DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4406

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

STUDY OF MEMORY ORGANIZATION AND MULTIPROCESSOR SYSTEM - USING THE CONCEPT OF DISTRIBUTED SHARED MEMORY, MEMORY CONSISTENCY MODEL AND SOFTWARE BASED DSM

Dhara Kumari
Mphil Scholar
Himalayan University
Arunachal Pradesh (India)
dharaioc@gmail.com

Dr. Rajni Sharma
Assistant Professor (Computer Science)
PT.J.L.N. Govt P.G College
Faridabad (India)
crownrajni@yahoo.com

Dr. Sarita Kaushik
HOD (Computer Science)
DAV College
Faridabad (India)
sarita_kaushik24@rediffmail.com

***Abstract*:** In current trend, performance and efficiency is the big issue of the memory organization and multiprocessor system whereas, A Memory Organization and Multiprocessor uses multiple modalities to capture different types of DSM (Software based, Hardware Based and it may be combine both Software & Hardware etc) because IT technology is greatly advanced and lot's of information is shared via the internet. To improve the performance and efficiency of that multiprocessor system and memory organization, we can use different type of techniques that is based on the concept and implementation of Hardware, Software, and Hybrid DSM. This paper provides an almost exhaustive survey of the existing problem and solutions in a uniform manner, presenting their memory organization, shared memory, distributed memory, distributed shared memory, Memory Consistency Model and software based DSM mechanisms and issues of importance for various DSM systems and approaches.

***Keywords*:** Performance, Efficiency, Memory, DSM, Shared Memory, Software Based DSM, Multiprocessor System. Memory Consistency Model

I INTRODUCTION

Traditionally, a major progress was recently made in the research and development of systems with multiple processors that are capable of delivering high computing power satisfying the constantly increasing demands of typical applications. It is more important to optimize the distributed system features to obtain the maximum possible performance and efficiency. Systems with multiple processors are usually classified into different groups, according to their memory system organization, shared memory, distributed memory, distributed shared memory, Memory Consistency Model and software based DSM.

A. Memory Organization

In recent years, power-efficiency has become a major design factor in systems. This trend is fueled by the ever-growing use of battery-powered hand-held devices on the one end, and large-scale data centers on the other end. To ensure high power-efficiency, all the resources in system (e.g., processor, caches, memory) must be used efficiently. So memory is internal storage areas in the computer system. The term memory identifies data storage that comes in the form of chips, and the word storage is used for memory that exists on tapes or disks. Moreover, the term memory is usually used as shorthand for physical memory, which refers to the actual chips capable of holding data. Some computers also use virtual memory, which expands physical memory onto a hard disk. Developments in technology and economies of scale have made possible so-called Very Large Memory (VLM) computers. [1] Every computer comes with a certain amount of physical memory, used with reference to computers generally refers to Random Access Memory or RAM. You can think of main memory as an array of boxes, each of which can hold a single byte of information. A computer that has 1 megabyte of memory, therefore, can hold about 1 million bytes (or characters) of information. A typical memory hierarchy for optimal performance is implemented in different levels with each level having higher speed, smaller size and lower latency closer to the processor than lower levels shown in figure 1.

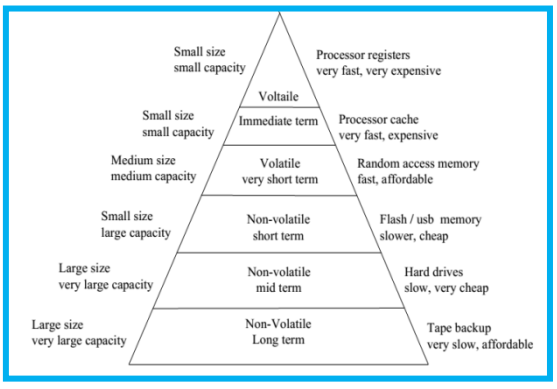

Figure 1: Memory Hierarchy

This figure show the number of levels in the memory hierarchy differs for different architectures and how all goals are achieved at a nominal cost by using multiple levels of memory. The fastest memories are more expensive per bit than the slower memories and thus are usually smaller. The price difference arises because of the difference in the capacity among different implementations for the same amount of silicon. But programs typically specified the location to write memory and what data to put there. This location was a physical location on the actual memory hardware. The slow processing of such computers did not allow for the complex memory management systems used today. Also, as most such systems were single-task, sophisticated systems were not required as much. This approach has its pitfalls. If the location specified is incorrect, this will cause the computer to write the data to some other part of the program. The results of an error like this are unpredictable. In some cases, the incorrect data might overwrite memory used by the operating system. Computer crackers can take advantage of this to create viruses and malware. So, to prevent this type of situation we can use the concept of Shared Memory System.

