Comparative Modeling and Evaluation of CC-NUMA and COMA on Hierarchical Ring Architectures

Xiaodong Zhang, *Senior Member, IEEE*, and Yong Yan

***Abstract*—Parallel computing performance on scalable shared-memory architectures is affected by the structure of the interconnection networks linking processors to memory modules and on the efficiency of the memory/cache management systems. Cache Coherence Nonuniform Memory Access (CC-NUMA) and Cache Only Memory Access (COMA) are two effective memory systems, and the hierarchical ring structure is an efficient interconnection network in hardware. This paper focuses on comparative performance modeling and evaluation of CC-NUMA and COMA on a hierarchical ring shared-memory architecture. Analytical models for the two memory systems for comparative evaluation are presented. Intensive performance measurements on data migrations have been conducted on the KSR-1, a COMA hierarchical ring shared-memory machine. Experimental results support the analytical models, and we present practical observations and comparisons of the two cache coherence memory systems. Our analytical and experimental results show that a COMA system balances the work load well. However the overhead of frequent data movement may match the gains obtained from improving load balance. We believe our performance results could be further generalized to the two memory systems on a hierarchical network architecture. Although a CC-NUMA system may not automatically balance the load at the system level, it provides an option for a user to explicitly handle data locality for a possible performance improvement.**

Index Terms—Cache coherence, CC-NUMA, COMA, performance modeling and measurements, slotted rings, shared-memory, the KSR1.

I. Introduction

LARGE scale shared-memory architectures provide shared address space supported by physically distributed memory. The strength of such systems comes from combining the scalability of network-based architectures with the convenience of the shared-memory programming model. Computing performance on such systems is affected by two important architecture and system design factors: the interconnection network and the memory system structure. The choice of interconnection networks to link processor nodes to cache/memory modules can make nonuniform memory access (NUMA) times vary drastically, depending upon the particular access patterns involved. A hierarchical ring structure is an interesting base on which to build large scale shared-memory multiprocessors. In large NUMA architectures, the memory system organization also affects communication latency. With respect to the kinds of memory organizations utilized, NUMA memory systems can be classified into the following three types in terms of data migration and coherence:

- **Non-CC-NUMA** stands for non-cache-coherent NUMA. This type of architecture either supports no local caches at all (e.g., the BBN GP1000 [1]), or provides a local cache that disallows caching of shared data in order to avoid the cache coherence problem (e.g., the BBN TC2000 [1]).
- **CC-NUMA** stands for cache-coherent NUMA, where each processor node consists of a processor with an associated cache, and a designated portion of the global shared memory. Cache coherence for a large scale shared-memory system is usually maintained by a directory-based protocol. Examples of such a system are the Stanford DASH [8], and the University of Toronto's Hector [13].
- **COMA** stands for cache-only memory architecture. Like CC-NUMA, each processor node has a processor, a cache, and a designated portion of the global shared memory. The difference, however, is that the memory associated with each node is augmented to act as a large cache. Consistency among cache blocks in the system is maintained using a cache coherence protocol. A COMA system allows transparent migration and replication of data items to the nodes where they are referenced. Example systems are the Kendall Square Research's KSR-1 [7], and the Swedish Institute of Computer Science's Data Diffusion Machine (DDM) [5].

A comparative performance evaluation between CC-NUMA and COMA models has been conducted by using simulations in [12], where dynamic network contention is not a major consideration. In addition, only 16 processors were simulated on relatively small problem sizes. However, both CC-NUMA and COMA systems are targeted at large scale architectures on large problem sizes. Another experimental measurement has been recently conducted to compare performance of the DASH (CC-NUMA on cluster networks) and the KSR-1 (COMA on hierarchical rings) [11]. We believe that further work needs to be done in order to more precisely and completely provide insight into the overhead effects inherent in the two memory systems. First, a comparative evaluation needs to carefully take into consideration the network contention which varies be-

Manuscript received Nov. 16, 1993; accepted Mar. 22, 1995.
The authors are with the High Performance Computing and Software Laboratory, University of Texas at San Antonio, San Antonio, TX 78249; e-mail: zhang@ringer.cs.utsa.edu, yyan@dragon.cs.utsa.edu.
To order reprints of this article, e-mail: transactions@computer.org, and reference IEEECS Log Number D95063.

tween the two memory system designs and among different interconnection network architectures. Second, a comparative evaluation between the two memory systems should be done based on a particular network structure, because different network structures can make the two memory systems perform and behave differently. COMA and CC-NUMA mainly differ in the requirements of the network and the difference in data locality, migration, and replication. Previous studies combine the two factors in the evaluation which limit the possibility to isolate the effects of either of them. Our study using a particular network is concerned with the isolated effects of the memory systems. Thirdly, a comparative evaluation should provide information to identify and distinguish the effects on computing performance caused by the memory systems and by the interconnection network structure. Finally, besides simulation and models, a comparative evaluation should also present experimental results on real CC-NUMA and COMA architectures to provide practical observations of program executions. This paper differs from the study cited in [11] and [12], with respect to the four points mentioned above. In addition, this modeling and evaluation work complements our experimental and application programming work on various network-based shared memory multiprocessors in the High Performance Computing and Software Laboratory. (See e.g., [14], [15], and [17]).

A hierarchical ring structure is the particular interconnection network in this comparative performance evaluation study of both the CC-NUMA and COMA systems. The organization of the rest of the paper is as follows. In Section II, we begin with detailed descriptions of the modeled ring network and cache coherence protocols of the CC-NUMA and COMA. The performance parameters of the analysis are also presented in Section II. The analytical models of the two memory systems on the hierarchical ring network are presented in Sections III and IV. Section V gives the comparative performance evaluation between the two memory systems in terms of data migration, load distributions, network contention and communication bandwidth. Section VI reports our experiments of cache coherence and data access patterns under the two memory systems on the KSR-1 to verify and support the analytical and simulation results presented in Section V. Finally, we give a summary and conclusions in Section VII.

II. The Definitions for the Hierarchical Ring Based CC-NUMA and COMA

In order to clarify and simplify the presentation without losing the generality of a hierarchical ring, we consider a two level ring-based shared-memory architecture for both the CC-NUMA and COMA systems.

A. The Architecture for the CC-NUMA and the COMA

The CC-NUMA and the COMA systems to be discussed in the following sections share the same ring interconnection network architecture shown in Fig. 1. This architecture has the following hardware and software structures, functions and parameters which are similar to the ones in [7] and [13]:

1) The architecture consists of a global ring and M local rings. Each local ring is connected to the global ring through an inter-ring port with a pair of buffers. The buffers are used to temporarily store and forward packets passing the port. The global ring has M equally sized slots connecting to the M local rings. Each local ring has N equally sized slots, each of which connects to a station module.
2) Each station module consists of one main memory module, one subcache and one processor. In the CC-NUMA system, the main memory module is a home addressed memory which occupies a unique contiguous portion of a flat, global (physical) address space. In the COMA system, the main memory module is a big-capacity cache which results in dynamic mapping between the context address and the system logic address through segment translation tables.
3) Both the global ring and local rings are rotated constantly. The rotation period between each slot is defined as t_r. A processor node in a local ring ready to transmit a message waits until an empty slot is available. The response and a request, such as a read/write will be rotated back to the requesting processor.
4) Each inter-ring port is able to determine the path of the message packets passed through it. In the COMA system, each port keeps a directory to record the mapping relationships among the cache memory modules in the corresponding local ring. In CC-NUMA, each port can determine the destination of the message packets according to the home address carried by the message packet.

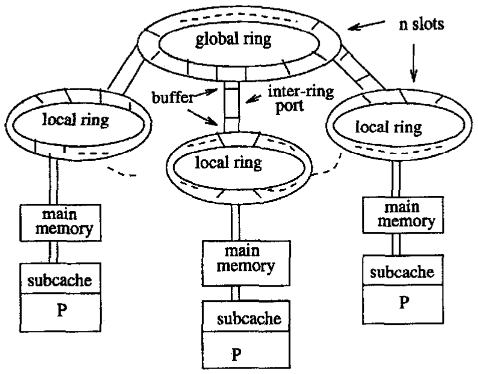


Fig. 1. The architecture of a two-level ring-based CC-NUMA/COMA system.

B. Cache Coherence Protocols of CC-NUMA and COMA

The cache coherence protocols in our models are based on the available ones which have been proposed/implemented on hierarchical ring architectures (see e.g., [2] and [4] and [7]). In order to compare the differences between the two memory systems, similar hierarchical data directories and cache coherence protocols are designed in each system. In both systems, sequential consistency is preserved.

B.1. Cache coherence protocol of CC-NUMA

1) Hierarchical directory
Each processor maintains a local directory in its local cache, which records the data allocation information in the local cache. Each local ring maintains a global directory built in the inter-ring port, which records the data allocation information in the local ring.
2) Ownerships of a shared data
- Share: there is more than one copy of the shared data existing in other memory modules.
- Exclusive: the current copy is the only one in the system.

3) Read/write protocol
- Reading the shared-data—the processor will get the data in its local memory if it is available there, otherwise it will get it from one of the memory modules in the local ring, or in a remote ring through searching. The newly loaded copy will have the "share" ownership.
- Writing the shared-data—the processor will either write the data locally if it is available or will do a remote-write in the destination memory module. The associated invalidation operations are defined as follows.

  The invalidation of shared data is conducted during the process of a write request traveling to the home node and returning from the home node. Each time a write request passes by a global directory which has copies of the requested data, it will produce a invalidation packet to invalidate the copies in the local ring.

B.2. Cache coherence protocol of COMA

1) Hierarchical directory
It has the same structure as the one defined for CC-NUMA.
2) Ownerships of cache segments
- Copy: there is more than one copy of a physical address segment in the system.
- Nonexclusive Copy: it has the same features as "Copy" except the location is the owner of the physical address segment.
- Exclusive: there is only one valid copy of the physical address segment in the system.
- Invalidation: The copy in the cache segment is not valid.

