ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

A Survey of Software based Distributed Shared Memory (DSM) implementation methodologies for Multiprocessor Environments

Hemant D. Vasava[1], Jagdish M. Rathod[2]

Assistant Professor, Dept. of CP, BVM Engineering College, V.V.Nagar, Gujarat, India[1]

Associate Professor, Dept. of Electronics, BVM Engineering College, V.V.Nagar,Gujarat, India[2]

Abstract: It is more important to optimize the distributed system features to obtain the maximum possible performance. A distributed shared memory (DSM) of distributed system is kind of mechanism that allowing system's multiple processors to access shared data without using interposes communication(IPC). A DSM is a simple yet powerful paradigm for structuring multiprocessor systems. It can be designed using hardware and/or software methodologies based on various considerations of data being shared in multiprocessor environments. It is better to design DSM in software because sharing data becomes a problem which has to be easily tackled in software and not in hardware as in multiprocessor systems. This paper discussed various design methodologies based on granularity of data being shared like object based, shared variable based and page based with its features and drawbacks. This study also describes DSM requirements, advantages of software based DSM design over hardware based design, Semantics of concurrent access and replication, DSM algorithms and various issues of DSM with respect to underlying parameters of multiprocessor environments.

Keywords: Interprocess Communication (IPC), Distributed Shared Memory (DSM), Distributed Global Address Space (DGAS), Very Large Scale Integration (VLSI)

I. INTRODUCTION

A distributed shared memory (DSM) is a kind of mechanism that allowing user's multiple processors to access shared data without using interprocess communications. To construct such a multiprocessor systems, the underlying hardware and/or software must move data items (memory contents) among processors in a way that provides the illusion of a globally shared address space. So, distributed shared memory also known as a distributed global address space (DGAS). As DSM system involves moving of data from on node to another node which is in traditional generated networks, so bandwidth and performance are the important criteria for design.

Recent developments in Very Large Scale Integration (VLSI) technologies create the possibility of multiple processors in a single chip. So, it is required to share a number of resources, especially off-chip resources, which creates new constraints in the design process [1]. These evolutions of VLSI technologies now it is possible to integrate whole systems with DSM on a single chip and new constraints have to be taken into account in this environment: quality of service, bandwidth, expansion, security, access latency, power consumption and memory, performance etc [1]. Distributed system have evolved using message passing as their main method of communication in loosely coupled system and tightly coupled system uses shared memory between processor.

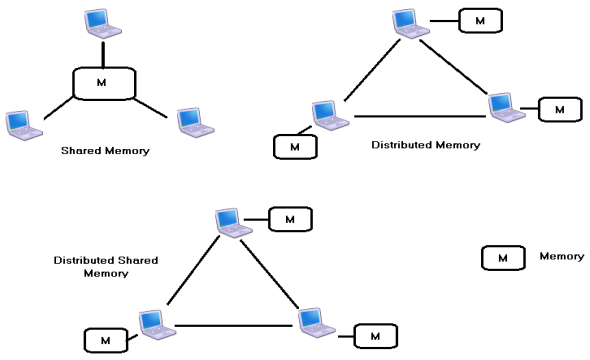

Figure 1: Shared Memory Organizations

Here, figure 1 shows memory organizations for distribute systems which having various advantages and drawbacks. Distributed shared memory (DSM) systems processes share data transparently across node boundaries, data faulting,

location, and movement is handled by the underlying system. So, DSM system represents a successful combination of two parallel computer classes i.e. shared memory and distributed memory. It provides the shared memory abstraction in system with physically distributed memories and consequently combine the advantages of both approaches [2].

Distributed shared memory can be implemented two ways. One with using hardware partitioning that also uses software components. Second is a software DSM system can be split into three classes: page-based, variable-based, object-based. The various DSM implementation are as given as below [3].

Object-based – It requires special programming language features to determine when a remote machine's memory is to be accessed.
Shared Variable-based – It requires custom compilers to add special instructions to program code in order to detect remote access.
Page-based – It uses the memory management unit (MMU) to trap remote access attempts.

II. NEED OF DSM

Sharing of data is essential requirement of any distributed system. Also, current advancement in processor design user still demand more performance with parallel processors so the processor design needs to be upgraded to get more performance with using shared memory. Eventually, single processor technologies must give way to multiple processors parallel computers like it is less expensive to run 10 inexpensive processors cooperatively than it is to buy a new computer 10 times as fast [4]. For this kind of design to reduce various overhead it is require designing distributed shared memory which can be access by all the parallel processors. Solutions to such distribute and parallel system communication is distributed shared memory (DSM). So, to optimize imbalance between processors and memory its is require to investigate new architectural design which should be easy to expandable. In future whenever it is required to change the system design or to add some new future then is should be expandable.

