rceil$ "closest" PMs, as well as locally. "Closest" is interpreted differently for hierarchical rings and meshes. For rings it is assumed that the processors are logically arranged in a sequence (by projecting them on to a line), and the memory is accessed in the $\lceil R(\cdot P - 1)/2 \rceil$ PMs on either side of the accessing PM, as well as locally. This corresponds to accessing a contiguous memory region centered at the local PM. For meshes, the closest PMs are determined by the number of hops required to reach

[4] In our NUMAchine prototype, the network and PMs operate at 50 MHz cycles.

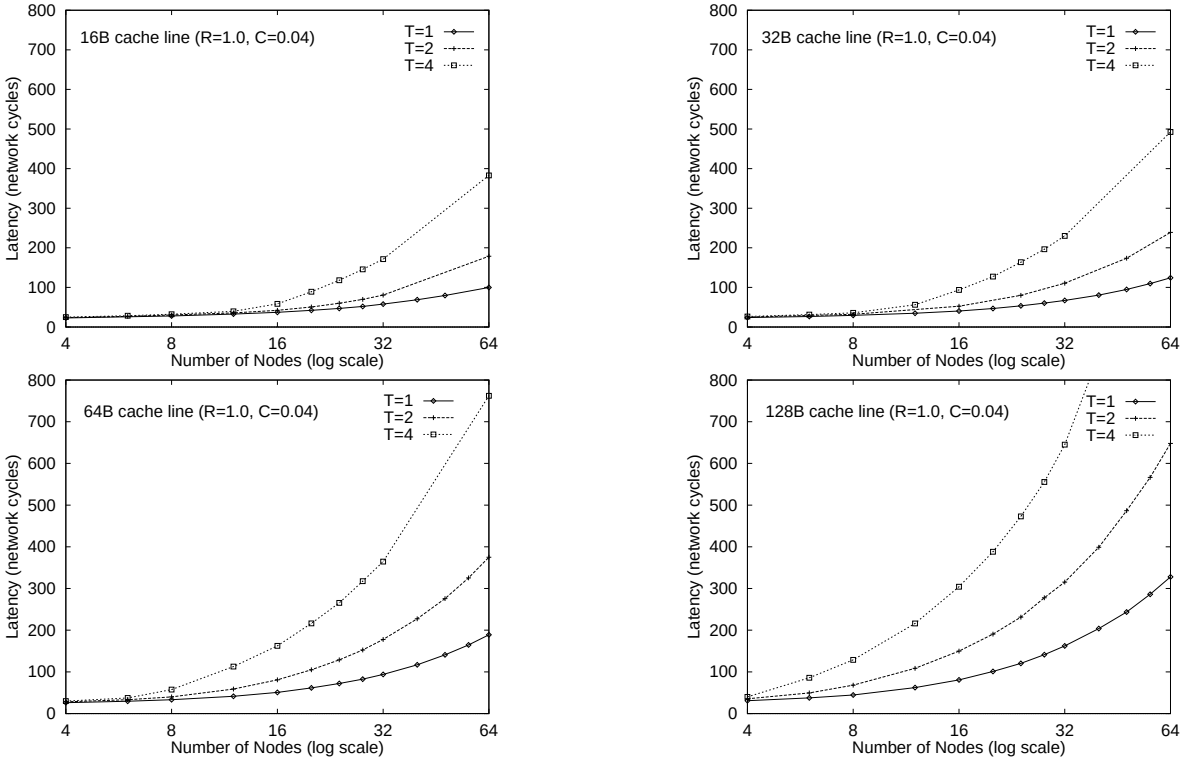


Figure 6: Latency for single rings with different cache line sizes.

them. This corresponds to accessing a non-contiguous memory region. In either case we assume that the sequence of memory references in a given memory region is uniformly distributed and independent across the region.

A processor is allowed to have T outstanding requests before it is required to block for a reply. This parameter is used to model processors with prefetching and/or multi-threading [14]. We assume that the rate at which requests are generated is independent of the number of outstanding requests, which mimics the behavior of multiple context processors.

The offered load is controlled by parameter C, the cache miss rate, and is assumed to be 0.04 (or 25 cycles between cache misses) for all our simulations. This rate is at the higher end of the predicted cache miss ratios of real applications [5, 25]. We assume the probability that the cache miss is a read as 0.7 throughout this study.

