Workshop on Computer Systems

A Novel Non-Hierarchical Cache Only Memory Architecture

Yeim-kuan Chang
Department of Information Management
Chung Hua University, Taiwan, R.O.C.
Email: ykchang@mi.chu.edu.tw

Abstract

Distributed shared memory multiprocessors with cache coherent non-uniform memory architectures (CC-NUMA) have become popular in the memory design of multiprocessors in recent years. The shared data can only be duplicated in caches for concurrent reads. The concept of fixed home node in CC-NUMA sometimes limits its performance. Capacity cache misses result from insufficient cache space for holding shared data. Capacity cache misses are forwarded to the home node to obtain the requested data with a longer network delay. Cache-Only Memory Architecture (COMA) treats the main memory as another level of caches. If the shared data items are too many to be hold in caches, the main memory can be a backup store for them. Thus, read misses in caches can be satisfied by local memory without a network delay. There is no notion of fixed home in COMA. Data items in the system can adapt to memory access behavior of programs with automatic data migration and greater data replication.

COMA performs better than CC-NUMA when the memory pressure is low, i.e., more attraction memory can be utilized to hold shared data. However, the protocol transaction overhead (especially replacement) increases when the memory pressure is high. In other words, CC-NUMA performs better than COMA in high memory pressure. In this paper, we propose a hybrid coherence protocol for COMA machines with a point-to-point network in mind. As in COMA-F protocol, the proposed protocol integrates the concept of fixed home node of NUMA into COMA system. It does not use a broadcast scheme to locate a data item. Thus it reduces the usage of network bandwidth for replacement messages. The proposed protocol is similar to the existing COMA-F protocol. But the difference between the proposed protocol and COMA-F is that the final copy of a shared data item will fall back to its home node when memory pressure is high. The proposed protocol performs similar to the existing COMA-F protocols in low memory pressure and similar to CC-NUMA in high memory pressure.

Keywords: Cache-only memory architecture (COMA), Cache-coherence non-uniform-memory architecture (CC-NUMA), shared memory, execution driven simulator.

Contact person: Yeim-kuan Chang

1 Introduction -NUMA and COMA-F Architectures

Distributed shared memory multiprocessors with non-uniform memory architecture (NUMA) have gained much attention in recent years in the area of parallel processing because they provide a simple programming model with a single address. Adding caches to NUMA machines (CC-NUMA) can improve the overall performance of the system. Many extra operations are needed to make the data in the caches consistent. Many existing CC-NUMA protocols such as the bit-mapped [1,5] and linked list protocols [2,3,4,6] intend to reduce the coherence overheads. The concept of the fixed home location also limits the performance improvement because all the read and write operations must go through these fixed home locations. Let's look at the following scenario: only one processor repeats a write after a read on the shared variables. All read and write requests must go through its home node. If the home node is remote, a lot of network messages will be generated and consequently waste of network bandwidth. Many different placement, replication, and migration strategies [8,13,17] were proposed to make most of read and write operations to be served locally.

One approach to solve this problem is to use Cache-Only memory architecture (COMA) [7]. Data blocks have no physical location. In other words, there is no concept of the home node. The data blocks can migrate and replicate dynamically s the code is in progress without special control scheme. Cache-only memory architecture multiprocessors organize the memory as another level of caches, called *attraction memory*. COMA allows a dynamic adaptation to the data reference pattern of applications. A COMA machine provides a programming model identical to that of shared memory architectures, but it does not require static distribution of execution and memory usage to run efficiently. Running an optimized NUMA program on a COMA machine results in a NUMA-like behavior, since the work spaces of the different processors migrate to their attraction memory. A COMA also adapts to and performs well for programs with dynamic or semi-dynamic scheduling. The working set migrates accordingly to the using processors regardless of the addresses. The COMA architectures also provide a way to increase the cache capacity that may improve the performance for most of the applications.

Three COMA machines have been proposed or built. KSR-1/2 [10], Data Diffusion Machine (DDM) [11], and simple COMA (S-COMA) [12]. KSR-1/2 machines are commercialized multiprocessors based on hierarchical ring. DDM relies on a hierarchical bus-based network structure. The higher level directory contains all the information of its lower level directories under it. The simple COMA design uses the processor's own memory management unit to implement the COMA caches. The memory management unit performs the function of locating data in the local memory of a processing node. S-COMA move part of the hardware for AM block replacement and relocation to software. Thus, the complex problem of relocation and replacement of data in the attraction memory become the responsibility of software.

One problem with S-COMA is the memory fragmentation causing low utilization of the memory space. Multiplexed Simple COMA (MS-COMA) [18] is proposed to eliminate this problem. In MS-COMA, multiple virtual pages are allowed to be mapped to the same physical page at the same time. This way increases the utilization of the memory space.

Other optimized NUMAs are NUMA with remote cache (NUMA-RC) [19] and Reactive NUMA [20]. NUMA-RC uses an additional DRAM space in CC-NUMA to act as level three cache in each node. The performance of NUMA-RC machines is comparable to COMA-F. COMA performs better than NUMA-RC in some situations, but worse in others. Thus, Reactive NUMA employs a mode switching method between NUMA-RC and S-COMA on a per-page basis. Reactive NUMA detects which of the two schemes is best for each page or block so that the best protocol can be used. Excel-NUMA [21] exploits the fact that, after a memory block is written and cached, the storage that kept the block in memory is unutilized. This storage is used to temporarily store remote copy displaced from the local caches.

This paper is organized as follows: Section 2 describes CC-NUMA and COMA-F architectures in detail and their advantages and disadvantages are discussed. The proposed protocol is detailed in section 3. Section 4 gives a simple analysis to compare the execution time of the proposed protocol with CC-NUMA and COMA-F. Finally, a concluding remark is given in section 5.

2 CC-NUMA and COMA-F Architectures

In this section we briefly describe two bit-mapped cache coherence protocols, namely CC-NUMA and COMA, for point-to-point networked shared memory multiprocessors. We assume that both the CC-NUMA and COMA machines have the same system organization as shown in Figure 1. A number of processing nodes are connected through a high bandwidth and low latency point-to-point interconnection network. Each processing node consists of a high performance processor, a cache (called processor cache), and a portion of global memory (called attraction memory, AM, in COMA) along with the corresponding directory. Cache coherence is maintained by a directory-based write-invalidate cache coherence protocol.

2.1 CC-NUMA Architecture

In this architecture, a memory block is always associated with a fixed location of a node called home node. The data in a memory block can only be duplicated among the processor caches of the processing nodes. Each memory block is associated with a directory for maintaining the consistency of the data among the processor caches and the corresponding home. In this paper, we assume the directory is fully bit-mapped [8]. There are *n* presence bits in the fully bit-mapped directory of a memory block. Each bit of the bit-mapped directory corresponds to a processing node. The directory stores the identity of any processing node that caches the data line by turning on the corresponding presence bit.

In addition, there are two additional bits to store the state information of the memory block. Three possible states of the memory blocks are *invalid*, *shared*, and *exclusive*. The cache blocks can also be in the invalid, shared, and exclusive state. A memory block that is not used by any processing node is in the invalid state and the data in the block is no valid. A shared memory block indicates that there are one or more than one processing nodes caching the line (in shared state) and all these processing nodes can read the data line concurrently. If the memory block is in the exclusive state, the data in the memory block is not valid and there is only one processing node caching the line exclusively. In other words, the processing node that exclusively owns the line can read and write locally.

The organizations of the cache and memory blocks are illustrated in Figures 1 and 2. We assume that there are $N = 2^n$ processors in the system. In each processor, there are $M = 2^m$ memory blocks and $C = 2^c$ cache blocks of $K=2^k$ bytes. The set associativity of caches is $S_c = 2^{s_c}$. The size of cache block is $K=2^k$ bytes, the same as memory block. There are $2^{c- s_c}$ sets in each cache and tag is of $n + m – c + s_c$ bits. Therefore, there are $2^{n+m-c+s_c}$ memory blocks mapping to a single cache set.

