Reducing the Replacement Overhead on COMA Protocols for Workstation-Based Architectures

Diego R. Llanos Ferraris, Benjamín Sahelices Fernández, and Agustín De Dios Hernández

Computer Science Department, University of Valladolid, Spain.
{diego, benja, agustin}@infor.uva.es

Abstract In this paper we discuss the behavior of the replacement mechanism of well-known COMA protocols applied to a loosely-coupled multicomputer system sharing a common bus. We also present VSR-COMA, a COMA protocol that uses an advanced replacement mechanism in order to select the destination node of a replacement transaction without introducing penalties due to the increase of the network traffic. Our comparative study of the behavior of different replacement mechanisms in the execution of Splash-2 programs confirm the effectiveness of the VSR-COMA protocol for this kind of systems.

1 Introduction

The use of workstation networks as loosely-coupled multicomputer systems allows to build distributed shared memory architectures with good price/performance ratio [1]. The main drawback of this kind of architectures is the use of a common bus: a slow transmission media that constrains the speedup and the scalability of these systems.

The design of a COMA protocol [4] for workstation-based architectures has already been proposed. COMA-BC [7] is a bus-based COMA protocol that reduces the network traffic using a snoopy-directories hybrid mechanism. This approach leads to a lower number of messages across the network. COMA-BC protocol has been evaluated running different Splash-2 programs [2], and good speedups have been obtained.

COMA-BC, however, presents a major drawback: there is no replacement mechanism. Instead, each attraction memory (AM) has the same size as the shared address space. This approach simplifies the protocol design, because there is no need to replace a block in order to make free space in the local AM, but it does not allow an efficient use of the AM space of each node.

There are two distributed COMA protocols, COMA-F [5] and DICE [3], that incorporate replacement strategies. As we will see, both approaches lead to a considerable overhead when are applied to a common bus-based COMA architecture.

A new bus-based COMA protocol that pretends to solve this problem has been developed. VSR-COMA (Valladolid Smart Replacement COMA) [6] is a

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 550–557, 2000.

COMA protocol that incorporates a new replacement mechanism. Every VSR-COMA cache-coherency controller knows the situation of each cache line in each remote AM of the system. This allows to choose the most appropriate destination node without introducing more traffic in the interconnection network. This solution also allows the local cache-coherency controller to make the decision based on more sophisticated, protocol-independent algorithms. In addition, this solution is not affected by the memory pressure.

The rest of this paper is organized as follows: Section 2 discusses in more detail the replacement strategies used in distributed COMA protocols. Section 3 introduces the VSR-COMA protocol. Section 4 describes the replacement strategy used in VSR-COMA. Section 5 makes a speedup comparison based on a simulation study of the protocols mentioned above. Finally, Section 6 shows our conclusions.

2 Replacement Strategies in COMA Protocols

The main problem in the replacement mechanism in distributed COMA protocols is the selection of the destination node. When a node needs to make free space and is the owner of every block in the corresponding set of the local AM, the need of sending the ownership of a block to another node arises. We have two problems. First, which block should be selected. Second, which node should be the destination for the replacement operation. The former is not so difficult to solve: first of all, we will try to transfer the ownership of a block that has another copy in a remote AM. If it is not possible, we need to transfer an "exclusive block", a single-copy block. The latter is more complex, because in a distributed COMA environment the nodes do not know which remote node has enough free space to accept the block.

Two approaches that pretend to solve this problem have been proposed: the random selection of the destination node, used in the COMA-F architecture [5], and the 4-level priority scheme of the DICE protocol [3]. The random selection consists on the selection of the destination node on a random basis. If the node accepts the block, it sends an acknowledgment message: the ownership has been transferred. If not, the remote node sends a negative acknowledgment and the ownership remains in the original node. In this case, the node should choose another node and try again. Note that at high memory pressures, the probability of choosing the "right" node (the one with enough free space) decreases when the number of nodes is increased.

The 4-level priority scheme used in DICE works as follows. The node sends a message asking for the state of the corresponding set of every remote AM. All the remote nodes answer this message with a 4-level code that reflects its situation: i) The node has a copy of the block; ii) the node does not have a copy but it has a free cache line; iii) the node has every block in use, but there is at least one block that could be overwritten (the node does not have the ownership of it) and iv) the node has the ownership of every block in the set. This 4-level code acts as a priority scheme. In the fourth case, the protocol establishes

that the replaced block must be exchanged with the required block -that is, a swapping technique. DICE replacement technique allows the node that starts the replacement to choose the best destination node, but at a high cost: for n nodes, $n + 2$ messages are needed to complete the replacement transaction (one request, $n - 1$ responses, the ownership transfer and its acknowledgment).

