Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 9, Number 1 (2016) pp. 37-40
© Research India Publications
http://www.ripublication.com

Design Space Exploration of Transactional Memory for efficient Computing in Emerging Concurrent System

Suresh.P[1] and Bharathi V Kalmath[2]

*[1,2] Department of Computer Science and Engineering,
Sri Venkateshwara College of Engineering, Bengaluru, India.
E-mail: [1] suresh.rvce@gmail.com, [2] sonubharathi@yahoo.com*

Abstract

With the advent of new technology, the data to be processed A data which is shared when processing, will undergo locking mechanism so that the data can be can be accessed by only one process. The data to be shared by different process will encounter the problem of mutual exclusion. To overcome the problem of mutex, the mechanism has to be done known as Transactional memory. As and when the data is required the data will be considered and the other data will be free from the locks. The efficient mechanism has been done in this paper to address the data when it is shared by the multiprocessors. The problem how the job has to be scheduled for different processor and how the data will be shared efficiently addressed and paper also discuss the problem of data synchronization which will access the data fast without the modification of the shared data and the appropriate data will be given to the appropriate process.

Keywords: Hardware Transactional Memory, Software Transactional Memory, Multiprocessor, Memory Management, Synchronization, locks, TxLinux, Multicores

INTRODUCTION

The Engineers have successfully worked for decades to improve single thread CPU performance, but we have now reached a peak in what a single thread can do. Therefore Thread Level Parallelism has become the need-of-the-hour especially to exploit the performance potential of Multicore Systems. Unfortunately, writing parallel programs is hard because synchronizing accesses to shared state is complex and error-prone. Many techniques have been experimented, however achieving performance and correctness simultaneously still requires expert programmers, and

the method of choice is decade old concept of "*locks*". Transactional Memory(TM) is a relatively new programming paradigm promising an easier road to correctness and performance using atomic code regions. These regions may then be speculatively executed in parallel, potentially providing performance gains.
TM is a memory management mechanism for distributed concurrent system that includes multi-threaded and multi-core systems, which are mainly transaction based in nature. The concept underlying in TM is eliminating the need of locks and semaphores for synchronization. In a multi-processor systems, thus eliminating the problems due to locks and semaphores causing priority Inversion, Convoying and Deadlocks. Many different transactional systems were proposed such as Speculative Lock Elision [1], where it suffers from the problems due to lock synchronization [2].The above problem can be solved by using Transactional Lock Free Execution, where the shared objects are never locked.
In this Scenario wherein priority based threads are used, there can be problems like thread contention and indefinite wait, that eventually may lead to deadlock. This problem can be resolved by approaches like Nested Parallelism [3], which uses shared memory and message passing constructs to exploit loop level parallelism [4].
The above specified lock based Synchronization problems can be resolved by Software Transactional Memory (STM) and Hardware Transactional Memory (HTM), of which STM is found to be the promising one. With the HTM approach the Scheduler is modified to overcome the problem of Scheduling faced with STM. Extensive research has been carried out on STM where most of them concentrated on program performance, leaving other metrics like space, task Scheduling, energy and throughput in the form of transaction per second.
Alternatively, to resolve the above problems the researcher proposed the Hybrid TM[5]. In Hybrid TM multiple instructions executing with complex operations will consume more latency, which is bounded in nature. The first Operating System TxLinux has been developed [6] which uses HTM as a Synchronization Primitive. The implementation of HTM suffers from overheads due to scheduling as there are conflicts possible in HTM primitives. Further research has been carried out to incorporate eager and lazy Transaction management. TM in the whole is a research toy [7]. There is open research issues like transaction prioritization, performance optimization and management in multicores etc, which will be addressed in the research work proposed.

MOTIVATION

From the Research we found that the conflict management is a key design dimension of HTM system. Implementation of efficient mechanism for detection and resolution becomes critical when conflicts are not a rare event.
Research has to be carried out in such a way that the transactions will continue even if the conflicts occur. Further works are proposed to make correct code work quicker, who want their multithreaded program to run fast will still need to tune their codes to minimize the amount of transaction conflicts[8][9].
While there has been much emphasis in STM bases system in the literature, little attention was found to be given to other important design metrics such as

space/memory, task Scheduling, throughput, conflict management and energy. Put together the STM and HTM are still in the initial stages of research, exploring further in this direction involves considerable design aspects such as conflict management, task scheduling, prioritization, space optimization, eager management and energy will find scope for wide variety of transaction based system including multicores, multithreaded and multiprocessors and other kind of Transaction based systems.
Existing approaches suffer from severe drawbacks due to limited or lack of the entirety of the available design space Exploration. Towards this end will help come up with better approaches for mapping of application to architecture with proper OS Support.

