© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

Issues of Consistency in Distributed Shared Memory System

Sandeep Yadav, Swati Sharma
Dronacharya College Of Engineering
Department of Information And Technology, Gurgaon, Haryana, India

Abstract- **Distributed shared memory is used to provide an environment where computers support a shared address space that is made by physically dispersed memories. The popularity of Distributed Shared Memory system is believed to be increasing in parallel computing, as it offers a single system memory view which makes the programming easy as well as the control of parallel computing using multiple processors. Consistency is an important issue because there might be some potential consistency problems when different processors access, cache and update the shared single memory space. The designers of distributed shared memory systems should decide the proper standards of memory coherence semantics and consistency protocols in order to get better performance and get accurate result of computation . In this paper, we first report the overview of distributed shared memory systems and reveal the consistency problems and their feasible solutions. We will also study the cases of several up to date implementations and their role in maintaining system memory consistency.**

I. INTRODUCTION

1.1 Overview

In 1986, Kai Li distributed his Phd thesis entitled, "Imparted Virtual Memory on Inexactly Coupled Microchips," in this manner opening up the field of research that is currently known as Appropriated Imparted Memory (DSM) frameworks.[1] From that point onward, heaps of scrutinizes in dispersed imparted memory frameworks have been proposed. In conveyed imparted memory frameworks, forms shared data crosswise over hub limits transparently. All hubs in the appropriated imparted memory framework see the same figment of a solitary location space. Any processor can get to any memory area in the location space specifically. Memory mapping administrators is in charge of mapping between nearby memories and the imparted memory location space. Other than mapping, their boss obligation is to keep the location space lucid at tall times; that is, the worth returned by a read operation is dependably the same as the quality composed by the latest compose operation to the same location[2]. There favorable circumstances of conveyed imparted memory frameworks including:

• Methodologies can run on distinctive processors in parallel.

• Memory mapping, page faulting, information development are overseen by circulated imparted memory without client mediation.

• Single location space disentangles programming tasks.

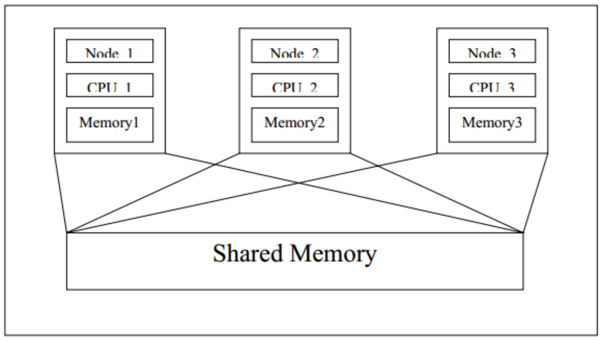


Fig 1: A single image illusion of distributed shared memory systems

1.2 Design Issues

A few outline issues need to be tended to before we go further into this study. Each of these components altogether influences the execution of the framework.

• Virtual memory and Distributed Memory System

Current machine frameworks utilize the idea of virtual memory to accomplish better execution. The virtual memory administration component is in charge of page substitution, swapping and flushing. Essentially, in fulfilling a remote memory ask for, the appropriated imparted memory would need to counsel the virtual memory supervisor to get a page outline, and so forth. The viability of the appropriated imparted memory standard depends vitally on how rapidly a remote memory access solicitation is adjusted and the processing is permitted to proceed.

• Granularity:

Processing granularity alludes to the measure of the imparting unit. It can be a byte, a saying, a page or other sort of unit. Picking the right granularity is a real issue in disseminated imparted memory on the grounds that it arrangements to the measure of calculation done between synchronization or correspondence focuses. Moving around code and information in the systems includes dormancy and overhead from system conventions. In this way, such remote memory gets to need to be incorporated some way or another with the memory administration at every hub. This regularly strengths the granularity of access to be an indispensable numerous of the central unit of memory administration (normally a page) or just exchange piece of the page to lessen the idleness[3].

• Memory Model and Intelligibility Conventions:

To guarantee right multiprocessor execution, memory models ought to be utilized with consideration. Two customary memory models are used in numerous conveyed imparted memory frameworks. Successive Consistency memory model guarantees that the perspective of the memory is predictable at all times from all the processors. The other is Discharge Consistency, which recognizes sorts of synchronization gets to, specifically, obtain and discharge, creating a predictable perspective of imparted memory at the discharge point[3]. A few cognizance conventions are utilized to keep up memory consistency and will be distinguished in subtle element in later segments.