As we can see, both systems require to send several messages to complete the replacement transaction. In addition, the number of required messages increases with the number of nodes in both systems. This leads to a considerable overhead in COMA systems that runs over networks of workstations, where the bus speed is the main bottleneck.

3 The VSR-COMA Protocol

The design goals of VSR-COMA are focused on the construction of a COMA protocol useful for a common bus multicomputer system. The main goal is to reduce the network traffic, based on the broadcast feature of the common bus. Since each message sent by a processing node is received by every node in the system, each node could theoretically know the block distribution in the remote AMs. This characteristic will allow the VSR-COMA cache-coherency controllers to trace the situation in all the remote AMs, and therefore to choose the best destination node for a replacement request without introducing extra messages.

The cache-coherency controller of a VSR-COMA node manages three basic data structures: the directory table, the state+tag table, and the replacement table. The directory table holds the information related to the owner of each block in the system. Every request in VSR-COMA protocol should be sent to the owner of the block. The state+tag table keeps record of the state of every cache line in the local AM, with the corresponding tags. Both structures are similar to those found in other directory-based COMA architectures.

The third data structure is the replacement table. The replacement table keeps track of the state of every cache line in the *remote* AMs, with the corresponding tags. This information is updated by the cache controller as follows: when the cache controller receives an event (remember that in a common bus architecture, every node sees every event generated in the system), it updates its directory and replacement tables with the information present in this event, i.e. if the event is a read request, every cache controller notices that the sender wants to read a block, and therefore it updates the corresponding data entry in the replacement table. To do this, each event carries on the tag information of the block requested and also the number of the cache line *inside the set* that will be updated. This information is enough for the cache-coherency controllers to keep track of the evolution of all the remote AMs.

This approach has a possible drawback. It would seem that we would have a considerable memory overhead in order to maintain such information replicated in every cache controller. This problem is not so significative: our results shows a memory overhead two to three times higher for our system than for a similar

system without replacement information, and does not exceed a 10% overhead [6].

3.1 Events, States, and Operations

There are two types of states in VSR-COMA: *stable* states and *transient* states. The stable states are similar to the states we found in DICE [3]: Inv (the cache block is not valid), Shared (the cache block is valid for reading, that is, the local node can read the information but it cannot overwrite it, since the local node is not the owner of the block), SharOwn (the local node is the owner of the block but possibly there are other copies of the block in remote AMs) and Excl (the local node is the owner of the block and there is exactly one copy of it).

Transactions can be overlapped in a single bus network. The utilization of this kind of bus (called "split-transaction bus") leads to the need of transient states in the protocol, indicating that a particular operation is still in progress. VSR-COMA uses three transient states: InvWExcl ("invalid, but waiting for an Exclusive block"), InvWExcl ("invalid, but waiting for a Shared copy of a block"), and WExport ("valid, but exists an export transaction in progress").

In VSR-COMA terminology, every message sent between the nodes is called an *event*. A pair of request-response events is called a *transaction*. Every event in VSR-COMA protocol has a source and a destination node. There are nine different events:

BusRreq : Bus read request.
BusRack : Bus read acknowledgment.
BusWreq : Bus write request.
BusWack : Bus write acknowledgment.
BusFInv : Bus fast invalidation. It is used by a node that has a SharOwn block and wants to modify it. This event invalidates every copy of this block in the remote AMs.
BusEXreq : Bus export request. This event includes the exported block, and is used to send the ownership of a block to another node.
BusEXack : Bus export acknowledgment.
BusEXnak : Bus export negative acknowledgment. The destination node cannot accept the incoming block due to space problems. The first node keeps the ownership of the block.
BusRACE : Bus race condition. This event is used when a node receives a request about a block that has another owner. This situation can be produced due to a transaction overlapping.

Finally, VSR-COMA establishes a simple memory protocol that allows the processors to request operations to the cache controller of each local node. This protocol has the following *memory operations*: PrRd (processor read), PrWr (processor write), PrTAS (processor test-and-set) and PrFAI (processor fetch-and-increment). The last two operations are used to implement synchronization operations: semaphores and barriers.

3.2 State Transition Diagram

Figure 1 shows the VSR-COMA state transition diagram. Note that in the figure we only consider the PrRd and PrWr memory operations, because PrTAS and PrFAI memory operations behave exactly like a PrWr operation from the network protocol point of view.