3) Read/write protocol
- Read shared data—if the copy exists in the local cache, the processor performs the read immediately. Otherwise, it sends a probe packet into the local ring or remote rings to search for a copy of its required physical address segment. The processor will receive a copy with the ownership of Copy. The ownership of the cache copy in the destination module will be changed into "Nonexclusive Copy" if its original ownership is Exclusive.
- Write shared data—the processor will first search the owner of the shared data through the entire ring hierarchy. As soon as the owner is found in a cache module, the processor loads the data back to its local cache and invalidates all the existing copies in the system. After the invalidation, the processor perform the write in the local cache. The updated data copy becomes "Exclusive."

C. Performance Parameters and Assumptions

In order to fairly compare the performance differences between CC-NUMA and COMA, it is necessary to define a common evaluation base. The models presented in the next sections are based on the following common performance parameters:

1) λ: request miss rate of each local cache.
2) λ_h: a fraction of λ directing to a hot-spot address segment.
3) λ_l: a fraction of $\lambda(1-\lambda_h)$ directing to memory modules on its local ring.
4) λ_r: read miss fraction of λ.
5) λ_w: write miss fraction of λ.
6) t_r: rotation period of each ring.
7) N: the number of stations connected to each local ring (or the number of slots in a local ring).
8) M: the number of local rings connected to the global ring.
9) N_c: the number of cache segments in each local cache memory (this parameter is only used for the COMA system).

Furthermore, we assume:

1) The request miss rate of each local cache follows a Poisson process.
2) The local request rate and the nonlocal request rate are uniformly distributed.
3) In the request sequence of each station, the read misses and write misses to a data location are uniformly distributed.
4) One message packet can be completely carried by one slot which only conveys this message packet. So the successive slots behave independently.
5) When a station receives a message packet from a slot, it will produce a reply into the same slot without any delay.

In this paper, major latency analyses for both memory systems on the ring architecture are based on evaluating two important performance factors. First, the ring network contention is modeled by studying hot spot effects. Second, analytical models of read/write miss latencies are constructed using the network contention models associated with the cache coherence protocols. The M/G/1 model is a major mathematical tool to derive the analytical latency formulas. In the following two sections, we present the objectives, assumptions and major results of each model to evaluate CC-NUMA and COMA systems on a hierarchical ring architecture. For detailed derivation process of the mathematical models, the interested reader may refer to Appendices A and B.

III. AN ANALYTICAL MODEL FOR THE HIERARCHICAL RING BASED CC-NUMA

A. Network Contention

In a hierarchical CC-NUMA system, network contention can be well characterized by a hot spot environment where a large number of processors try to access a globally shared variable across the network. In this case, a hierarchical ring is divided into three regions in terms of network activities: *hot local ring* which is the local ring where the hot spot is located, *cool local rings* which are the rest of the local rings without the presence of the hot spot, and the global ring. A comprehensive access delay model for the entire hierarchical ring in the presence of the hot spot is presented based on contention in each of the three parts of the rings. In Appendix A, the following CC-NUMA latency factors are obtained:

- $\overline{d}_n$: the mean waiting time for a message to find an empty slot in a cool local ring.
- $\overline{q}_{cool\_lport}$: the queuing time of a message in the interface port from the global ring to a cool local ring.
- $\overline{d}_h$: the mean waiting time for a message to find an empty slot in the hot local ring.
- $\overline{q}_{hot\_lport}$: the mean queuing time of a message in the interface port from the global ring to the hot local ring.
- $\overline{q}_{hot\_gport}$: the mean queuing time of a message in the interface port from the hot local ring to the global ring.
- $\overline{q}_{cool\_gport}$: the mean queuing time of a message in the interface port from a cool local ring to the global ring.

B. Latency of a Remote-Write to the Hot Spot

A remote-write to the hot spot will be satisfied in the following two possible situations:

1) The write request is from the hot local ring, with probability of $\frac{1}{M}$:

This request only needs to travel the hot local ring for one circle. The traveling time, denoted by $\overline{T}_{l\_numa}$, consists of the time for the source processor to find an empty slot on the hot local ring and the time for the request to travel the hot local ring for one circle:

$$\overline{T}_{l\_numa} = \overline{d}_h + Nt_r. \tag{3.1}$$

2) The write request is from a cool local ring, with probability of $\frac{M-1}{M}$:

This write will access the hot memory remotely. The remote-write time, denoted by $\overline{T}_{g\_numa}$, consists of four parts: the time from the source cool ring to the global ring, the time from the global ring to the hot ring, the time for searching the destination processor in the hot ring, and the time for the data packet to go back to the source processor:

$$\begin{aligned}\overline{T}_{g\_numa} &= \overline{d}_n + \overline{q}_{cool\_gport} + \overline{q}_{hot\_gport} \\ &\quad + \overline{q}_{hot\_lport} + \overline{q}_{cool\_lport} + t_r(M+2N).\end{aligned} \tag{3.2}$$

Therefore, the average latency of a remote-write to the hot spot is

$$\overline{T}_{w\_numa} = \frac{\overline{T}_{l\_numa}}{M} + \frac{(M-1)\overline{T}_{g\_numa}}{M} \tag{3.3}$$

C. Latency of a Remote-Read to the Hot Spot

A read-miss process is more complex than that of a write-miss because multiple copies of the data may exist in the system. In general, the process of a remote-read can be described by the state transition graph shown in Fig. 2. In Fig. 2, O represents the initial state, L represents the state where the requesting processor receives the data from the local ring, G represents the state where the requesting processor receives data from a remote ring, and T_l and T_g represent the latency in states L and G, respectively.

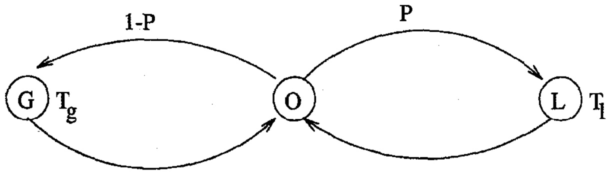


Fig. 2. Transition graph of a remote-read to the hot spot.

Because we have assumed that read misses and write misses are uniformly distributed in the request sequences, whether a data item has multiply copies distributed in other memory modules is determined by the relative ratio of the read miss rate to the write rate. The transition probability P can be determined as follows:

1) When $\lambda_r \leq \lambda_w$, each read miss must be preceded by a write miss. So the probability for a hot read to get the data item from a copy of the home data is zero. In this case, the probability P of reading the hot memory module in the local ring equals to the probability of the local ring's becoming the hot ring, which is $\frac{1}{M}$.

2) When $\lambda_r > \lambda_w$, each write miss follows $\frac{\lambda_r}{\lambda_w}$ read-misses where only the first read-miss visits the home data and the other read-misses visit a copy of the home data. So the probability for a read miss to visit the home data is $\frac{\lambda_w}{\lambda_r}$. Moreover, the probability for the hot data item to be located on a different ring as a request is $\frac{M-1}{M}$. Therefore, the probability P for a request to get the data item from a copy of the home data or from the home data on the local ring is $1-\frac{(M-1)\lambda_w}{M\lambda_r}$.

Concluding the above analyses, the transition probability P can be represented as

$$P = \begin{cases} \frac{1}{M} & \lambda_w \geq \lambda_r, \\ 1-\frac{\lambda_w(M-1)}{M\lambda_r} & \text{otherwise.} \end{cases} \tag{3.4}$$

By (3.4), the latency of a read-miss to the hot spot is

$$\overline{T}_{r\_numa} = PT_l + (1-P)T_g, \tag{3.5}$$

where T_l and T_g are computed under the following two conditions:

1) $\lambda_w \geq \lambda_r$:
A read always visits the hot data item in the home node because no copies of the hot data item exist in this situation. Hence, by (3.1) and (3.2), we have $T_l = \overline{T}_{l\_numa}$ and $T_g = \overline{T}_{g\_numa}$.

2) $\lambda_w < \lambda_r$:
In this case, the remote-read miss latency T_g is $\overline{T}_{g\_numa}$. The local-read miss latency T_l is

$$T_l = \frac{1}{M}\left(\overline{d}_h + NT_r\right) + \frac{(M-1)\left(\overline{d}_n + Nt_r\right)}{M} \qquad (3.6)$$

where $(\overline{d}_n + Nt_r)$ is the searching time of a read-miss in a nonhot local ring and $(\overline{d}_h + Nt_r)$ is the searching time of a read-miss on the hot local ring.

IV. An Analytical Model for the Hierarchical Ring Based COMA

Based on the cache coherence protocols defined in Section II, there are only two types of packets running in the ring: probe packets which carry access requests to search their destinations and data packets which carry the data segments back to the source processors. In a steady state, each ring can be considered to have the same number of probe packets and data packets. Moreover, a COMA system may cause a physical address segment to be moved in different local caches at a different time. Therefore a physical address segment can be assumed to have the same probability to reside on every local cache at the same time under the condition that each processor requests a physical address segment with the same probability during a unit period of time. Based on this unique COMA data migration feature, we can assume that each local ring has the same contention pattern in a steady state which is independent of the contention differences among physical address segments. This is the major difference between the home addressed CC-NUMA (a data item can be fixed in a memory module, and access to it is conducted by remote-read and remote-write), and the changeably addressed COMA. However, the data migration feature of a COMA system makes the read/write miss process more complicated than that in a CC-NUMA system. In a COMA system, a read/write miss will dynamically chase the data because the request data does not have a home address and will be dynamically moved in a local cache by a write access. In the following, the latency of a read/write miss is derived mainly by modeling the dynamically chasing process of a read/write miss.

A. Modeling the Network Contention in COMA

For all local rings, each interface port from the global ring to a local ring contributes an equal amount of traffic to the local ring, which is not affected by the hot spot effects because of the data dynamic migration feature. So the fraction of accesses to the hot spot in each processor should be considered uniformly distributed among the memory modules in N local rings. Based on the above analysis, the packet arrival rate, denoted as λ_{lp}, in each interface port from the global ring to a local ring can be expressed as

$$\lambda_{lp} = \frac{2N(M-1)\lambda\lambda_h}{M} + 2N(1-\lambda_h)(1-\lambda_l)\lambda, \qquad (4.7)$$

where the hot request rate to a local ring is $\frac{N(M-1)\lambda\lambda_h}{M}$, the nonhot request rate to a local ring is $N(1 - \lambda_h)(1 - \lambda_l)\lambda$, the data packet rate to respond to the hot requests of a local ring is $\frac{N(M-1)\lambda\lambda_h}{M}$, and the data packet rate to respond to the nonhot requests of a local ring is $N(1 - \lambda_h)(1 - \lambda_l)\lambda$.

