Journal of Engineering and Technology Vol. 3 No. 2, 2004 *ISSN : 1684-4114*

THE WRITE-OWNER PROTOCOL FOR ENSURING SEQUENTIAL CONSISTENCY OF DSM IN DISTRIBUTED COMPUTING SYSTEM

Al-Mukaddim Khan Pathan*

ABSTRACT

The existing protocols used to implement replicated, migrating blocks (RMB) strategy for ensuring the sequential consistency in distributed shared memory (DSM), suffers from excessive cost. In this paper, an efficient protocol named as the Write-Owner Protocol has been proposed using the binding agent for RMB strategy. The goal is to minimize the cost as well as to increase the performance. The proposed protocol uses a hash table to maintain the modification as well as location and owner information of the data blocks with the version number(s). Though some overhead is associated with it, other factors tend to dominate and this small overhead is usually negligible.

Keywords: replicated migrating block, distributed shared memory (DSM), sequential consistency model, hash table, interprocess communication.

1. INTRODUCTION

The use of shared memory paradigm for interprocess communication [Sinha 2001] is natural for distributed processes running on tightly coupled shared-memory multiprocessors. However, for loosely coupled distributed memory systems, no physically shared memory is available to support the shared memory paradigm for interprocess communication. But some recent loosely coupled distributed-memory systems have implemented a software layer on top of the message passing communication system to provide a shared-memory abstraction to the programmers. The term Distributed Shared Memory (DSM) refers to the shared-memory paradigm applied to loosely coupled distributed memory systems [Fitzgerald et al. 1986].

The DSM [Nitzberg and Virginia Lo 1991] is basically an abstraction that integrates the local memories of different machines in a network environment into a single logical entity shared by cooperating processes executing on multiple sites. The shared memory itself exists virtually, where each node in the system consists of one or more CPUs and a memory unit. The nodes are connected by high speed communication network. A simple message passing system is used to exchange messages with each other. A software memory mapping manager routine in each node maps the local memory onto the shared virtual memory. To facilitate the mapping operation, the shared memory space is partitioned into *blocks*. Data blocks can be migrated on demand from one node to another, when a block fault occurs.

A consistency model [Tanenbaum 1995] for the distributed shared memory (DSM) system basically refers to the degree of consistency that has to be maintained for the shared-

* *CIT Dept. Islamic University of Technology, Gazipur-1704, Bangladesh.*

memory data, for the memory to work correctly for a certain set of applications. There are several consistency models for the DSM [Mosberger 1993]. Among them the most commonly used model in DSM systems is the sequential consistency model. It can be implemented; it supports the most intuitively expected semantics for memory coherence, and does not impose any extra burden on programmer Protocols for implementing the sequential consistency model in a DSM system depend to a great extent on whether the DSM system allows replication and/or migration of shared data block

Among the replication and migration strategies, the replicated, migrating blocks (RMB) strategy uses the write invalidate and the write update protocols for ensuring sequential consistency model. But they are not the efficient one. To overcome this, an efficient protocol named as the Write-Owner Protocol has been proposed using the binding agent for RMB strategy with a view to minimize the cost and increase performance. Some of the existing protocols are discussed here to make our assumption clearer:

1.1. The write invalidate protocol [Sinha 2001]

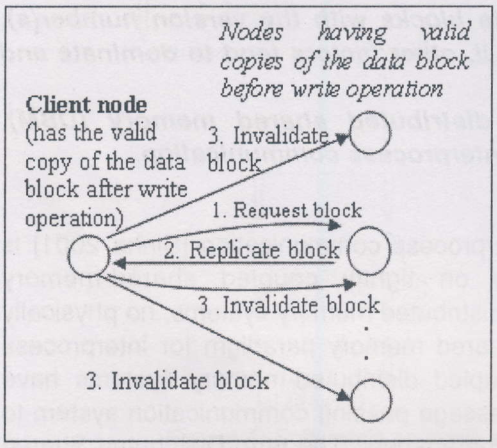
Nodes having valid copies of the data block before write operation
Client node (has the valid copy of the data block after write operation)
3. Invalidate block
1. Request block
2. Replicate block
3 .Invalidate block
3. Invalidate block

