Indian Journal of Science and Technology, *Vol 10(28), DOI: 10.17485/ijst/2017/v10i28/112308, July 2017*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improving Performance of Distributed Shared Memory (DSM) on Multiprocessor Framework with Software Approach

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

[1]School of Engineering, RK University, Rajkot – 360020, Gujarat, India; hemantdvasava@yahoo.co.in
[2]BVM Engineering College, Vallabh Vidyanagar, Anand – 388120, Gujarat, India; jmrathod@bvmengineering.ac.in

Abstract

Objectives: To design Distributed Shared Memory (DSM) for the multiprocessor distributed framework using a different software parametric approach that provides significant performance improvement against convention software based architectures. **Methods/Statistical Analysis:** Software distributed shared memory can be architected by using a different concept of an operating system, by utilizing a programming library and by extending underlying virtual address space architecture. It incorporates various design options like granularity, consistency model, implementation level, data organization, algorithms, protocols, etc. We have proposed few software parameter choices and impact which gives significant performance improvement compared to past designs to manage software distributed shared memory. This paper also discusses various issues that exist while moving toward software distributed shared memory implementation. **Findings:** There are two methodologies by which it is possible to achieve distributed shared memory design are first in hardware like cache coherence circuits and network interfaces and the second is software. Here the proposed system architecture makes major impact on programming, performance, design and cost. An algorithm is designed such a unique manner which resides in memory controller and make efficient global virtual memories. It is using variable as granularity which are shared that is more flexible for complex data structure and large database. It is defined using unique identifier which makes its mapping and retrieval more manageable using proposed consistency mechanism. **Application/Improvements:** Distributed shared memory optimization is a most important area of improving distributed system performance. By taking care of good choice on underlying issues and according to system's design requirement, it possible to gain advantages of improved architecture which can be more used for various distributed applications where shared data plays a major role.

Keywords: Distributed Shared Memory, Distributed Systems, Granularity, Interprocess Communication, Virtual Address Space

1. Introduction

A Distributed Shared Memory (DSM) system sometimes called a multicomputer system where multiple independent process nodes with locally allocated memory are connected by a general high speed network. A Software DSM systems can be implemented can be at the operating system level, or by using a programming library and can be thought of as expending the underlying virtual address space architecture that depends upon the requirements.[1] When memory is implemented insight operating system, such systems are transparent, means that the underlying distributed address space is completely hidden to its users. However, software distributed shared memory system is implemented at the programming library or language. Its language paradigm or libraries are not transparent, but developers need to program them differently. These shared memory systems offer a more portable approach for implementations. A Distributed Shared Memory (DSM) system uses a shared memory model of a physically distributed independent memory for Interprocess Communication (IPC).

**Author for correspondence*

A shared memory system controls separate, independent memory into virtual shared segments which is distributed amongst all nodes, main memory and distributes all memory between nodes.[2,3] It is required to choose an appropriate coherence protocol in accordance with a consistent model that maintains the memory coherence. Conventionally, all memory programming can create bottlenecks because of multiple accesses by process in critical regions in memory address space. This is because of large amounts of data processing will requires integrity, consistency and stability of the application and data with system performance. The pre and post development efforts can be more significant for shared data items and message communication of distributed programming to access its operating system kernel, thus increasing overhead to the system at all levels.[2] Figure 1 shows distributed memory architecture. This implementation of creating asynchronous application for distributed tasks which can reduce the performance in bottlenecks and improve the consistency and scalability in the memory system.[2] The Distributed Shared Memory (DSM) can operate on a uniform and non uniform memory architecture that have different access times to processor's memory. The core factors like consistency, synchronization and coherency are required to consider when going designing distributed memory architecture for large scale data processing in a distributed environment. After this right pre and post development programming efforts and efficient tuning like memory issues, it is possible to minimize all the bottlenecks in the system.[2]