1.3. Consistency Issues in Distributed Memory System

To get adequate execution from a Disseminated Imparted Memory Framework, information must be set close to the processors who are utilizing it. This is carried out by imitating and substituting information for read and compose operations at various processors. Since a few duplicates of information are put away in the neighborhood reserve, read and compose access can be performed effectively. The reserving system expands the proficiency of Appropriated Imparted Memory Frameworks, yet it additionally raises the consistency issues, which happens when a processor composes (changes) the reproduced imparted information. How and when this change is noticeable by different processors who additionally have a duplicate of the imparted information turns into a paramount issue.

A memory is predictable if the worth returned by a read operation is dependably the same as the quality composed by the latest compose operation to the same location. In a conveyed imparted memory framework, a processor needs to get to the imparted virtual memory when page flaws happen. To lessen the correspondence expense started by this reason, it appears to be regularly to build the page size. In any case, vast page size delivers the discord issue when various procedures attempt to get to the same page and it likewise triggers the false imparting issue, which, thus, may expand the quantity of messages due to aggregation[4].

False offering is created by the vast size of the memory page and thought to be an execution bottleneck to conveyed imparted memory frameworks. False imparting happens when two irrelevant variables (each one utilized by diverse procedures) are set in the same page. The page seems imparted, despite the fact that the first variables were not. Routine programming typically obliges methods to increase restrictive access to a page before it begins adjustment. Hence, false offering prompts a race condition where numerous processors go after responsibility for page while really they are altering entirely unexpected sets of information.

A few methods are acquainted with decrease the impact of false offering including: Loose memory consistency model and compose imparted conventions. We will explore these arrangements and the executions in later segments.

Numerous arrangements are proposed to decrease or even kill these consistency issues. We will examine some of them in later segments

II. MEMORY INTELLIGIBILITY MODELS

2.1. Consecutive Consistency

Lamport characterized the framework to be consecutively (strictly) reliable if:

The aftereffect of any execution is the same as though the operations of every last one of processors were executed in some consecutive request, and the operations of every individual processor show up in this grouping in the request defined by its program.

The framework guarantees that all gets to of the imparted memory from diverse processors interleave in a certain way so that the weighty execution is the same as though these gets to are executed in some successive request. While this model ensures that each compose is promptly seen by all processors in the framework, it additionally creates more messages for keeping up this sort of consistency and, accordingly, higher idleness [7]. In addition, deciding consecutive consistency is a NPcomplete issue, which may prompts genuine framework stoppage in huge scale Distributed shared memory System

2.2. Processor Consistency

Processor consistency permits composes from diverse processors to be seen in distinctive requests, in spite of the fact that composes from a solitary processor must be executed in the request that they happened. Unequivocal synchronization operations must be utilized for gets to that ought to be all around requested. The primary playing point of processor consistency is that it permits a processor's peruses to sidestep its composes and henceforth build the framework execution.

2.3. Relaxed Consistency

Loose (feeble) consistency does not oblige changes to be noticeable to different processors quickly. At the point when certain synchronization gets to happen, all the past composes must be seen in the project request. Two methodologies are said to be contending if no less than one of them is a compose. Imparted memory gets to are classified either as standard or synchronization gets to, with the recent class further partition into secure and discharge gets to [8].

Two well-know methodologies executing the loose consistency are:

• Discharge Consistency (RC):

Discharge consistency is a manifestation of loose memory consistency. A framework is discharge predictable if:

- Before a common access is permitted to perform as for some other processor, all past secures must be performed
- Before a discharge is permitted to perform as for whatever other processor, all 6 past common peruses and composes must be performed.
- Uncommon gets to are successively reliable concerning each other.

The preference of this manifestation of consistency is that it postpones the consistency overhaul with synchronization occasions. In this manner, overhauls happen just when required by application and unnecessary messages will be diminished. Nonetheless, most discharge predictable frameworks require the software engineer to make unequivocal utilization of gain and discharge operation.

• Sluggish Discharge Consistency (LRC):

In Sluggish Discharge Consistency, the engendering of changes is further put off until the time of the obtain [10]. A framework in LRC needs to fulfill the accompanying conditions [11]:

- Before a customary read or compose access is permitted to perform regarding an alternate process, all past procure gets to must be performed with deference to that different procedure
- Before a discharge access is permitted to perform regarding some other procedure, all past normal read and store gets to must be performed with deference to that different methodology, and
- Sync are consecutively predictable concerning each other.

2.4. Passage Consistency

In passage consistency, information must be expressly announced accordingly in the project message, and connected with a synchronization protest that secures access to that imparted information. Entrance consistency exploits the relationship between particular synchronization variables which secure discriminating areas and the imparted information got to inside those basic segments. Forms must synchronize by means of framework supplied primitives. Synchronization operations are partitioned into secures what's more discharges. In the wake of finishing a secure, passage consistency guarantees that a procedure sees the latest rendition of the information connected with the obtained synchronization variable.

