A Memory Approach to Consistent, Reliable Distributed Shared Memory

Niels Christian Juul
Copenhagen Business School
Copenhagen, Denmark

Brett D. Fleisch
University of California
Riverside, CA, USA

Abstract

Fault-tolerant distributed shared memory systems do not always need to support a complete and consistent recovery *after a failure. We describe a framework, within which different approaches to, and different degrees of consistency and recoverability can be understood. The addition of consistent failure recovery may be approached from two different viewpoints: either by an application-oriented view or a memory-oriented view. The major characteristics used in our framework are variations of* availability, consistency, *and* application support.

This paper explains the basic model, which is used in Reliable Mirage+, *and describes how the framework can be used by other researchers to understand and classify solutions to the reliable DSM problem. The model distinguishes a* recoverable *system, which must be able to survive any* single-site failure, *from a* reliable *system which also ensures consistency after the recovery. Since* consistency requirements *may impose a high penalty on standard operational performance, various relaxed recoverability consistencies are described by the multi-level model. Recovery under this model may be accomplished by applications specifying consistency and availability requirements.*

1 Introduction

A serious problem for most *distributed shared memory* (DSM) systems is that as the size and number of participating sites grows, the possibility of a site failure increases. Our group at the University of California, Riverside has been investigating the issue of Reliable Distributed Shared memory from three different perspectives. First, we have developed a stochastic model of reliable DSM which is the basis for our analytical work in the area[27]. Second, we have developed a DSM reliability simulator to help analyze the effects of coherence protocols on reliability. Lastly, we are extending an experimental DSM system[7, 11] to incorporate changes to support reliability based on these approaches and the unifying framework presented in this paper[10]. This framework identifies existing and new solutions that make DSM systems robust to failure. We define two different approaches to reliable DSM systems and a multi-level model for the degree of reliability and consistency supported by each approach.

This paper focuses on concerns of making DSM systems reliable and suggests an approach that permits weaker consistency guarantees. In particular, when compared to sequential consistent DSM systems, weaker consistency is often motivated by improved performance. In a similar way, applications may benefit from subscribing to a lower level of robustness with limited consistency guarantees. For example, consider a singly linked list stored in DSM. Several processes attempt to enqueue and dequeue items on the linked list. The failure of any one site does not mean that consistent reliability is needed. If a process attempts to enqueue or dequeue an item which causes a page fault, it would be unacceptable for the memory to be inaccessible and other processes unable to complete their corresponding operations due to the site failure. Whereas recoverability is required, a strictly consistent reliable system with additional attendant overheads is not. As another example, consider a portion of the DSM address space acting as a blackboard. In this area of the address space, agents place and remove information as it is gathered from real-time sensors. If a site fails which is updating the blackboard with sensor data, it may not be necessary to recover the updates with a strictly reliable consistency scheme. Indeed, in this type of system, the most recent data gathered from the sensors should be placed on the blackboard. Therefore, as long as the blackboard memory is restored, other agents can place timely data on the blackboard and continue work.

2 Background

Previous DSM systems[16, 17, 2, 1, 23, 8, 21, 18] did not address the issue of reliability, mainly because their size and applications did not require it.

Often, the systems executed applications whose runtimes were short when compared to the mean time between failures of the system components. The development of more powerful networks and workstations makes DSM feasible for larger—and thus longer running—parallel programs. Thus, reliability is a key issue as DSM systems are scaled to larger configurations. To provide a reliable DSM system, the system must be able to cope with failures and continue to service applications effectively.

More recently, recoverability and consistency in reliable DSM system have been studied[14, 26, 28, 9, 24] including studies of specific systems like *Recoverable Virtual Memory* (RVM)[25, 5], *Munin*[4], *ickp*[22], *DiSOM*[20], and RELIABLE MIRAGE+[6, 11]. Almost all studies use, however, an application-oriented approach.