III. WHY SOFTWARE BASED DSM?

There are various advantages of programming distributed shared memory for multiprocessor environment as stated below:

- Sharing data becomes a problem which has to be tackled in the software and not in hardware as in multi-processor systems.
- Shared memory programs are usually shorter and easier to understand.
- Large or complex data structures may easily be communicated.
- Programming with shared memory is a well-understood problem.
- Shared memory gives transparent process-to-process communication.
- Compact design and easy implementation and expansion.

IV. DSM DESIGN ISSUES

The distributed shared memory to present the global view of the entire address space to a program executing on any machine [4]. A DSM manager on a particular machine would capture all the remote data accesses made by any process running on that machine. A implementation of a DSM would involve various choices. Some of them are as below[5].

- DSM Algorithm
- Implementation level of DSM Mechanism
- Semantics for concurrent access
- Semantics (Replication/Partial/ Full/R/W)
- Naming scheme has to be used to access remote data
- Locations for replication (for optimization)
- System consistency model & Granularity of data
- Data is replicated or cached
- Remote access by HW or SW
- Caching/replication controlled by HW or SW

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

- DSM controlled by memory management SW, OS, language run-time system.

A. DSM Algorithm

An algorithm of implementing DSM deal with two basic problems [6]. First, static and dynamic distribution of shared data across the system to minimize the latency. Second is to preserve a coherence view of shared data while minimizing coherence management overhead. Two frequently used strategies for distributing shared data are replication and migration [6]. Replication allows multiple copies of the same data item to reside in different local memories. Migration implies that only a single copy of a data item exists at any one time. So, to decrease coherence management overhead, users prefer this strategy when sequential patterns of write sharing are prevalent.

B. Naming Scheme [7]

When a processor wants to access remote data it has to know on which machine does the data reside and fetch it from there. So, all shared data are visible to all the machines, it has to be a unique naming mechanism to avoid any conflicts. The possible solution is to have a logical global address space. The VM manager at each node performs the translation of the logical address to get the location on a remote machine. But such an approach would not be useful if the granularity of shared data is less than a page. Here, the calling process will have to possess explicit knowledge of the remote location of the data which it wants to access.

C. Choice of Consistency Model

It is a very important parameter in a design of DSM. The possible replications of shared data across multiple nodes mean that different machines have different copies of the data. Thus, the DSM has to maintain consistency in the state of these copies of the same data[7]. This problem is similar to cache coherency in multiprocessors.

D. Granularity

Granularity is used to refers the size of a data unit exists in the distributed shared memory. This is an important decision which essentially governs the implementation of a DSM. The immediate successor of shared memory from multiprocessor would have a page as the unit for data transfer. But it has its own Disadvantages [7].

E. Remote Access by HW or SW

Efficient sharing of memory in a distributed system has the promise of greatly improving the performance and cost-effectiveness of the system when running large memory intensive jobs. A point of interest is the hardware support required for good memory sharing performance. We evaluate the performance of two models: the software-only model that runs on a traditional distributed system configuration, and requires support from the operating system to access remote memory; and the hardware-intensive model that uses a specialized network interface to extend the memory system to allow direct access to remote memory.

F. Semantics for Concurrent Access

As it pertains to storage, it is the ability to access a particular data from either two different paths towards memory simultaneously (load balanced), or by two separate hosts distributed environment (distributed lock management). The problem to overcome is when two hosts try to "change" the data at the same time. This is where a lock manager is needed to allow cooperation between the hosts for data access. So, appropriate semantics about concurrent access is require to implement for managing concurrent access.

V. DSM DESIGN METHODOLOGIES

To design DSM useful to multiple processors in a distributed system, we chose to allow multiple readers access to the same region of shared memory. This improves performance and saves space by allowing multiple processes on possibly different nodes to work together in the same shared address space [4]. Based on the granularity of data DSM is divided in to three categories viz. object based, shared variable based and page based.

A. *Object based DSM*

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

An object is a programmer defined encapsulated data structure. It consists of internal data, object state and procedures. The procedures are called methods or operations that operate on the object state. To access or operate on the internal state, the program must invoke one of the methods. The method can change the internal state, return the state, or something else. It also supports information hiding. Each object state having auxiliary variables which uses various methods. Figure 2 below shows the concept of object based DSM [7].