The above consistency models can be compressed and represented in the accompanying table:

Strictness	Consistency Models
More Strict	Sequential consistency
↓	Processor consistency
	Weak consistency
	Release consistency
Less Strict	Entry consistency

(Table1) Summary of Consistency

III. CONSISTENCY CONVENTIONS

Storing imparted information presents expands the framework execution in circulated imparted memory frameworks. Nonetheless, to keep up memory consistency, extraordinary outlined conventions required to be executed to

1) engender a recently composed worth to all reserved duplicates of the changed area,

2) discover when a compose is finished and

3) safeguard the atomicity for composes regarding different operations.

3.1. Compose Distributed Convention

The compose imparted convention cradles the compose gets to in this way permits numerous essayists upgrade simultaneously. Two or more essayists can adjust their nearby duplicates of the same imparted information in the meantime and the changed duplicates are consolidated in the following synchronization occasion.

The dispersed imparted memory programming at first compose secure the memory page containing the compose imparted information. At the point when some processor needs to change this page, appropriated imparted memory programming makes a duplicate of the page containing the compose imparted information and take off the compose insurance so further overhaul operations is possible without dispersed imparted memory programming intercession.

The first information page is placed in a deferred redesign line. At discharge time, the framework performs a correlation of the first page and its duplicate and run-length encodes the consequences of this distinction into the space distributed to the duplicate. Each one encoded overhaul comprises of a tally of indistinguishable words, the quantity of contrasting words that take after, and the information connected with those varying words. At that point each one hub that has a duplicate of an imparted question that has been adjusted is sent a rundown of the accessible overhauls. The getting hubs will then translate the upgrades and consolidation the progressions into their variant of the imparted information. This convention kills the sick impacts of false-imparting and subsequently brings down the correspondence connected with it.

3.2. Lazy Diff Creation Convention :

Fundamentally, LDC is indistinguishable to compose imparted convention is the feeling of make diffs for uniting further overhaul. The time of making diff in LDC is delayed until the adjustments are asked for, which contrasts from that of compose imparted convention. This altogether diminished the quantity of diffs made and enhanced execution.

3.3. . Eager Invalidate Protocol:

Excited conventions push adjustments to all hubs that store the information at synchronization variable discharges. In the event that remote duplicate is perused just, it is basically negated; if the duplicate is checked as readwrite, the remote hub adds the diff to the answer and afterward negates the page. At the point when the locking processor discharges its composes, all other storing hubs are told that they must negate their duplicates. The procurement inertness is long when lock appeal pending at discharge, short generally.

3.4. Lazy Invalidate Convention

In apathetic nullify, the proliferation of changes is postponed until the time of the procure.

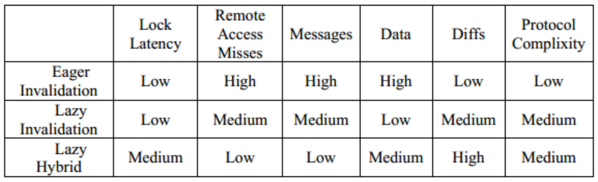
The releaser informs the acquirer, of which pages have been adjusted, bringing about the acquirer to discredit its neighborhood duplicates of these pages. A processor acquires a page blame on the first get to an discredited page, and gets diffs for that page from past releasers. The execution of each methodology is isolated into mostly requested interims, which is typically spoken to by timestamps.

Each time a methodology performs a discharge or an obtain, another interim starts. Neighborhood duplicates of pages for which a compose notice with a bigger timestamp is gotten are discredited. This convention has most brief lock obtaining dormancy (single message) when solicitation pending, additionally great when not pending.

3.5. Lazy Hybrid Protocol:

This convention is like apathetic nullify convention with the exception of that sluggish half breed upgrades a portion of the pages at the time of an obtain as opposed to nullifying the changed page. The releaser sends to the acquirer all the alterations that it feels that the acquirer is intrigued by. The acquirer negates pages for which compose notices were gotten however no changes were incorporated in the lock award message. Single pair of messages in the middle of acquirer and releaser, just have overhead head of piggybacks. Measure of information is more diminutive than for the overhaul convention. Lessened number of access misses.

The trade off between these conventions can be delineated in the accompanying table.



