CHAPTER 4 SOFTWARE BASED DSM APPROACHES

4.1 Introduction

There are numerous systems studied and developed in various countries around the world (19,24). Few of them are described in this chapter are several developments which achieve their requirement and understanding. Various researchers have used diverse techniques in previous developments of research centers and universities and its comparison of the proposed system is described in this section. This makes clear differences between all distributed shared memory system configurations. Until the previous decade, dispersed computing systems widely engaged message passing communication (46). To measure the accuracy of the projected technique with variable based granularity described in chapter 3. Conversely, this performed to be much less suitable than the shared space programming method since the developer must be responsive of data spreading and explicitly achieve data interchange via messages. In addition, such designs present severe complications in passing composite data constructions and process migration in various address spaces are encouraged. Therefore, the knowledge of constructing a software methodology that delivers the shared memory pattern to the designer on the upper of message passing occurred.

4.2 Software Based DSM Systems

Many researchers have used different software based distributed shared memory operations in previous research projects of research centers and universities are described subsequently. In these all DSM projects, programmers have tried to make different DSM implementation adoptions that can distinguish with proposed architecture.

IVY famous as integrated shared virtual memory at Yale is employed in an Apollo domain architecture (19) i.e. token ring by Apollo workstations. It utilizes granularity of 1 kb page. IVY contains five components. From these five three components, memory allocation, process utilization and organization starting client interface defined by a set of primitives used by an
application. The operation system provides lower level support to memory mapping functions and remote execution. There is virtual memory space and private local address space to process. So, local address space remains private to each machine or process. The memory mapping manager does mapping between virtual memory and local address space that executes page fault and block that process if the page is not local. After, page fault it demands remote access to the manager and acquire page from local memory of another site. The central manager scheme is implemented for read and write consistency with strict consistency model and write update protocol.

The Mirage was established at the University of California, Los Angeles, kernel modified to provide distributed shared memory operation (19,24). In, IVY page is transferred between several processes. To manage thrashing mirage extent the coherence protocol of IVY system. Each process access the page for ‘delta’ seconds after acquiring requested page. Mirage decreases locality of reference and thrashing are an important benefit of design. The page segmentation mechanism most significantly utilized by mirage. Each process creates a segment by size, access shield and name and cooperating process find and access that by using name only. In Mirage, all segments are managed by library site as single node and it processes queue and process subsequently.

Clouds is implemented at the faculty of technology, Georgia (24). The global distributed shared memory is formed and observed of logical address of all site. Shared data are shipped to requesting site in case of remote request of the site. The distributed shared memory controller (DSMC) manages overall data transmission. The actual data transference is completed by the distributed shared memory controller (DSMC). This DSM have a collection of segment which is basically comprise object set and those have an access characteristics like to read only, read write, weak read, or none. Since the site to acquire a copy of the page with no assurance that the page will not be changed during read the shared memory behavior of clouds without any definite limitations in week read and that leads to memory in an inconsistent state.
Munin approach is weaker in comparison of the sequential consistency model, but inexpensive to develop (19). It utilizes locks, barriers and semaphores to accomplish sufficient consistency. Granularity is achieved by using objects and it’s provisions heterogeneous system framework. There are some boundaries in the intersection between memory executions. The munin DSM system includes two important features: type specific coherence mechanisms and the release consistency model. It is a dynamic system application, although it too needs a preprocessor that translates the program annotations, some library routines, a modified linker and operating system support. It employs different coherence protocols well suitable to the expected access design for a shared data object type.

The Mermaid has produced shared data conversion routine automatically in distributed object based framework instead to provide fault tolerance. To implement shared memory mechanism on heterogeneous environment, it has used sun workstations (19,24). It is also provided illusion of global distributed memory with pages as a shared memory contents. The Mermaid is developed by write invalidate protocol in combination of strict consistency model. The signal /wait and semaphore messages are utilized for the shared data organization. The distributed shared memory method is developed at the user level as a library package for linking to the application. Negligible deviations to the Sun OS, operating system kernel comprise situation the access authorization of the user level memory pages, as sending the address of a shared memory page to its user level fault handler. In addition to data swapping, the essential for data conversion arises due to heterogeneity of clusters.

Midway maintenances a distinctive consistency method for memory recognized as entry consistency (19). The data items become consistent at a site when the processor acquires a coordinated data contents acknowledged to protect the data is a guaranty of entry consistency. Entry consistency (EC) is weaker than other structural design designated in the earlier, such as processor and release consistency, but it improves potential greater
performance programming of the entire consistency protocol. Midway cares many consistency models like entry, processor and release that can convert dynamically in the similar application. It is implemented in C programming language. Against granularity of different shared data contents midway also protect processor and stronger release methodologies.