On a read miss on the cache, the request is sent to the home node. If the data is valid in the home node, the requested block is returned. Thus, two messages are needed to complete the read miss. If the data is exclusively owned by another node, and four messages are needed. The time taken for invalidation is proportional to the number of valid copies that can be large for applications with large degrees of sharing.

2.2 COMA Architecture

COMA architecture is motivated by the idea that data items should not be statistically allocated to fixed memory locations. COMA treats the main memory as another level of caches, called attraction memory. Data items dynamically migrate and replicate at attraction memory level as they do at the cache level in CC-NUMA. Most COMA systems use an inclusion approach to maintaining the cache consistency between processor cache and attraction memory. The inclusion approach means that the data items contained in processor cache must be contained in attraction memory. Data items in attraction memory can adapt to the application’s memory access pattern with automatic data migration and replication without the intervention from the operating system.

In contrast to CC-NUMA machines, data items in COMA machines have no fixed home. COMA machines must have a way to search for the data item being accessed. The interconnection networks for traditional COMA machines are normally hierarchical networks or the ones with efficient broadcasting capability. Take a hierarchical directory structure as an example. It is required to traverse the hierarchy to locate a data item when a miss in attraction memory occurs. The hierarchy traversal results in an increased internode miss latency. Broadcast may also be used to locate a data item in a non-hierarchical directory design. Note that in general broadcast is not an efficient function in a point-to-point connected network.

Additionally, replacing the last copy of a data item in attraction memory must be handled correctly and efficiently without generating too many replacement messages on the network. The costs for searching for a data item and handling the replacement of the last copy of a data item in a hierarchical network are usually very high. High memory pressure can lead to a devastating replacement traffic.

To reduce the replacement costs, a non-hierarchical COMA, called COMA-Flat (COMA-F) is proposed in [9]. COMA-F protocol takes the advantage of the concept of fixed home to reduce the overhead for data search and replacements. The organization of the cache blocks is identical to that of CC-NUMA as shown in Figure 3.

As traditional COMA architectures do, COMA-F uses physical addresses as the identifier for a data item in the system. The organization of the attraction memory is given in Figure 4. The memory consists of two fields. The first field is the data block. Additional cache tags and other related information are added in order to implement another level of the caches. The address of a data block is formed from the tag and the corresponding set index. As shown in Figure 4, the i^{th} data field in set j of processor *p* contains the memory block of address Y. The other field is the directory containing bit-map, state information, and master link. The directory records the coherence information of the data block whose address is determined by the memory block index. As shown in Figure 4, the i^{th} directory field in set j of processor *p* records the coherence information of a memory block with address X. We assume that the set associativity of attraction memory is $S = 2^s$. Therefore, there are 2^{m-s} sets in each attraction memory and tag is of n + s bits. There are 2^{n+s} memory blocks that can be mapped to a single attraction memory set.

Three possible states of the cache blocks are *invalid*, *shared*, and *exclusive*, the same as that in CC-NUMA. Similarly, attraction memory can have one of the following states: *invalid*, *shared*, *exclusive* and *master.* The states of attraction memory in COMA-F are similar to that in CC-NUMA, except the master state. The master state is used to keep record of the last copy of a data item in the system. Among many shared copies of a data item, there must be one and only one master copy. The directory of a memory block includes the following fields: an N-bit presence bit-map, a master link, and a 2-bit state field. The master link is used to point to the node containing the master or exclusive copy of the shared data block.

The allowable cache and attraction memory state combinations that follow the inclusion property are shown in Table 1. An inclusion bit is used to maintain the intranode cache consistency between cache and attraction memory. For a shared copy of a data block in attraction memory, setting the corresponding inclusion bit means the data block is also stored in the cache. The intranode coherence transactions are briefly described as follows.

Intranode transactions:

Consider the case that an invalidation message arrives at the attraction memory to invalidate a shared copy of a data block. If the inclusion bit is set, attraction memory forwards the invalidation message to invalidate the copy in the cache and get an acknowledgement back before invalidating the copy in the attraction memory.

Being in exclusive state coupled with the inclusion bit set implies that the data block in attraction memory is not valid. The valid exclusive data is in the cache. Thus, the read/write operations from processor can be performed locally without the intervention of attraction memory. An external read request to exclusive copy with the inclusion bit set requires the following transactions. First, the read request is sent to cache to get the up-to-date data. Then the cache changes the state from exclusive to shared and replies the requested block to attraction memory. After receiving the reply from cache, attraction memory stores the data block, sets the state to shared, and sends the requested block to the requester. The internode coherence protocol of COMA-F is briefly described as follows.

Internode transactions:

On a read miss, the request will be forwarded to attraction memory to locate the requested data block. If the requested data block cannot be found in attraction memory, the read request is then forwarded to the corresponding home node. If the recorded state of the requested block in directory is shared, the home node forwards the read miss request to the master node. In the mean time, the home node also sets the master link to the requester and turns on the presence bit corresponding to the requester. After receiving the read miss request, the master node sends the requested block to the requester and sets its state to shared. When the requester receives the requested data, it uses a replacement algorithm to select a place to hold the data block and set its state to master. The replacement algorithm will be discussed later.

If the state of the requested block is exclusive, the home node forwards the read miss request to the exclusive node. The home node also sets the master link to the requester, turns on the presence bit corresponding to the requester, and changes the state to shared. When the exclusive node receives the read miss request, it performs the intranode coherence transactions if necessary. The home node sets the state to shared and forwards the requested block to the requester. Transactions similar to the above situation are then followed after the requester receives the requested data.

When a write miss occurs in a cache, the request will be forwarded to attraction memory to locate the requested data. If the state of the requested block is in exclusive state, the write miss can be satisfied locally in attraction memory. The requested block is simply returned to the cache and the inclusion bit the memory block is set. Processor changes the state to exclusive and writes the data locally after receiving the requested block from attraction memory.

Now consider the situation that the write miss can not be satisfied locally in attraction memory. The write miss is first forwarded to the corresponding home node. If the state of the requested block recorded in the directory is shared, the home node sends invalidation messages to other shared copies, based on the bitmap in the directory. The master node sends the requested block to requester, changes state to invalid, and sends an acknowledgement back to home node after receiving an invalidation message. The shared node changes state to invalid and sends an acknowledgement back to home node after receiving an invalidation message. After the invalidation acknowledgement messages come back, the home node grants the exclusive right to the requestor by returning the requested block. The home node then sets the master link to requester, sets the state to exclusive, and turns on the presence bit corresponding to the requester. The coherence transactions are similar when the state of the requested block recorded in the directory is exclusive.

When a miss occurs in cache, a similar replacement policy to the traditional NUMA cache is followed. The replacement policy in cache is simple because of inclusion property between cache and attraction memory. Any data in cache is also stored in a corresponding attraction memory block. Thus, a replaced cache block in exclusive state always find a placeholder in attraction memory.

The replacements in attraction memory will not be as simple as the cache replacements. A master or exclusive block being replaced needs to find a place to reside. Determining which attraction memory has a space to hold the replaced memory block is not an easy task. If there is no special design for this problem, the number of replacement messages grows rapidly as the memory pressure increases in the system. COMA-F optimizes replacements by the following observation. A replacement message is generated only when a memory request misses in attraction memory and the attraction has no appropriate invalid or shared block to hold the requested block. The requested block is fetched from a remote location that has the state changing from master or exclusive to shared. Thus, the remote location should be able to accept the replaced block.