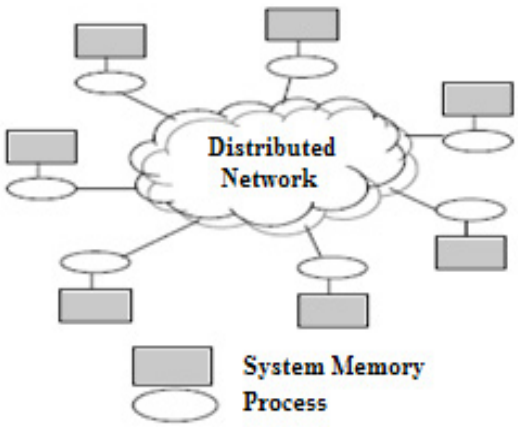


Figure 1. Distributed Memory Architecture.

The distributed shared memory programmings is well understood and focus the interest in parallel and distributed architecture systems for reasons of fault tolerance, availability, and increasing computational power. Motivation for shared memory is the recent trend in design distributed systems using a collection of disk less main computing servers like workstations and at other data servers or file server architectures.[3] In this framework the data for executing program has been cached from the main server or remote node. There can be too kind of issues here. First is a scheduling decision of where of execution of program is best left to a higher level policy making decision. And the second is the task of fetching in the required program code and data.[2] In this paper we have also discussed our experience of implementing a software DSM system using granularity based approach. This is also discussed various designing parameters and its impact on software distributed shared memory architecture. Various suitable parameters or design choices taken into account while designing is considered according to granularity. We have presented an implementation of this model in the context of a Linux operating system.

2. DSM Requirements

In any distributed network system sharing of data will be an essential requirement. Also the recent trends of technology advancement of processor configuration, user will still demand for more performance within parallel running processes.[2] So, it is required to upgrade processor design to get good performance with the distributed shared memory environment.[4] However, the single processor design provides a way to create environment in that multiple processors run simultaneously to get good performance at low cost. It is better to run 15 or more inexpensive processors in parallel rather to design or buy 15 times faster processor.[5-8] This type of design is required to decrease various overhead in distributed virtual address space which can be accessed by simultaneous parallel running processors.[9,10] Solutions to such parallel and distributed communication is DSM which required investigating new architecture design by optimizing imbalance between processors and memory which should be upgradable eventually.[11]

3. Implementation Level

The level where the Distributed Shared Memory (DSM) is implemented is one of most important decisions for designing distributed shared memory system because it affects

both overall programming and system performance with its cost.[4,12-14] Distributed shared memory shared space is distributed among local node boundaries. It is required to execute lookup on each access to find whether requested data are in local memory or located remotely.[2] If it is not in local memory, then the node must fetch data into local memory. Here, the system must require executing an action to preserve and manage coherence view of shared data items.[3] So, both lookup and action can be implement using software, hardware, and combination of both. According to this implementation choice design falls into three group software, hardware, and hybrid methodology.

The implementation choice usually depends on requirements and price performance trade-offs. Software based distributed shared memory implementation originated during 1980s and is to provide the shared memory parameters to the developer using various factors. Traditionally, it can be architects in user level, the OS, at run time library routines or using a programming language.[13] Thus, if the requested data is not in local memory, then memory controller will retrieve the data or address of local memory of another node, or disk of remote node by mapping. Typical representatives of approach are IVY, Mermaid, Munin, Midway, Tread Marks, Orca, Linda, Mirage and Clouds. IVY is few initially proposed software distributed memory solutions with different design parameters.[15-17] This design generally more flexible in comparison with hardware support which enable better tailoring of consistency model to the application characteristics but cannot compete with hardware methodology in terms of cost and performance.[18]

4. Software DSM Model

There are no of reasons to program DSM for distributed environment are stated in the past. Like complex and large data structures can be easily communicated.[2] Shared data item becomes a big issue which can easily managed in software, but not in hardware independent multiprocessor distributed system. It gives transparent process to process communication. Shared data programs are usually smaller and easier to understand. Program implementation of shared memory is a well understood problem. Compact program design, easy implementation and expansion. The distributed shared data memory can be implemented two different ways one by using hardware components, which also utilizes software involvements.[4,16] Another is software methodology which also provide flexibility in organization of shared data segment in different ways. Like page based method organization of shared memory into fixed size pages. The object based or shared variable based approach organization of the shared data region storing data of variable sizes. One other commonly described implementation type that uses a tuple space where the unit is sharing is a tuple.