3 Failure Impact on DSM Systems

Many DSM systems share by coupling each application's private memory together with a library implementation of DSM. MIRAGE+ provides a memory that is potentially shared by all processes on a site and across sites, subject to protection constraints. Although both systems must be able to recover and to continue to operate, the failure impact for each can be quite different. The former need only cater to single affected applications (and their DSM), whereas the later must take a more holistic approach to DSM recovery. These later systems must affect recovery for *all* sharing applications affected.

A robust DSM system must be able to survive any *single-site failure*. There are two major areas where site failures cause problems for a DSM system. First, the system internals may be compromised by the loss of necessary global system state information. This includes part of the DSM directory state and part of the user data stored in DSM. Second, user applications may be compromised because state information from an application may be lost or rolled back inconsistently.

A *recoverable* DSM system must be able to recover all user and system data. It is, however, not always necessary to provide a consistent recovery for user data in the DSM and the application. A *reliable* DSM should support user applications that survive failures *consistently*. This additional support can be either transparent to the application or can be offered as a set of tools that can be used by applications. To implement *fully reliable applications*, the implementation of a reliable DSM must preserve a consistent picture of the shared memory (the global data) and provide mechanisms for the processes of a failed site to be recovered consistently (process state and private data). Furthermore, tools that ensure consistency with external data including input/output, e.g., by using *process checkpoints*, are needed.

In some DSM schemes, redundant copies of the shared memory are more easily obtained. A system with an *eager* or *competitive update* protocol[1, 13, 3, 12] has extra copies that could be used for recovery in most cases, whereas a system using a *lazy update* protocol would not have the same amount of up-to-date copies. Even competitive update protocols may be vulnerable to single copy instances. In *write-invalidate* systems[15, 19, 8, 7], the lack of multiple copies in the network is more probable since a write-invalidate DSM system designates one site to hold the write-access-right and data for parts of the global address space. The loss of that site introduces a gap in the global shared memory address space. A recoverable write-invalidate DSM system must ensure recovery from such a loss. In release and entry consistent systems, data in shared memory need not be lost when a failure occurs. Instead these systems must be able to recover from the loss of any currently held, unreleased locks at the failed site.

Memory access presents another problem: sites requesting shared memory expect to access the data eventually. There are two types of systems pertinent to the design of a reliable DSM system. In the first type of system, a user process requesting a page is blocked after requesting the page and that process is never resumed if a critical site fails. This is a *non-timeout page fault system*. In the second type of system, a user process requesting a page is blocked for a time and eventually re-requests the page if it is not received after a given time interval. This type of system is a *client-retry page fault system*. The assumption of whether clients will be allowed to retry their fault requests influences the design of the reliable DSM system considerably.

4 Reliability and Consistency Models

We have designed a framework to model the different solutions to achieve various degrees of robustness and consistency across failures in DSM systems. The framework is based upon two different approaches to robustness and consistency: A *memory-oriented* and an *application-oriented* approach. For both, a multi-level model with increasing constraints on consistency is presented. Whereas our research group has taken a memory-oriented design approach, most other researchers have taken the application-oriented approach.

The two approaches to robust DSM systems are

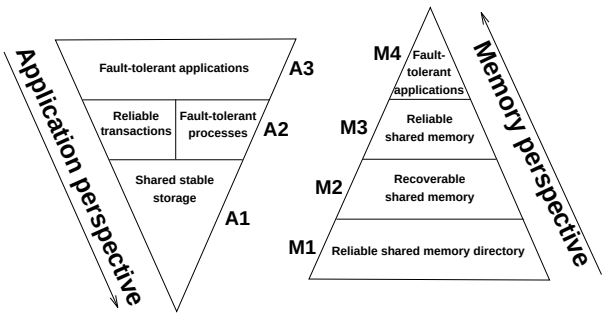

Figure 1: A Top-down or Bottom-up Approach to Reliability

outlined in Figure 1. Both are illustrated as multi-level models but their levels are asymmetric. The application approach starts from the top with the goal of running applications in a fault-tolerant manner(A3). That goal can be reached by using reliable transactions or individual, fault-tolerant processes (A2) combined with consistency requirements. In turn, reliable transactions and fault-tolerant processes are built using a consistent shared stable storage (A1). In contrast, the memory approach starts from the bottom with a reliable directory service for shared memory (M1). At the second level, data recovery is added to achieve a recoverable shared memory (M2). With additional consistency mechanisms, reliable shared memory (M3) is achieved at level three. To provide fault-tolerance at the application level (M4), recovery of the state of the distributed application stored outside the shared memory must also be provided consistently.