To implement this replacement optimization, the replacement algorithm must be performed after the requested block is fetched, but before the requested block is actually stored. Let's call the remote master node that provides the requested data the *old-master* node. The replacement message containing the replaced data block is augmented with the ID of the old-master node. The replacement message is first sent to the home node of the replaced block. Consider the case that there is another sharer in the system. The home node simply selects one of the sharers of the requested block to be the new master node by doing the following. The home node sets master link to the selected sharer and informs the selected sharer to be the new master by sending it a replacement message. The new master will acknowledge the home node after receiving the replacement message. Here, the replaced data block and the ID of the old-master node in the replacement message are not used.

The replaced data block and the old-master node ID are needed when there is no any other sharer of the replaced block in the system. In this case, the home node of the replaced block first sends the replacement message containing replaced data to the old master node and then goes to a temporary state to wait for an acknowledgement from the old master node. After receiving the replacement message, the old master node selects an invalid or a shared memory block as the victim for replacement and sends an acknowledgement back to its home directory. If the victim block is shared, a replacement acknowledgement is sent to the home directory of the victim block. If the inclusion bit of the victim block is set, a replacement message is sent to the local cache to invalidate the copy in cache. The selected victim block is then updated with the replaced data and changes to master state. Figure 5 illustrates the cache coherence transactions when a read miss on block A occurs in processor P1. Notice that the blocks of addresses A and B are mapped onto the same set of attraction memory blocks, the set *i* in this example.

2.3 Discussion of CC-NUMA and COMA-F

One noticeable advantage of CC-NUMA compared to other similar cache protocols is that most read misses take only two network messages to complete. The capacity misses are insensitive to the memory pressure in the system because the data in the memory blocks of CC-NUMA can only be duplicated in memory. As shown in Figure 3, there are $2^{n+m-c+s_c}$ memory blocks that can be mapped to a single cache set. If the number of $n + m - c + s_c$ is large, probability of having capacity misses can be very high.

COMA utilizes the unused memory blocks to back up the data being replaced in cache by treating the memory as another level of caches. When the memory pressure is low, the system has a lot memory space to hold the replaced data from cache. The performance of COMA will perform better than CC-NUMA. However, the coherency overhead can be large when there is no much free space to hold the shared data. In this situation, the performance of COMA will perform worse than CC-NUMA.

As stated in the replacement optimization algorithm of COMA-F, the old master node of the requested block is able to hold the replaced block. However, it is possible that by the time the replacement message arrives at the old master node, the shared copy of the requested block in old master node has been occupied by some other memory block. This situation can occur when the old master node makes a new request that is satisfied before the replacement message arrives.

There is no easy way to know which attraction memory is able to hold the replaced memory block after the interception of the shared copy of the memory block by a read miss in the old master node. To solve this problem, the system can follow the default replacement handling procedure by forwarding the replacement message to each processor in a round robin fashion until one attraction memory accepts it. The worst case number of replacement messages may be the same as the number of the attraction memory. The frequency of this occurrence is expected to be small when the memory pressure of the system is light. However, the worst case number of replacement messages incurred by an attraction memory miss can be very large if the memory pressure is high.

3 Proposed Protocol

In COMA, data items can dynamically migrate and replicate as the code is in progress without any special control scheme. Since there is no fixed location for a data item, a broadcast mechanism or a hierarchical directory is always employed for locating the data item. Broadcast wastes a lot of network bandwidth for point-to-point interconnection networks that do not have the broadcast hardware support. A hierarchical directory also incurs longer delays for coherence transactions.

In this section, we will develop a new coherence scheme that avoids using the broadcast mechanism and the hierarchical directory to locate a data item in the system. The organizations of attraction memory and cache are similar to that of COMA-F. The aims of the proposed protocol are to have similar performance as COMA-F in low memory pressure and as CC-NUMA in high memory pressure. In other words, when the system has a lot of free memory space, the proposed protocol will duplicate the shared data as it is needed. Also, when there is a little or even no memory space to duplicate data, the proposed protocol forces the master copy of a data block to be stored in its home node. Thus, in high memory pressure, only two network messages are needed to complete a read miss in attraction memory. To best utilize the space in caches for duplicating shared data, we need to break the inclusion property between cache and attraction memory. Otherwise, as we will show later, the proposed protocol will be the same as NUMA memory systems.

As stated, there is no fixed location for a data item in COMA. A special care must be taken if the last copy of the data is being replaced. Recall that there are two fields in an attraction memory block shown in Figure 4. The first one is the directory field that records the coherence information of a data block whose addresses are determined by its memory block index. The second field is the data field that keeps the contents and other related information of a data block whose addresses are determined by its tag field. If an attraction memory block is not physically allocated to a program, the corresponding directory field will not be used in any coherence operation. We distinguish whether the directory of an attraction memory block is used or not by checking the state of the corresponding directory. An attraction memory block is used by a program if its directory field is invalid state. Otherwise, it is used by a program.

As in COMA-F, the master copy of a data block in the proposed protocol can reside in the data field of any block in an appropriate attraction memory set. However, there is one constraint in the proposed protocol: only the invalid or shared data block can occupy the data field of another attraction memory block if the corresponding directory is used. The attraction memory block can be occupied by any data block in any state if the corresponding directory is not used. Consider the example shown in Figure 4. When the data block of address X is allocated to a program, the associated directory field is used to maintain the coherency of the data block. The data block of address Y occupying the associated data field can only be in invalid or shared state. If a master data block is in the i^{th} block of attraction memory set j, it must have address X. We call the attraction memory block has its own data block. As in COMA-F, we keep the same policy that the copy of a data block that newly joins the sharing list becomes the master. Thus, if a master copy of a data block resides in its home node, it cannot be replaced. Notice that the initial place of a physically allocated data block is in its home node. The states of attraction memory and cache are the same as that of COMA-F. The above constraint forces the master copy of a data item fall back to its home node when the memory pressure is high.

It is possible that all the blocks of an attraction memory set contain their own data block in master or exclusive state when the memory pressure is high. We call this kind of attraction memory set a *highest pressured set*. Based on the above constraint, no attraction memory block in a highest pressured set can be used to store the requested block. Only cache can store the requested block in this situation. This is the reason why in the proposed protocol the inclusion property is broken. We introduce a new flag called *set pressure flag* to record which memory set is the highest pressured set. Using set pressure flag avoid checking the state of all the blocks in a attraction memory set. Set pressure flag is helpful when the memory pressure in the system is high.

The proposed protocol can be implemented by using a bit-mapped or linked list directory structure. Assuming a full-map directory, only the coherence transactions of the proposed protocol that are different from COMA-F are described as follows.

When a miss occurs in the local cache, the requested data is obtained from the local attraction memory or from the remote master or exclusive copy. The replacement operation on a shared block is performed by simply returning a replacement message to the corresponding home directory, which is the same as COMA-F. If a master block is selected for replacement, it must not be in its home node. In this case, the replaced data is sent back to its home location. When a home node receives its own data block being replaced, it first checks if there is a shared data block occupying the data field. If yes, the home node performs another replacement operation to replace the shared block. Then the home node writes the replaced data block in its home data field. Notice that the replacement operation can be done before or after the requested data block is replied. At most two network messages are generated for replacing a master or exclusive block.

If there is no appropriate attraction memory block to be selected for replacement, the requested block will be forwarded to the cache of the requester. When a cache block being in exclusive state is replaced, it is first written back to the corresponding attraction memory. If there is no appropriate memory block to hold the replaced data, the replaced data is forwarded to its home node. When a shared cache block is replaced, it first sends a replacement message to the corresponding attraction memory. If there is a valid copy of the replaced block in the attraction memory, the replacement operation is done by setting the corresponding inclusion bit off. Otherwise, the replacement message is forwarded to its home node.