B. Shared Memory

Shared Memory is an efficient that means passing data between programs. One program will create a memory portion which other processes (if permitted) can access. To realize a shared memory system, it is necessary to avoid memory access contention, maintain cache coherency [2, 3], and realize synchronization between computing nodes or data to enable collaboration. In multiprocessor environment, Shared memory systems cover a broad spectrum, from a system that maintain consistency entirely in hardware to those that do it entirely in software and makes a global physical memory equally accessible to all processors. This system also known as tightly coupled multiprocessor [4] that enable simple data sharing through a uniform mechanism of reading and writing shared structures in the common memory. Figure 2 shows the general structures of Bus-based shared memory system.

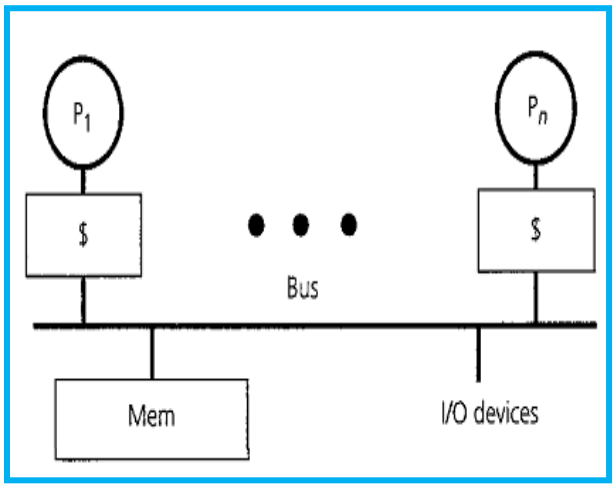

Figure 2: Bus-based shared memory

According to the bus-based shared memory approach (Figure 2), the interconnect is a shared bus located between the processor's cache hierarchies and the shared main memory subsystem This approach has been widely used for small to medium-scale multiprocessors consisting of up to 20 or 30 processors. This system has advantages of ease of programming and portability. However, shared-memory multiprocessors typically suffer from increased contention and longer latencies in accessing the shared memory, which degrades peak performance and limits scalability compared to distributed systems. Memory system design also tends to be complex. Thus, in 1986, Kai Li proposed a different scheme in his PhD dissertation entitled, "Shared Virtual Memory on loosely Coupled Microprocessors", it opened up a new area of research that is known as Distributed Shared Memory (DSM) systems [5] that support in multiple computer environments.

C. Distributed Shared Memory System

A DSM system logically implements the shared memory model on a physically distributed-memory system (distributed memory refers to a multiprocessor computer system in which each processor has its own private memory). So we can say that DSM is a model of inter-process communications in distributed system. Distributed shared memory (DSM) is also a form of memory architecture where the (physically separate) memories can be addressed as one (logically shared) address space distributed shared memory (DSM) is a form of memory architecture where the (physically separate) memories can be addressed as one (logically shared) address space. Figure 3 shows the general structure of distributed shared memory system.