4.1 The Memory-Oriented Approach

The memory-oriented approach to robust DSM does not require applications to support fault-tolerance. Hence, conservative estimates of the need to checkpoint data consistently are made by the DSM system.

Level M1 robustness is the basic fault-tolerance which is always provided. It ensures a *sound DSM system*, but consistency of user data is not ensured. Recovery of the user data in shared memory is not provided although the integrity of shared address space is preserved. To provide this level of service, a *reliable directory service* is needed.

Level M2 robustness extends the basic fault-tolerance with guarantees of recovery of the shared memory without consistency guarantees for shared data. A page-oriented DSM system with this level provides recovery *of each page* of the DSM individually. Although all data is recovered at this level, the availability requirements of the DSM system define whether the recovery may be done *lazily*, or *eagerly*. Eager recovery shortens the time where the system is vulnerable to a second failure at the cost of added recovery latency.

Level M3 robustness adds consistency requirements to the shared data. Many variants are possible depending on the degree of consistency required. For instance, *consistency for a collection of user data* in the shared memory from level M2 may be enforced by requiring a page-oriented DSM to recover a number of pages up to all pages, i.e., *the entire shared memory space*, consistently. Thus, this level could potentially guarantee a fully consistent recovery of the entire user DSM address space.

Level M4 robustness extends a fully consistent recovery of the user data in DSM by adding recovery and consistency of application processes. The DSM system may provide upcalls indicating when applications are to checkpoint or perform checkpoints on behalf of the applications. If the application is based on general facilities for process recovery, these facilities can be tailored to cooperate with DSM upcalls. Consistency between the shared memory and the distributed user application provides *fault-tolerance for applications*.

4.2 The Application-Oriented Approach

The application-oriented approach is also described by a set of layers. Although it usually considered a top-down approach, the following description is outlined bottom-up as with the memory approach.

Level A1 robustness provides a *reliable shared stable storage* in which individual checkpoints and logs of various processes or transactions are placed. The storage is guaranteed to survive a site failure and to be available even if the site remains unavailable. The individual checkpoints are usually thought of as internally consistent, whereas additional mechanisms are needed to ensure consistency across multiple checkpoints which together constitute a global state. A reliable shared stable store is often provided by replicated filesystems or dual-ported disks.

Level A2 robustness provides *reliable atomic actions*. These may be described from a transactional or process viewpoint:

- *reliable transactions* ensure that a transaction is either done or undone at commit and that consistency is maintained in both cases.
- *process reliability* ensures that a process and its local state is maintaining checkpoints and able to roll back and replay from the last checkpoint.

This level provides consistency within each transaction/process but no global consistency.

Level A3 robustness built on level A2 and ensures consistency across multiple transactions or processes after recovery to ensure a *fault-tolerant application*. This enables a distributed application to recover a memory consistently after a site-failure and its processes to resume consistently. A trend has been to piggyback support for this level on messages associated with the DSM protocol.

5 Summary and Conclusions

RELIABLE MIRAGE+ is currently operational and provides robustness at levels M1 and M2. MIRAGE+ is based upon a distributed directory service for shared memory pages grouped in segments. Each segment has a *library* and each page has a current primary copy at its *clock* site. Backup copies are stored at its *trailer* site(s). Given a library failure for any segment, a new library site is elected and all sites having knowledge about that segment send their directory state to the new library site. Given a clock site failure for any page, the trailer site is upgraded to be the clock site. This provides the basis for level M2 robustness. In our current implementation, if we lose a page, it is replaced lazily. In the future, we plan to be aggressive and to instrument the cost of this change in policy. We have designed the level M3 robustness from a memory perspective and plan to use upcall mechanisms to achieve level M4.

