DIMM: Architectural Support for Data Isolation and Memory Monitoring

Arrvindh Shriraman
ashriram@cs.rochester.edu, 4th year graduate student
Department of Computer Science, University of Rochester
Advisor: Sandhya Dwarkadas

1 Introduction

In recent years, multi-core and many-core processors based on the shared-memory model have become increasingly common and appear to be the choice of hardware platform for future systems. Typical software kernels load all extensible modules into a single address space for fast shared memory based communication. This improves the performance but compromises the safety and reliability aspects. The naive shared memory paradigm is partially at fault; it makes it very expensive to defend against memory corruption since hardware coherence and value propagation is almost completely transparent to software. Ideally the underlying system would sandbox the module, locally exposing an untrusted module's updates, but hiding it from the core kernel address space until the system approves.

With the influx of multi-core processors, "Parallel Programming" is another challenge that programmers have to contend with. There is a general consensus that writing multi-threaded code with all the synchronization presents a significant challenge for the average programmer. The transactional memory (TM) programming paradigm hopes to ease some of these concerns and there is significant pressure on hardware vendors to include support components in future processors. Our research develops generic hardware support that seeks to support these seemingly unrelated requirements from the TM, security, reliability, and debugging application domains. We propose decoupled hardware mechanisms for (1) memory monitoring and (2) data isolation. Our thesis is that,

Hardware primitives for memory monitoring and data isolation will facilitate wide-spread usage of shared memory multiprocessors via enhanced concurrent programming models. It is possible to export these primitives to software in a manner that enables flexible use of hardware support for multiple purposes.

We informally define these terms below.

Data Isolation allows software to control the visibility of a write in a shared memory system, i.e., execute a write operation, make it visible to the local thread, but hide the written value from remote threads until some commit point in the future.

Memory Monitoring provides support for (1) summarizing the read/write accesses of the program and (2) event-based notification of remote memory operations on special addresses marked by software.

2 Work in Progress

Our proposal explores lightweight, generic, fine-grained support for DIMM (Data Isolation and Memory Monitoring).

Phase 1: Bounded DIMM- Our first implementation of the hardware support was under the umbrella of the "RTM" transactional memory (TM) project [2]. We developed architectural mechanisms for bounded capacity DIMM, (1) Alert-on-update (AOU) marks the L1 cache lines and leverages cache coherence for event notification, and (2) Programmable data isolation (PDI) allows the L1 cache to hold speculative data until software decides to make them visible. These components provided acceleration for a hardware-software based TM system. We also began exploring API extensions that would allow use in non-transactional applications (e.g., security, debugging).

Phase 2: Unbounded DIMM- The results from the RTM project seemed to indicate that software metadata management required to handle L1 overflows is a significant performance overhead. We extended the DIMM components to target these overheads by adding (1) a hardware bloom-filter based signature that can represent an unbounded number of locations for monitoring, and (2) an overflow buffer (in virtual memory) filled by hardware to support cache overflows. We also included a new hardware event-logger, Conflict Table, which noted the processors from which accesses were made to monitored locations. These DIMM components aided in constructing (1) a high performance transactional memory system that could virtualize effectively [1] and (2) a memory bug detector tool [1].

Phase 3: Using DIMM hardware- Most recently, our research has focused on employing DIMM components to provide fine-grain monitoring and isolation support for applications in the reliability and security domain. We are employing DIMM to checkpoint, bound resource usage, and buffer memory state of untrusted/unverified modules to allow easy safe reclamation (e.g., when killing a thread). In the past these requirements have been met using the large-granularity OS process paradigm that makes it difficult to share state through memory and introduces significant performance penalty.

3 Summary

The success of multi-core chips depends strongly on convincing the software vendors of their value. Our research work has defined and developed two generic hardware primitives, memory monitoring and data isolation, which we believe software developers will be able to deploy across different application domains (e.g., security, reliability, programming models). Decoupling the DIMM components and giving each a well defined API will also make it easier for hardware vendors to implement and refine them. We think ASPLOS with its confluence of software and hardware developers is the right platform to advertise our research.

References

[1] Arrvindh Shriraman, Sandhya Dwarkadas, and Michael L. Scott. Flexible Decoupled Transactional Memory Support. Technical Report TR 910, Department of Computer Science, University of Rochester, December 2006.

[2] Arrvindh Shriraman, Michael F. Spear, Hemayet Hossain, Sandhya Dwarkadas, and Michael L. Scott. An Integrated Hardware-Software Approach to Flexible Transactional Memory. In *Proc. of the 34th Intl. Symp. on Computer Architecture*, Jun 2007.