Consider the operation when an attraction memory receives an invalidation message. If the associated block exists in the attraction memory, the copy in cache is first invalidated when the corresponding inclusion bit is set. After receiving an acknowledgement from the cache, the copy in the attraction memory is invalidated and an acknowledgement is sent back to its home node. If the associated block does not exist in the attraction memory, the invalidation message is forwarded to the cache. Then the copy in the cache is invalidated and an acknowledgement is sent back to its home node directly.

Figure 6 illustrates an example of a read miss operation. Processor P1 issues a read request to block A. Since all the blocks in the corresponding attraction memory set are in master state, a replacement operation is performed first. Suppose that the first block B is selected for replacement. P1 sends a replacement message to the home node of block B. After receiving the replacement message, the block U in shared state is replaced by sending another replacement message to U’s home. Then the data of block B is written into its home data field. After block B in P1 is replaced, P1 starts to read the data of the requested block A by sending the read request to block A’s home. A’s home sets the master link to P1, sets the presence bit corresponding to P1, and forwards the request to P2. P2 then changes the state of block A from master to shared and forwards the data of block A to P1. After P1 receives the requested data, it saves the data in the invalid block and forwards the data to local cache.

4 Performance Evaluation

This section begins with a discussion of relationship between memory pressure and the number of messages required for misses on cache and attraction memory in CC-NUMA, COMA-F, and the proposed protocol. We also discuss the number of replacement messages from a reference miss on attraction memory. Afterward, we analyze the execution time for the three protocols given the memory pressure and memory associativity. The analysis assumes that references to memory are uniformly distributed and independent of each other.

In CC-NUMA, only caches are used to store the most recently referenced data. The data is always obtained through the directory that is associated with the home memory block. For a read miss on the cache, two network messages are needed if the data at home is valid and four network messages are needed if the data is exclusively owned by another cache. The number of messages required for serving a read miss does not depend on the type of misses, i.e. capacity and coherence misses. CC-NUMA's performance does not depend on the memory pressure in the system either. There is no network message generated by replacing the shared copy of a cache block. Only one replacement message sent to the home node is generated by replacing the exclusive copy of a cache block. The replacement overhead of CC-NUMA is in fact minimal.

In COMA-F, the traditional memory is used as the second level of caches, the attraction memory. For the coherence misses, the requested data is obtained from the master copy of the block through the home directory. COMA-F allows the master copies of data blocks freely migrate from one processor to another. The home location of a memory block is normally different from the location of the master copy. Therefore, two network messages are needed for serving the coherence read miss if the master or exclusive copy of the block is in its home node. However, four messages are needed when the requested block is exclusively owned by a node that does not locate in its home node. If the memory pressure is low in the system, there are more non-allocated memory blocks. Most capacity misses can be serviced by the attraction memory. No network message is generated. However, when the memory pressure is high, there are less unused memory blocks to store the requested data. Similar to the coherence miss, three messages are normally needed in this case.

Consider the replacement overhead in COMA-F. When replacing the shared copy of a memory block, one replacement message is generated. The replacement message is sent to the home node that in turn updates the directory by turning off the corresponding presence bit. When the master or exclusive copy of the block is being replaced, one replacement message is also generated and sent to the home node. If there are other shared copies in the system, the home node selects one to be the new master. Then the home node sends a confirmation message to the selected node and waits for the acknowledgement. Three messages are needed in this case. Notice that the directory at the home node acts as a central point to coordinate the replacement operation. The way of using a central directory avoids the access through it before the replacement operation is complete. If the replaced block is exclusive or it is the master copy without other sharer in the system, the candidate node is the one providing the requested data for a reference that causes this replacement. The candidate node also needs to send a replacement message to its home node. Four replacement messages are needed as illustrated in Figure 5. However, it is possible that by the time the replacement message arrives at the candidate node, the shared copy of the block has been occupied by some other memory block. In this situation, the worst case number of replacement messages required for finding a place to hold the replaced block can be as large as 2N+2, where N is the number of attraction memories in the system.

In the proposed protocol, the number of network messages generated for serving a read miss and the number of incurred replacement messages are almost the same as that of COMA-F when the memory pressure is low. There is a high possibility that the master copy of a memory block can stay in its home node when the memory pressure is low. Thus, the number of a read miss on attraction memory can be two that is better than COMA-F. In high memory pressure, at most two replacement messages are generated for replacing a master copy of a block, as shown in Figure 6. The worst case situation as in COMA-F will not happen. The number of network messages needed to serve a read miss will be two when the memory pressure is high. The comparisons of the number of messages needed for a miss are summarized in Table 3, where we assume that the requesting node, the home node, and the node holding master or exclusive copy are different.

We now analyze the execution time of these three different protocols as follows. The various notations used in the analysis are listed in Table 4. We ignore the difference between read and write misses, the invalidation process, and the replacement process because they all have the same impact on all these three protocols. The analysis below is just for comparison purpose.

The expected execution time for CC-NUMA is given in equation (1).

$$h \times t_{cache} + (alpha + beta)\left\{\begin{array}{l} t_{cache} + \frac{1}{N}\left[t_{dir} + t_{mem}\right] + \\ \frac{N-1}{N}\left[t_{net.cmd} + t_{dir} + t_{mem} + t_{net.data}\right] + t_{cache} \end{array}\right\} \qquad (1)$$

The expected execution time for COMA-F is given equation (2). The probability that the requested block from a miss on the cache can not be found in the attraction memory is equal to P_s. P_s is the probability that M items can be randomly distributed among B locations where a pre-selected group of S locations all contain one f the M items.

$$h \times t_{cache} + alpha\left\{\begin{array}{l}(1-P_s)\times(t_{cache}+t_{mem}+t_{cache})+\\ P_s \times\left(\begin{array}{l}t_{cache}+t_{mem}+\frac{1}{N}\left[t_{dir}+t_{net.cmd}+t_{mem}+t_{net.data}\right]+\\ \frac{N-1}{N}\left[t_{net.cmd}+t_{dir}+\frac{N-2}{N-1}t_{net.cmd}+t_{mem}+t_{net.data}\right]+t_{mem}+t_{cache}\end{array}\right)\end{array}\right\}+$$

$$beta\left\{\begin{array}{l}t_{cache}+t_{mem}+\frac{1}{N}\left(t_{dir}+t_{net.cmd}+t_{mem}+t_{net.data}\right)+\\ \frac{N-1}{N}\left(t_{net.cmd}+t_{dir}+\frac{N-2}{N-1}t_{net.cmd}+t_{mem}+t_{net.data}\right)+t_{mem}+t_{cache}\end{array}\right\} \qquad (2)$$

The expected execution time for the proposed protocol is given by equation (3). The time t_{pflag} is taken because the protocol checks it before accessing the attraction memory. The probability that the requested block from a miss on the cache can not be found in the attraction memory is assumed to be P_s. The reason is that the migration freedom of master memory blocks in the proposed protocol is roughly the same as that of COMA-F.

$$h \times t_{cache} + alpha\left\{\begin{array}{l}(1-P_s)\times(t_{cache}+t_{pflag}+t_{mem}+t_{cache})+\\ P_s \times\left(\begin{array}{l}t_{cache}+t_{pflag}+(1-P_s)\times t_{mem}+\frac{1}{N}\left[t_{dir}+t_{net.cmd}+t_{mem}+t_{net.data}\right]+\\ \frac{N-1}{N}\left[t_{net.cmd}+t_{dir}+(1-P_s)\times t_{net.cmd}+t_{mem}+t_{net.data}\right]+\\ (1-P_s)\times t_{mem}+t_{cache}\end{array}\right)\end{array}\right\}+ \qquad (3)$$

$$beta\left\{\begin{array}{l}t_{cache}+t_{pflag}+(1-P_s)\times t_{mem}+\frac{1}{N}\left(t_{dir}+t_{net.cmd}+t_{mem}+t_{net.data}\right)+\\ \frac{N-1}{N}\left(t_{net.cmd}+t_{dir}+(1-P_s)\times t_{net.cmd}+t_{mem}+t_{net.data}\right)+(1-P_s)\times t_{mem}+t_{cache}\end{array}\right\}$$

We plot the expected execution time for CC-NUMA, COMA-F, and the proposed protocol based on latencies listed in Table 5. Plots are illustrated for various configurations of hit rate, capacity miss rate, coherence miss rate, and memory pressure (M/B ratio). We use the hit, capacity miss, and coherence miss rates of three benchmark programs [15,16] to compute the performance of CC-NUMA, COMA-F, and the proposed protocol.