Then, using the same method described in Appendix A, we can obtain the following three important performance results:

1) U_{l_coma}, the utilization of a local ring in COMA is

$$U_{l\_coma} = \frac{N\lambda}{M}\left(M + 2(M-1)\lambda_h + 2M(1-\lambda_h)(1-\lambda_l)\right). \quad (4.8)$$

2) $\overline{q}_{wait\_coma}$, the waiting time for a message to find an empty slot in a local ring is

$$\overline{q}_{wait\_coma} = \sum_{i=0}^{\infty} iU_l^i(1-U_l)t_r = \frac{U_l t_r}{1-U_l}. \qquad (4.9)$$

3) $\overline{q}_{l\_coma}$, the queuing time in the interface port for a message to enter a local ring from the global ring is

$$\overline{q}_{l\_coma} = \frac{t_r}{1 - \frac{N\lambda t_r\left(M+3(M-1)\lambda_h+3M(1-\lambda_h)(1-\lambda_l)\right)}{M}}. \qquad (4.10)$$

For the global ring, each interface port from a local ring to the global ring can be modeled as a $M/G/1$ queue with packet arrival rate of $2N\lambda((M - 1)\lambda_h/M + (1 - \lambda_h)(1 - \lambda_l))$. Then, we can obtain the following two important performance results in the same way:

1) U_{g_coma}, the utilization of the global ring is

$$U_{g\_coma} = 2N\lambda((M-1)\lambda_h + M(1-\lambda_h)(1-\lambda_l)). \qquad (4.11)$$

2) $\overline{q}_{g\_coma}$, the queuing time in the interface port for a message to enter the global ring from a local ring is

$$\overline{q}_{g\_coma} = \frac{t_r}{1 - \frac{2Nt_r(M+1)\left((M-1)\lambda_h + M(1-\lambda_h)(1-\lambda_l)\right)}{M}}. \qquad (4.12)$$

B. Latency of Remote-Write to a Hot Memory

In a COMA system, the searching process of a write-miss can be expressed as the state transition graph shown in Fig. 3 based on the dynamic migration feature of data. State O represents the initial state of a write miss at its source processor. States LS and GS represent two possibilities for a write-miss to find the owner of its required address segment, where LS is the process of searching and getting the hot segment in the local ring with probability of $1/M$, and GS is the process of getting the hot segment in a remote memory with probability of $(M-1)/M$. States INV_1 and INV_2 represent the corresponding invalidation states of LS and GS respectively. We use t_{ls}, t_{inv_1}, t_{gs}, and t_{inv_2} to represent the time spent in each corresponding state. Based on Fig. 3, the latency of a remote-write to a hot spot, denoted as $\overline{T}_{w\_coma}$, can be expressed as

$$\overline{T}_{w\_coma} = \frac{t_{ls} + t_{inv\_1}}{M} + \frac{(M-1)(t_{gs} + t_{inv\_2})}{M} + \overline{q}_{wait\_coma}, \quad (4.13)$$

where the detailed derivation process of t_{ls}, t_{inv_1}, t_{gs}, and t_{inv_2} is listed in Appendix B.

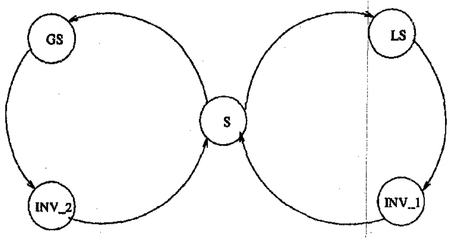


Fig. 3. Transition graph of a COMA write-miss.

C. Latency of Remote-Read to a Hot Memory

In a COMA system, the remote-read process can be described by the same state transition graph as shown in Fig. 2. The transition probability P also has the following expression:

$$P = \begin{cases} \frac{1}{M} & \lambda_w \geq \lambda_r, \\ 1 - \frac{\lambda_w (M-1)}{M\lambda_r} & \text{otherwise}. \end{cases} \quad (4.14)$$

The hot read missing latency is

$$\overline{T}_{r\_coma} = PT_l + (1-P)T_g, \quad (4.15)$$

where the computation of local search latency T_l and global search latency T_g is more complicated than that in a CC-NUMA system because a read miss in a COMA system involves in a process of dynamically chasing for data. In the following, we calculate T_l and T_g under two conditions:

1) $\lambda_w \geq \lambda_r$:
Each read miss to data must be preceded by a write miss to the data, which means that no copies of a data item exist when a read miss to the data occurs. In this situation, the data search procedure of a read miss is the same as that of a write miss except that a read miss does not involve an invalidation process. Hence, we have

$$T_l = t_{ls} + Nt_r/2,$$
$$T_g = t_{gs} + (M + N)t_r/2. \quad (4.16)$$

2) $\lambda_w < \lambda_r$:
Each write miss follows $\frac{\lambda_r}{\lambda_w}$ read misses. So the average number of copies of a data item in the system is $(1 - \lambda_r/\lambda_w)/2$ when a read miss to the data occurs, which reduces the global search latency T_g to

$$T_g = \max\left\{(2N+M)t_r + \overline{q}_{wait\_coma} + \overline{q}_{l\_coma} + \overline{q}_{g\_coma}, \frac{t_{gs} + (M+2N)t_r/2}{1 + (1 - \lambda_r/\lambda_w)/2}\right\}, \quad (4.17)$$

where $(2N+M)t_r + \overline{q}_{wait\_coma} + \overline{q}_{l\_coma} + \overline{q}_{g\_coma}$ is the least time of remotely getting data, and $\frac{t_{gs}+(M+2N)t_r/2}{1+(1-\lambda_r/\lambda_w)/2}$ is the reduced global search latency by $(1 - \lambda_r/\lambda_w)/2$ data copies.

Furthermore, the average number of copies of a data item in one of M local rings is $(1 - \lambda_r/\lambda_w)/(2M)$, which reduces the local search latency T_l to

$$T_l = \max\left\{Nt_r + \overline{q}_{wait\_coma}, \frac{t_{ls} + Nt_r/2}{(1 + (1 - \lambda_r/\lambda_w)/(2M))}\right\}, \quad (4.18)$$

where $Nt_r + \overline{q}_{wait\_coma}$ is the least time for traveling a local ring for a circle, and $\frac{t_{ls}+Nt_r/2}{(1+(1-\lambda_r/\lambda_w)/(2M))}$ is the reduced search time by the multiple copies.

V. Comparative Performance Evaluation between CC-NUMA and COMA Based on the Analytical Models

In this section, we provide analytical results dependent on various architecture effects such as access-miss latency and bandwidth. The analysis of the bandwidth is based on the analysis of the upper bound of the request rate per processor. The architectural factors to be considered are the size of a ring and the rotation speed of the ring. The system factors to be considered are the data locality, the hot spot effects and the ratio of between read-miss and write-miss.

A. The Analysis on Access-Miss Latency

A.1. The hot spot effects

Here we choose 32 as the number of slots in each local ring and the global ring. The rotation period is one unit time. The effects of the hot spot to miss latency under this condition in CC-NUMA and COMA systems are shown in Fig. 4. Fig. 4 indicates that the hot spot has little effect on remote-read latency in both systems and remote-write latency in CC-NUMA. But it affects remote-write latency in COMA to a certain degree because of more frequent data migration. The results show that the structure of the hierarchical ring network can well balance network traffic when a hot spot occurs.

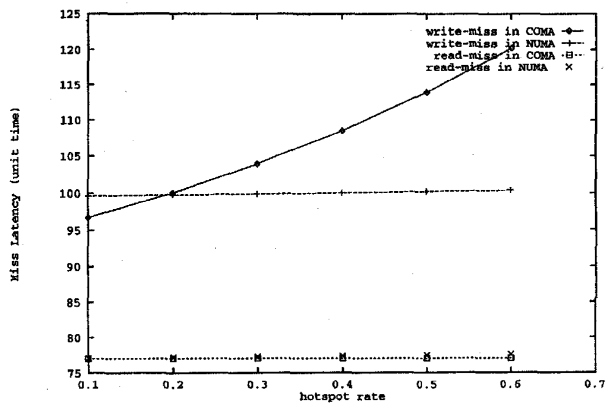


Fig. 4. Effects of changing the hot spot rate on access-miss latencies.

A.2. The locality effects

The locality is defined as the ratio between the number of accesses to a local ring and the total number of nonhot memory accesses in the system. The performance parameters are selected as in Section V.A.1 except that total miss rate is 0.0005. Fig. 5 presents the effects of the locality on the miss latencies in both memory systems. It shows that the increase of the locality significantly reduces the miss latencies in both systems. In particular, the read-miss latencies in both systems present almost the same curves. But the write-miss latency in a COMA is less than that in a CC-NUMA system.

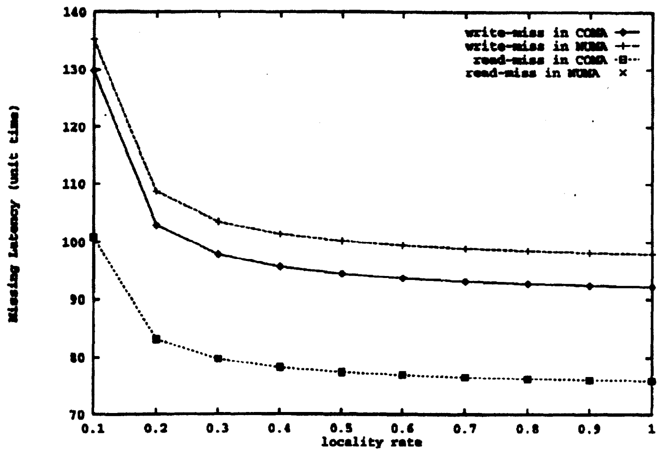


Fig. 5. Effects of changing the program locality on miss latencies.

A.3. The effects caused by different miss rates and request rates

The network contention is mainly determined by the miss rate in each processor. Assuming a uniform distribution of miss rates in each processor, the effects of the miss rate on access-miss latencies in both systems are shown in Fig. 6. The results show that the increase in miss rate causes higher read/write miss latency in both systems. But the write-miss latency in the COMA is slightly smaller than the write-miss latency in the CC-NUMA because of a more balanced load in the COMA.