	Lock Latency	Remote Access Misses	Messages	Data	Diffs	Protocol Complixity
Eager Invalidation	Low	High	High	High	Low	Low
Lazy Invalidation	Low	Medium	Medium	Low	Medium	Medium
Lazy Hybrid	Medium	Low	Low	Medium	High	Medium

(Table 2) Protocol Trade off [8]

IV. DISCUSSION AND CONCLUSION

From the discussion above, we can find that distributed shared memory system gives an environment to simple programming and parallel processing. Yet the correspondence expense inherited in the underlying system is extremely lavish, hence restricts the adaptability of appropriated shared memory system and make different issues. Due to this trademark, the granularity of memory unit is limited in a certain reach to anticipate false-offering or inordinate message-passing. Be that as it may, the unbend ability of granularity has a negative impact on calculation speedup for some system with high correspondence necessity.

For the correlation of the framework plan, memory models and conventions utilized as a part of Treadmarks and Halfway, the result can be finished up as takes after:

• For the programming straightforwardness, Treadmarks needs no uncommon necessity while Halfway requires the developers to unequivocally relate a lock with an imparted information object.

• For compose recognition, Treadmarks framework needs to output the whole imparted information area, albeit just a little parcel of it may have been upgraded. In Halfway, framework just outputs the grimy bits of the imparted information object.

• Halfway just make those information connected with the lock predictable at a lock get stage. Conversely, Treadmarks needs to guarantee consistency for all information objects, which brings about less information being moved in Halfway than in Treadmarks.

• The evasion of TCP/IP convention stack harms the versatility of Treadmarks, particularly in the Web time where TCP/IP is a prevailing convention.

Clearly, there is no overwhelming framework between these two examined in the paper. Case in point, Section Consistency beats Languid Discharge Consistency on the off chance that its rationality unit is bigger than a page. In the event that Entrance Consistency's intelligibility unit is littler than a page, then Section Consistency beats Languid Discharge Consistency if there is a false-offering while Sluggish Discharge Consistency outflanks Passage Consistency if there is spatial region bringing about a prefetch impact. Accordingly, the decision of usage must be made as per the need of clients or developers and additionally different conditions.

In addition to algorithms, conventions and memory models, new system innovations may assume an imperative part in enhancing Appropriated Imparted Memory Frameworks proficiency since the correspondence expense is still the central point that influences framework.

REFERENCE

[1] John B. Carter, Dilip Khandekar, and Linus Kamb, Distributed Shared Memory: Where We Are and Where We Should Be Headed, the Fifth Workshop on Hot Topics in Operating Systems, May 1995.

[2] Kai Li and Paul Hudak, Memory Coherence in Shared Virtual Memory Systems, ACM Transactions on Computer Systems, Vol. 7, No. 4, November 1989

[3] Ajay Mohindra and Umakishore Ramachandran, A Comparative Study of Distributed Shared Memory Design Issues, GIT-CC-94/95, August 1994.

[4] Cristian Amza, Alan Cox, Karthick Rajamani, and Willy Zwaenepoel, Tradeoffs between False Sharing and Aggregation in Software Distributed Shared Memory, Proceedings of ACM SIGPLAN Conference on Principles and Practices of Computer Programming, 1997.

[5] B Nitzberg and V Lo, Distributed Shared Memory: A Survey of Issues and Algorithms, IEEE Computer August 1991, pp. 52-60.

[6] John Hennessy, Mark Heinrich and Anoop Gupta, Cache-Coherent Distributed Shared Memory: Perspectives on Its Development and Future Challenges, Proceedings of the IEEE, VOL. 87, No. 3, March 1999.

[7] Masaaki Mizuno, Michel Raynal, James Z. Zhou, Sequential Consistency in Distributed Systems, Proc. of the Int'l Workshop on Theory and Practice in Distributed Systems, October 1994.

[8] Pete Keleher, Alan L. Cox, Sandhya Dwarkadas, and Willy Zwaenepoel. An evaluation of software-based release consistent protocols. Journal of Parallel and Distributed Computing, 29(2):126--141, September 1995.

[9] Vijay Karamcheti, Architecture and Programming of Parallel Computers, Lecture 11, Future Directions Project presentations: December, 1998

[10] Pete Keleher, Sandhya Dwarkadas, Alan Cox, and Willy Zwaenepoel. Treadmarks: Distributed shared memory on standard workstations and operating systems. In Proceedings of the 1994 Winter Usenix Conference, pages 115 131, January 1994.

[11] Pete Keleher, Lazy Release Consistency for Distributed Shared Memory, PhD thesis of Rice University, Huston, Texas, January, 1995.

[12] A. Judge, P.A. Nixon, V.J. Cahill, B. Tangney, S. Weber, Overview of distributed shared memory, October, 1998.