Figure 7 shows the expected execution for Barnes-Hut. As we can see, the COMA-F and the proposed protocol perform equally well in the low memory pressure. The CC-NUMA performs the best when memory pressure is greater than 0.9. The proposed protocol performs very close to CC-NUMA in the high memory pressure. Figure 8 shows the expected execution for FFT which is similar to Barnes-Hut. Figure 9 illustrates the results for Cholesky. CC-NUMA performs better than the other two protocols in all cases. The reason is that the capacity miss rate is very small. In Figure 9, the performance of the proposed protocol is better than COMA-F.

5 Conclusion

In this paper, we have presented a new COMA protocol for shared-memory multiprocessors with non-hierarchical architectures. There are two advantages of the proposed protocol over COMA-F. The first advantage is that the replacement overhead is low in all possible cases. The second one is that the proposed protocol performs well both in the low and high memory pressure. We also gave a simple model for analyzing the execution time of three cache coherence protocols. The analysis shows that the performance of the proposed protocol is what we expected: the proposed protocol performs as well as COMA-F in low memory pressure and performs close to CC-NUMA in high memory pressure.

References

[1] L. M. Censier and P. Feautrier, “A New Solution to Coherence Problems in Multicache Systems,” IEEE Transactions on Computers, pp. 1112--1118, December 1978.

[2] M. Thapar, B. Delagi, and M. J. Flynn, “Linked List Cache Coherence for Scalable Shared Memory Multiprocessors,” In Proc. International Parallel Processing Symposium, pp. 34--43, April 1993.

[3] IEEE, IEEE Std 1596-1992: IEEE Standard for Scalable Coherent Interface, IEEE, Inc., 345 East 47th Street, New York, NY 10017, U.S.A., August 1993.

[4] D. Chaiken, J. Kubiaowicz, and A. Agarwal, “LimitLESS Directory: A Scalable Cache Coherence Scheme,” In Proceedings of ASPLOS-IV, pp. 224—234, April 1991.

[5] Y. Chang and L.N. Bhuyan, "An Efficient Hybrid Cache Coherence Protocol for Shared Memory Multiprocessors", IEEE Transactions on Computers, pp. 352-360, March 1999.

[6] P. Stenstrom, “A Survey of Cache Coherence for Multiprocessors”, IEEE Computer, Vol. 23, No. 6, pp. 12-24, June 1990.

[7] Fredrik Dahlgren and Josep Torrellas, "Cache-Only Memory Architectures", IEEE Computer Magazine, pp.72-79, June1999.

[8] W. D. Weber and D. Gupta, “Analysis of Cache Invalidation Patterns in Multiprocessors”, ASPLOS-III, pp.243-256, 1989.

[9] Kendall Square Research Corporation, Kendall Square Research: Technical Summary, Kendall Square Research Corporation, Waltham, MA, 1992.

[10] E. Hagersten, A. Landin, and S. Haridi, “DDM – A Cache-Only Memory Architecture”,IEEE Computer, pp. 44-54, September, 1992.

[11] A. Saulsbury, T. Wilkinson, and J. Carter, “ An Argument for Simple COMA”, First IEEE symposium on High Performance Computer Architecture, pp. 276-285, Jan., 1995.

[12] P. Stenstrom, T. Joe, and A. Gupta, “Comparative Performance Evaluation of Cache-Coherent NUMA and COMA Architectures”, Proceedings of the 19th Int’l Symposium on Computer Architecture, pp. 800-91, 1992.

[13] David Lilja, “Cache Coherence in Large-Scale Shared Memory Multiprocessors: Issues and Comaprisons”, ACM Computing Surveys, Vol. 25, No. 3, pp. 303-338, September 1993.

[14] Truman Joe, “COMA-F: A Non-Hierarchical Cache Only Memory Architecture”, Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, March 1995.

[15] J.P.Singh, A. Gupta, and M. Levoy, “Parallel Visualization Algorithms: Performance and Architectural Implementation”, IEEE Computer, 27(7):45—55, July 1994.

[16] J.P. Singh, W.D. Weber, and A.Gupta, “SPLASH: Stanford Parallel Applications for Shared Memory,” Technical Report CSL-TR-92-526, Stanford University, Stanford, CA, 1992.

[17] Q. Yang, L. N. Bhuyan, and B.-C. Liu, “Analysis and Comparison of Cache Coherence Protocols for a Packet-Switched Multiprocessor,” IEEE Transactions on Computers, vol. 38, no. 8, pp. 1143--1153, August 1989.

[18] S. Basu and J. Torrellas, "Enhancing Memory Use in Simple Coma: Multiplexed Simple Coma", Fourth International Symposium on High-Performance Computer Architecture, February 1998.

[19] Z. Zhang and J. Torrellas, "Reducing Remote Conflict Misses: NUMA with Remote Cache versus COMA", Third International Symposium on High-Performance Computer Architecture, January 1997.

[20] Babak Falsafi and David A. Wood, "Reactive NUMA: A Design for Unifying S-COMA with CC-NUMA", ACM/IEEE International Symposium on Computer Architecture (ISCA), June 1997.

[21] Z. Zhang, M. Cintra, and J. Torrellas, "Excel-NUMA: Toward Programmability, Simplicity and High Performance", IEEE Transactions on Computers, Special Issue on Cache Memory and Related Problems, vol. 48, no. 2, pp. 256-264, Feb. 1999.

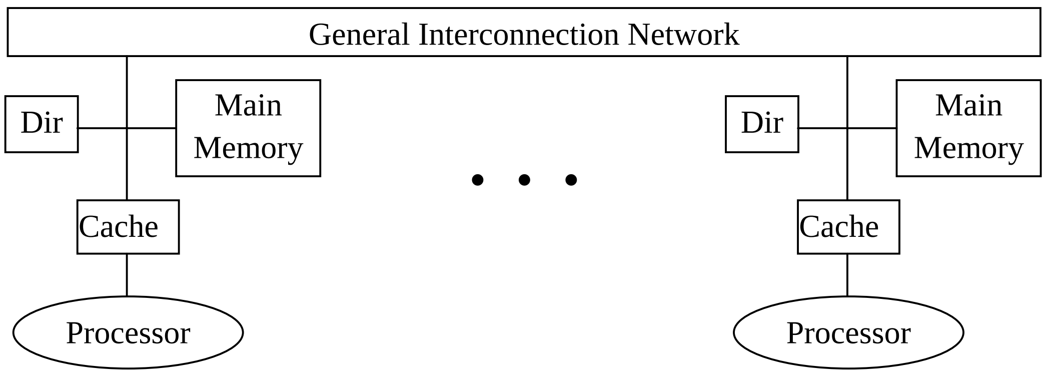


Figure 1: System organization for shared memory architecture.