TreadMarks is famous distributed shared memory mechanism developed at rice university (19,24). The standard Unix systems such as Ultrix and Sun OS incorporated in a structural distributed memory design of Treadmarks. To preserve latency rate TreadMarks project emphases on preserving shared memory consistency. So, it is characterized as a lease consistency memory model. Against straight models this requires a slow latency to keep logical address space consistent, but uses a similar interface. TreadMarks utilize multiple writer memory protocols to stating the problem. The multiple writer protocol used by this project to address the defined problem. The several information required to organize pages by process in multiple writer protocol.

Another portable, object based, distributed shared memory structural design is well known as Orca (19,24). When read/write ration is small for replicating mixture of shared data contents Orca utilize a write update protocol transmission which is very efficient for this environment. With the totally ordered segment latency method overhead for performance is less. This environment is intelligent to yield near enhanced verdicts for object placement and replication. The distributed development depends on migration, discerning replication and an update methodology. Diverse deviations of the update methodologies are offered, depending on the communication delivered from the point to point, reliable and unreliable multicast fundamental distributed system. For application designing predominantly Orca project was developed. In comparison, of CRL and Treadmarks the Orca application sends less messages and less and also gain improved speedups.

Linda’s great data shared data organization is well known for its characteristics of distributed computer (19). With a minor syntax extension set of primitives are combined into programing language which makes shared
memory access possible. A group of tuples of varying sizes and types recognized as the tuple space recognized as distributed shared memory in Linda. Each tuple may be different in terms of its size, value, the type and sequence of its components. Linda reflected as the type of an object based shared memory as it is more organized in comparison or page and shared variable.

Mether is a distributed shared memory that runs on sun terminals under the sun os 4.0 (24). Employer programs access the mether shared space in a manner in different from other memory. Mether was motivated by the MemNet shared memory, but not like MemNet, mether contains of software communicating over a conservative ethernet. The kernel portion of mether really does no data negotiation over the network. Data communication is accomplished by a user level server. The kernel driver has no partiality for a server and certainly does not know that the servers be present. The kernel driver has been complete very safe and in fact fear is not in its library. It is dispersed in the sense that the pages of memory are not all at one workstation, but somewhat transfer around the network in a demand paged fashion. It is shared in the reason that processes concluded the network share read, write, and execute access. And it is memory in the logic that user application access the data in a mode similar to other shared memory. The shared memory is certainly not paged to disk, but the delay of retrieving a page over the network is almost the identical as a paging disk.

Including above research Brizzard, Plus, Agora, Shasta, CRL, Memnet, Dash, KSR1, DDM, RMS, SCI, Merlim, JIAJIA is various distributed shared memory research conceded in hardware, software and hybrid approaches with appropriate necessity and design goals (19,24). Table 4.1 illustrates few software based distributed shared memory representations and its comparisons.

4.3 Proposed DSM Technique

In this section, we have presented a novel software DSM technique to handle distributed data on physically divided sites. Based on granularity specification
and other parameters mentioned in chapter 2 distributed shared memory can be designed in software using various approaches. When to consider design moderate or lower scale distributed memory architecture. It is better to design memory in software in comparison of other approaches because sharing of data may be a problem which is easily tackled in software not in hardware. To select specific constraints and approach is more suitable to make development of shared memory design against project goal which results eliminated some shortcomings. The configuration and selection of various parameters and its impact on DSM briefly discussed in chapter 5. Here, it is presenting architecture difference with typical representatives in terms of its structural design.

![Figure 4.1 Configuration of Memory Controller](image1)

**Figure 4.1 Configuration of Memory Controller**

![Figure 4.2 Mapped Region from Local Memory](image2)

**Figure 4.2 Mapped Region from Local Memory**

In Fig. 4.1 different nodes specify sites with are connected via high speed communication links. Each site has its own local memory and processors. The
memory controller is interconnected with each site and creates an illusion of
the shared region shown in Fig. 4.2. Process wishing to access remote data
will send a request to the memory controller. Shared data are organized by
using shared variable. Sequential consistency well manages the coherence
semantics. A research investigated to make system most effective at different
point of view, but required to add more effort to make commercial use.
Implementations are also done with GUI based platforms using object
oriented concept and languages. As mention concepts in chapter 3 several
research project has been carried out in literature which are acceptable at a
certain level of benefits and drawbacks. Meanwhile, the message passing
structure is more difficult to implement and easy to build and expand. (47). A
message passing and the shared memory structure require to extend at par.
The possible solution to parallel system architecture is distributed shared
memory which can be considered after clear, thoughtful of the prerequisite
and its description.

4.4 Summary

Software provision for distributed memory is usually more flexible than
hardware provision and allows better adapting of the consistency method to
the application behavior. Though, it typically cannot contend with hardware
employments in performance. Usually, is can be accomplished in the run time
library, the operating system, user level or a programming language. Several
distributed memory architectures utilize three fundamental techniques. Higher
sizes in gain are distinctive for software resolutions, because the shared
memory organization is usually reinforced concluded local memory. Thus, if
the demanded content is not accessible in private memory, a fault manager
will acquire the contents either from the local memory of another system or
from the setup.