Our implementation has been tested with a variety of failure scenarios. We introduce artificially generated site failures and code failures (panics) to test the reliability and resiliency of the DSM system. The current implementation is robust and handles failures well.

Our design for RELIABLE MIRAGE+ uses the memory-oriented perspective. One reason for using this approach is that we have much more autonomy in providing consistency as memory system designers as compared to application-oriented fault-tolerance designers. In the latter approach, application processes will affect the memory and may often be required to roll back.[1] These may require cascading rollbacks of other processes. Although the rollbacks may cascade, a point may be reached where a rollback is required for a process over which an application-oriented designer may not have autonomy. Application-oriented robustness designers must provide hooks to the memory system to enable consistency when privileged autonomous cooperating applications are used. Our approach provides a stronger consistency guarantee because the memory system has autonomy over all of the state of all memory at all sites. Those processes over which we do not have autonomy simply ignore our upcalls.

We believe a memory-oriented approach to consistent recovery is the most promising for reliable DSM. After all, it is the memory which the consistency constraints are aimed at.

Acknowledgements

This research was sponsored by NSF CCR-9209405, DEC External Research Program, and the Danish Research Council of Natural Science, SNF J.nr. 11-0484.

References

[1] J. K. Bennett, J. B. Carter, and W. Zwaenepoel. Munin: Distributed shared memory based on type-specific memory coherence. In *Proceedings of the 1990 Conf. Principles and Practice of Parallel Programming*, pages 168–176, New York, NY, USA, 1990. ACM Press.

[2] R. Bisiani and M. Ravishankar. PLUS: A distributed shared-memory system. Technical report, School of Computer Science, Carnegie Mellon University, 1990.

[3] S. Dwarkadas, P. Keleher, A. L. Cox, and W. Zwaenepoel. Evaluation of release consistent software distributed shared memory on emerging network technology. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 244–255, May 1993.

[4] E. N. Ehnozahy, D. B. Johnson, and W. Zwaenepoel. The performance of consistent checkpointing. In *Proceedings of the* 11^{th} *Symposium on Reliable Distributed Systems*, pages 39–47, Houston, Texas, USA, 1992. IEEE-CS Press.

[5] M. J. Feeley, J. S. Chase, V. R. Narasayya, and H. M. Levy. Integrating coherency and recoverability in distributed systems. In *Proceedings of the First Symposium on Operating Systems Design and Implementa-*

[1] Transactional approaches may vary with respect to the degree they suffer from this problem.

tion (OSDI'94), pages 215–227, Monterey, CA, USA, Nov. 1994. USENIX.

[6] B. D. Fleisch. Reliable Distributed Shared Memory. In *Proceedings of the Second IEEE Workshop on Experimental Distributed Systems*, pages 102–105, Huntsville, AL, USA, 1990. IEEE-CS.

[7] B. D. Fleisch, R. L. Hyde, and N. C. Juul. MIRAGE+: A kernel implementation of distributed shared memory on a network of personal computers. *Software—Practice & Experience*, 24(10):887–909, Oct. 1994.

[8] B. D. Fleisch and G. J. Popek. Mirage: A coherent distributed shared memory design. In Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, published in *Operating Systems Review* 23(5) Special Issue, pages 211–223, The Wigwam, Litchfield Park, Arizona, USA, Dec. 1989. ACM Press.

[9] B. Janssens and W. K. Fuchs. Relaxing consistency in recoverable distributed shared memory. In *Proceedings of the Twenty-Third Annual International Symposium on Fault-Tolerant Computing: Digest of Papers*, pages 155–163, June 1994.

[10] N. C. Juul and B. D. Fleisch. A framework for consistency and recoverability in distributed shared memory. Mirage Research Note 10, Department of Computer Science, University of California, Riverside, CA, Feb. 1995. Submitted for publication.