3 Hierarchical rings

In this section we analyse the performance of hierarchical ring networks. The topology of a hierarchical ring system greatly affects its performance. In our study, we use the topology that performs best for applications with relatively poor memory access behavior, i.e. no locality and high cache miss rates ($R = 1.0$, $C = 0.04$, $T = 4$). This topology is dependent on the cache line size. Table 2 depicts the best topology for

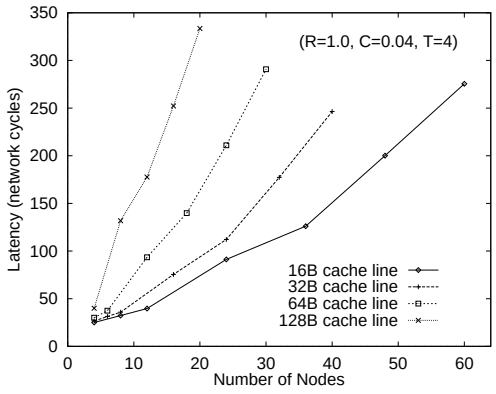
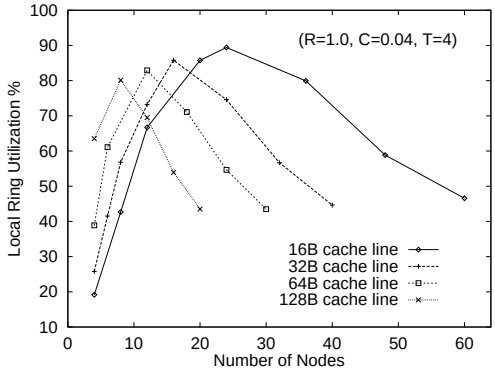


Figure 7: Latency for 2-level ring hierarchies.

the given number of processors and cache line size. The notation "2:3:4" refers to a 3-level hierarchy with 1 global ring, 2 intermediate rings, 3 local rings per intermediate ring, and 4 PMs per local ring.

Figures 6–11 show how we arrived at these topologies. The performance of single ring systems is depicted in Figure 6 for cache line sizes of 16, 32, 64 and 128 bytes, respectively. We present latency curves for PMs supporting 1, 2 and 4 outstanding transactions (T). The results show that single ring systems with 16, 32, 64 and 128 bytes cache line sizes can conserva-



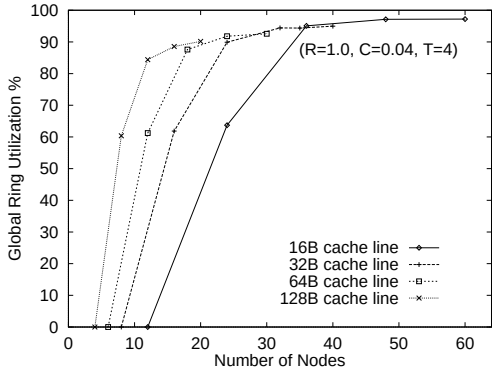


Figure 8: Local and global ring utilization for 2-level ring hierarchies.

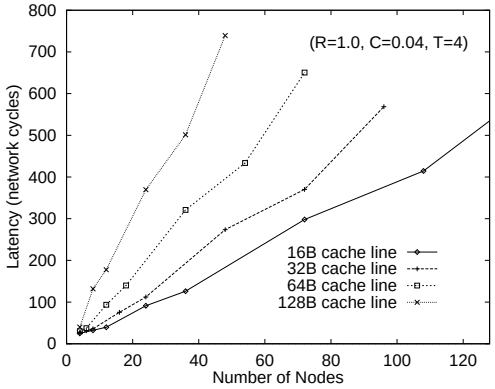


Figure 9: Latency for 3-level ring hierarchies.

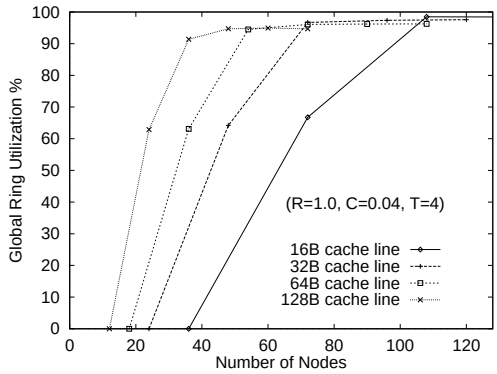


Figure 10: Global ring utilization for 3-level ring hierarchies.

tively sustain 12, 8, 6 and 4 nodes, respectively, with almost no performance degradation.

Figure 7 depicts the performance of systems with two levels of rings, given the same workload with $T = 4$. A second level, *global ring* ring connects local rings, each containing the maximum number of PMs as determined above for the single ring case. The latency curves for values of T less than 4 show the same trends and are not shown. There are two places where there is an increase in the slope of the latency curve for a given cache line size. The first increase occurs when the size forces us to connect two local rings to a second level global ring, causing an increase in the round trip distance. This happens at 12, 8, 6, and 4 processors for cache line sizes 16, 32, 64 and 128 bytes, respectively.

The second increase occurs after we connect three local rings to the global ring which happens at 36, 24, 18 and 12 processors for cache line sizes 16, 32, 64 and 128 bytes, respectively. This second increase is primarily due to bisection bandwidth limitations, as can be seen in Figure 8 which depicts utilization of local and global rings. The global ring utilization almost reaches its full capacity when we connect 3 local rings, and any attempt to connect additional local rings saturates the global ring. The local rings utilization decreases as we connect more local rings to the global ring. This clearly shows that the system performance is bisection bandwidth limited. We conclude that up to three local rings can be sustained in a two level ring hierarchy and, more importantly, is independent of the cache line size.