With the great locality of references coarse grain pages are profitable for
programs which also decrease the essential library storage. But, fine grain
distribution utilized by parallel applications which unfavorably affected,
because of the thrashing and false sharing. The research can depend on
broadly accessible programming languages and operating systems on the systems of terminals, numerous implementations of software distributed shared memory have developed. The translation of regular data types, the user must offer translation routines and an index mapping data types to specific routines besides user defined data types. Just one data type is permitted per page. Subsequent Table 4.1 summarizes these implementations: IVY, Mirage, Clouds, Munin, Mermaid, Midway, TreadMarks, Orca, Linda and Mether. It is guaranteed that the variable page size appropriate to data access configurations in Mermaid project.

Table 4.1 describes typical representative where its depict the implementation type, consistency model, type of algorithm, granularity mechanism and coherent policy. All representative is using different implementation approaches based on their requirement from mentioning technique. While the algorithm may be separate from each other may be multiple read or write or single read or write. Consistency model fulfills the system requirement. Granularity mechanism is also described in Table 4.2. Where on the software based implementation page and object based mechanisms are widely implemented. Coherency policy comparison depicts that invalid method is more suitable to maintain data more consistency. Basically, models are implemented in a homogeneous environment except a few. The sequential consistency mechanism is far better to implement in software in comparison of other models. The overall design choice will be depend on operating environment and project goals.
### Table 4.1 Some Software DSM Implementations

<table>
<thead>
<tr>
<th>Imp.</th>
<th>Type of Operation</th>
<th>Kind of Algorithm</th>
<th>Consistency Model</th>
<th>Granularity Unit</th>
<th>Coherence Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVY</td>
<td>User level library + OS modification</td>
<td>MRSW</td>
<td>Sequential</td>
<td>1 kb</td>
<td>Invalidate</td>
</tr>
<tr>
<td>Mirage</td>
<td>OS kernel</td>
<td>MRSW</td>
<td>Sequential</td>
<td>512 bytes</td>
<td>Invalidate</td>
</tr>
<tr>
<td>Clouds</td>
<td>OS, Out of kernel</td>
<td>MRSW</td>
<td>Inconsistent, Sequential</td>
<td>8 kb</td>
<td>Discard Segment when Unlocked</td>
</tr>
<tr>
<td>Munin</td>
<td>Runtime System+Linker+Library+Preprocessor+OS Modification</td>
<td>Type specific (SRSW, MRSW, MR-MW)</td>
<td>Release</td>
<td>Variable sized Objects</td>
<td>Type Specific (Delayed Update, Invalidate Update)</td>
</tr>
<tr>
<td>Mermaid</td>
<td>User level library + OS modification</td>
<td>MRSW</td>
<td>Sequential</td>
<td>1 kb, 8 kb</td>
<td>Invalidate</td>
</tr>
<tr>
<td>Midway</td>
<td>Runtime system + Compiler</td>
<td>MRMW</td>
<td>Entry, Release, Processor</td>
<td>4 kb</td>
<td>Update</td>
</tr>
<tr>
<td>TreadMarks</td>
<td>User level</td>
<td>MRMW</td>
<td>Lazy Release</td>
<td>4 kb</td>
<td>Update, Invalidate</td>
</tr>
<tr>
<td>Orca</td>
<td>Language</td>
<td>MRSW</td>
<td>Sequential</td>
<td>Variable (tuple size)</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>Linda</td>
<td>Language</td>
<td>MRSW</td>
<td>Sequential</td>
<td>Variable (tuple size)</td>
<td>Implementation dependent</td>
</tr>
<tr>
<td>Mether</td>
<td>User Level+Language</td>
<td>MRMW</td>
<td>Release</td>
<td>32 bytes</td>
<td>Update, Invalidate</td>
</tr>
<tr>
<td>Imp.</td>
<td>Heterogeneity</td>
<td>Granularity Type</td>
<td>Consistency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------------------</td>
<td>----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVY</td>
<td>No</td>
<td>Pages</td>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mirage</td>
<td>Yes</td>
<td>Pages</td>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clouds</td>
<td>No</td>
<td>Pages</td>
<td>Inconsistent, Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Munin</td>
<td>Yes</td>
<td>Variable sized objects</td>
<td>Release</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mermaid</td>
<td>Yes</td>
<td>Objects/Pages</td>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Midway</td>
<td>Yes</td>
<td>Variables</td>
<td>Entry, Release, processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TreadMarks</td>
<td>No</td>
<td>Pages</td>
<td>Lazy Release</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orca</td>
<td>No</td>
<td>Objects</td>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linda</td>
<td>No</td>
<td>Tuples</td>
<td>Sequential</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mether</td>
<td>No</td>
<td>Pages</td>
<td>Release</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>