

Research Statement

Jayaram Bobba

January 23, 2010

The computer industry has thrived upon decades of exponential growth in hardware and software capabilities. As hardware becomes more powerful and cost-efficient (primarily facilitated by Moore’s Law), software expands to utilize the additional capacity. This expansion of software in turn drives the development of better hardware. Underlying this synergistic evolution is an increasing human ability—assisted by better programming models and software tools—to build bigger and more complex systems. In the future too, increases in human productivity are going to be key to sustaining this model of evolution and expansion of the computer industry.

Recent industry trends, however, have cast a shadow on programmer productivity gains in software development. The shift to chip multiprocessors (CMP) as the vehicles for general-purpose computing has signalled a move to writing parallel software. However, thus far, parallel software has been hard to write and bug-prone. This problem is exacerbated by the lack of efficient software tools to facilitate parallel programming. Unless we innovate in parallel programming models and software tools, we are likely to run into a *productivity wall* that would halt the evolutionary cycle driving our industry.

My research focusses on providing **hardware support to ease software programming** and thus improving programmer productivity. This support can be in the form of either facilitating better programming models like Transactional Memory (TM) [12, 13] or more efficient software tools like dynamic atomicity checkers [14]. In my dissertation research, I provide a foundation for a broad class of such proposals, known as *supervised systems*, along with exploring applications of supervised systems.

Dissertation Research

Supervised Memory (Under Submission)

Supervised systems use additional bits in hardware to store metadata that is used for monitoring and controlling accesses to memory. Examples of supervised systems include hardware TM (HTM) systems, memory-typestate trackers [20, 21, 23, 27] like empty/full-bits [21], log-based architectures [7, 8], deterministic shared memory [10], hardware-assisted garbage collectors [9, 17, 25]. While existing work presents a wide range of uses for supervised systems, it is still incomplete and leaves many questions unanswered. For instance, most of the work builds on top of slow sequentially-consistent (SC) systems and takes an informal approach to specifying memory consistency that could lead to ambiguities.

My thesis builds a firm foundation for current and future supervised systems by addressing drawbacks with existing proposals. First, it demonstrates implementation and correctness issues that could arise with supervised systems on a non-SC system. Using empty/full-bits and a deterministic shared memory proposal as examples, it identifies three issues that are relevant to building supervised systems on top of a relaxed memory system. To address correctness concerns, my thesis proposes **Supervised Memory**, a formal model that provides program metadata in hardware and forms the substrate for many supervised systems. It also formally defines two consistency models similar to the well-known TSO model [11]. In order to resolve the classic tension between performance and ease of reasoning in the design of memory models, I propose **safe supervision**, a program property akin to *data-race-freedom* [1, 2] that simplifies memory models for most novice programmers. Finally, my research investigates implementing a relaxed supervised system on top of a real industry design. Using the OpenSPARC T2 [16], an industrial-strength RTL-level design of a CMP, it demonstrates mechanisms for handling low-level issues like late exceptions and load buffer bypassing using RTL modifications and hypervisor support.

TokenTM [ISCA2008] and StealthTest [PACT2009]

I also examine HTM systems more extensively in my thesis. Transactional Memory is a promising language model that provides an ‘atomic’ construct which enables programmers to easily and declaratively specify the synchronization intent corresponding to a block of code. However, existing TM systems suffer from a self-justifying ‘small-transaction’ assumption that could restrict their applicability to software.

My thesis redresses the small-transaction assumption by demonstrating an efficient TM virtualization solution. Going against convention, it takes a tagged-memory approach and uses supervised memory’s metadata to carry TM state. It combines existing ideas of token counting [15] and per-thread software logs [18] to build **TokenTM [3], an unbounded HTM system that gracefully handles virtualization events** while still imposing little performance overhead on the execution of small transactions. As an added benefit, TokenTM requires minimal changes in complex systems like cache coherence and virtual memory.

Finally, my thesis expands the applicability of TM systems to the critical area of software testing. While software testing is a hard problem, the emergence of CMPs and the proliferation of bug-prone parallel software makes testing even harder. Recently, researchers are exploring methods to continue testing software after deployment. These *on-line* techniques typically fork new processes to hide the functional impact of testing [19, 22]. Unfortunately, the high overhead of `fork()` significantly degrades performance.

I propose StealthTest [6], an interface that exposes TM transactions as the key mechanism for executing on-line tests. StealthTest allows on-line tests to work in isolation on a consistent view of memory. Moreover, explicitly aborting the test transaction after it is done guarantees that its changes are invisible to the rest of the system. To demonstrate the utility of StealthTest, I apply it to two existing on-line testing frameworks that previously relied on forking processes: *in vivo* testing [19] and Delta Execution [22].