For 3-level rings, we assume that a global ring connects a number of second level rings, each in a maximum configuration as determined above. Figure 9 shows the average latency as a function of the number of nodes for $T = 4$ and different cache line sizes. Similar to the two level system, there is an increase in the slope of the latency curve as we increase the system size to require a third level ring. A second increase occurs when we try to accommodate more than three second-level rings. This behavior is again independent of the cache line size. We conclude, we can reasonably support 108, 72, 54 and 36 nodes in a system with 3 levels of hierarchy and cache line sizes of 16, 32, 64 and 128 bytes, respectively.

Figure 10 shows the utilization of the global rings for $T = 4$ and different cache line sizes. The global ring saturates once we connect more than 3 2-level rings to the global ring, reinforcing the bisection bandwidth constraints of hierarchical rings.

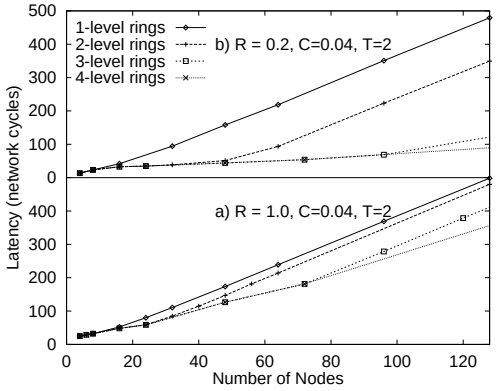


Figure 11: Latency for hierarchical ring systems with different number of levels and 32B cache lines for workloads with a) poor memory locality, R=1.0, b) high memory locality, R=0.2

Figure 11 shows the benefit of having hierarchies in ring-based systems for two different workloads: the first with no memory locality ($R = 1.0$, $C = 0.04$, $T = 2$) and the second has some ($R = 0.2$, $C = 0.04$, $T = 2$). It is evident that each additional level in the hierarchy shifts the latency curve further to the right, thereby allowing a larger number of nodes to be accommodated. For the workload with locality, we see that the benefit of having a hierarchy is significant.

In our further study of rings we limit ourselves to three levels of hierarchy.

4 Mesh networks performance

In this section, we evaluate the performance of mesh networks. As described in Section 2, we assume square, 2-dimensional bi-directional wormhole routed meshes with no end-around connection and simple, deterministic *e-cube* routing. We are brief in our presentation, since our results are compatible with those obtained by other researchers [1, 2, 8].

Figure 12 presents latency curves for the access pattern with $R = 1.0$, $C = 0.04$, and $T = 4$, assuming buffers sizes of 1, 4 or cl flits, where cl is the size required to accommodate a packet containing a cache line. One significant observation from the figure is that the increase in latency as a function of the number of nodes is more moderate than that obtained with hierarchical ring networks. As we increase the size of a mesh network, both the aggregate network bandwidth and the bisection bandwidth scales, whereas in ring networks the bisection bandwidth remained constant even though the aggregate local ring bandwidth increased. In mesh systems with cache line sized buffers, the latency increases by only a factor of between 5 and 7 (depending on cache line size) when the size of the system is scaled by a factor of 30 from 4 to 121 processors. For systems with 4-flit buffers, latency increases by a factor of between 6 and 8 for the same increase in system size, and for systems with 1-flit buffers there is a factor of 9 to 12 increase in latency.

As is evident from Figure 12, the performance im-

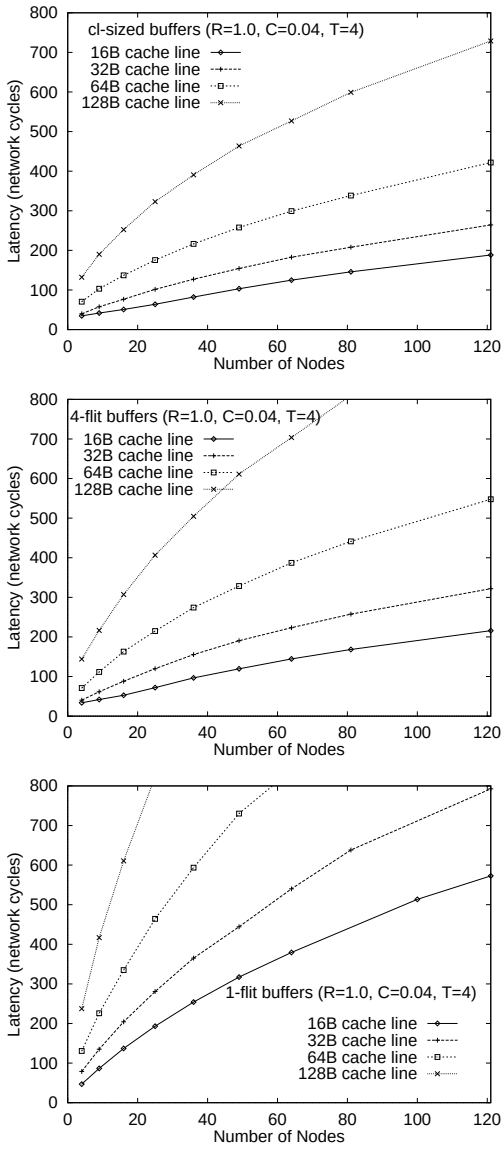


Figure 12: Latency for 2D meshes.