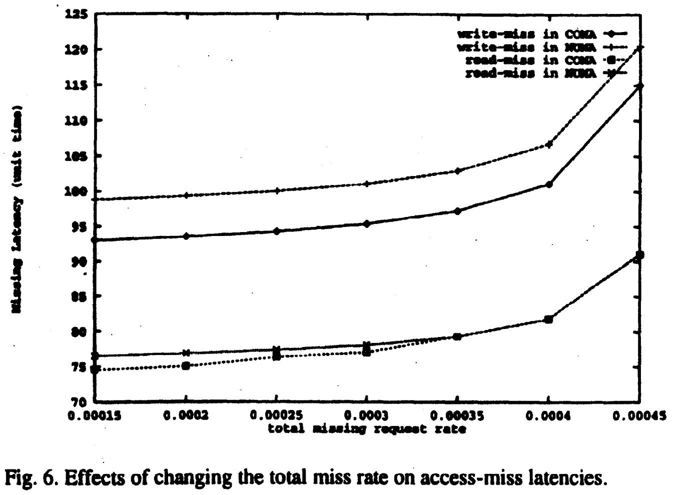


Fig. 6. Effects of changing the total miss rate on access-miss latencies.

A.4. The effects of read/write miss distributions

The effects of read/write miss distributions to the miss latency are measured by changing the ratio between read-misses and write-misses. Fig. 7 shows that, by increasing the read-miss rate, the read-miss latencies in both systems reduce significantly; but the write-miss latencies in both systems increase because there are more cache copies to be invalidated. On the other hand, the COMA handles read/write misses slightly more effectively than that in the CC-NUMA.

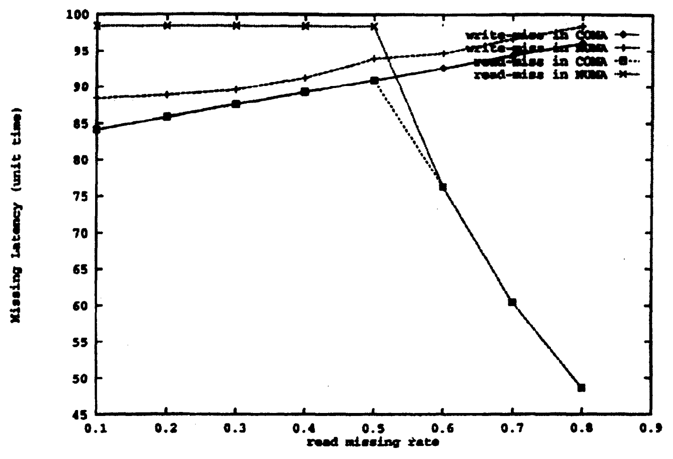


Fig. 7. Effects of changing read miss distribution on miss latencies.

A.5. System effects by changing the ring size

The size of a ring is an important architecture factor. Assuming the miss rate in each processor is uniformly distributed, and the rotation period of the ring is in a unit time, Fig. 8 shows that, by increasing the size of a ring, read-miss latencies in both systems have the same increasing curves, but the write-miss latency in COMA increases slightly slower than that in the CC-NUMA.

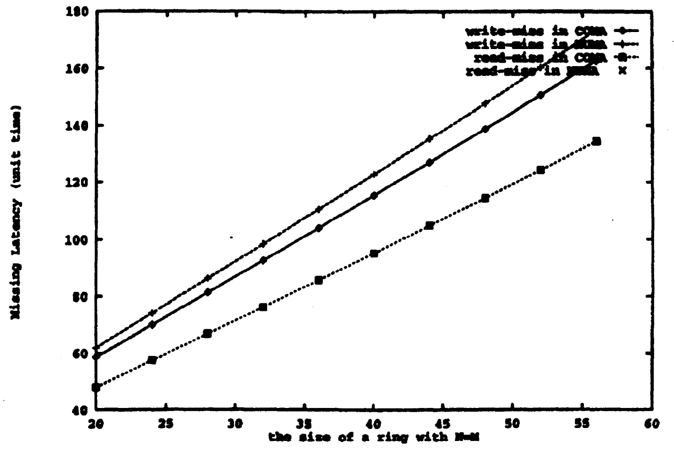


Fig. 8. Effects of changing the ring size to miss latencies.

A.6. System effects by changing the rotation period

The rotation period of the rings is another important architecture factor affecting performance. Fig. 9 plots the latencies of both systems by slowing down the rotation speed step by step. The results indicate that the COMA performs slightly better than the CC-NUMA in terms of changing the rotation period.

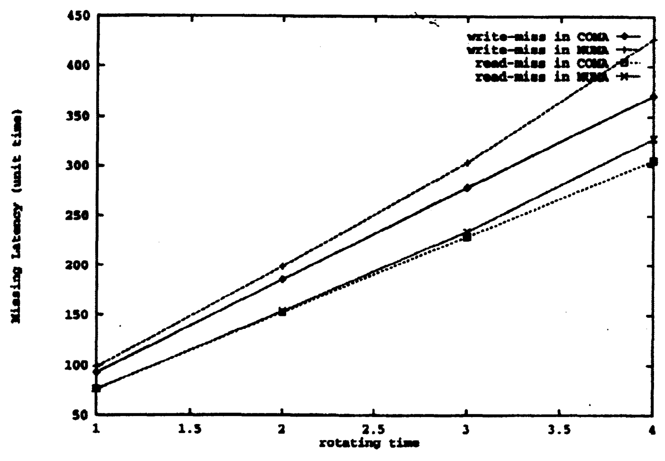


Fig. 9. Effects of changing the rotation period on missing latencies.

B. Bandwidth Analysis

In CC-NUMA and COMA systems, the access-miss request on each processor is bounded by the network contention. It is important to compare the different effects on the request bound by changing the performance parameters between the two memory systems. The detailed mathematical models for bandwidth analysis are given in Appendix C.

B.1. Effects of locality

Fig. 10 presents the effects of localities to the upper bound of the miss rate. It shows that the upper bound of the miss rate in the COMA increases slightly faster than that in the CC-NUMA. Fig. 11 shows that the effects of the hot spot on the upper bound of the miss rate in both CC-NUMA and COMA are almost identically significant.

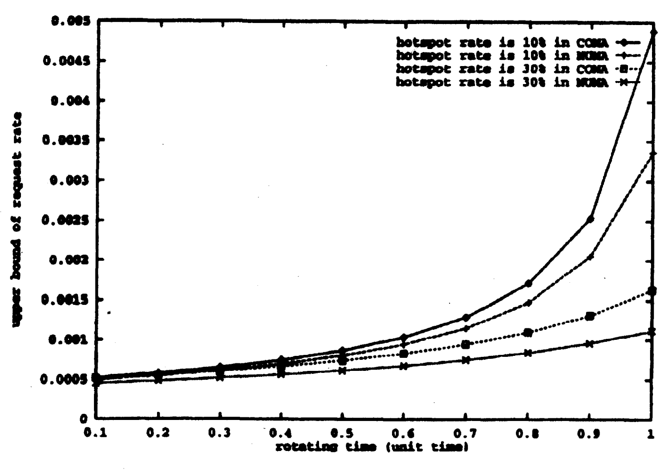


Fig. 10. Effects of changing program locality with a specific hot spot rate on the request bound.

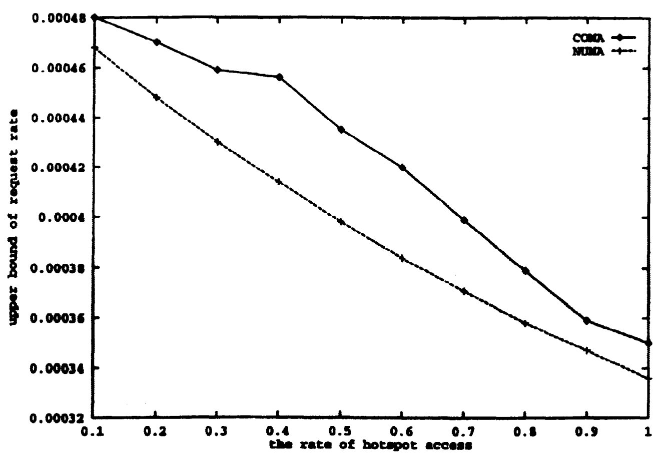


Fig. 11. Effects of changing the hot spot rate on the request bound.

B.2. Effects of architectural factors

Fig. 12 presents the effects by changing the size of the rings with a certain hot spot rate to the upper bound of the access-miss rate. The curves shows that both systems have nearly the same performance. Fig. 13 shows that by decreasing the rotation speed, the upper bound of the miss rates in both the CC-NUMA and COMA systems is almost identically affected.

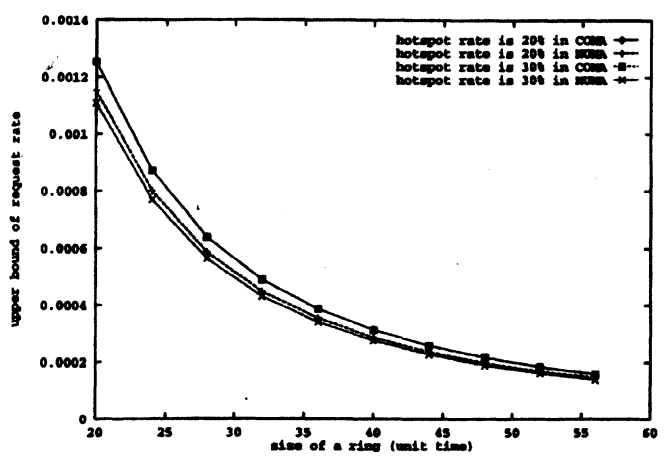


Fig. 12. Effects of changing the ring size with a specific hot spot rate to the request bound.

VI. Experiment-Based Validation on the KSR-1

A. An Overview of the Experiment-Based Validation

To validate analytical models, execution-driven simulation and real-machine-based experiments are two alternatives. Although the execution-driven simulation can measure a variety of architecture features by flexibly changing various architectural parameters, it may be difficult or even impossible to verify whether the simulator has correctly modeled a real multiprocessor system. As pointed out in [10], the validation to a simulation only shows whether it can produce results similar to another simulator. Therefore, we decided to conduct experiments on a real hierarchical ring system to validate our analytical results.

