Designing a Distributed Software Transactional Memory System

Christos Kotselidis[*,1], Mohammad Ansari[*,1], Kim Jarvis[*,1], Mikel Lujan[*,1], Chris Kirkham[*,1], Ian Watson[*,1]

[*] *School of Computer Science, The University of Manchester, Oxford Road, M13 9PL, Manchester*

ABSTRACT

Distributed systems are widely used in high performance computing for performing computationally extensive calculations. In programming terms, distributed systems have usually been exploited through the use of dedicated APIs such as OpenMP and MPI. With these programming models, synchronization is achieved using locks and barriers, and is complex to implement correctly. Transactional memory is a new promising parallel programming model that aims to replace conventional locking mechanisms with transactions. Major transactional memory research has focused on Chip MultiProcessors(CMPs), leaving the area of distributed systems unexplored. In this paper a prototype distributed software transactional memory framework is described.

KEYWORDS: Transactional Memory; Distributed Systems; Clustering; Multithreading

1 Introduction

Parallel programming for distributed systems is a difficult and challenging task. With the conventional parallel programming models and languages, the programmer must ensure memory consistency via explicit coordination and synchronization. This explicit synchronization is achieved by the use of locks, semaphores, barriers etc. Parallel and distributed programming has been the domain, until now, of a small number of specialists who build tailor-made custom applications. The introduction of multicore architectures [OH05], however, necessitates the creation of new parallel programming models that abstract away from the user the error-prone complexities of explicit synchronization.

Transactional memory (TM) is an alternative paradigm to lock-based concurrent programming. Derived from transactional databases, TM uses transactional semantics for critical code regions that require synchronization. Programmers utilizing TM just have to enclose segments of code that access shared variables in transactions. Consequently, the TM system guarantees the atomicity, consistency and isolation of executing critical regions. If

[1]E-mail: {kotselidis,ansari,jarvis,lujan,chris,watson}@cs.manchester.ac.uk

successfully executed (i.e. no conflicts detected), transactions “commit”. In the presence of a conflict, a contention manager is consulted in order to resolve the conflict. After conflict resolution, a single conflicting transaction will continue execution, while the remaining conflicting ones will be “aborted”.

A number of hardware [NCW+07] and software [HLMWNS03, HF03] TM systems have been developed testing different approaches. Researchers have not yet concluded on a universal TM approach and attention is focused on establishing high performance TM systems.

This paper describes the preliminary design and implemenation of a distributed software transactional system. The system focuses on exploiting transactional memory on clusters with distributed memory (i.e. no underlying shared memory system).

2 Motivation

The majority of the systems developed examine TM in the domain of CMPs. However, substantial research has also been conducted in order to exploit TM on clusters [MMA06]. While in CMPs we have a single shared memory, in clusters, if not supported by underlying mechanisms (DSM [KDCZ94]), the memory is distributed among the processors. The presence of non-shared memory creates new research challenges. Memory coherence and consistency must be preserved by updating remote objects, which involves network delays. Hence, efficient consistency models must be designed in order to minimize expensive network communications. Furthemore, efficient contention management algorithms must be employed in order to resolve local and remote conflicts.

3 Preliminary Design

Concerning the memory model, the transactional coherence and consistency (TCC) [HWC+04] model is adopted. According to TCC, when a transaction attempts to “commit” it broadcasts its modification set to the system in order to detect any conflicting transactions. The fact that in TCC transactions broadcast their write set in one single packet and only once (before they commit) makes it suitable for clusters as it minimizes network communications.

In the case of a direct update TM system, updates must be redirected to the master node where conflict detection takes place. The contention manager resolves conflicts and “aborted” transactions are re-executed. Although easier to implement, a significant bottleneck will be created on the master node, degrading the system’s performance and scalability.

In the case of a deferred update TM system, remote nodes can maintain local copies of the global data. Distributed systems offer the luxury of maintaining copies of data since there is an abundance of physical memory. Upon distributed “commit”, if transactions modify remote objects, they broadcast their changes to the other nodes and wait for an acknowledgement in order to commit. If a directory of the remote objects does not exist then a transaction that attempts to commit must broadcast to all remaining nodes. If a directory does exist, the committing transaction should broadcast to the nodes that keep a copy of the modified objects. If the transaction commits, remote nodes update their local copies in order to be consistent with the modified global objects.