5. Achieving Methods and Implementation

From the usual ways of architecting DSM system like hardware as cache coherence configuration and network interfaces and based on application , software distributed shared memory systems can be implemented in programming library, at the operating system level or at underlying memory architect can be used to achieve the goal with different software parameters.[19,20] Implementation at the operating system level will remain transparent to programmer that means distributed shared memory will remain completely hidden to its user but system can take benefits of design. Here, software distributed shared memory systems implemented as language or library level are not completely transparent and hidden to programmers usually they have to do programming separately. But advantage is that these kinds of systems are more portable and easy updatable software approach to DSM system design.

The DSM systems implementation of virtual shared memory model of independent physically distributed memory system. If we consider one of its design parameter granularity, then by using three ways it is possible to design a software DSM like page based method that use the system's virtual address space, shared variable method which utilize programming routines to share and access shared variables and object based approach which access shared data through object oriented concepts.[21] It also involves various other implementation choices depend upon its design requirements which includes distributed shared memory algorithms, level of implementation like in hardware, software or hybrid, critical region semantics, semantics of replication level, naming scheme specification has to be used to access remotely located shared data, replication location for optimization, system consistency

model, the granularity of shared data, remote access and caching/replication control of hardware or software, distributed shared memory controlled by virtual memory management software, OS or language run time system etc.[2] According to design choices and issues memory can be implemented.[22] Here, subsequently our distributed shared memory system design considerations systems are explained in brief with its advantages over conventional designs which provide another methodology to implement DSM that provide significant performance improvement. However the design choice may vary according to system design requirement and performance, cost and scaling parameters.

5.1 Algorithm Architecture

An algorithm for implementing Distributed Shared Memory (DSM) is the most significant part of the architecture. Conceptually, these algorithms make extension of the local virtual address spaces to extend multiple host machine connected by a Local Area Network (LAN), and many of them can be integrated in virtual memory systems. The distributed shared memory implementation algorithm deal, with two basic issues first is static and dynamic allocation of shared data across distributed network system to minimize its latency.[2] Another is to preservation of coherence view of shared data when designer minimize coherence management overhead. There are two strategies can be used to manage shared data which are migration and replication.[23] In migration implementation methodology only one copy of a data exists at time in the system while in replication multiple copies of similar data item reside at different local memories. So, to decrease memory coherence overhead, developer more prefer this method when sequential consistency of writing sharing is prevalent.[5,19,24]

There is basically four algorithms client server; migration read replication, full replication based on migration and replication.[2] Each having a different methodology to manage distributed shared data. Here distributed shared memory algorithm will manage all memory management activity through memory controller using some protocol. If requested data not available in local memory, then request go to memory controller that will check look up to serve requests for remotely located data. Multiple processes can read and write simultaneously to share data. If it is required to update the system to improve system then it will be easily manageable.

5.2 Data Organization

To work distributed system efficiently it is required to organize and manage data using proper conditions. Any process can share large amount of shared variables and data structures which are utilize by more than one node in the environment.[2] So, it is required to decide changes from paging on the network to how to maintain a potentially distributed, replicated data items consisting of the shared variables and data structures. Granularity is one of the parameters are the size of data unit that exists in distributed systems.[2] This is a very important decision which makes another big impact on distributed system design. Based on granularity parameter data can be organized using shared variables and data structure, pages or by objects as the unit for data transfer. All granularity design having its own advantages and disadvantages. A distributed shared memory using shared variable is to let the applications or developer decide as to which variables are to be shared in an environment and memory controller will manage a database of the shared variables and data structures in the distributed framework.[2] By using a memory controller the developer to explicitly declare data variables are to be shared globally in the application[2]. It removes the issues of thrasing and false sharing issues. All process of different independent machine can access shared variables and data structures by requesting through memory controller.