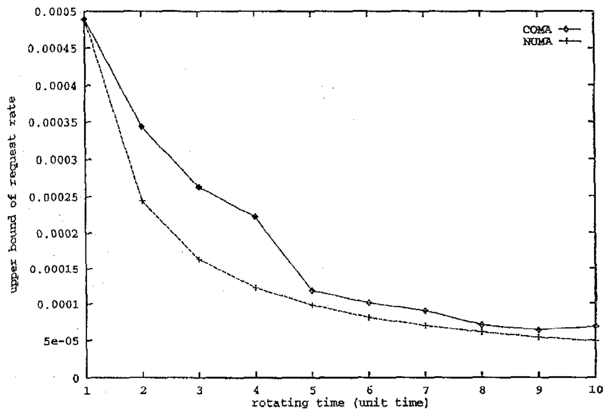


Fig. 13. Effects of changing the rotation period to the request bound.

KSR-1 [7], introduced by Kendall Square Research, is a hierarchical ring based COMA multiprocessor system which provides a direct testbed for validating the analytical results of the COMA system. To validate the analytical results on a hierarchical ring based CC-NUMA system, we simulate its memory operations on KSR-1. A key part of simulating a CC-NUMA system in a COMA system is to generate the CC-NUMA memory access patterns on the KSR-1. While in a CC-NUMA system data is home addressed, in a COMA system a data item is dynamically duplicated and moved upon read/write requests. In order to fix a data item, we use an array on the KSR-1 to simulate a home addressed variable in a CC-NUMA system. The array is called the extension vector of the variable. The memory access pattern of a CC-NUMA read/write operation sequence is simulated by directing a read/write operation to each independent element of the variable's extension vector based on the following rule:

Let s be a variable, $s[m]$ be the extension vector of s, where m is the length of the vector for multiple accesses to s. Let $a_1(s)$, $a_2(s)$, ..., $a_t(s)$ be a read/write sequence on s in a CC-NUMA system, where t is the length of the sequence. This sequence is simulated in a COMA system by sequence $a'_1, a'_2, \ldots, a'_t$ which is constructed as follows:

1) $a'_1 = a_1(s[1])$;
2) For any $i > 1$, if a'_{i-1} is a write operation on $s[j]$, then $a'_i = a_i(s[j+1])$; if a'_{i-1} is a read operation on $s[j]$, then $a'_i = a_i(s[j])$.

The above rule guarantees that a series of consecutive read operations access the same variable, and a write operation does not move the location of the data item. Hence, a CC-NUMA access pattern can be rigorously simulated with the support of the extension vector. The cache coherence protocol proposed for the CC-NUMA system in Section II indicates that a remote write always has the same executing trace which is independent of the number of data copies in the system. This is because the system uses multiple parallel invalidation packets to invalidate the copies in local rings. If two processors in a CC-NUMA system produce two similar operation sequences on a shared variable, the similar memory access pattern can be simulated by making each processor produce its own operation sequence on two different variables in the same memory module.

In the rest of this section, we report two sets of experiments on KSR-1 to validate our analytical results, and to give more comparative results between the CC-NUMA and the COMA which complement the results from analytical models. The first set of experiments designed for performance validation, called uniform experiments where the memory access patterns in the analytical models are simulated and the effects of cache miss rate, read/write rate, cache coherence protocol, hot spot, and locality are measured for comparisons with the analytical results. Based on the analytical models, this set of experiments were uniformly constructed such that all the 64 processors on two rings were employed. In execution, each processor generated read/write request misses to the memory modules on its local ring and on the remote ring respectively. The changes of memory access patterns were adjusted by the four parameters: request rate, read (or write) rate, hot spot rate, and locality rate. In the second set of experiments, additional memory access patterns were generated to measure the effects of hot spot and the two cache coherence protocols.

The objective of running these experiments is to validate analytical results presented in early sections. Since the analytical models and experiments were performed on two different bases, the absolute latency measures are different. However, we can still well present performance model validations and comparisons based on performance tendencies and implications from both analytical and experimental results.

B. Cache Coherence Effects

KSR-1 maintains consistency of the data in each cache using a write-invalidate cache coherence strategy. Whenever a data item is requested by a processor for an update, every other copy of that data item, in every other cache where that subpage is located, is marked "invalid" so that the proper value of the data item can be maintained. The distributed cache directories maintain a list of each subpage in the local cache, along with an indication of the "state" of that subpage. To validate the analytical results on cache coherence effects, we performed the following three experiments.

The first experiment was designed to validate the Assumption 5 in Section II.C. In our experiment, one processor writes an array of 500,000 elements into its local cache. All other processors then read that array into their local caches, leaving the exclusive ownership of this array with the processor which originally wrote it and leaving copies of the array located in every other cache. The original processor updates the array again, requiring that all other copies be invalidated. The tests were conducted two different ways:

1) the number of processors was scaled from 1 to 62 continuously, 0-31 on one ring, then moving to the second ring for 32-62; and
2) the number of processors was scaled in pairs, one processor on each of the two rings being a member of the pair.

With each increment of the number of processors, one processor was added to each ring. Consequently, the number of processors scaled by 2; e.g., 2, 4, 6, 8, ..., 62. The results in Fig. 14 reflect these two different ways of scaling the problem and conclusions will be drawn from these two different configurations. Fig. 14 shows that the maintenance cost of cache coherence bears no additional cost in KSR-1, over and above the latency of the ring rotation itself, which does not reflect an increase because of the number of processors, but an increase due to the distance the data invalidation must travel from one ring to another ring. This is consistent to the Assumption 5 in Section II.C where the ring rotation is clocked so that any and all actions that could possibly take place during a stop at each cell can be accomplished.

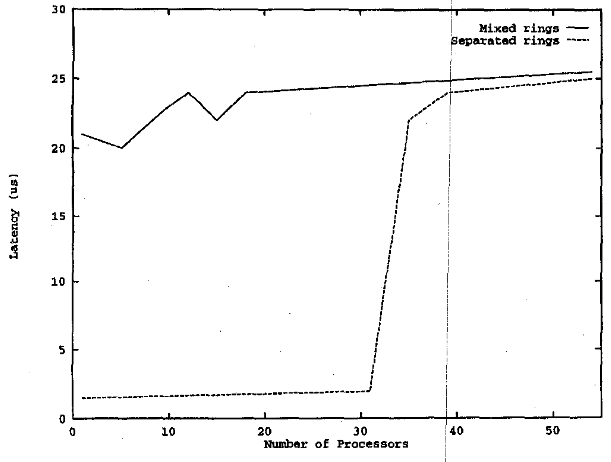


Fig. 14. Cache coherence timing changes as the number of processors is increased on the KSR-1.

In the second and the third experiments, the uniform experiments were conducted under different sets of parameters. To validate the effect of request miss rate on network contention, in the second experiment, we used the same set of performance parameters used in the analytical model where the hot spot rate was set to 0, the read rate was 0.7 and the locality rate was uniformly distributed. The write miss latencies and the read miss latencies were measured while the request rate was changed from 0.00015 to 0.00045 through a delay function. The measurements are given in Table I, which shows the similar varying tendencies of miss latency to the analytical results in Fig. 6.

TABLE I
MEASUREMENT RESULTS: EFFECTS OF CHANGING THE TOTAL MISS RATE ON ACCESS-MISS LATENCIES (IN μs)

request rate	0.00015	0.00025	0.00035	0.00045
w-miss in COMA	27.5	29.0	31.3	36.2
w-miss in NUMA	32.4	34.2	36.0	38.5
r-miss in COMA	22.1	23.1	25.0	26.2
r-miss in NUMA	24.4	25.6	27.1	26.9

To validate the effects of different read/write miss distributions on miss latency, in the third experiment, we set the following conditions: the request rate was fixed at 0.0005, the locality rate was uniformly distributed, the hot spot rate was set to 0 and the read rate was changed from 0.1 to 0.8. Table II lists the measurement results showing that the read-miss latency is very close to the write-miss latency when the read rate is less than 0.5, then decreases significantly with the increase of read rate beyond 0.5. This is due to the effect of multiple copies of data items in both COMA and CC-NUMA systems. These experimental results are identical to the analytical results given in Fig. 7 in terms of the system effects.

TABLE II
MEASUREMENT RESULTS: EFFECTS OF CHANGING READ/WRITE MISS DISTRIBUTION ON MISS LATENCIES (IN μs)

read rate	0.1	0.3	0.5	0.6	0.7	0.8
w-miss in COMA	29.2	32.2	35.1	35.7	37.0	37.2
w-miss in NUMA	32.1	33.5	36.4	37.2	38.0	39.2
r-miss in COMA	29.2	32.2	35.1	27.0	17.0	8.3
r-miss in NUMA	34.1	34.8	35.2	27.0	17.3	8.8

C. Locality Effects

The effect of locality rate was measured through the uniform experiments under the following condition: The request rate was fixed at 0.0005, the hot spot rate was set to 0, and the read rate was set to 0.7. The locality rate was changed from 0.1 (where 90% of requests sent by a processor directed to a remote processor in the remote ring) to 0.9 (where 90% of requests sent from a processor directed a remote processor on the local ring). The measurement results reported in Table III show that the decrease rate of miss latencies is similar to the analytical result given in Fig. 5. For example, both analytical and experimental results show that the miss latencies reduce about 25% when the locality rate increases from 0.1 to 0.5.

TABLE III
MEASUREMENT RESULTS: EFFECTS OF CHANGING LOCALITY RATE ON MISS LATENCIES (IN μs)

locality rate	0.1	0.3	0.5	0.7	0.9
w-miss in COMA	31.2	27.3	24.1	20.2	19.2
w-miss in NUMA	29.4	24.1	22.3	20.2	19.2
r-miss in COMA	23.0	20.6	16.5	11.8	10.2
r-miss in NUMA	24.3	21.2	16.5	11.9	11.0

D. Hot Spot Performance on the KSR-1

In practice, a hot spot may occur under different memory access patterns, resulting in different performance degradation. Hence our measurements were conducted not only by the uniform experiments for validating the analytical results presented in Fig. 4, but also by three additional experiments for studying the hot spot effects under practical memory access patterns.