The logic of the deferred update TMs is adopted in order to benefit from the amount of physical memory available. Subjects such as communication protocols, contention manage-

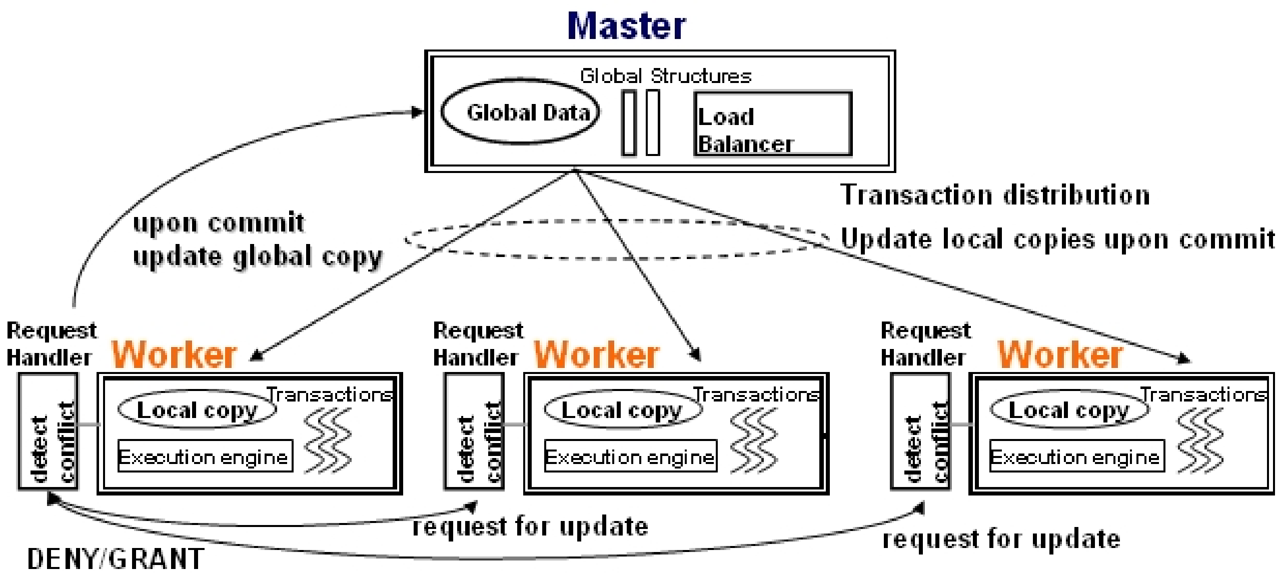

Figure 1: Design of Distributed STM system

ment and concurrency control are ongoing research. A preliminary snapshot of the implemented system is depicted in Figure 1.

4 Experimental Setup

The distributed software transactional memory system implemented uses the ProActive grid middleware framework [BBC+06]. Transactions are wrapped in active objects and distributed across the nodes. The transaction execution engine is a modified version of DSTM2 [HLM06]. Experiments will be carried out on a 4-way node cluster utilizing in total 40 Opterons 2.4Ghz cores. The boards of the cluster are connected to each other via a Gigabit ethernet switch.

References

[BBC+06] Laurent Baduel, Françoise Baude, Denis Caromel, Arnaud Contes, Fabrice Huet, Matthieu Morel, and Romain Quilici. *Grid Computing: Software Environments and Tools*, chapter Programming, Deploying, Composing, for the Grid. Springer-Verlag, January 2006.

[HF03] Tim Harris and Keir Fraser. Language support for lightweight transactions. In *OOPSLA '03: Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications*, pages 388–402, New York, NY, USA, 2003. ACM Press.

[HLM06] Maurice Herlihy, Victor Luchangco, and Mark Moir. A flexible framework for implementing software transactional memory. In *OOPSLA*, pages 253–262, 2006.

[HLMWNS03] Maurice Herlihy, Victor Luchangco, Mark Moir, and III William N. Scherer. Software transactional memory for dynamic-sized data structures. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 92–101, New York, NY, USA, 2003. ACM Press.

[HWC+04] Lance Hammond, Vicky Wong, Mike Chen, Brian D. Carlstrom, John D. Davis, Ben Hertzberg, Manohar K. Prabhu, Honggo Wijaya, Christos Kozyrakis, and Kunle Olukotun. Transactional memory coherence and consistency. *SIGARCH Comput. Archit. News*, 32(2):102, 2004.

[KDCZ94] P. Keleher, S. Dwarkadas, A. L. Cox, and W. Zwaenepoel. Treadmarks: Distributed shared memory on standard workstations and operating systems. In *Proc. of the Winter 1994 USENIX Conference*, pages 115–131, 1994.

[MMA06] Kaloian Manassiev, Madalin Mihailescu, and Cristiana Amza. Exploiting distributed version concurrency in a transactional memory cluster. In *PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 198–208, New York, NY, USA, 2006. ACM Press.

[NCW+07] Njuguna Njoroge, Jared Casper, Sewook Wee, Teslyar Yuriy, Daxia Ge, Christos Kozyrakis, , and Kunle Olukotun. Atlas: A chip-multiprocessor with transactional memory support. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE), Nice, France, April 2007*, 2007.

[OH05] Kunle Olukotun and Lance Hammond. The future of microprocessors. *Queue*, 3(7):26–29, 2005.