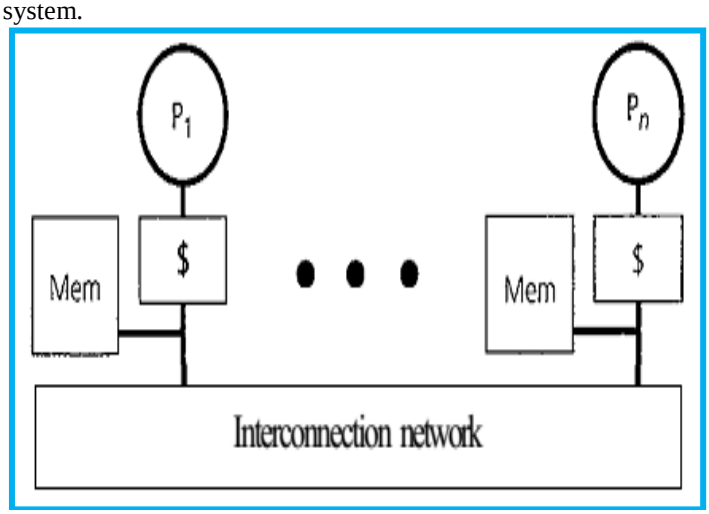

Figure 3: distributed shared Memory System

In this figure 3, distributed-memory, is not symmetric. A scalable interconnect is located between processing nodes or data, but each node or data has its own local portion of the global main memory to which it has faster access. During this step, processes running on separate hosts can access a shared address space. The underlying DSM system provides its clients with a shared, coherent memory address space. Each client can access any memory location in the shared address space at any time and see the value last written by any client. So the main advantage of DSM is the simpler abstraction it provides to the application programmer.

IMPLEMENTATION OF DISTRIBUTED SHARED MEMORY

DSM can be implemented in hardware DSM as well as software DSM.

- ✓ According to Hardware implementation, it requires addition of special network interfaces and cache coherence circuits to the system to make remote memory access look like local memory access. So, Hardware DSM is very expensive.
- ✓ According to Software implementation, a software layer is added between the OS and application layers and kernel of OS may or may not be modified. Software DSM is more widely used as it is cheaper and easier to implement than Hardware DSM.

Design issues of DSM

The distributed shared memory is to present the global view of the entire address space to a program executing on any machine [6]. A DSM manager on a particular machine would capture all the remote data accesses made by any process running on that machine. An implementation of a DSM would involve various choices. Some of them are as below [7].

- ✓ 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

The value of distributed shared memory depends upon the performance of Memory Consistency Model. The consistency model is responsible for managing the state of shared data for distributed shared memory systems. Lots of consistency model defined by a wide variety of source including architecture system, application programmer etc.

D. Memory Consistency Model

Although, shared-memory systems allow multiple processors to simultaneously read and write the same memory locations and programmers require a conceptual model for the semantics of memory operations to allow them to correctly use the shared memory. This model is generally referred to as a memory consistency model or memory model. So we can say that the memory consistency model for a shared-memory multiprocessor specifies the behavior of memory with respect to read and write operations from multiple processors. According to the system designer's point of view, the model specifies acceptable memory behaviors for the system. Thus, the memory consistency model influences many aspects of system design, including the design of programming languages, compilers, and the underlying hardware.

A memory model can be defined at any interface between the programmer and the system whereas the system consists of the base hardware and programmers express their programs in machine-level instructions. There are two type of interface:

- ✓ At the machine code interface, the memory model specification affects the designer of the machine hardware and the programmer who writes or reasons about machine code.
- ✓ At the high level language interface, the specification affects the programmers who use the high level language and the designers of both the software that converts high-level language code into machine code and the hardware that executes this code.

The computer researcher proposed different memory models to enhance distributed shared memory systems (like sequential consistency model, processor consistency model, weak consistency model, release consistency model etc.). These models, to increase the memory access latency, the bandwidth requirements, and simplify the programming. It also provides better performance, at the expense of a higher involvement of the programmer in synchronizing the accesses to shared data.

E. Software Based DSM

A distributed shared memory 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 but 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. The memory organization of a software DSM system determines the way shared virtual memory is organized on top of the isolated memory address space. 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 multiprocessor 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.

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. 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.

Software implementations of the DSM concept are usually built as a separate layer on the top of message passing model.

According to the implementation level, several types of software-oriented DSM approaches can be recognized in Figure 4.

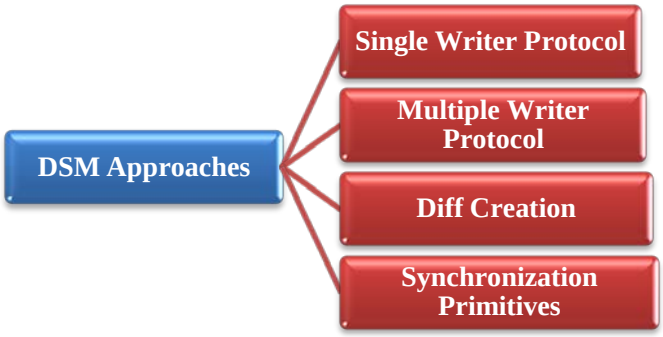

Figure 4: software-oriented DSM approaches

During the implementation of software based DSM, these approaches provide many advantages in design of multiprocessor systems. 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.