5.3 Memory Controller

Memory controller mapping and manage virtual memory used by different processes to communicate via shared variables to each other. It will make interprocess communication via global virtual address space. Mapped shared data or memory tuning between the environment of remotely located shared data and process's own local memory. Some programs or as divide file into variable size chunks and copy them into virtual address space so that can be available in a process's address space. After that process can read/write remote contents with ordinary memory access. This permits fast access of shared content. Figure 2 shows the memory mapping mechanism done by different process.

Here, memory controller will plays major role insight distributed system. It will dynamically manages all the activities of distributed memory system like managing indexing/mapping memory segment or shared data items, handle requests from processors, converting instruc-

tion format sets, locking mechanism, retrieving data and location from remote memories, scheduling and creates virtual global address space.

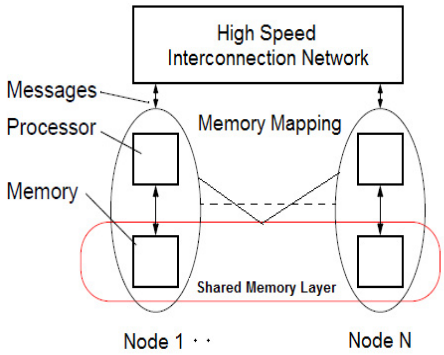


Figure 2. Illustration Distributed Shared Memory Mapping.

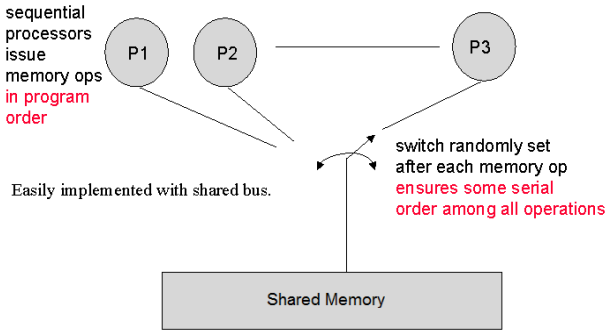
5.4 Naming Scheme Specification

In a distributed shared memory systems if the process wishing to access remotely located data then it required knowing on which system data resides and fetch it from remote node. System will take the help of memory controller to manage the shared memory system activity. So, all shared data is visible to the memory controller, it is also managing lookup or indexing. The memory controller has required using unique naming mechanism to avoid any conflicts while shared data items are distributed over remote location. And possible solution is to assign a logical global naming. The memory manager at each machine performs the conversion of the logical address to provide the location of the data segment on a remote node. But this approach is not being useful if the granularity units of shared data items are less than a page.[2] Memory controller plays an important role between local and remote node to access requires shared data. So, first process will use data items from its own local memory. If required data items are not in local memory then requested transfer to the memory controller. Memory controller satisfies the request by mapping mechanism and to respond to requests made. Shared data items having unique global identification that is registered with memory controller. All process wishing to access shared data items use the same identifier.

5.5 Consistency Model

Today's modern high speed processor executes operations in a different order than specified in past literature to increase execution speed and decrease idle waiting time. This program instruction execution order often categories in to different types. In this kind of shared memory system the execution is in the correct order if the result same as one in which the instruction were executed in program order. Such kind of system design known as sequential.