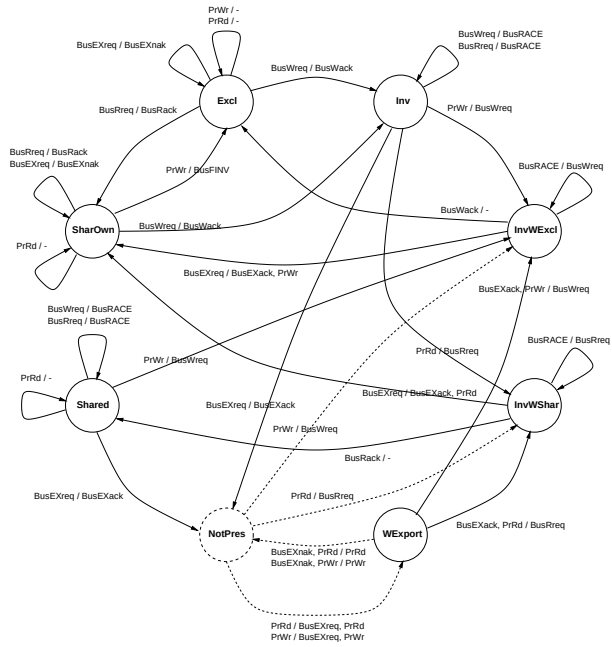


Fig.1. VSR-COMA state transition diagram. The transitions are labeled as Operation-or-Event received / Event generated.

The diagram of fig. 1 has eight different states. The eighth state is Not Present. This is not a block state. Instead, this state reflects the possibility of that the block is currently not present in the system. In this case we need to know the situation of the rest of the cache lines in our set, in order to decide what to do. If the block is not present and we have at least one block in Inv or Shared state, this block will be overwritten, changing its state to InvWShar or InvWExcl state. If the block is not present and every block in our set has the Excl or SharOwn states, we need to perform a replacement operation. Note that the selection of the destination node of the exported block does not depend on the protocol. This characteristic allows the designer to explore different replacement strategies without modifying the protocol, leading to a more flexible behavior. In the following section we will examine the replacement strategy currently used in VSR-COMA.

4 VSR-COMA Replacement Strategy

As was explained in section 3, VSR-COMA allows each cache-coherency controller to know the situation of every remote AM in the system. Note that this information is not complete, because new events can be produced at the same time the controller is checking this information, due to the race conditions inherent to the bus architecture. This information, however, can be effectively used to select the destination node for an export operation.

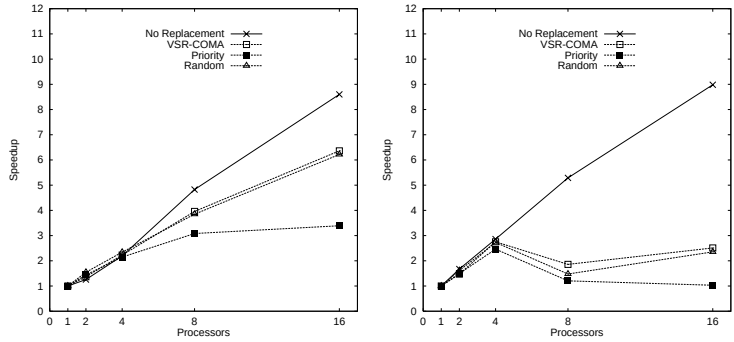
VSR-COMA uses the following selection algorithm:

1. If the block we want to export is in SharOwn state, a node with a Shared copy of this block is chosen.
2. Otherwise, we look for a node with the block we want to export in InvWExcl state (that node is currently requesting our block to the wrong owner).
3. Otherwise, we look for a node with the block we want to export in InvWShar state (that node is currently requesting a copy of our block to the wrong owner).
4. Otherwise, we look for a node with the block we want to export in Inv state (that node has been using our block in the recent past).
5. Otherwise, we look for a node with *any* block of the set in Inv state.
6. Otherwise, we look for a node with *any* block of the set in Shared state. This selection will force the destination node to discard a Shared block.
7. At this point, it seems that every block in the corresponding set of the remote AMs is in Excl, SharOwn or a transient state. This situation is possible at high memory pressures. The solution is to look for a node with any block in InvWShar state: when the replacement request arrives to that node, it is possible that the node has already completed its BusRreq and has a Shared copy that could be overwritten. If not, the node will respond with a BusEXnak event and the local node will start the node selection process from the beginning.
8. Otherwise, we look for a node with a block in InvWExcl state. The replacement request will probably be denied, but at this point there is no alternative. However, this is an extremely infrequent and transient situation: after this attempt the situation will surely change.

We would have more than one node that meet the requirements in each step. In this case, the node with less blocks in ownership in the set is chosen. This selection method leads to a better balance of the ownerships between the AMs.