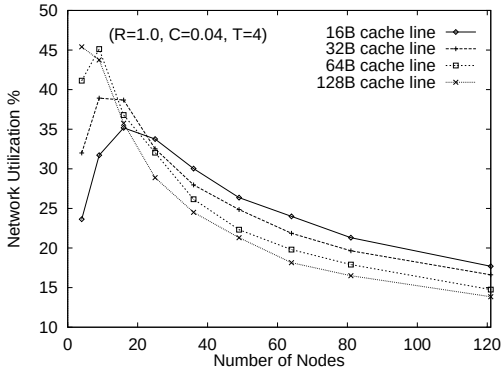


Figure 13: Network util. for meshes with 4-flit buffers.

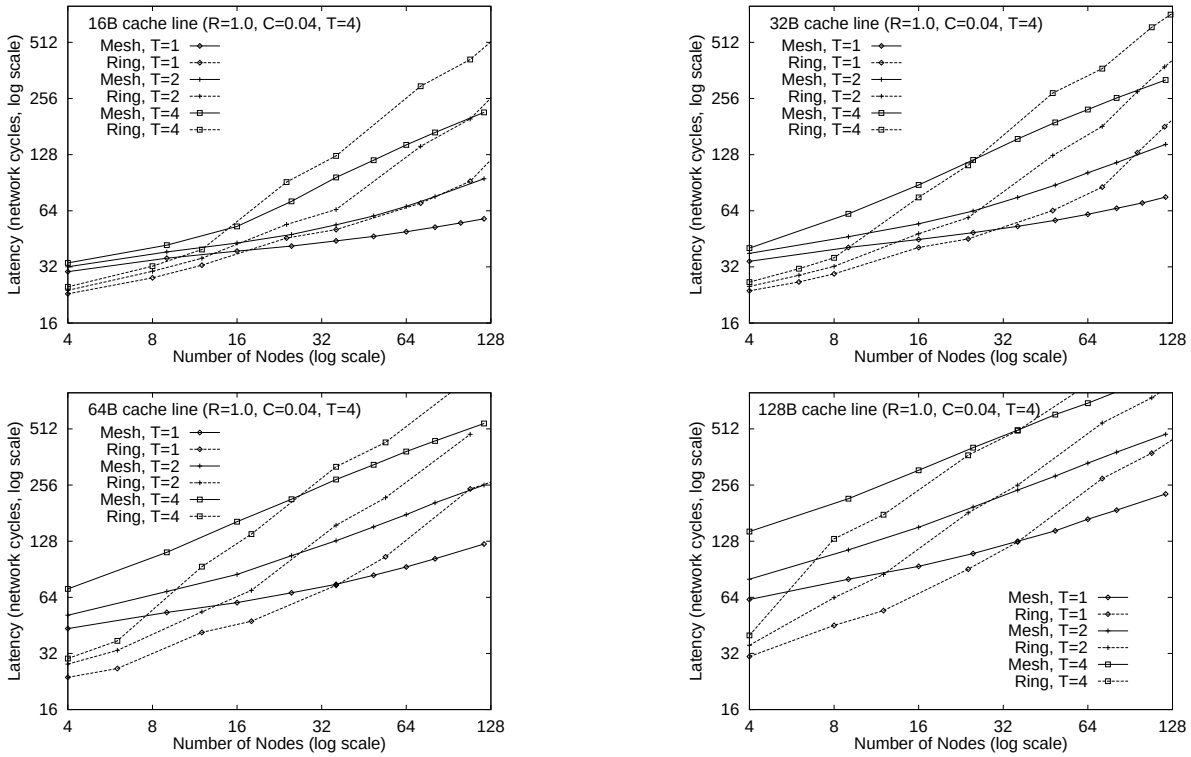


Figure 14: Comparing the latency of meshes with 4-flit buffers and rings for different cache lines.

pact of buffer sizes on mesh networks is quiet appreciable. A 64 processor system with 128 byte cache lines will exhibit 3 times higher latency with 1-flit buffers and 1.3 times higher latency with 4-flit buffers when compared to that of cl-sized buffers. However, the memory requirements for cache line sized buffers are 144 times higher than that for 1-flit buffers (with a 128-byte cache line size), whereas the memory requirements for 4-flit buffers are only 16 times higher than that for 1-flit buffers for the same cache line size. In other words, for a system with 128 byte cache lines and 64 processors there is only a 25% decrease in latency when we go from 4-flit buffers to cache line sized buffers, but there is a 9 times increase in the total on-chip memory requirements.