[11] N. C. Juul, B. D. Fleisch, and C. DeMatteis. Reliable Distributed Shared Memory. Mirage Research Note 7, Department of Computer Science, University of California, Riverside, CA, Dec. 1994. Unpublished research note on the design of RELIABLE MIRAGE+ with multiple consistency levels.

[12] A. R. Karlin. Competitive snooping caching. In *Proceedings of the 27th Annual Symposium on the Foundations of Computer Science*, pages 244–254, Oct. 1986.

[13] P. Keleher, S. Dwarkadas, A. Cox, and W. Zwaenepoel. Treadmarks: Distributed shared memory on standard workstations and operating systems. Technical Report COMP TR93-214, Department of Computer Science, Rice University, Houston, Texas, USA, Nov. 1993.

[14] R. Koo and S. Toueg. Checkpointing and roolback-recovery for distributed systems. *IEEE Trans. Softw. Eng.*, SE-13(1):23–31, Jan. 1987.

[15] K. Li. IVY: A shared virtual memory system for parallel computing. In *Proceedings 1988 International Conference on Parallel Processing*, volume 2, pages 94–101, Aug. 1988.

[16] K. Li and P. Hudak. Memory coherence in shared virtual memory systems. In *Proceedings 5th ACM SIGACT-SIGOPS Symposium of Principles of Distributed Computing*, pages 229–239, Canada, Aug. 1986. ACM Press.

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

[18] V. Lo. Operating systems implementations of distributed shared memory. *Advances in Computers*, 39, 1994.

[19] S. Möding. Distributed Shared Memory für das PANDA-Laufzeitsystem. Fachbereich Informatik, Universität Kaiserslautern, Germany, May 1993. (in German).

[20] N. Neves, M. Castro, and P. Guedes. A checkpoint protocol for an entry consistent shared memory system. In *Proceedings of the 13th ACM Symposium on Principles of Distributed Computing (PODC'94)*. ACM Press, Aug. 1994.

[21] B. Nitzberg and V. Lo. Distributed shared memory: A survey of issues and algorithms. *IEEE Computer*, 24(8):52–60, Aug. 1991.

[22] J. S. Plank and K. Li. ickp: A consistent checkpointer for multicomputers. *IEEE Parallel & Distributed Technology*, 2(2):62–67, Summer 1994.

[23] U. Ramachandran, M. Ahamad, and M. Y. A. Khalidi. Unifying synchronization and data transfer in maintaining coherence of distributed shared memory. Technical Report GIT-ICS-88/23, Georgia Institute of Technology, Atlanta, GA, USA, June 1988.

[24] G. G. Richard, III and M. Singhal. Using logging and asynchronous checkpointing to implement recoverable distributed shared memory. In *Proceedings of the* 12^{th} *Symposium on Reliable Distributed Systems*, pages 86–95, Princeton, New Jersey, USA, Oct. 1993. IEEE-CS Press.

[25] M. Satyanarayanan, H. H. Mashburn, P. Kumar, D. C. Steere, and J. J. Kistler. Lightweight recoverable virtual memory. *ACM Transactions on Computer Systems*, 12(1):33–57, Feb. 1994.

[26] M. Stumm and S. Zhou. Fault tolerant distributed shared memory algorithms. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, pages 719–724. IEEE Press, Dec. 1990.

[27] O. E. Theel and B. D. Fleisch. Design and Analysis of Highly Available and Scalable Coherence Protocols for Distributed Shared Memory Systems based on Stochastic Modeling. Submitted for publication, also available as Technical Report THD-BS-1995-02, University of Darmstadt, Department of Computer Science, Institute for System Architecture, Germany, Jan. 1995.

[28] K.-L. Wu and W. K. Fuchs. Recoverable Distributed Shared Virtual Memory. *IEEE Trans. Comput.*, 39(4):460–469, Apr. 1990.