Figure 1: Write-invalidate memory coherence approach for replicated, migrating blocks (RMB) strategy [Sinha 2001]

In this scheme, all copies of a piece of data except one are invalidate before a write is performed on it. Therefore, when a write fault occurs at a node, a fault handler copies the accessed block from one of the block's current nodes to its own node, invalidates all other copies of the block by sending an invalidate message containing the block address to the nodes having a copy of the block, changes the access of the local copy of the block to write, and returns to the faulting instructions as shown in Figure 1:

After returning, the node owns that block and can proceed with the write operation and other read/write operations until the block ownership is relinquished to some other node. If one of the nodes that has a copy of the clock before invalidation tries to perform a memory access operation (read/write) on the block after invalidation, a cache miss will occur and the fault handler of that node will have to fetch the block again from a node having a valid copy of the block. Thus, this scheme achieves sequential consistency. In the basic implementation of this protocol a status flag is associated with each block to indicate whether the block is valid, whether it is shared, and whether it is read-only or writable. But the main difficulty in this scheme is to know the location of the current owner of the data block at a particular time.

1.2. The Write-update protocol [Sinha 2001]

In this scheme, a write operation is carried out by updating all copies if the data on which the write is performed. Therefore, when a write fault occurs at a node, the fault handler copies the accessed block from one of the block's current nodes to its own node, updates

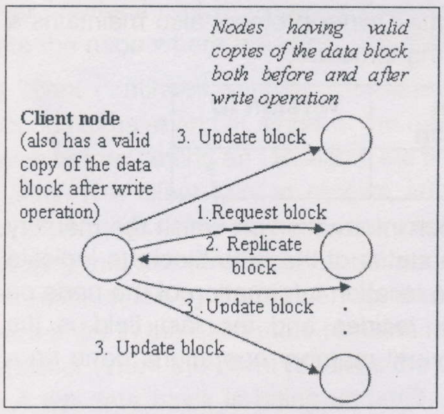
Nodes having valid copies of the data block both before and after write operation
Client node (also has a valid copy of the data block after write operation)
3. Update block
1. Request block
2. Replicate block
3. Update block
3. Update block

Figure 2: Write-update memory coherence approach for replicated, migrating blocks (RMB) strategy [Sinha 2001].

all copies of the block by performing the write operation on the local copy of the block and sending the address of the modified memory location and its new value to the nodes having a copy of the block, and then returns to the faulting instruction as shown in Figure 2:

The write operation completes only after all the copies of the block has been successfully updated. In this method, sequential consistency can be achieved by using a mechanism to totally order the write operations of all the nodes so that all processes agree on the order of writes.

One method to do this is to use a global sequencer to sequence the write operations of all nodes. In this method, the intended modification of each write operation if first sent to the global sequencer. The sequencer assigns the next sequence number to the modification and multicasts the modification with this sequence number to all the nodes where a replica of the data block to be modified is located. This scheme is depicted in Figure 3:

Figure 3: Global sequencing mechanism to sequence the write operations of all nodes [Sinha 2001].

The write operations are processed at each node in sequence number order. The main drawback of the write-update approach is the excessive cost for use with loosely coupled distributed-memory systems because it requires a network access on every write operation and updates are only propagated when data are read, and several updates can take place before communication is necessary.

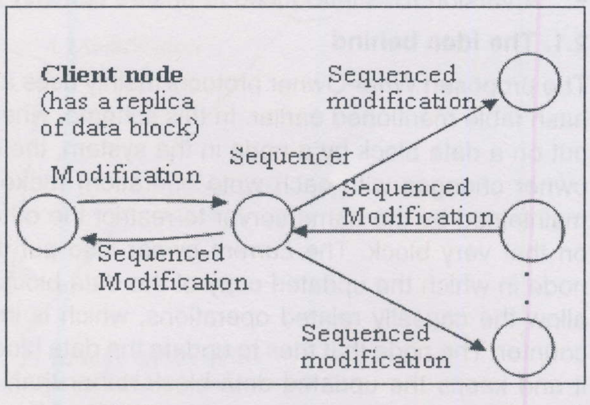

Figure 3: Write-update memory coherence approach for replicated, migrating blocks (RMB) strategy [Sinha 2001].