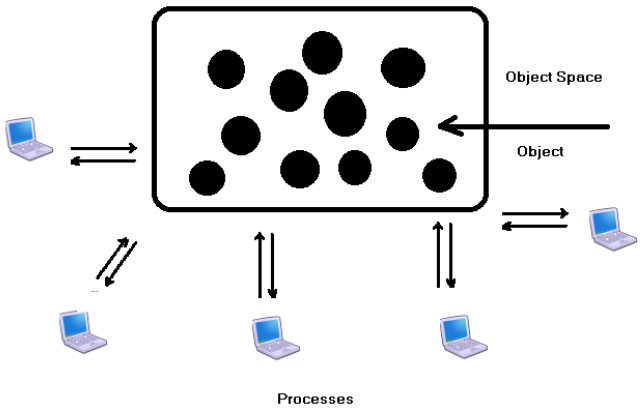

Figure 2: Object Based DSM

The object based shared memory is designed as a collection of separate objects instead of as a linear address space, there are many other choices to be made. If replication is not used, all accesses to an object go through the one and only copy, which is simple, but may lead to poor performance. Allowing objects to migrate from machine to machine, as needed, it may be possible to reduce the performance loss by moving objects to where they are needed [8].

Object based DSM has three advantages over the other methods: more modular, more flexible and Synchronization and access can be integrated together cleanly. Object-based DSM also has disadvantages [8]. For one since all accesses to shared objects must be done by invoking the objects, methods, extra overhead is incurred that is not present with shared pages that can be accessed directly. It is managed by language runtime system. Second disadvantage, it cannot be used to run old "dusty deck" multiprocessor programs that assume the existence of a shared linear address space that every process can read and write at random.

B. Shared Variable based DSM

Shared variable based DSM is to share only certain variables and data structures that are needed by more than one process in to the environment. This way, the problem changes from how to do paging over the network to how to maintain a potentially replicated, distributed data base consisting of the shared variables. Using shared variables that are individually managed also provides considerable opportunity to eliminate false sharing. If it is possible to update one variable without affecting other variables, then the physical layout of the variables on the pages is less important. The most important example of such system is Munin [8].

A DSM using shared variable is to let the applications decide as to which variables are to be shared and the DSM manager will maintain a database of the shared variables in the distributed environment. This concept expects the programmer to explicitly declare which variables are to be shared globally in the program [8]. This removes the problem of false sharing. Any process on a different machine can access this variable by requesting through that machine's DSM manager.

C. Page based DSM

In this approach a memory page as the unit of data sharing. Page based DSM closely emulate the shared memory in multiprocessor that is managed by operating system. The entire address space is divided into pages (Chunks). Whenever the virtual memory manager finds a request to an address space which is not local, it asks the DSM manager to fetch that page from the remote machine [8]. Such kind of page fault handling is simple and similar to what is done for local page faults. The basic system that can improve performance considerably is to replicate chunks that are read

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 2, Issue 7, July 2013

only, read-only constants, or other read-only data structures. Another possibility is to replicate not only read-only chunks, but all chunks. The inconsistency is prevented by using some consistency protocols [8].

In this design the owner finding is by doing a broadcast, asking for the owner of the specified page to respond. The owner can then send a single message transferring ownership and the page is well, if needed. Broadcasting has the disadvantage of interrupting each processor [8]. To remove this drawback one of these process is designated as the page manager. It is the job of the manager to keep track of who owns each page. A problem with this protocol is the potentially heavy load which can be reduced by having multiple page managers instead of just one. Another possible algorithm is having each process keep track of the probable owner of each page. Requests for ownership are sent to the probable owner, which forwards them if ownership has changed. If ownership has changed several times, the request message will also have to be forwarded several times[8]. At the start of execution and every *n* times ownership changes, the location of the new owner should be broadcast, to allow all processors to update their tables of probable owners.

Another important task is how all the copies are found when they must be invalidated. The first is to broadcast a message giving the page number and ask all processors holding the page to invalidate it [8]. It works only if broadcast messages are totally reliable and can never be lost. The second possibility is to have the owner or page manager maintain a list or copyset telling which processors hold which pages. When a page must be invalidated, the old owner, new owner, or page manager sends a message to each processor holding the page and waits for an acknowledgment. When each message has been acknowledged, the invalidation is complete.

The page replacement in a DSM system, as in any system using virtual memory, it can happen that a page is needed but that there is no free page frame in memory to hold it. When this situation occurs, a page must be evicted from memory to make room for the needed page. Two sub problems immediately arise [8]: which page to evict and where to put it. The choice of which page to evict can be made using traditional virtual memory algorithms, such as some approximation to the least recently used algorithm. The second best choice is a replicated page that the evicting process owns. It is sufficient to pass ownership to one of the other copies but informing that process, the page manager, or both, depending on the implementation. The page itself need not be transferred, which results in a smaller message [8].

VI. DSM COMPARISONS

Here, table 1 given below shows the comparison of these three types of software based DSM design against various parameters.