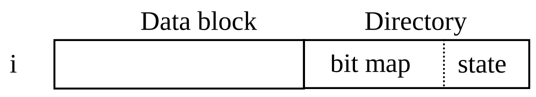


Figure 2: Organization of memory blocks for CC-NUMA

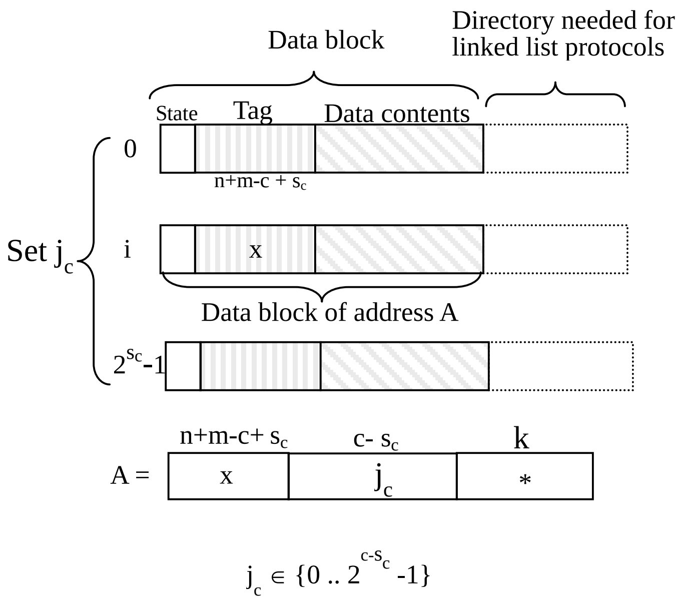


Figure 3: Organization of cache blocks for CC-NUMA and COMA-F

		Cache State		
		Invalid	Shared	Exclusive
Attraction Memory State	Invalid	X		
	Shared	X	X *	
	Master	X	X *	
	Exclusive	X	X *	X *

Table 1: Allowable state combinations of cache and attraction memory in COMA-F, where * means that the inclusion bit is turned on.

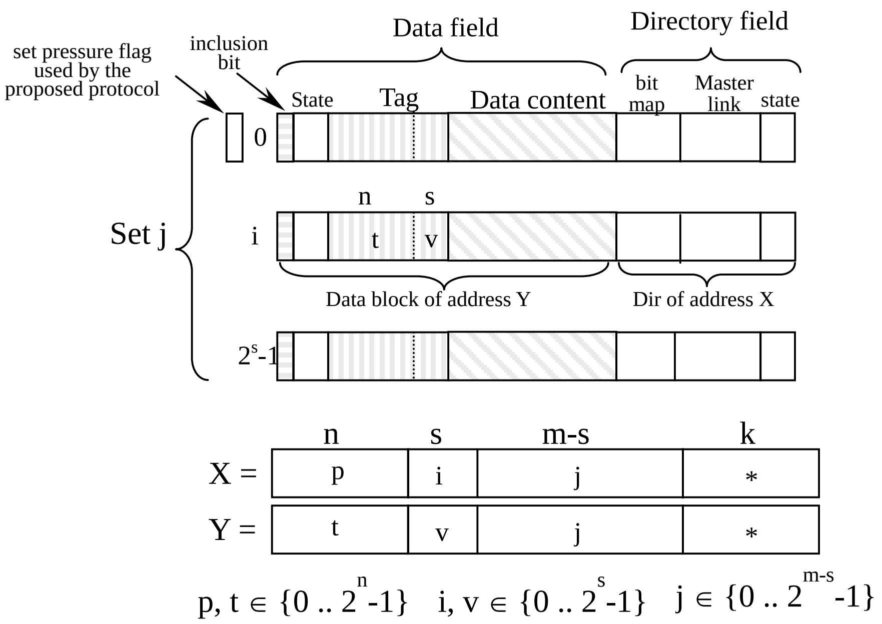
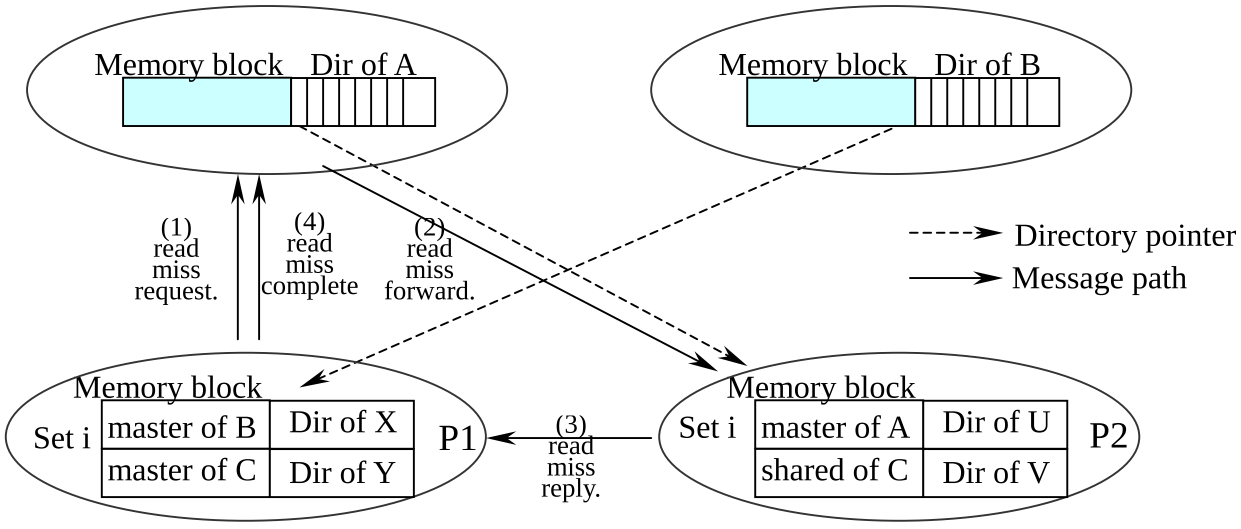


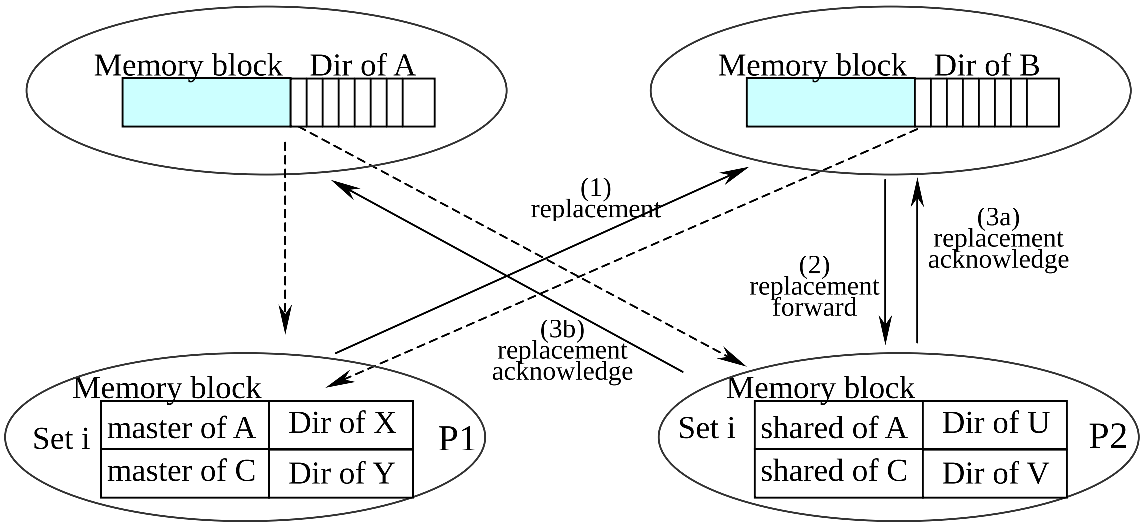
Figure 4: Organization of attraction memory of processor *p* in COMA-F and the proposed protocol.