D.1. Effects on memory access delay

The hot spot on the KSR-1 is allocated either in a fixed location, called *fixed hot spot* for CC-NUMA or in movable locations, called *movable hot spot* for COMA. The fixed hot spot remains physically on one processor as other processors try to read it with a single variable or a block of data. The movable hot spot will be migrated around the ring on demand of any processor which does a read with a single variable or a block of data. This data migration is a feature of the KSR-1 intended to enhance data locality.

To validate our analytical results, we first evaluated the hot spot effects through the uniform experiments under the following conditions: each processor in the two rings generated request misses at the fixed rate of 0.0005 where the locality rate was uniformly distributed, the read rate was fixed at 0.7, and the hot spot rate was changed from 0.1 to 0.6. The read/write miss latencies were averaged over 10000 test cases and are listed in Table IV. The measurement results show that a write-miss in the COMA system is more sensitive to a hot spot than in the CC-NUMA due to the overhead of more frequent data movement in the COMA system. This conclusion is consistent to the analytical results reported in Fig. 4 in terms of hot spot effects.

TABLE IV
MEASUREMENT RESULTS: EFFECTS OF CHANGING HOT SPOT RATE ON MISS LATENCIES (IN μs)

hot spot rate	0.1	0.2	0.3	0.4	0.5	0.6
w-miss in COMA	30.2	33.3	36.1	39.2	42.1	45.2
w-miss in NUMA	34.4	33.8	33.3	33.6	34.1	33.7
r-miss in COMA	25.0	25.1	25.2	25.0	25.1	25.2
r-miss in NUMA	25.3	25.2	25.5	25.1	25.2	25.5

In practice, a hot spot is usually generated only by part of processors in the system. Experiments reported in [16] simulate this type of memory access patterns, which used 57 out of 64 processors in a KSR-1 system to generate the hot spot on another remote cache module, remaining 6 remote cool cache modules. The miss latencies of remote reads and remote writes of one word, one block, two blocks, and three blocks were respectively measured under an environment without any hot spots, an environment with the hot spot generated by cache references in a word unit, and an environment with the hot spot generated by cache references in a block unit. In comparison between the fixed and movable hot spot experiments, the results in [16] indicate that a movable hot spot slightly increases the access delay to cool variables in the cool cache modules due to heavier traffic caused by more data movement.

Another experiment to verify our modeling work is to see if a hot spot can affect remote readings among processors that are not involved in the process of generating the hot spot. Again, two rings were used for the experiment. The difference between this experiment and the previous one is that all of the processors contributing towards generating the hot spot are on the same ring (the *hot ring*). While the other processors on the other ring (the *cool ring*) are involved in generating the hot spot. The hot spot is fixed to one processor within the hot ring. We varied the experiment by increasing the number of processors to be involved in the hot ring to generate the hot spot. We chose to use 50% and 97% of the available processors on the hot ring to generate the hot spot for these variations. Thus, for 50% usage of the processors on the hot ring there were also 16 processors that were not involved in generating the hot spot. At the same time, there were also 16 processors to be used on the cool ring. These two sets of 16 processors were used to do remote readings of their counter processors, respectively. These remote readings were also unidirectional among the two rings during any run of the experiment. Thus, the 16 processors on the cool ring read the 16 processors on the hot ring during the hot spot activity in one run of the experiment. Then we reversed the remote reading and had the 16 processors on the hot ring read remotely the 16 processors on the cool ring during the hot spot activity in another run. The hot spot was generated by either reading a single variable or by reading a block of data. Thus, there were four different timings from the hot spot activity that were compared to the remote readings when there was no hot spot. For 97% of processors on the hot ring in usage there was one available processor on the hot ring and 1 available processor on the cool ring. As shown on Tables V and VI for the different variations, the hot spot has very little effect on all the remote reads in this experiment. This additional experiment on the KSR-1 further strengthens our analysis that a hierarchical ring based architecture, such as the KSR machine handles the hot spot activity efficiently, as presented by the analytical models in Fig. 4.

TABLE V
READING MEASUREMENTS (IN μs) WHEN HOT SPOT IS GENERATED BY 50% OF PROCESSORS ON THE HOT RING ON THE KSR-1

	(1 var)	(1 blk)	(2 blks)	(3 blks)
from cool$_R$ to cool$_R$	31.48	36.47	66.71	97.57
from hot$_R$ to cool$_R$ hot spot (var)	32.19	37.77	66.99	100.33
from hot$_R$ to cool$_R$ hot spot (block)	32.74	37.68	67.59	99.60
from cool$_R$ to hot$_R$ hot spot (var)	31.55	38.67	66.55	98.94
from cool$_R$ to hot$_R$ hot spot (block)	32.17	37.50	68.08	99.13

TABLE VI
READING MEAUREMENTS (IN μs) WHEN HOT SPOT IS GENERATED BY 97% OF PROCESSORS ON THE HOT RING ON THE KSR-1

	(1 var)	(1 blk)	(2 blks)	(3 blks)
from cool$_R$ to cool$_R$	27.83	32.14	61.63	93.84
from hot$_R$ to cool$_R$ hot spot (var)	27.80	32.50	62.70	93.18
from hot$_R$ to cool$_R$ hot spot (block)	27.94	32.08	63.09	93.05
from cool$_R$ to hot$_R$ hot spot (var)	27.49	32.10	62.65	93.99
from cool$_R$ to hot$_R$ hot spot (block)	27.92	32.22	62.28	92.73

D.2. Effects on normal parallel computations in cool nodes

In this experiment we measured the effects that a hot spot may have on a matrix multiplication application. Again, there were two forms of hot spots, *fixed* for CC-NUMA and *moveable* for COMA.

We used 64 processors for our experiments. We increased the number of processors doing the matrix multiplication from 1, 2, 4, 8, and 16 processors. The matrices to be multiplied are $A \times B$ and the result is put in matrix C. The size of these matrices are 224×224. The computation resided on the same ring. The matrices A and C are distributed so that each processor has to access from its neighbor to do its share of the computation. Each of the contributing processors has a local copy of the matrix B. When there is a hot spot present, the number of processors that contribute towards generating the hot spot is 48 of the 64 processors (75%). The ring where the hot spot resides is the hot ring (this is true only for the fixed hot spot implementation). The other ring is the cool ring. As in our other experiments, the hot spot was generated by either reading a single variable or reading a block of data.

Table VII presents the timing results of fixed hot spot effects on the matrix multiplication. The first row gives timings of the matrix multiplication (MM) without a hot spot present. The second and third rows show the timings during the following setup. The processors doing the matrix multiplication all reside on the same ring (cool ring) while the hot spot resides on the hot ring. The fourth and fifth rows show the opposite set up of the previous two rows. That is, all processors involved in the matrix multiplication reside on the hot ring (where the hot spot is located). The sixth and seventh rows show the same setup as in the previous two rows with one difference. One of the processors that contribute towards the matrix multiplication will be the processor where the hot spot resides. This is shown in the table with a one in parenthesis (1) to signify this. As we predicted, there was virtually no difference in the timings during the presence of a fixed hot spot with any of the implementations.

TABLE VII
READING MEASUREMENTS (IN SECS) OF MATRIX MULTIPLICATION OF SIZE 224 × 224 DURING THE PRESENCE OF A *FIXED* HOT SPOT

Processors	1	2	4	8	16
MM (no hot spot)	56.75	28.95	14.93	7.77	4.75
MM at cool$_R$ hot spot (var)	56.57	29.25	15.06	7.80	4.79
MM at cool$_R$ hot spot (block)	55.76	28.42	14.66	7.65	4.70
MM at hot$_R$ hot spot (var)	56.78	29.17	14.92	8.01	4.56
MM at hot$_R$ hot spot (block)	56.75	28.85	15.01	8.01	4.58
MM at hot$_R$ (1) hot spot (var)	56.70	28.60	15.16	8.05	4.58
MM at hot$_R$ (1) hot spot (block)	56.82	28.58	15.38	8.04	4.56

TABLE VIII
READING MEASUREMENTS (IN SECS) OF MATRIX MULTIPLICATION OF SIZE 224 × 224 DURING THE PRESENCE OF A *MOVEABLE* HOT SPOT

Processors	1	2	4	8	16
MM (no hot spot)	56.68	29.39	15.03	7.85	4.77
MM hot spot (var)	56.70	29.35	15.08	7.87	4.80
MM hot spot (block)	56.71	29.06	14.89	7.81	4.76

In Table VIII, we show the timings during the presence of a moveable hot spot. Row one shows the timings without the presence of the hot spot while rows two and three shows the timings with the moveable hot spot. Again, there was virtually no difference in the timings in the presence of the hot spot. This group of experiments further support our analytical performance evaluation in Section V.A.1.

VII. CONCLUSION

In this paper, our analytical models provide performance differences between the CC-NUMA and the COMA on a hierarchical ring architecture. The model considers the interconnection network and the memory systems, which are two important factors affecting the shared-memory performance. We also conducted experiments on the KSR-1, a hierarchical ring COMA system to verify some of the analytical results. We summarize performance evaluation results as follows:

1) In a hierarchical ring based architecture, a slotted ring orders and delays remote data access requests. This structure naturally reduces network contention for programs with hot spots. Analysis indicates that in the presense of hot spots overall ring traffic will be moderately increased but it will be distributed evenly in the ring network. Analytical results have been verified by the experiments on the KSR-1. When the hot spot memory access rate is increased, the write-miss latency in a COMA system will become slightly bigger than that in the CC-NUMA.
2) In the presence of a hot spot, COMA generates higher write-miss latency due to more frequent data migrations and a larger number of invalidations.
3) Our analysis indicates that COMA would have slightly lower write-access latency by changing the degree of localities of programs, but it would have the same read-miss latency in most cases as that in CC-NUMA.
4) Our analyses and experiments show that for applications with dominant read-misses at either high or low rates, COMA and CC-NUMA have nearly identical performance. In contrast, the simulation results in [12] indicate the two systems have the nearly identical performance only for the applications with low miss rates. A main reason for the different performance results is related to the different evaluation testbeds. The constant latency assumption on the flat network architecture simulator causes longer delay for COMA, and is likely to make the network contention independent of memory access patterns of applications. In addition, a flat network architecture is more hot spot sensitive than a hierarchical network. In comparison, the KSR ring architecture allows the system to exploit hierarchical locality of reference by moving referenced data to a local cache and satisfying data references from nearby copies of a data item whenever possible.
5) We show that CC-NUMA handles coherence misses only slightly more efficiently than COMA in the ringe architecture, while the simulation results in [12] indicate the difference is significant. Again, this is related to the different network architectures used for the evaluations.