Item	Object Based	Shared Variable based	Page Based
Transfer Medium	Network	Network	Network
Transfer Unit	Object	Variable	Page
Data Migration done by	Software	Software	Software
Linear, Shared Virtual Address Space?	No	No	Yes
Possible Operations	General	R/W	R/W
Encapsulation and Methods	Yes	No	No
Is Remote Access Possible in hardware?	No	No	No
Is unattached memory possible?	No	No	No
Who converts Remote Memory Access to message?	Runtime System	Runtime System	OS

Table 1: Software based DSM Comparisons

VII. CONCLUSIONS

A software based DSM provide many advantages in design of multiprocessor systems. A distributed shared memory mechanism allowing user's multiple processors to access shared data efficiently. As DSM system increase the bandwidth and performance are the important criteria for design. A DSM various implementation helps to design various kinds of system using hardware and/or software approaches for multiprocessor environments as logically shared, local physically distributed, paged-based, shared variable based and object based architectures. A DSM having no memory access bottleneck and large virtual memory space can accommodate more no of processors. Its programs are portable as they use common DSM programming interface, But still having some disadvantages like programmers need to understand consistency models, to write correct programs. So, this study is very useful to build new shared memory system against designing constraints and other languages.

ACKNOWLEDGMENT

We are very thankful to our principal Dr. F.S. Umrigar and Prof. P.B. Swadas, Head Computer Engineering Department, BVM Engineering College, V.V.Nagar for encouraging us to write this review paper.

REFERENCES

[1] Carlos Macian, Sarang Dharmapurikar and John Lockwood, "Beyond Performance: Secure and Fair Memory Management for Multiple System on a Chip".
[2] Prof. Bal Gopal, Rizwan Beg and Pankaj Kumar, "Memory Management Techniques for Paging on Distributed Shared Memory Framework," *International Journal of Computer Science and Information Technology*, Volume 2, April 2010.
[3] Tomas Seidmann, "Distributed Shared Memory in Modern Operating Systems," *Ph.D. thesis*, Slovak University of Technology, pp.812-19, Bratislava.
[4] Song Li, Yu Lin, and Michael Walker, "Region-based Software Distributed Shared Memory," CS 656 Operating Systems, May 5, 2000.
[5] Ajay Kshemkalyani and Mukesh Singhal, *Ch12: Distributed Computing: Principles, Algorithms, and Systems*, Cambridge University Press, CUP 2008.
[6] Jelica Protic, Milo Tomasevic and Viljko Milutinovic, " Distributed Shared Memory: Concepts and Systems," IEEE Parallel and Distributed Technology, Summer 1996.
[7] Varun Chandola, "Design Issues in Implementation of Distributed Shared Memory in User Space".
[8] Distributed Shared Memory, Available at: *http://www.cs.gmu.edu/cne/modules/dsm/*
[9] Milind R. Penurkar and Rekha S. Sugandhi, "OPDSM:A Combinational Page based and Object based DSM Model," *International Journal of Computer Applications (0975 – 8887)*,Volume 1 – No. 6,2010.
[10] Masaaki KONDO and Motonobu FUJITA, "Software-Controlled On-Chip Memory for High-Performance and Low-Power Computing," *ACM Computer Architecture News*, Vol.30, No.3, pp.7–8, June 2002
[11] Bil Nitzburg and Virginia Lo, " Distributed shared memory: A survey of issues and algorithms," University of Oregon.
[12] Lionardo Dagum and Ramesh Menon, " Open MP: An industrial standard API for shared memory programming," *IEEE computational science and engineering 1998*.
[13] Pradeep K. Sinha, *Distributed Operating Systems: Concepts and Design*, pp.1-90, PHI Learning (2009).
[14] Andrew S. Tanenbaum and Maarten Van Steen, *Distributed Systems: Principles and Paradigms*, PHI Learning (2011).
[15] Paul Krzyzanowski, *Lectures on distributed system: taxonomy of distributed system*, Rutgers University – CS 417: Distributed Systems ©2000-2003.
[16] Abdel fatah Aref Yahya and Rana Mohamad Idrees Bader , "Distributed Shared Memory Consistency Object based Model," *Journal of Computer Science* 3 (1): 57-61, 2007,ISSN1549-3636.
[17] Lionardo Dagum and Ramesh Menon, " Open MP: An industrial standard API for shared memory programming," *IEEE computational science and engineering 1998*.
[18] Distributed Shared Memory System, D.S.M., Available at*: http://www.ens-lyon.fr/LIP/RESO/Software/Dosmos /DSM.html*
[19] M. Stumm and S. Zhou, " Algorithms implementing distributed shared memory," *Computer* Vol 23, No 5, pp. 54-654,May 1990.
[20] www.ibm.com
Available at: *http://publib.boulder.ibm.com/infocenter/txformp/v6r0m0/index.jsp?topic=%2Fcom.ibm.cics.te.doc%2Ferziaz0015.htm*