		Cache State		
		Invalid	Shared	Exclusive
Attraction Memory State	Invalid	X	X	X
	Shared	X	X *	
	Master	X	X *	
	Exclusive	X	X *	X *

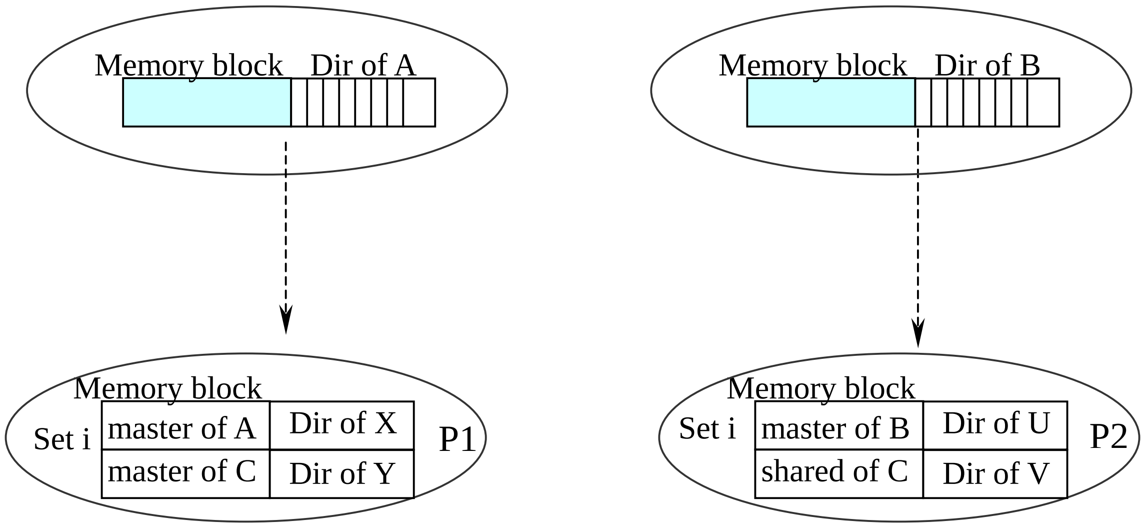
Table 2: Allowable state combinations of cache and attraction memory in the proposed protocol, where * means that the inclusion bit is turned on.



(a) Read miss transactions.

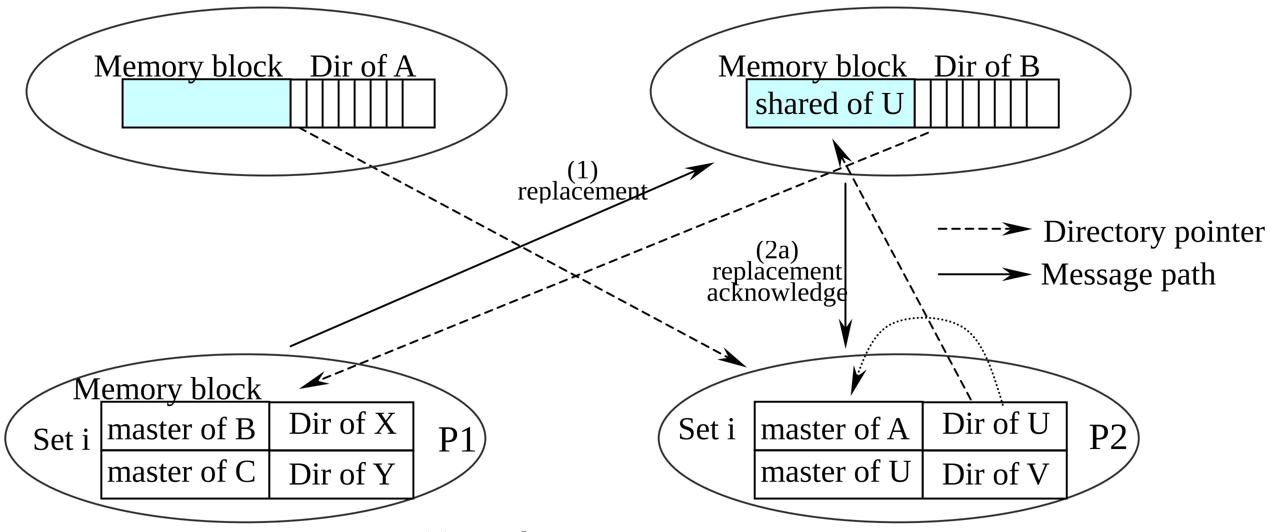


(b) Replacement transactions.

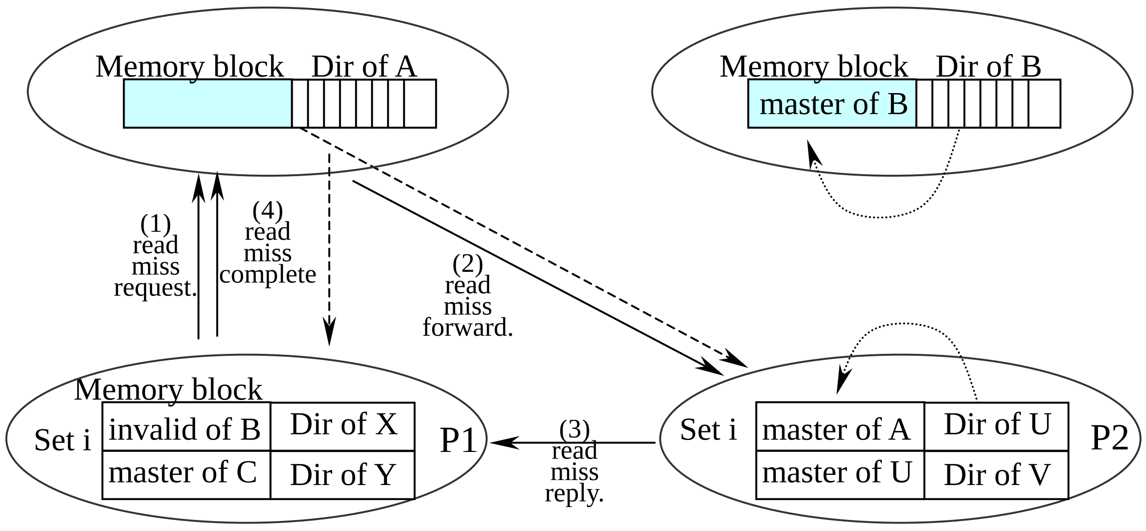


(c) Final state after completing the read miss.

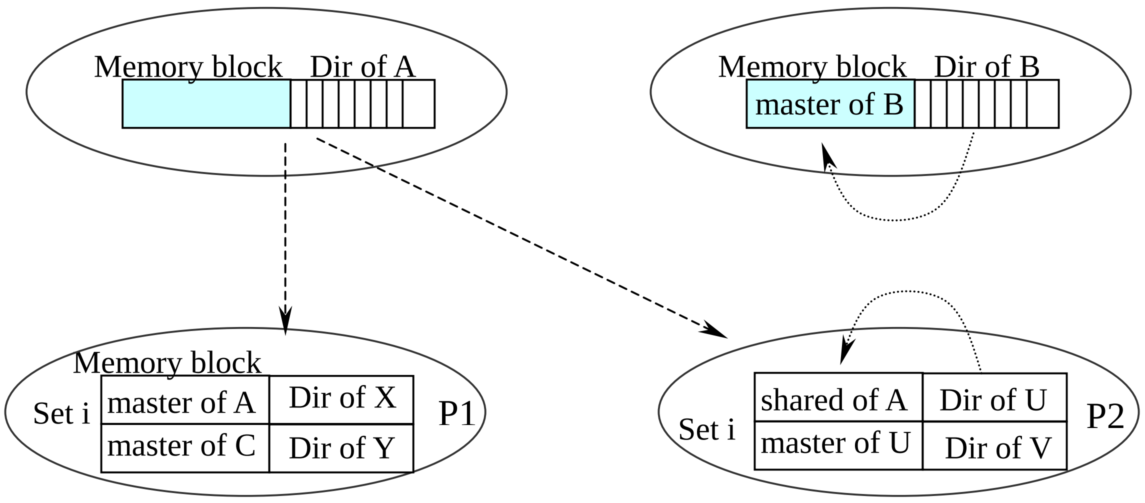
Figure 5: Coherence transactions for COMA-F when a read miss occurs on memory block A in P1.