Figure 13 presents the network utilization curves for a system with 4-flit buffers. The network utilization reaches a peak relatively early and begins to decrease monotonically thereafter as the system becomes larger. Peak network utilization occurs at 16, 9, 9 and 4 nodes when the cache line size is 16, 32, 64 and 128 bytes, respectively. The utilization drops to less than 20% for 121 processor systems regardless of the cache line sizes considered. The network utilization is lower for larger systems because the average distance a packet must travel increases, with an attendant increase in the probability of blocking (confirmed by the increase in the average NIC delay of a packet).

5 Comparative performance of rings and meshes

In this section, we compare the performance of hierarchical rings and meshes, both for the workload with no locality we have been considering so far ($R = 1.0$, $C = 0.04, T = 4$) and also for a workload with a higher degree of locality ($R \leq 0.3$, $C = 0.04$, $T = 4$). In our comparison, we generally try to be fair, although we slightly favor meshes. For example, our memory access locality model favors meshes, as it minimizes the number of hops in meshes, but not so for ring-based systems. We assume the rings are wormhole routed, even though slotted rings tend to perform somewhat better [21]. Also, we assume the same routing speed for both mesh and ring NICs, although routing in rings is simpler due to fewer number of connections and can generally be done faster. When considering on-chip memory requirements for buffers, assuming cache line sized buffers for meshes favors meshes, while assuming 1-flit buffers for meshes favors hierarchical rings.

5.1 Access patterns with no memory locality

Figure 14 compares the latency of rings and meshes. In these experiments, the mesh NICs all have 4-flit buffers. Each figure represents a different cache line size[5] Generally, rings perform better than meshes

[5] We use log-log graphs here, since most of the cross-over points are before 36 processors.

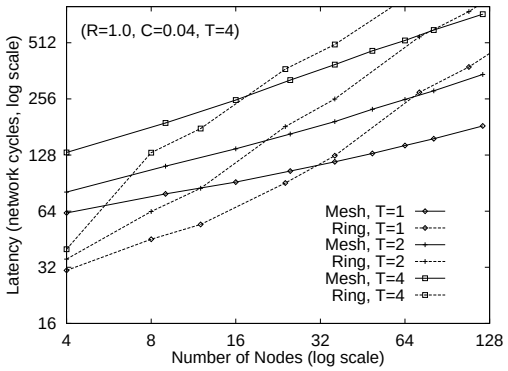


Figure 15: Comparing the latency of meshes with cl-sized buffers and rings for 128B cache lines.

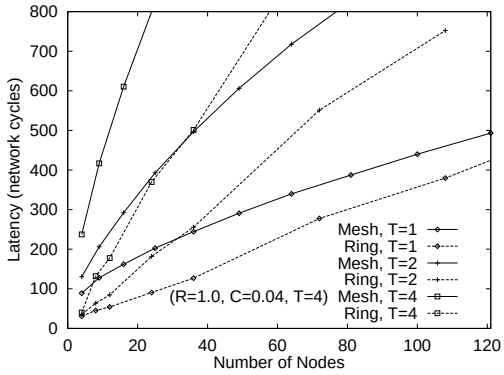


Figure 16: Comparing the latency of meshes with 1-flit buffers and rings for 128B cache lines.

when connecting a small number of nodes, but meshes perform better in systems with a larger number of nodes. We define the *cross-over* point as the number of nodes where the switch over occurs. The results show that the cross-over point depends on the cache line size; the larger the cache line size, the higher it is. The cross-over points are 16, 25, 27 and 36 nodes for 16, 32, 64 and 128 bytes cache line systems, respectively. With larger cache lines, the relative length of a worm in a mesh network is higher than in a ring network when compared to smaller cache lines, so the probability of blocking increases. The cross-over point is pretty much independent of the number of outstanding transactions (with the exception of $T = 1$ where the cross-over point is higher). However, the relative difference in performance between the two network types becomes larger with larger values of T.

For meshes with cache line sized buffers, the cross-over points are lower, lying between 16-30 nodes depending on the value of T. This is shown in Figure 15 for a cache line size of 128 bytes. Systems with other cache line sizes have the same cross-over points (not shown), because there is zero probability that a worm will stall more than one link.

For meshes with 1-flit buffers, then rings outperform meshes for all cache line sizes: the cross-over point lies above 121 nodes for all cache line sizes considered. The mesh performs poorer than hierarchical rings, even for 16 byte cache line sizes (not shown), primarily because of the increased probability of worms blocking over multiple links. Figure 16 shows these curves for a 128 byte cache line size system.