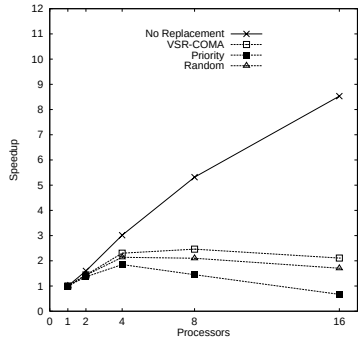
5 Results

Figure 2 shows the speedup comparison using six well-known programs of the Splash-2 benchmark suite [2]. The simulation results have been obtained considering a set of RISC workstations at 167 MHz, and a Myrinet-type interconnection network. We have considered 4-way associative AMs, 256 bytes block size and

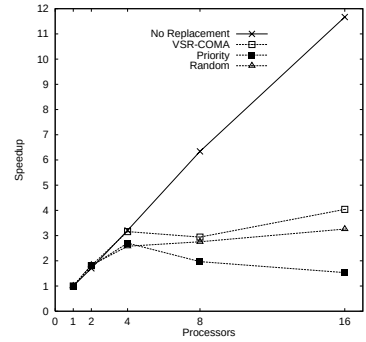


(a) FFT, 65k points

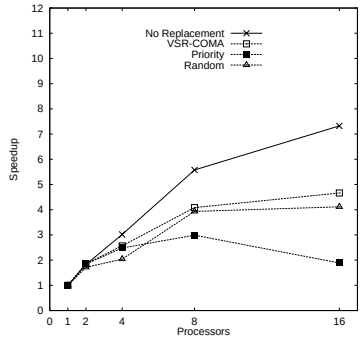
(b) LU, 256x256 matrix



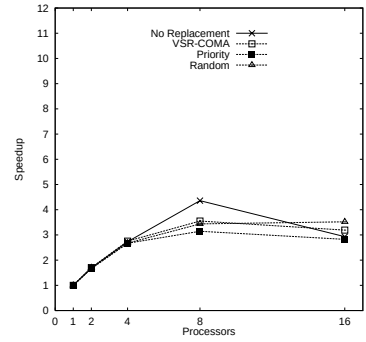
(c) Radix, 1M points



(d) Barnes-Hut, 4096 particles



(e) Ocean, 258x258 km.



(f) Radiosity, "ROOM" model

Fig.2. Speedup comparison for the three replacement strategies studied above with a no-replacement approach.

an 80 percent memory pressure. At this high memory pressure there are many remote misses, and the overhead due to replacement operations increases.

Figure 2 shows the behavior of VSR-COMA protocol using four different replacement algorithms: i) the random node selection used in COMA-F; ii) the four-level priority scheme used in DICE; iii) the selection algorithm for VSR-COMA described in section 4, and iv) the no-replacement approach proposed by COMA-BC. In this sense, COMA-BC acts as an infinite-way associative system, and therefore there is always enough space to avoid a replacement. The speedup results are heavy influenced by the use of a loosely-coupled, workstation-based architecture, but we can see that the VSR-COMA destination node selection algorithm works better than the rest of replacement algorithms at high pressures, providing a speedup over random selection that reaches 124% in Radix and a speedup over priority-based mechanisms that reaches 315%, again in Radix.

6 Conclusions

Our results confirm that VSR-COMA is a valid alternative to build COMA machines with a network of workstations that share a common bus. We have also proposed a replacement algorithm that leads to better results than other well-known replacement algorithms for this kind of systems.

The design of the VSR-COMA protocol allows the designer to explore different replacement algorithms without modifying the protocol, leading to a more flexible behavior.

References

[1] Thomas E. Anderson, David E. Culler, and David A. Patterson. A case for NOW (networks of workstations). *IEEE Micro*, pages 54–64, February 1995.

[2] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 programs: Characterization and metodological considerations. In *Proceedings of the 22nd Anual International Symposium on Computer Architecture*, pages 24–36, June 1995.

[3] Sangyeun Cho, Jinseok Kong, and Gyungho Lee. Coherence and Replacement Protocol of DICE - A Bus Based COMA Multiproccesor. *Journal of Parallel and Distributed Computing*, pages 14–32, April 1999.

[4] Fredrik Dahlgren and Josep Torrellas. Cache-only memory architectures. *IEEE Computer*, pages 72–79, June 1999.

[5] Truman Joe. *COMA-F: A Non-hierarchical Cache Only Memory Architecture*. PhD thesis, Department of Electrical Engineering, Stanford University, 1995.

[6] Diego R. Llanos Ferraris. *VSR-COMA: Un protocolo de coherencia cache con reemplazo para sistemas multicomputadores con gestión de memoria de tipo COMA*. PhD thesis, Departamento de Informática, Universidad de Valladolid, España, Abril 2000.

[7] Benjamín Sahelices Fernández, Juan Illescas, and Luis Alonso Romero. COMA-BC: A cache only memory architecture multicomputer for non-hierarchical common bus networks. In *Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Processing*, pages 502–508, 1998.