Here, distributed shared memory model utilizing sequential consistency order which ensures no memory operation execution begins until all the previous operation have been completed. Figure 3(a) and (b) shows sequential consistency implementation among different process. In a distributed system an efficient, ordered effective broadcast mechanism, shared variables could be grouped together on one or more variable, and operations of shared variable require to be broadcasted and all nodes have to require seeing the similar order of memory pointers.



(a)

P1: W(x)a			W(x)c	
P2:	R(x)a	W(x)b		
P3:	R(x)a		R(x)c	R(x)b
P4:	R(x)a		R(x)b	R(x)c

(b)

Figure 3. (a) and (b) Sequential Operation Exexution.

5.6 Protocol Mechanism

A Distributed system needs communication between multiple nodes insight system. In this proposed architecture internet domain sockets implemented to make connections between processes on different nodes which are connected by a high speed network. Machines are connected via local area network. In this write invalidate protocol write access to the shared data item will invalidation of all replication excluding one before the write begins. Invalidated replicated data units are not accessible through any node of system. So, the major drawback of writing the

invalidate protocol implementation is that invalidation require to send each node whether they have used shared data or not. It is better suitable for programs where several updates held during multiple reads, as well as when an application exhibits in which high degree per node locality of reference. However, sometimes this write invalidate protocol is inefficient when no of nodes frequently access data items, because an updated data will require to be copied back to many processes on nodes immediately after every invalidation occurs. No of distributed memory systems have used to write invalidate protocol like IVY which support strict consistency model, Clouds that have used release consistency model and many more in the past.

5.7 Advantages and Disadvantages

This software based distributed shared memory system having various advantages as below:

- Provides large virtual memory space
- Well expansion with increase in process nodes
- Underlying message communication between nodes will remain invisible outside.
- Complex structure and large database support
- Generally cheaper than using a multiprocessor system
- More portable because of common interfaces
- Protect its developer from sending and receiving primitives

It also gives few disadvantages like becomes slower to access in comparison with non distributed shared memory and it is requiring additional concurrency control mechanism against concurrent accesses towards shared data.

6. Conclusion and Future Work

A distributed system can be implemented using various distributed model where shared memory plays major role in architecture. In this research paper we have described a new way of architecture using software parameters of shared memory system design. So, shared memory implementing requires the careful evolution of several design issues such as interaction with shared data by node, data organization, consistency model and algorithm, protocols, scheduling etc. It requires complex aspects in core architecture and close association between virtual address management and mechanism that make distributed shared memory possible. Latency plays an important role while shopping the protocol. This implementation uses sequential consistency which is the strongest consistency model that is practicable when there is delay in communication over network. The motivation for this design was to implement using shared variable and data structure, its DSM algorithm, memory controller concept and specific issues, kernel programming, inter-process communication and protocol involved. These goals have been largely met to get improved performance and reliability.

Future work can be done to improve algorithm and memory controller activities, respectively to design issues that will add more benefits to distribute shared memory management. It can manage more no of processors simultaneously, increased execution speed and also provide security. More effort can be added to increase the consistency mechanism of system to make the design more reliable against the failure.

7. References

1. Sinha KP. Distributed Operating Systems Concepts and Design; 2009. p. 1–90.
2. Vasava HD, Rathod JM. A Survey of Software based Distributed Shared Memory implementation methodologies for Multiprocessor Environments. International Journal of Innovative Research in Science Engineering and Technology. 2013 Jul; 2(7):3055–60.
3. Tanenbaum SA, Steen MV. Distributed Systems Principles and Paradigms. 2nd Edition. PHI Learning; 2011. p. 1–705.
4. Naeem A, Jantsch A, Lu Z. Scalability analysis of memory consistency models in NoC based distributed shared memory SoCs. IEEE transaction on computer aided design of integrated circuits and systems. 2013 May; 32(5):760–73.
5. Jin X, Guan N, Lv M, Deng Q. Improving the Performance of Shared Memory Communication in Impulse C. IEEE Embedded Systems LETTERS. 2010 Sep; 2(3):49–52. Crossref
6. Choi I, Zhao M, Yang X, Yeung D. Experience with Improving Distributed Shared Cache Performance on Tileras Tile Processor IEEE Computer Architecture. 2011 Dec; 10(2):45–8.
7. Ibrahim KZ, Hofmeyr S, Lancu C. The Case for Partitioning Virtual Machines on Multicore Architectures. IEEE Transactions on Parallel and Distributed Systems. 2014 Oct; 25(10):2683–96. Crossref
8. Yu Z, Xiao R, You K. A 16-Core Processor with Shared Memory and Message-Passing Communications. 2014 Apr; 61(4):1081–94.