5.2 Access patterns with locality

In this section we consider memory access patterns with higher degrees of locality. Figure 17 compares the latency of rings and meshes for 16, 32, 64 and 128 byte cache line systems with 4-flit mesh buffers and $T = 4$. Latency curves are shown for three different values of R, namely 0.1, 0.2, and 0.3. The results show that even with moderate locality (i.e. $R = 0.3$), rings outperform meshes for systems with up to 121 processors. This is true for all cache line systems considered with the exception of 16 byte cache line systems, where the performance of rings and meshes are similar.

For 32 byte cache line systems, rings perform on average 20% better than meshes for access patterns with $R = 0.3$ or less. For 64 and 128 byte cache line systems, rings perform on average 30% better than meshes. Interestingly, the difference in performance is larger for $R = 0.2$ than for $R = 0.1$. This is because in meshes, most of the target PMs are direct neighbors when $R = 0.1$, while with $R = 0.2$ larger distances must be traversed with increased attendant contention.

The above discussion assumed 4-flit buffers. For *cl*-sized mesh buffers, increased locality has the effect of increasing the cross-over point, as shown in Figure 18 for 128 byte cache line systems. For $R = 0.3$ or less, the cross-over point is 45 processors or more. This shows that even with *cl*-sized mesh buffers, meshes perform worse than rings for small and medium scale systems for workloads with moderate to good locality.

6 Increasing the bisection bandwidth of rings

The performance and scalability of hierarchical rings are clearly limited by their constant bisection bandwidth. In this section, we show that by increasing the bandwidth of the global ring (and thus the bisection bandwidth), we can connect additional lower level rings to the global ring without worsening the average memory access latency. Targeting just the global ring is effective, because the utilization of the lower level rings is low, especially when the global ring is saturated.

The bandwidth of the global ring can be increased either by increasing the width of the ring or the speed of the ring. Moreover, using faster and more expensive technology is viable in this case, since it does not affect the total cost of the system by much with the global ring constituting a small portion of the system. For example, the global ring of the NUMAchine multiprocessor will be implemented using free-space optical technology [24], which will give us an aggregate global ring bandwidth of 1 Tb per second.

Here, we explore the option of clocking the global

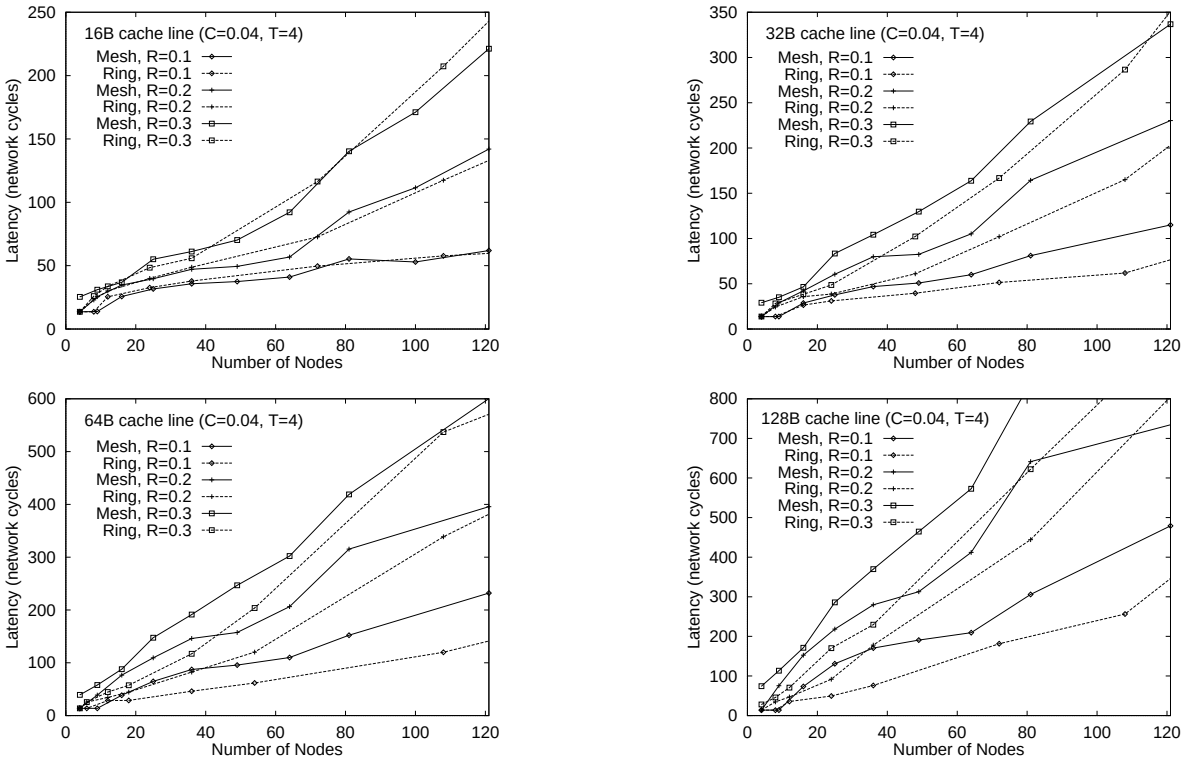


Figure 17: Comparing the latency of meshes with 4-flit buffers and rings for different values of R and cache line sizes.