2. THE PROPOSED WRITE-OWNER PROTOCOL

The proposed Write-Owner protocol is mainly based on the binding agent [Tanenbaum 1995] method used for server locating. So, the proposed protocol has some features which are same as the binding agent. These features are:

- The binding agent is basically a name server used to bind a client to the server by providing the client the location information of the desired server. In this proposed Write-Owner protocol, actually, this name server is used to locate the particular node on which the latest update block resides.

- Like the binding agent method the proposed Write-Owner protocol also maintains a hash table in the name server, having the following structure:

Data Block	Status	Location Information	Version ID

The first field of the hash table contains the data block information on which the memory operations are done. The second field indicates the status of the data block, to indicate whether it is modified by a node, the third field is the location information of the node on which the latest updated copy of the data block resides and the last field is the identification information that is used to identify several memory operations done on a particular data block by other node(s).

The additional features of the proposed Write-Owner protocol can be pointed out as follows:

- The owner node is responsible to supply the data block to be updated, to the client.
- The node that is modifying the block becomes the owner of the data block after modification is completed.
- Only one replication of the data block is needed for each write operation.
- A version ID is introduced to ensure causally related operations.

2.1. The idea behind

The proposed Write-Owner protocol mainly uses a centralized name server containing the hash table mentioned earlier. In this scheme, whenever, a write operation is to be carried out on a data block by a node in the system, the current owner of the data block (as the owner changes with each write operation) makes an entry in the hash table, which is maintained by the name server to restrict the other nodes to perform memory operation on that very block. The current owner also put the location information (handle) of the node in which the updated copy of the data block will be found and a version number to allow the causally related operations, which is implemented using a global systemwide counter. The node that tries to update the data block makes a replica of the block, updates it and keeps the updated data block rather than returning it to the previous owner and becomes the owner of the data block.

Whenever a client wants the data block to read, a timer of 150s is set and it contacts to the name server with the information about the data block that it wants to access. The name server has a fixed address which is known to all nodes. The name server checks in the hash table to see whether the data block for which the client has made request, is currently updating or not. If the status flag of the data block in the hash table is set (that is, the data block is being updated by anyone of the nodes), the name server simply returns a reply message before the timer expires; to the client containing the handle to locate the node that contains the updating data block and the status of the data block. It also restricts the client to access the data block during update. This is done by checking the status field in the reply message. If the data block is not being updated, the name server returns a reply message containing the status of the data block and the handle to locate the node where the latest updated data block resides.

The client continues sending messages periodically (15s) to the name server until it succeeds to have an OK signal in the reply message to access the data block. If the timer expires before getting an OK signal the timer is reset and the client continues with sending requests. If a client fails to receive an OK signal from the name server after 5 timer expiration, it gives up and an error message is sent to the client by the name server indicating that the data block is not accessible. At the same time the entry in the hash table (maintained by the server) for that particular data block, is removed. Thus a data block's entry in the hash table remains for 150s. By this way the access time can also be optimized. When the client receives the OK signal, it also gets the handle of the node by which the data block is being updated. Having received a green signal from the name server the client directly sends request to the node that contains the latest updated data block. In turn, the current owner of the data block returns the desired block to the client. The whole method is depicted in Figure 4.