Problem of Software Based DSM

During the analysis of Research paper, we found two types of problem:

1. Poor application programming interfaces for solving complicated synchronization problems.
2. Inefficiencies in multiple writer protocol.

II OBJECTIVE

The main objective of software DSM not only depends on performance, but also on Persistence, Interoperability, Security, Resource Management, Scalability, and Fault Tolerance" that are presented in our proposed work for improving access control application in design of multiprocessor systems. During the review of paper, we identify two weaknesses of contemporary software distributed shared memory systems: poor application programming interfaces for programmers who need to solve complicated synchronization problems and inefficiencies in the traditional multiple writer protocols. in future work, these two type of problem can be solve by using different type of methodology and implementation that to provide a strong guarantee of Performance, Persistence, Interoperability, Security, Resource Management, Scalability, and Fault Tolerance during the read and write operation onto memory organization and multiprocessor system.

III LITERATURE SURVEY

Literature Survey is vital part of the any research area for gathering a new type of information during the analysis that produce new technology, application and other beneficial information. In this paper, it includes a review of different techniques and algorithms of Software Based Distributed Shared Memory System.

✓ ***Review of consistency models***

A consistency model is used to express the semantics of memory as observed by the programs sharing it. Traditionally, only computer architects designing multiprocessor systems were interested in memory consistency. However, the study of memory consistency became increasingly popular in recent years and many publications about theoretical aspects appear in the literature. For example, Adve and Hill [8] and Sindhu et al. [9] introduce formal specifications of consistency models. Also, Raynal and Schiper [10] propose a set of formal definitions for several consistency models. The specification of a consistency model provides answers to such questions as: (1) what behavior is expected by the system (i.e., what is the value returned by every read operation performed by a user)? (2) How does the system adhere to the expected consistency of shared data? and (3) what are the constraints imposed on the ordering of shared data accesses performed by two or more processors?

✓ ***Review of software implementations***

During the past decade, several prototypes have been built that provide a DSM abstraction at the system level. System level implementations usually integrate the DSM as a region of virtual address space in the participating programs using the virtual memory management system of the underlying operating system. Li's shared virtual memory [11] provides users with an interface similar to the memory address space on a multiprocessor architecture. Later, he expanded this idea to other architectures and developed a prototype called Shiva on a hypercube [12]. Munin [13] is a runtime system and a server mechanism to allow programs written for shared memory multiprocessors to be executed efficiently in a distributed memory environment. The runtime system handles faults, threads, and synchronization mechanisms and provides support for multiple consistency protocols [14], while the server mechanism handles the correct mapping of shared segments into local memories.

✓ *Review of performance evaluation and analysis*

Performance evaluation of distributed shared memory systems (both hardware and software approaches) is not an easy task. Keleher et al. [15] evaluate three implementations of software-based release consistent protocols.. Levelt et al. [16] compare a language based DSM with a IVY–like system level implementation. Sun and Zhu [17] propose "generalized speedup" as a new performance metric. Also, some work was done measuring the performance of parallel applications that run on distributed shared memory systems [18].

TreadMarks [19] provides a virtual shared address space like IVY and Munin. TreadMarks implement lazy release consistency. A program with a data race condition might get results which programmers do not expect. However, a program without a data race condition runs as if in a sequentially consistent memory model. Unlike Munin, TreadMarks does not have different types of shared variables. All of shared memory follows lazy release consistency. TreadMarks supports two synchronization primitives, locks and barriers.

IV CONCLUSIONS AND FUTURE WORK

In this paper according to analysis, we found that modern software distributed shared memory systems have some

weaknesses that do not increase performance and efficiency during the implementation of software based distributed shared memory system. These weaknesses are:

- There are no high level synchronization primitives provided. In this case, Programmers have to use basic synchronization primitives for example, barriers and locks, to solve synchronization problems.
- If many writers write to the page and read the page then current multiple writer protocols suffer from the high cost of making a stale page current.