ring at a speed higher than that of the local and intermediate rings. Figure 19 presents latency as a function of the number of nodes for a 3-level hierarchical ring system with the global ring being clocked at twice the speed of the local and intermediate rings. To allow easy comparison, the figure also depicts the latencies of the corresponding systems with the global ring running at normal speed. The curves are presented for 32, 64 and 128 byte cache line systems for a value of $R = 1.0$, $C = 0.04$, and $T = 4$. The results show that a third level global ring can then sustain up to 5 second-level rings, as compared to 3 second-level rings when global rings run at normal speed. Five and not six second-level rings can be sustained, because the amount of traffic a second level ring routes through the global ring increases with the number of second-level rings in our memory access model. A system with 5 second-level rings has 180, 120, 90 and 60 processors when the cache line sizes are 16, 32, 64 and 128 bytes respectively; this compares to 108, 72, 54 and 36 processors for a system with normal speed global rings. From Figure 20, it is evident that the utilization of the double speed global rings increases more slowly and in a more linear fashion than of normal speed global rings.

Figure 21 compares the latencies of meshes and hierarchical rings with the global ring running at twice the normal speed for a workload with no memory locality (i.e. $R = 1.0$, $C = 0.04$, $T = 4$). The figure shows the latency curves for 32, 64 and 128 byte cache line systems and $T = 4$. Hierarchical rings with 128 byte cache line sizes perform 10-20% better than meshes for systems of these sizes. However, for 32 and 64 byte cache line systems there is no relative improvement in ring-based systems; the cross-over points are largely the same as with normal speed global rings, because the cross-over points occur before there is a need for a third level global ring. Hence, clocking the

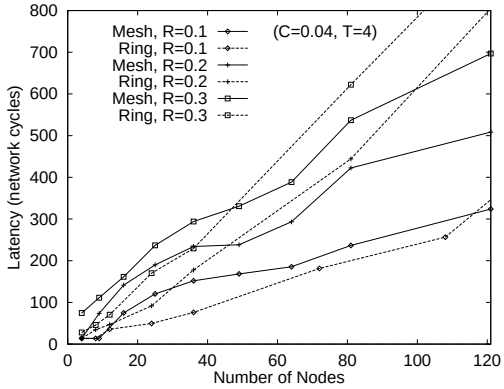


Figure 18: Comparing the latency of meshes with *cl*-sized buffers and rings for 128B cache lines.

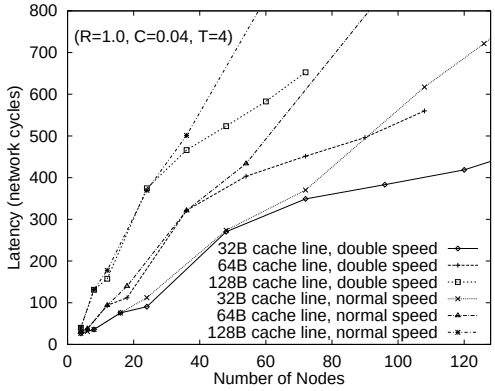


Figure 19: Latency of 3-level ring hierarchies with normal and double speed global rings.

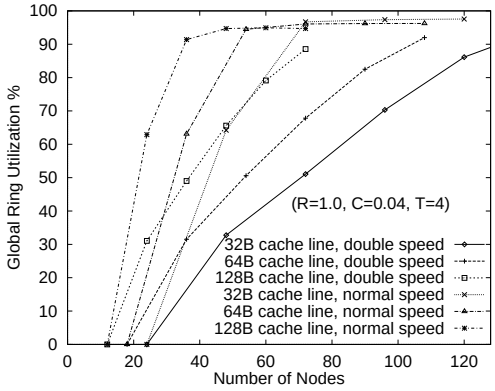


Figure 20: Global ring utilization for 3-level ring hierarchies with normal and double speed global rings.

global ring at twice the speed has no effect.

We conclude that by doubling the speed of the global ring, hierarchical ring systems can be made to scale to reasonably large system sizes and they perform better than meshes for large cache line sizes, even for workloads that exhibit no locality.

7 Conclusion

Our simulation study has shown that:

• meshes have superior scaling characteristics relative to hierarchical rings.

• hierarchical rings perform significantly better than meshes for system sizes up to 121 processors if the workload exhibits moderate to high memory access locality.

Even if there is no memory locality:

• hierarchical ring systems perform better than meshes for systems with large cache lines either if the system is small, or if the global ring has double the normal bandwidth.

We conclude that hierarchical ring systems are an attractive alternative to meshes, especially for system sizes of 128 processors or less, sizes that correspond to those system that are expected to make up the vast majority of the shared-memory multiprocessor market.