We conclude that both CC-NUMA and COMA systems behave similarly on a hierarchical ring architecture. Two main reasons for this are that overhead of data migration in COMA matches the saving from improving locality in CC-NUMA; and that a slotted ring architecture balances the network contention. Our study indicates that a hierarchical ring network is a reasonable candidate for both COMA and CC-NUMA systems. We believe our performance results could be further generalized to the two memory systems on a hierarchical network architecture. Although a CC-NUMA system may not automatically balance the load at the system level, it provides an option for a user to explicitly handle data locality for a possible performance improvement. Therefore, the decision of using or making CC-NUMA or COMA systems on a hierarchical ring should be determined by the programming and manufacturing cost of the systems. Finally, based on our study, we believe a fair and valuable comparative performance evaluation of COMA and CC-NUMA systems should be conducted on each particular network architecture.

Appendix A Modeling Network Contention Using the M/G/1 Queue Theory

There are N stations and one interface port in a local ring which are assumed to be independent of each other. Each independent station contributes an equal amount of traffic to the ring at the same miss request rate of λ which follows a Poisson process. The interface port also inputs traffic to the local ring at the rate of λ_{cool_lport}, which is expressed as the sum of the probe packet arrival rate and the data packet arrival rate:

$$\lambda_{cool\_lport} = N\lambda\lambda_h + 2N(1-\lambda_h)(1-\lambda_l)\lambda. \quad (7.19)$$

A general network utilization is defined by:

$$U = \lim_{t\to\infty}\frac{C}{t}, \quad (7.20)$$

where C is the total number of bytes transmitted to the network from all the connected stations and the interface port during a period of time t. Based on (7.19) and (7.20), the utilization of a nonhot local ring is formulated as

$$U_l = t_r(\lambda_{cool\_lport} + N\lambda) = N\lambda t_r(1 + \lambda_h + 2(1-\lambda_h)(1-\lambda_l)). \quad (7.21)$$

Because the successive slots have been assumed to behave independently, the probability of a slot on a cool local ring to be full is U_l. The time needed for a packet in each station to find an empty slot can be approximated to have a geometric distribution. The analyses in [3] and [9] show that the error caused by the independence assumption is trivial. Hence, the average time, termed as $\overline{d}_n$, to find an empty slot on a cool local ring is:

$$\begin{aligned}\overline{d}_n &= \sum_{i=0}^{\infty} it_r(1-U_l)U_l^i = \frac{t_rU_l}{1-U_l} \\ &= \frac{N\lambda t_r^2(1+\lambda_h+2(1-\lambda_h)(1-\lambda_l)}{1-N\lambda t_r(1+\lambda_h+2(1-\lambda_h)(1-\lambda_l))}.\end{aligned} \quad (7.22)$$

The interface port from the global ring to a cool local ring uses a queue to buffer the message packets. In order to calculate the average waiting time for a request in the queue, we can model the port as a M/G/1 queue with packet arrive rate of λ_{cool_lport}. The average queue length in the buffer may be calculated by Little's law,

$$Q = \lambda_{cool\_lport}\overline{q}_{cool\_lport}, \quad (7.23)$$

Where $\overline{q}_{cool\_lport}$, an average waiting time in the queue may be calculated by

$$\overline{q}_{cool\_lport} = \left(\overline{d}_n + t_r\right) + Q\left(\overline{d}_n + t_r\right). \quad (7.24)$$

Combining (7.23) and (7.24), $\overline{w}$ becomes

$$\overline{q}_{cool\_lport} = \frac{\overline{d}_n + t_r}{1-\lambda_{cool\_lport}\left(\overline{d}_n + t_r\right)}. \quad (7.25)$$

In the hot local ring, the interface port from the global ring to the hot local ring contributes traffic to the hot local ring at the rate of λ_{hot_lport} which is calculated as:

$$\lambda_{hot\_lport} = (M-1)N\lambda\lambda_h + 2N(1-\lambda_h)(1-\lambda_l)\lambda. \quad (7.26)$$

By (7.20), (7.22), (7.23), (7.24), and (7.26), $\overline{d}_h$, the mean time to find an empty slot on the hot local ring, and $\overline{q}_{hotlport}$, the mean queueing time in the interface port between the global ring and the hot local ring before a message enters the hot ring, are derived as:

$$\overline{d}_h = \frac{N\lambda t_r^2\left(1+(M-1)\lambda_h+2(1-\lambda_h)(1-\lambda_l)\right)}{1-N\lambda t_r\left(1+(M-1)\lambda_h+2(1-\lambda_h)(1-\lambda_l)\right)}, \quad (7.27)$$

and

$$\overline{q}_{hot\_lport} = \frac{t_r}{1-N\lambda t_r\left(1+2(M-1)\lambda_h+4(1-\lambda_h)(1-\lambda_l)\right)}. \quad (7.28)$$

In the global ring, there are $N-1$ cool interface ports connected to $N-1$ cool local rings, and one hot interface port connected to the hot local ring. Each cool interface port can be modeled as an M/G/1 queue with packet arrival rate of λ_{cool_gport}:

$$\lambda_{cool\_gport} = N\lambda\lambda_h + 2N(1-\lambda_h)(1-\lambda_l)\lambda. \quad (7.29)$$

The hot interface port can be modeled as an M/G/1 queue with packet arrival rate of λ_{hot_gport}:

$$\lambda_{hot\_gport} = (M-1)N\lambda\lambda_h + 2N(1-\lambda_h)(1-\lambda_l)\lambda. \quad (7.30)$$

Then, using the same method as the above, we can get the mean queuing times $\overline{q}_{cool\_gport}$ in the cool ports and $\overline{q}_{hot\_gport}$ in the hot port, respectively, as follows:

$$\overline{q}_{hot\_gport} = \frac{t_r}{1-N\lambda_r\left(3(M-1)\lambda_h+2(M+1)(1-\lambda_h)(1-\lambda_l)\right)}. \quad (7.31)$$

$$\overline{q}_{cool\_gport} = \frac{t_r}{1-N\lambda t_r\left((2M-1)\lambda_h+2(M+1)(1-\lambda_h)(1-\lambda_l)\right)}. \quad (7.32)$$

APPENDIX B
MODELING READ/WRITE MISS LATENCIES IN COMA SYSTEM

A. Modeling the Search Time t_{ls} in *LS* State in Fig. 3

Using a local ring, we know that the probability for a slot to be non-empty in a local ring is U_{l_coma}. Since there are only two types of packets running on the rings, the probe packets and the data packets, the probability for a slot to have a probe packet is $U_l/2$. In addition, the probability for a probe packet to be a hot write miss is $(\lambda_h + \frac{(1-\lambda_h)}{NMN_c})\lambda_w$. Therefore, the probability for a slot to have a hot write probe packet is

$$P_{whl} = \frac{U_{l\_coma}}{2}\left(\lambda_h + \frac{(1-\lambda_h)}{NMN_c}\right)\lambda_w. \tag{7.33}$$

In state *LS*, let i_1 be the initial distance (number of slots) from the hot segment to the write miss request, which is a random value in [0, *N*]. We assume that the probability for the hot segment to be located in local cache *j* (*j* = 1, 2, ⋯, *N*) is 1/*N*. Then the average of i_1 is *N*/2. The probability for the hot write request to catch up with the hot segment at distance i_1 is $(1-P_{whl})^{i_1}$, which is the probability for none of the other write requests to write the hot segment in advance. If the write-miss has not found the hot segment at distance i_1, the hot segment must have been carried onto another local cache which has distance i_2 from the hot write probe block. Variable i_2 is a random value in [0, *N*/2] with the average of *N*/4. The probability for the write request to catch up with the hot segment at the second local cache is $(1-(1-P_{whl})^{i_1})(1-P_{whl})^{i_2}$. The write request will repeat this process until it gets the hot-spot segment. Thus, the search process of a write request can be expressed as the state transition graph shown in Fig. 15.

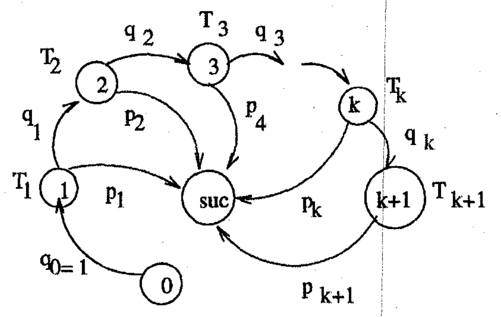


Fig. 15. Searching process of a hot write on a local ring.

In Fig. 15, *k* = log(*N*), $p_i = (1-P_{whl})^{N/2^i}$, $q_i = 1 - p_i$ and the time T_i can be approximately evaluated as $Nt_r/2^i$ because the new owner of the hot spot can be any local cache except the old ones and the source node at each state *i* (*i* = 1, 2, ⋯, *k* + 1). So the search time t_{ls} is

$$t_{ls} = \sum_{i=1}^{k+1} t_i * p_i * (q_1 * \cdots * q_{i-1}). \tag{7.34}$$

B. Modeling the Invalidation Time t_{inv_1} in State INV_1 in Fig. 3

When a hot write request changes from state *LS* to state *INV*_1, the invalidation process in *INV*_1 is determined by the ownership changes of the owner of the hot data:

CASE 1. The owner's status of the data copy is changed from "Exclusive" to "Invalidation" with probability λ_w: The invalidation block carries the hot data directly to the source processor. Hence the invalidation time t'_{inv_1} is $Nt_r/2$.

CASE 2. The owner's status of the data copy is changed from "Nonexclusive" to "Invalidation" with probability λ_r: the invalidation packet must travel the global ring for one circle to invalidate the other copies and then goes back to the source processor. The invalidation time t''_{inv_1} consists of the following components:

1) the time traveling from a local ring to the global ring.
2) the time traveling one circle of the global ring.
3) the time traveling from the global ring to the local ring.
4) the time traveling to the source processor in the local ring.