(a) Read miss transactions.



(b) Replacement transactions.



(c) Final state after completing the read miss.

Figure 6: Coherence transactions for the proposed protocol when a read miss occurs on memory block A in P1.

	# of network messages needed for serving a read miss on cache		# of network messages needed for serving a read miss on attraction memory		# of replacement messages generated by replacing a memory block	
	Best Case	Worse Case	Best Case	Worse Case	Best Case	Worse Case
CC-NUMA	2	4	N/A	N/A	0	1
Optimized COMA-F	0	4	4	4	3	2N+2
Proposed Protocol	0	4	4	4	3	3

Table 3: Comparisons of number of messages used in CC-NUMA, COMA-F, and the proposed

h	:hit rate on processor cache
alpha	:capacity miss rate on processor cache
beta	:coherence miss rate on processor cache
M	:the number of master blocks in each attraction memory
B	:the number of memory blocks in each attraction memory
N	:The number of processors in the system
S	:set associativity of an attraction memory
P_s	: $\dfrac{\binom{B-S}{M-S}}{\binom{B}{M}}$ probability that an attraction memory miss also occurs when there is a capacity miss in processor cache.
t_{cache}	:cache access time
t_{mem}	:memory access time
t_{dir}	:directory access time (assume implemented with cache)
t_{pflag}	Set pressure flag in the proposed protocol (assume implemented with cache)
$t_{net.cmd}$	:communication network latency
$t_{net.data}$	:data network latency

Table 4: Notations for the performance evaluation.

P_s	$= \sim(M/B)^S$
t_{cache}	= 1 processor clock
t_{mem}	= 32 processor clocks
t_{dir}	= 1 processor clock
t_{pflag}	= 1 processor clock
t_{netcmd}	= 12 processor clocks
$t_{netdata}$	= 20 processor clocks

Table 5: Memory and communication latencies.

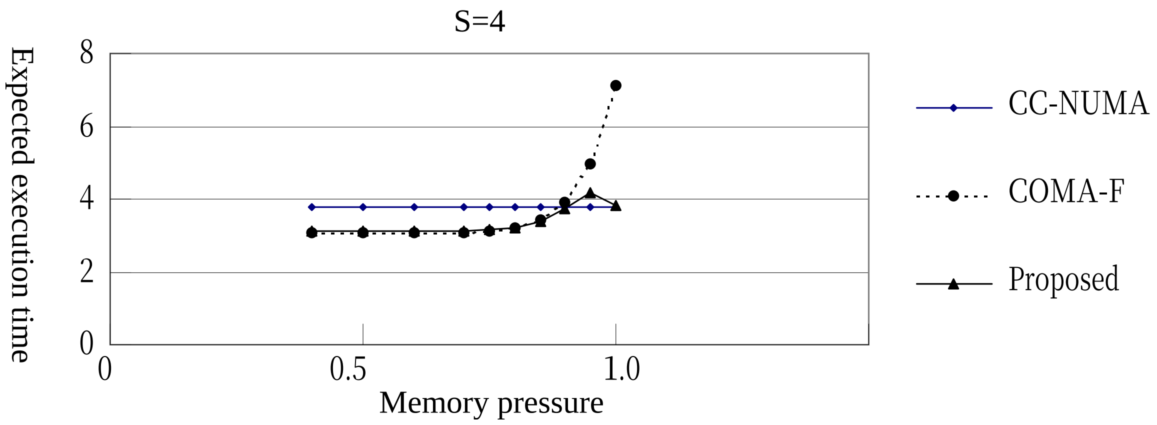


Figure 7: Plot for Conjugate Gradient with *h*=96%, *aloha*=3.5%, *beta*=0.5%.

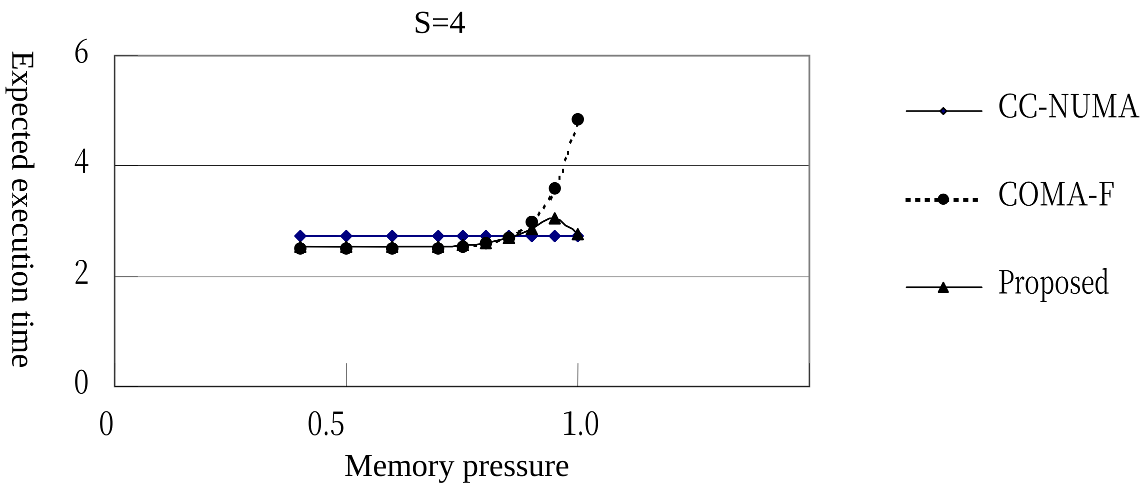


Figure 8: Plot for FFT with *h*=97.5%, *alpha*=2%, *beta*=0.5%.

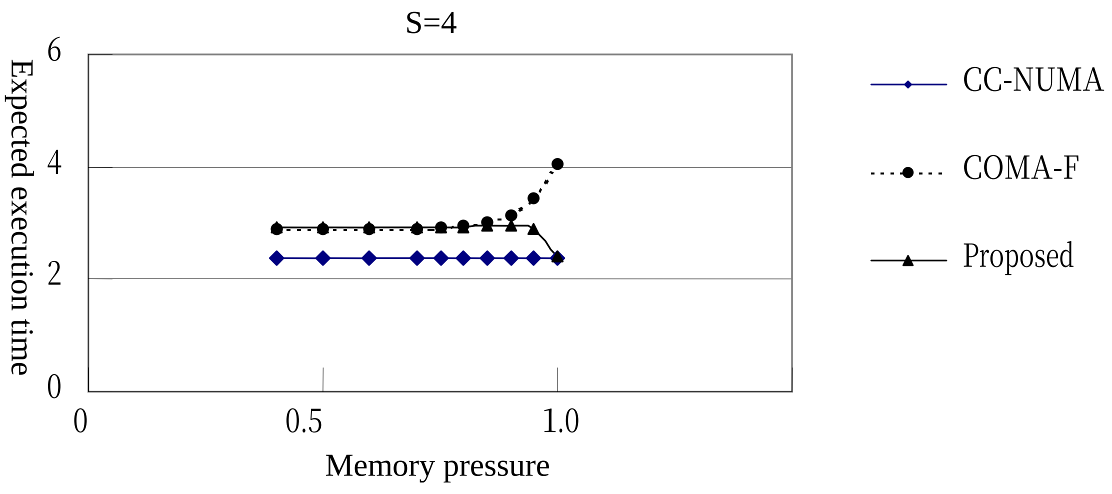


Figure 9: Plot for Cholesky with *h*=98%, *alpha*=1%, *beta*=1%.