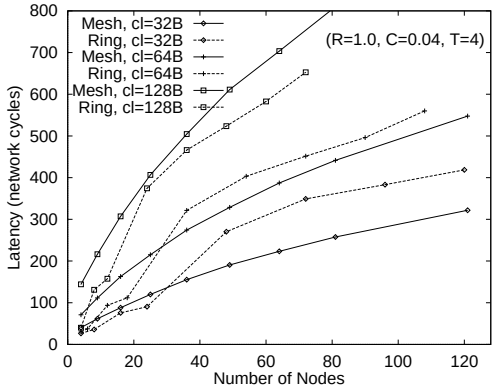


Figure 21: Comparing the latency of meshes with 4-flit buffers and 3-level ring hierarchies with double speed global rings.

References

[1] V.K. Adve and M.K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 3, pp. 225-246, March 1994.

[2] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398-412, April 1991.

[3] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B. Lim, K. Mackenzie, and D. Yeung, "The MIT Alewife machine: Architecture and performance," *Proc. Intl. Symp. on Computer Architecture*, pp. 2-13, 1993.

[4] L.A. Barroso and M. Dubois, "Performance evaluation of the slotted ring multiprocessor," *IEEE Trans. on Computers*, vol.44, no.7, pp. 878-890, July 1995.

[5] B. Boothe and A. Ranade, "Improved multithreaded techniques for hiding communication latency in multiprocessors," in *Proc. Intl. Symp. on Computer Architecture*, pp. 214-223, May 1992.

[6] S. Brown et al., "The NUMAchine multiprocessor", CSRI-TR-324, CSRI, University of Toronto, 1995.

[7] H. Burkhardt, "Overview of the KSR1 computer system," KSR-TR 9202001, Kendall Square Research, 1992.

[8] A.A. Chien, "A cost and speed model for k-ary n-cube wormhole routers," in *Proc. Hot Interconnects '93*, pp 3.1.1-3.1.7, August 1993.

[9] Convex Computer Corporation. *Convex Exemplar system overview*. URL: http://www.convex.com/

[10] Cray Research Inc. *Cray Origin2000 system overview*. URL: http://www.cray.com/

[11] W.J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp 187-196, March 1986.

[12] W.J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, vol. 39, no. 6, pp. 775-785, June 1990.

[13] K.I. Farkas, Z. Vranesic, and M. Stumm, "Scalable cache consistency for hierarchically structured multiprocessors," *The Journal of Supercomputing*, vol. 8, no. 4, pp. 345-369, 1992.

[14] A. Gupta, J. Hennessy, K. Gharachorloo, T. Mowry, and W. D. Weber, "Comparative evaluation of latency reducing and tolerating techniques," in *Proc. Intl. Symp. on Computer Architecture*, pp. 254-265, May 1991.

[15] V.C. Hamacher and H. Jiang, "Comparison of mesh and hierarchical networks for multiprocessor," *Proc. Intl. Conf. on Parallel Processing*, Vol. I, pages 67-71, August 1994.

[16] M. Holliday and M. Stumm, "Performance evaluation of hierarchical ring-based shared memory multiprocessors", *IEEE Trans. on Computers*, vol. 43, no. 11, pp. 52-67, Jan 1994.

[17] 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," *Proc. Intl. Symp. on Computer Architecture*, pp. 302-313, April 1994.

[18] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, and J. Hennessy, "The DASH prototype: Logic overhead and performance," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 41-61, Jan 1993.

[19] M.H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, MIT Press, 1987.

[20] G. Ravindran and M. Stumm, "Hierarchical ring topologies and the effect of their bisection bandwidth constraints," in *Proc. Intl. Conf. on Parallel Processing*, pp. I/51-55, 1995.

[21] G. Ravindran and M. Stumm, "A comparison of blocking and non-blocking packet switching techniques in hierarchical ring networks," in *IEICE Trans. on Information and Systems*, vol. E79-D, no. 8, pp. 1130-1138, August 1996.

[22] R.H. Saavedra, "Characterizing the performance space of shared memory computers using micro-benchmarks," *Proc. Hot Interconnects '93*, August 1993.

[23] K.G. Shin and S.W. Daniel, "Analysis and implementation of hybrid switching," in *Proc. Intl. Conf. on Computer Architecture*, pp. 211-219, 1995.

[24] T.H. Szymanski and H.S. Hinton, "Reconfigurable intelligent optical backplane for parallel computing and communications," in *Journal of Applied Optics*, vol. 35, no. 8, pp. 1253-1268, March 1996.

[25] J. Tsai and A. Agarwal, "Analyzing multiprocessor cache behavior through data reference modelling," in *Proc. ACM SIGMETRICS Conf. on Measurement and Modelling of Computer Systems*, pp. 236-247, 1993.

[26] Z.G. Vranesic, M. Stumm, D. Lewis, and R. White, "Hector: A hierarchically structured shared-memory multiprocessor," *IEEE Computer*, vol. 24, no. 1, pp. 72-78, Jan. 1991.