ASeD: Availability, Security, and Debugging Support using Transactional Memory

JaeWoong Chung, Woongki Baek, Nathan Grasso Bronson, Jiwon Seo
Christos Kozyrakis, Kunle Olukotun
Computer Systems Laboratory
Stanford University
{jwchung, wkbaek, nbronson, jiwon, kozyraki, kunle}@stanford.edu

ABSTRACT

We propose ASeD that uses the hardware resources of transactional memory systems for non transactional memory purpose. We show that the hardware components for register checkpointing, data versioning, and conflict detection can be reused as basic building blocks for reliability, security, and debugging support.

Categories and Subject Descriptors: C.1.4 [Processor Architecture]: Parallel Architecture

General Terms: Reliability, Security, Design

Keywords: Transactional Memory, Reliability, Security, Debugging

1. INTRODUCTION

The turn toward chip-multiprocessors (CMPs) has motivated research on architectural support for practical parallel programming. *Transactional Memory (TM)* is emerging as a promising technology in this area [1]. Hardware TM (HTM) systems provide mechanisms to manage multiple versions of data and detect conflicts between concurrent transactions. [1, 2]. With the help of software, these mechanisms can provide the three necessary features for transactional execution: *atomicity* (the ability to roll back to a safe system state), *consistency* (the ability to enforce system-level invariants such as execution order), and *isolation* (the ability to limit the propagation of side effects).

2. ASED

The main thesis of this work is that atomicity, consistency, and isolation (ACI) can be used to address significant system challenges beyond concurrency control. Specifically, we use them to improve *availability*, *security*, and ease of *debugging*. For many applications, these three metrics are equally important as raw performance. In the past few years, there have been several architectural efforts to develop techniques that independently provide support for availability, security, or debugging. While they are elegant and effective schemes, throwing them all into a system creates a hodge podge that is difficult to use and expensive to support in terms of hardware resource. In this paper, we provide an *integrated* hardware design for parallel programming, availability, security, and debugging, First, instead of providing an end-to-end hardware solution for each issue, we provide a core set of hardware primitives that system vendors can use *flexibly* and combine with complementary software mechanisms. Second, to provide availability, security, and debugging support in a *cost-effective* manner, we generalize the proposed support for atomicity, consistency, isolation in HTM systems and reuse them to also support availability, security, and debugging features.

This paper presents *ASeD*, an integrated design that provides architectural support for **a**vailability, **se**curity, and **d**ebugging within HTM systems. For availability, ASeD supports local and global recovery schemes that can deal with permanent loss of cores and caches and transient faults in communication or computation. For security, ASeD provides fine-grain canaries and read/write barriers without significant performance overhead. It also supports isolated execution that can limit the effects of suspicious code or data until its safety is verified. For debugging, ASeD supports an arbitrary number of hardware watchpoints, the ability to simultaneously bookmark all threads in a program, and the ability to step all threads forward or backward in time. These features are available to both existing programs (parallel or sequential) and new parallel programs that take advantage of memory transactions.

The mapping between the ASeD features and the hardware resources in transactional memory systems is done in two steps. First, instead of trying to implement each feature as a piecemeal solution, we reduce the features to four basic primitives : thread-wide isolated execution, system-wide global checkpoint, fine-grain address protection, and user-level software handler. This approach allows us to lower the hardware requirement on underlying transactional memory systems without losing much the potential performance benefits from the hardware resources by accelerating the common cases in supporting the features. Second, we implement the primitives using the hardware primitives of transactional memory. Isolated execution is supported by transaction. Global checkpoint requires additional hardware to record the values that belong to the last checkpoint. Fine-grain address protection uses the transaction conflict detection mechanism. Software handler is based on the event handlers such as commit handler and abort handler.

Our evaluation results show that ASeD successfully reuses the hardware resources in transactional memory to provide reliability, security, debugging support at a reasonable additional hardware cost.

3. REFERENCES

[1] L. Hammond, V. Wong, et al. Transactional Memory Coherence and Consistency. In *the Proc. of the 31st Intl. Symp. on Computer Architecture (ISCA)*, Munich, Germany, June 2004.

[2] K. E. Moore, J. Bobba, et al. LogTM: Log-Based Transactional Memory. In *the Proc. of the 12th Intl. Conf. on High-Performance Computer Architecture*, Feb. 2006.