So t''_{inv_1} can be expressed as

$$t''_{inv\_1} = (2N+M)t_r + \overline{q}_{wait\_coma} + 2\overline{q}_{l\_coma} + \overline{q}_{g\_coma}.$$

Combining the above two cases, the invalidation time in state *INV*_1 is:

$$t_{inv\_1} = \lambda_w t'_{inv\_1} + \lambda_r t''_{inv\_1}.$$

C. Modeling the global search time t_{gs} in state GS in Fig. 3

Similar to (7.33), the probability for a slot on the global ring to have a hot write probe block is

$$P_{whg} = U_{g\_coma}\left(\lambda_h + \frac{(1-\lambda_h)}{NMN_c}\right)\lambda_w/2. \tag{7.35}$$

The time t_{gs} consists of the following three components:

1) t_{tra}, the time traveling from a local ring to the global ring, which is

$$t_{tra} = Nt_r/2 + \overline{q}_{l\_coma},$$

2) t_{sea}, the time searching the directory along the global ring. Initially we can consider *M*/2 (the number of the lots) to be the average distance from the source processor to the hot global directory which connects to the hot local ring. When the request, denoted by r_1, reaches the directory, the directory may have become cool, which means that there was another write request entering the hot ring before this one. The request r_1 must continue searching along the global ring, to wait for another global directory to become hot, and then to repeat the above procedure until r_1 catches up with the hot global directory. Each time a write request runs into the hot global directory, the hot directory becomes cool. There will be no hot directory until the write request carries the hot segment into the source local ring, which makes the global directory on the source local ring hot. The delay, denoted by *d*, in

the system between generating two hot directories is the sum of the time for the write request to enter the hot local ring, the time to search the hot segment, the time to carry the hot segment to the global ring and the time to travel to the interface port connected to the source local ring. Hence, this delay can be expressed as

$$d = \bar{q}_{g\_coma} + \bar{q}_{l\_coma} + Mt_r/2 + t_{ls}. \qquad (7.36)$$

The probability for the write request, r_1, to successfully catch up with the hot global directory each time is $(1 - P_{whg})^{M/2}$. So the average searching time t_{sea} is

$$\begin{aligned} t_{sea} &= \sum_{i=0}^{\infty} id\left(1-\left(1-P_{whg}\right)^{M/2}\right)^i \left(1-P_{whg}\right)^{M/2} + \frac{Mt_r}{2} \\ &= \frac{d\left(1-\left(1-P_{whg}\right)^{M/2}\right)}{\left(1-P_{whg}\right)^{M/2}} + \frac{Mt_r}{2}, \end{aligned} \qquad (7.37)$$

3) t_{ts}, the time of the request's entering the hot local ring and searching the hot segment is

$$t_{ts} = \bar{q}_{g\_coma} + t_{ls}.$$

Combining the above three components, we have

$$t_{gs} = \bar{q}_{g\_coma} + \bar{q}_{l\_coma} + \frac{(M+N)t_r}{2} + t_{ls} + \frac{d\left(1-\left(1-P_{whg}\right)^{M/2}\right)}{\left(1-P_{whg}\right)^{M/2}}. \qquad (7.38)$$

D. Modeling the invalidation time t_{inv_2} in state *INV_2* in Fig. 3

In the *INV_2*, the invalidation process has the same two alternatives as those in *INV_1*. Using similar analysis techniques, the invalidation time in *INV_2* can be expressed as

$$t_{inv\_2} = Nt_r + \bar{q}_{l\_coma} + \bar{q}_{g\_coma} + Mt_r(\lambda_r + 1)/2. \qquad (7.39)$$

APPENDIX C
MATHEMATICAL MODELS FOR BANDWIDTH ANALYSIS

Because the queuing time is larger than zero, by (7.25), (7.28), (7.31), and (7.32), we can derive the upper bound of the request rate in a CC-NUMA system, denoted as λ_{numa}, as follows:

$$\lambda_{numa} = min\left\{\frac{1}{Nt_r\left(1+2(M-1)\lambda_h+4(1-\lambda_h)(1-\lambda_l)\right)}, \frac{1}{Nt_r\left(3(M-1)\lambda_h+2(M+1)(1-\lambda_h)(1-\lambda_l)\right)}\right\}. \qquad (7.40)$$

By (4.10) and (4.12), the upper bound of the request rate in a COMA system, denoted as λ_{coma}, is given as follows:

$$\lambda_{coma} = min\left\{\frac{M}{Nt_r\left(M+3(M-1)\lambda_h+3M(1-\lambda_h)(1-\lambda_l)\right)}, \frac{M}{2Nt_r(M+1)\left((M-1)\lambda_h+M(1-\lambda_h)(1-\lambda_l)\right)}\right\}. \qquad (7.41)$$

Formulas (7.40) and (7.41) are the basic models for bandwidth analysis presented in Section V.B.

ACKNOWLEDGMENTS

We are grateful to the anonymous referees for providing important suggestions and comments to improve the technical quality, clarity and readability of the paper. We appreciate our colleague Neal Wagner for carefully reading the paper and contributing many helpful comments and Robert Castañeda for contributing some hot spot experiments on the KSR-1. Finally, many thanks go to Fredrik Dahlgren at the Lund University, Sweden, for the discussions and his many useful technical comments on this work.

This work is supported in part by the National Science Foundation under research grants CCR-9102854 and CCR-9400719, by the U.S. Air Force Office of Scientific Research under grant AFOSR-95-1-0215, by a grant from Cray Research, and by a fellowship from the Southwestern Bell Foundation. Part of the experiments were conducted on the KSR-1 machines at Cornell University and at the University of Washington.

REFERENCES

[1] BBN Advanced Computer Inc., *Inside the GP1000 and the TC2000*, 1989.

[2] L.A. Barroso and M. Dubois, "The performance of cache-coherent ring-based multiprocessors," *Proc. 20th Int'l Symp. Computer Architectures*, pp. 268-277, May 1993.

[3] L.N. Bhuyan, D. Ghosal, and Q. Yang, "Approximate analysis of single and multiple ring networks," *IEEE Trans. Computers*, vol. 38, no. 7, pp. 1,027-1,040, 1989.

[4] K. Farkas, Z. Vranesic, and M. Stumm, "Cache consistency in hierarchical ring-based multiprocessors," *Proc. Supercomputing 92*, pp. 348-357, Nov. 1992.

[5] E. Hagersten, A. Landin, and S. Haridi, "DDM—A cache-only memory architecture," *Computer*, vol. 25, no. 9, pp. 44-54, Sept.1992.

[6] Kendall Square Research, *KSR1 Technology Background*, 1992.

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

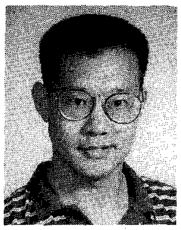
[8] W.M. Loucks, V.C. Hamacher, B. Preiss, and L. Wang, "Short-packet transfer performance in local area ring networks," *IEEE Trans. Computers*, vol. 34, no. 11, pp. 1,004-1,014, 1985.

[9] S.K. Reinhardt, M.D. Hill, and J.R. Larus, "The Wisconsin wind tunnel: virtual prototyping of parallel computers," *Proc. 1993 ACM SIGMETRICS Conf.*, pp. 48-60, May 1993.

[10] J. P. Singh, T. Joe, A. Gupta, and J. Hennessy, "An empirical comparison of the KSR and DASH multiprocessors," *Proc. Supercomputing 93*, pp. 214-225, Nov. 1993.

[11] P. Stenstrom, T. Joe, and A. Gupta, "Comparative performance of cache-coherent NUMA and COMA architectures," *Proc. 19th Int'l Symp. Computer Architectures*, pp. 80-91, 1992.

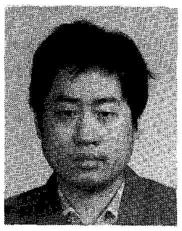
[12] Z.G. Vranesic, M. Stumm, D.M. Lewis, and R. White, "Hector: A hierarchically structured shared-memory multiprocessor," *Computer*, vol. 24, no. 1, pp. 72-79, 1991.
[13] X. Zhang, R. Castañeda, and W.E. Chan, "Spin-lock synchronization on the Butterfly and KSR-1," *IEEE Parallel and Distributed Technology*, vol. 2, no. 1, pp. 51-63, spring 1994.
[14] X. Zhang, K. He, and G. Butchee, "Execution behavior analysis and performance improvement in shared-memory architectures," *Proc. Fifth IEEE Symp. Parallel and Distributed Processing*, pp. 23-26, Dec. 1993.
[15] X. Zhang, Y. Yan, and R. Castañeda, "Comparative performance evaluation of hot spot contention between MIN-based and ring-based shared-memory architectures," *IEEE Trans. Parallel and Distributed Systems*, vol. 6, no. 8, pp. 872-886, Aug. 1995.
[16] X. Zhang, Y. Yan, and K. He, "Latency metric: An experimental method for measuring and evaluating program and architecture scalability," *J. Parallel and Distributed Computing*, vol. 22, no. 3, pp. 392-410, 1994.



Xiaodong Zhang received the BS degree in electrical engineering from Beijing Polytechnic University, China, in 1982, and the MS and Ph.D. degrees in computer science from the University of Colorado at Boulder in 1985 and 1989, respectively.

He is an associate professor of computer science and director of the High Performance and Computing and Software Laboratory at the University of Texas at San Antonio. He has held research and visiting faculty positions at Rice University and Texas A&M University. His research interests are parallel and distributed computing, parallel architecture and system performance evaluation, and scientific computing.

Dr. Zhang has served on the program committees of several conferences and is the program chair of the Fourth International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'96). He currently serves on the editorial board of *Parallel Computing* and is an ACM National Lecturer.



Yong Yan received the BS and MS degrees in computer science from Huazhong University of Science and Technology, Wuhan, China, in 1984 and 1987, respectively. He is currently a PhD student of computer science at the University of Texas at San Antonio. He has been a faculty member at Huazhong University of Science and Technology since 1987. He was a visiting scholar at the High Performance Computing and Software Laboatory at the University of Texas at San Antonio from 1993-1995. Since 1987, he has published extensively in the areas of parallel and distributed computing, performance evaluation, operating systems, and algorithm analysis.