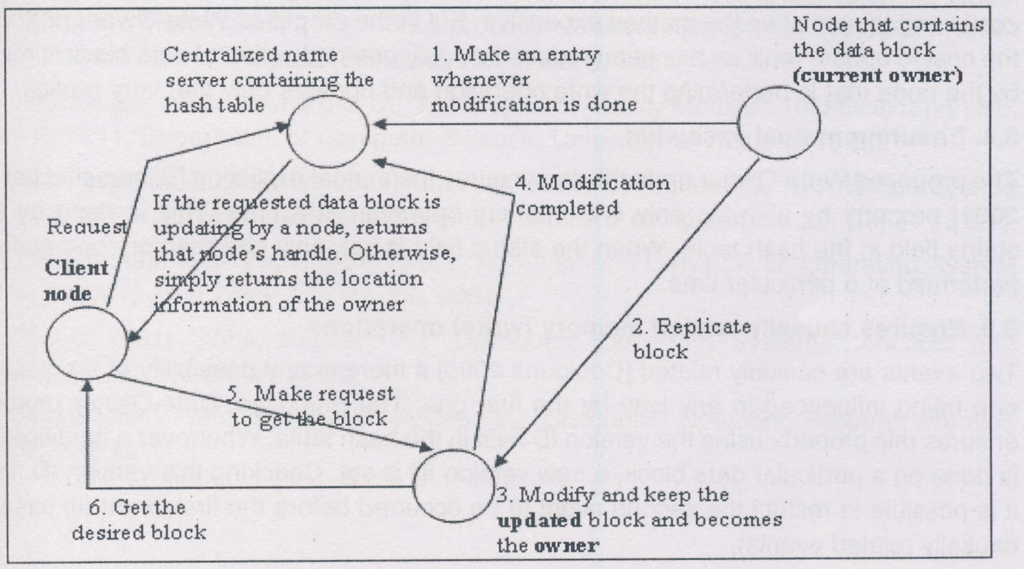
Node that contains the data block (current owner)
Centralized name server containing the hash table
1. Make an entry whenever modification is done
4. Modification completed
If the requested data block is updating by a node, returns that node's handle. Otherwise, simply returns the location information of the owner
Request
Client node
2. Replicate block
5. Make request to get the block
6. Get the desired block
3. Modify and keep the updated block and becomes the owner

Figure 4: The pictorial representation of the proposed Write-Owner protocol.

3. ADVANTAGES OF THE PROPOSED WRITE-OWNER PROTOCOL

The proposed Write-Owner protocol is an efficient one that has a number of advantages. Some of the advantages are:

3.1. Reduces overhead for message passing

In the Write-Invalidate protocol, when a client wants to make a write operation on a data block, it has to explicitly send a message to all the other nodes in the system to indicate the block as invalid for any memory operation [Sinha 2001]. But in the proposed protocol, we are not facing this problem. There is no need to send messages to the nodes to restrict access. The name server sends in the reply message, the status of the data block. If it is currently updating, by simply checking this field the client restrict itself to access the block. Thus the overhead associated with message passing to all the nodes like the write invalidate protocol is reduced.

3.2. Need for data block replication is reduced

In the proposed Write-Owner protocol, unlike the write-update protocol, there is no need to make a number of replication of the data block. When the write operation is performed on this block, only then replication of the data block is made, on that very node that has made the request to perform write operation. Thus, the cost for replication is reduced in the proposed protocol.

3.3. Reduces cost for replica update

In the existing Write-Update protocol, a write operation is carried out by updating all copies of the data on which the write is performed. It is known that, replica update is very costly and thus, makes the method expensive. But in the proposed Write-Owner protocol, the cost to update replicas has been reduced as only one replication of data block is made by the node that is performing the write operation and updates only that very replica.

3.4. Ensuring mutual exclusion

The proposed Write-Owner protocol also ensures the mutual exclusion [Silberschatz et al. 2001] property by allowing only one memory operation at a time. This is done by the status field in the hash table. When the status field is set, only one memory operation is performed at a particular time.

3.5. Ensures causally-related memory (write) operations

Two events are causally related [Coulouris 2000] if there is any possibility of the second one being influenced in any way by the first one. The proposed Write-Owner protocol ensures this property using the version ID field in the hash table. Whenever a modification is done on a particular data block, a new version ID is set. Checking this version ID field, it is possible to restrict the second event to be occurred before the first event (in case of causally related events).

3.6. Ease of locating the owner of a data block

It is necessary to reduce the consistency-related communication in DSM [Carter et al. 1994]. So, the process of locating the owner of a data block should not impose extra burden to the system. It is difficult to locate the owner of the data block in the write-invalidate method as the owner of the block is changing with each write operation. The proposed Write-Owner protocol has overcome this limitation. Though, in the proposed protocol also, the owner of the data block can be changed, it is not a problem to locate the owner, as in the hash table in the name server; the location information of the current owner is maintained.