Thus in future work, these two type of weaknesses can be solve by using different type of methodology and implementation that to provide a strong guarantee of Performance, Persistence, Interoperability, Security, Resource Management, Scalability, and Fault Tolerance during the read and write operation onto memory organization and multiprocessor system.

V REFERENCES

[1] Stanek, William R. (2009). Windows Server 2008 Inside Out. O'Reilly Media, Inc. p. 1520. ISBN 978-07356-3806-8. Retrieved 2012-08-20. [...] Windows Server Enterprise supports clustering with up to eight-node clusters and very large memory (VLM) configurations of up to 32 GB on 32-bit systems and 2 TB on 64-bit systems.

[2] H. Amano, Parallel Computer. Shoukoudou, June 1996

[3] N. Suzuki, S. Shimizu, and N. Yamanouchi, An Implemantation of a Shared Memory Multiprocessor. Koronasha, Mar. 1993.

[4] M. J. Flynn, Computer Architecture: Pipelined and Parallel Processor Design, Jones and Barlett, Boston, 1995.

[5] Kai Li, "Shared Virtual Memory on Loosely Coupled Microprocessors" PhD Thesis, Yale University, September 1986.

[6] Song Li, Yu Lin, and Michael Walker, "Region-based Software Distributed Shared Memory," CS 656 Operating Systems, May 5, 2000.

[7] Ajay Kshemkalyani and Mukesh Singhal, Ch12: Distributed Computing: Principles, Algorithms, and Systems, Cambridge University Press, CUP 2008.

[8] S. V. Adve and M. D. Hill. A Unified Formalization of Four Shared-Memory Models. IEEE Trans. on Paralleland Distributed Systems, 4(6):613–624, June 1993.

[9] P. S. Sindhu, J-M. Frailong, and M. Cekleov. Formal Specification of Memory Models. In M. Dubois and S. S. Thakkar, editors, Scalable Shared Memory Multiprocessors, pages 25–41. Kluwer Academic Publishers, 1992.

[10] M. Raynal and A. Schiper. A Suite of Formal Definitions for Consistency Criteria in Shared Memories. In Proc. of the 9th Int'l Conf. on Parallel and Distributed Computing Systems (PDCS'96), pages 125–131, September 1996.

[11] K. Li and P. Hudak. Memory coherence in shared virtual memory systems. ACM Transactions on Computer Systems, 7(4):321– 359, November 1989.

[12] K. Li and R. Schaefer. Shiva: An Operating System Transforming A Hypercube into a Shared-Memory Machine. Technical Report CS-TR-217-89, Dept. of Computer Science,Princeton University, April 1989.

[13] J. B. Carter, J. K. Bennett, and W. Zwaenepoel. Techniques for reducing consistency-related communication in distributed shared memory systems. ACM Transactions on Computer Systems, 13(3):205–243, August 1995.

[14] J. B. Carter. Design of the Munin distributed shared memory system. Journal of Parallel and Distributed Computing on Distributed Shared Memory, 1995.

[15] P. Keleher, A. L. Cox, S. Dwarkadas, and W. Zwaenepoel. An Evaluation of Software-Based Release Consistent Protocols. Journal of Parallel and Distributed Computing, 29(2):126–141, September 1995.

[16] W. G. Levelt, M. F. Kaashoek, H. E. Bal, and A. S. Tanenbaum. A Comparison of Two Paradigms for Distributed Shared Memory. Software—Practice and Experience, 22(11):985–1010, November 1992. Also available as Free University of the Netherlands, Computer Science Department technical report IR-221.

[17] X-H. Sun and J. Zhu. Performance Considerations of Shared Virtual Memory Machines. IEEE Trans. OnParallel and Distributed Systems, 6(11):1185–1194, November 1995.

[18] R. G. Minnich and D. V. Pryor. A Radiative Heat Transfer Simulation on a SPARCStation Farm. In Proc. of the First IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-1), pages 124–132, September 1992.

[19] P. Keleher, A. L. Cox, S. Dwarkadas, and W. Zwaenepoel. TreadMarks: Distributed shared memory on standard workstations and operating systems. In the 1994 Winter USENIX Conference, 1994.