9. Vasava HD, Rathod JM. Software based Distributed Shared Memory (DSM) model using shared variables between Multiprocessors. In Proceeding of IEEE International Conference on Communication and Signal Processing; 2015. p. 1431–5. Crossref
10. Gopal B, Beg R, Kumar P. Memory Management Techniques for Paging on Distributed Shared Memory Framework. International Journal of Computer Science and Information Technology. 2010 Apr; 2(2):141–53. Crossref
11. Dhruv S, Mehta, Iye VS. Study and Implementation of Distributed Shared Memory. NIT: Nagpur; 2004-2005.
12. Seidmann T. Distributed Shared Memory in Modern Operating Systems. University of Technology in Bratislava; 2004.
13. Kshemkalyani A, Singhal M. Distributed Computing: Principles Algorithms and Systems. Cambridge University Press; 2011 Mar. p. 1–756.
14. Penurkar MK, Sugandhi R. OPDSM: A Combinational Page based and Object based DSM Model. International Journal of Computer Applications. 2010 Feb; 1(6):1–5. Crossref
15. Castanie L, Mion C, Bruno L. Distributed Shared Memory for Roaming Large Volumes. IEEE Transactions on Visualization and Computer Graphics. 2006 Oct; 12(5):1299–306. Crossref
16. Navratil PA, Childs H, Fussell SD. Exploring the Spectrum of Dynamic Scheduling Algorithms for Scalable Distributed-Memory Ray Tracing. Transactions on Visualization and Computer Graphics; 2013. p. 1–14. PMid:23831285
17. Marongiu A, Benini L. An Open MP Compiler for Efficient Use of Distributed Scratchpad Memory in MPSoCs. IEEE Transactions on Computers. 2012 Feb; 61(2):222–36. Crossref
18. Lee M, Ahn M, Kim EJ. Fast Secure Communications in Shared Memory Multiprocessor Systems. IEEE Transactions on Parallel and Distributed Systems. 2011 Oct; 22(10):1714–21. Crossref
19. Krzyzanowski P. Distributed Shared Memory and Consistency Model. Rutgear University; 2009. p. 1–10.
20. Dash A, Demsky B. Integrating Caching and Prefetching Mechanisms in a Distributed Transactional Memory. IEEE Transactions on Parallel and Distributed Systems. 2011 Aug; 22(8):1284–98. Crossref
21. Shi Q, Khan O. Toward Holistic Soft-Error-Resilient Shared-Memory Multicores. IEEE Computer Society Press. Los Alamitos, USA. 2013 Oct; 46(10):56–64. Crossref
22. Tumeo A, Villa O, Chavarria-miranda DG. Aho-corasick stirng matching on shared and distributed memory parallel architecture. IEEE Transaction on Parallel and Distributed Systems. 2012 Mar; 23(3):436–43. Crossref
23. Apostolakis A, Gizopoulos D, Psarakis M, Pachalis A. Software based Self Testing of Symetric Shared Memory Multiprocessors. IEEE Transactions on Computers. 2009 Dec; 58(12):1682–94. Crossref
24. Diaz J, Caro CM, Nino A. A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. IEEE Transactions on Parallel and Dist Systems. 2012 Aug; 23(8):1369–86. Crossref