4. CONCLUSION

From the above discussion, it is evident that the proposed Write-Owner protocol is expected be a superior one than the existing protocols as it overcomes almost all the limitations of these existing protocols as well as provides some other advantages. One can treat this centralized approach as a candidate for single point of failure. But it is possible to overcome it by distributing the name server function among several name servers and replicating information among them. Though some overhead is associated, other factors tend to dominate and this small overhead is usually negligible.

5. REFERENCES

[1] [Carter et al. 1994] Carter, J. B., Benette, J. K., and Zwaenepoel, W., "Techniques for Reducing Consistency-Related Communication in Distributed Shared Memory Systems," ACM Transactions on Computer Systems, Vol. 12 (1994).

[2] [Coulouris 2000] Coulouris, George "Distributed Systems Concepts and Design", Pearson Education Asia, 2000.

[3] [Fitzgerald et al. 1986] Fitzgerald R. and Rashid R.F; "The Iintegration of Virtual Memory Management and Interprocess Communication in Accent", ACM Transactions on Computer Systems, Vol. 4, No. 2, 1986

[4] [Mosberger 1993] Mosberger, D., "Memory Consistency Models", Technical report No. TR 93/11, Department of Computer Science, University of Arizona (1993).

[5] [Nitzberg and Virginia Lo 1991] Nitzberg, N., and Virginia Lo, "Distributed Shared Memory: A Survey of Issues and Algorithms", IEEE computer, Vol.24, No. 11 (1991).

[6] [Silberschatz et al. 2001] Silberschatz, Abraham and Galvin, P. B;"Operating System concepts", John Wiley & Sons, Inc, 2001.

[7] [Sinha 2001] Sinha, Pardeep K.; "Distributed Operating Systems-Concepts and Design", Prentice Hall, 2001.

[8] [Tanenbaum 1995] Tanenbaum, Andrew S.; "Distributed Operating System", Prentice Hall, 1995.

(Contd. from 2nd cover page)

8. SI units must be used in the mauscript. However, other units may be used in parenthesis.
9. Tables should be referred to in consecutive Arabic numerical. Each table must have a table caption.
10. Line drawings must be in a form suitable for reproduction e.g., laser print, drawn in indian ink on white paper or on tracing paper. Photographs should have a glossy finish. Each figure must have a number and a figure caption. Electronic mode is prefered.
11. References should be set out in alphabetical order of the author's last name in a list at the end of the article. They should be given in standard form as in the following examples:

(a) Journal
Bloomer G. and Write A., "Scheduling of Vehicles from Factory to Depot." Operations Research, Vol. 12, January, pp. 590-598, 1984.

(b) Book
Best, John., and Kahn, James V., Research in Education, Prentice-Hall, New Jersey, 1986.

(c) Monograph
Syedali, M.M. "Computer Controlled Car", Thesis, M.Sc. Engineering, Department of Mechanical Engineering, BUET, 1990.

(d) Conference paper
Hasan, M. and Ullah, M.S. "Tourism development in the Chittagong Hill Tracts of Bangladesh after the peace agreement of 1997", a paper submitted to the Regional Conference on physical mobility and development in the mountains, at Tribuvan University, 15-17, March, 2000 Kathmandu, Nepal, pp. 12.

(e) Unpublished paper
Ahmadi, R and Tang : Production Allocation with Dual Provisioning. Working Paper, Anderson Graduate School of Management, UCLA (1991)

12. The University does not accept responsibility for loss or damage of manuscript while in mail.
13. The responsibility for opinoins in the contributions rests entirely on their authors.
14. The author(s) must submit declaration that the paper was not published elsewhere.
15. In case of joint papers, communication will be made with the first author.
16. The University will reserve the copyright of the paper once it is accepted for publication in the journal. The authors must obtain written permission from REASP, IUT for publication elsewhere.

Procedure for acceptance of papers and publications :

1. Papers submitted for publication will be referred to the listed reviewers for assessment. However, the Editorial Board will make initial screening of the papers.
2. After the assessment, the authors may be requested to modify/clarify certain points.
3. Accepted (modified/corrected) papers will be published in the next issue of the journal.