GOALS

Key Objective of the proposed research work is to carry out a Design Space Exploration of transactional memory for efficient computing in emerging concurrent system.
As a first step, by way of thorough survey of the literature and state of art Technologies. A trade-off between TM, key design parameters will be obtained. We begin with hardware implementation of TM. We then study qualitatively and quantitatively the large design space for hardware TM as defined by primary options such As version management and conflict detection, and vary the secondary options such as the structure of the memory hierarchy, the instructions per cycle, and the configuration of the interconnect. We will determine the semantics and interfaces needed by any hardware TM system to support rich software functionality in modern operating systems and programming languages. Finally, we will extend hardware support for transactional execution to create a multi-core architecture that provides cache coherence and memory consistency at the granularity of atomic transactions.
Transactional architecture implementing lazy versioning and optimistic conflict detection is the preferred method of implementing TM in software due to its simplicity and good performance across a wide range of contention scenarios. Also, to support rich semantics, four mechanisms are added: two-phase transaction commits software handlers, nested transactions, and non transactional loads and stores. Finally, a continuously transactional architecture called Transactional Coherence and Consistency(TCC) maintains performance benefits while simplifying the hardware and software implementation of TM. Experiments for design space exploration will be arrived by incorporating STM and TCC mechanisms in software simulator developed or such as OpenMP, IBM IXL compilers, JVM compilers, MicSoc and other similar tools.

KEY RESEARCH CONTRIBUTIONS

Survey of existing transaction based systems, Transactional memory and state of art the art Technologies and arrived at design parameters.

- Characterization and modeling of Transactions in Multicore, Multithread, Database, Data mining and Mobile Computing using the different parameters.

- Analysis of the Characterization obtained and developing a Transaction based system.
- Simulation of Transaction processing scenarios on the TM system developed and experimentation using Benchmark Applications.
- Verification of the system and analysis of results.

REFERENCES

[1] Ravi Rajwar and James R. Goodman, “Speculative Lock Elision: Enabling Highly Concurrent Multithreaded Execution”, Appears in the proceedings of the 34th International Symposium on Microarchitecture (MICRO), Dec. 3-Dec. 5, 2001, Austin, Texas.

[2] Nir Shavit and Dan Touitou. “Software transactional memory. distributed Computing”, 10(2):99–116, 1997.

[3] Kunal Agarwal, Jeremy T Fineman, Jim Sukha, “Nested Parallelism in Transactional Memory” ACM SIGPLAN workshop on Transaction Computing. Transact 2007

[4] Woongi Baek, Nathan Bronson, Christos Kozyrakis, Kunle “Implementing and Evaluating Nested Parallel Transactions in Software Transactional Memory”, Olukotun, SPAA’10, June 13-15, 2010.

[5] Peter Damron, Alexandra Fedorova, Yossi Lev, “Hybrid Transactional Memory” ASPLOS October 21–25, 2006, San Jose, California, USA.ACM 1-59593-451- 0/06/0010.

[6] Christopher J. Ross Bach, Hany E. Ramadan, Owen S. Hofmann, Donald E. Porter,Aditya Bhandari and Emmett Witchel “TxLinux and MetaTM: Transactional Memory and the Operating System”, 2009.

[7] Calin Cascaval, Colin Blundell, Maged Michael, Harold W. Cain, Peng Wu, Stefanie Chiras, and Siddhartha Chatterjee. “Software transactional memory: why is it only a research toy?”, Commun. ACM, 51(11):40{46, 2008.

[8] Victor Luchangco, Maurice Herlihy, Mark Moir,“A Flexible Framework for Implementing Software Transactional Memory”, 2007

[9] M. Herlihy and J.E.B. Moss. Transactional Memory: Architectural Support for Lock- *Free Data Structures*. In the 20th Intl. Symposium on Computer Architecture (ISCA), May 1993.