Other Research

[HPCA2006,ASPLOS2006,HPCA2007,ISCA2007]

In addition to my dissertation research, I have contributed to the design and development of a range of HTM system proposals from LogTM [18] to LogTM-SE [26]. In the process, I co-led the development and release of a HTM simulator built on the GEMS multiprocessor simulation framework [24] leading to at least 13 external publications. I also performed the first extensive comparison between various design points in the HTM design space. Instead of a raw comparison between existing systems, this work makes a more long-lasting contribution by identifying pathological workload behaviors that could arise with each of the design points [4]. This work was well-received by the community and was selected for the 2008 IEEE Micro Top Picks as one of the 10 most significant research publications in

Computer Architecture in 2007, based on novelty and industry relevance [5].

Future Research

In the immediate future, I could continue exploring newer techniques for using hardware metabits to tackle the productivity wall. In this effort, I will focus on two areas—richer programming models (e.g., object-oriented languages, implicitly parallel programming models) and whole-program analysis tools. First, richer (and arguably easier to use) language models could use hardware support in order to be implemented efficiently. For example, hardware support has been shown to enable fast and scalable garbage collection in object-oriented languages. This line of research has precedents in the 1970s and early 1980s when hardware support was added for dynamically-typed languages like LISP. Second, many whole-program analysis algorithms like static dynamic escape analysis are intractable. However, with run-time support, these algorithms can be efficiently implemented dynamically enabling many optimizations like stack allocation. For single-threaded programs, run-time support could possibly be implemented entirely within software. However, for multi-threaded programs, efficient run-time support would most likely require hardware support. Moreover, given the foundation provided by my Supervised Memory model, I can provide more formal and accurate specifications of such a hardware feature such that it can be readily incorporated on top of existing systems.

More broadly, I am interested in the field of Computer Architecture with specific interest in designing hardware/software interfaces. These interfaces represent a complex trade-off between many factors like simplicity, performance and compatibility and hence provide exciting research opportunities. Programmability challenges are already leading to a re-examination of the hardware/software interface. Hardware reliability issues will soon need to be exposed to software and hence will require an interface modification. As computation systems expand in both dimensions, we are witnessing the emergence of huge warehouse-size computers and tiny mobile computation devices. This could also necessitate a re-examination of older CISC/RISC interfaces in the context of these new systems.

References

- [1] Sarita V. Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. 29(12):66–76, December 1996.
- [2] Sarita V. Adve and Mark D. Hill. Weak ordering - a new definition. In *Proc. of the 17th Annual Intl. Symp. on Computer Architecture*, pages 2–14, May 1990.
- [3] Jayaram Bobba, Neelam Goyal, Mark D. Hill, Michael M. Swift, and David A. Wood. TokenTM: Efficient execution of large transactions with hardware transactional memory. In *Proc. of the 35th Annual Intl. Symp. on Computer Architecture*, Jun 2008.
- [4] Jayaram Bobba, Kevin E. Moore, Haris Volos, Luke Yen, Mark D. Hill, Michael M. Swift, and David A. Wood. Performance pathologies in hardware transactional memory. In *Proc. of the 34th Annual Intl. Symp. on Computer Architecture*, June 2007.
- [5] Jayaram Bobba, Kevin E. Moore, Haris Volos, Luke Yen, Mark D. Hill, Michael M. Swift, and David A. Wood. Performance pathologies in hardware transactional memory. *IEEE Micro*, 28(1):32–41, Jan/Feb 2008.
- [6] Jayaram Bobba, Weiwei Xiong, Luke Yen, Mark D. Hill, and David A. Wood. StealthTest: Low overhead online software testing using transactional memory. In *Proc. of the 18th Intl. Conf. on Parallel Architectures and Compilation Techniques*, September 2009.
- [7] Shimin Chen, Babak Falsafi, Phillip B. Gibbons, Michael Kozuch, Todd C. Mowry, Radu Teodorescu, Anastassia Ailamaki, Limor Fix, Gregory R. Ganger, Bin Lin, and Steven W. Schlosser. Log-based architectures for general-purpose monitoring of deployed code. In *ASID '06: Proceedings of the 1st workshop on Architectural and system support for improving software dependability*, pages 63–65, New York, NY, USA, 2006. ACM.
- [8] Shimin Chen, Michael Kozuch, Theodoros Strigkos, Babak Falsafi, Philip B. Gibbons, Todd C. Mowry, Vijaya Ramachandran, Olatunji Ruwase, Michael Ryan, and Evangelos Vlachos. Flexible hardware acceleration for instruction-grain program monitoring. In *Proc. of the 35th Annual Intl. Symp. on Computer Architecture*, Jun 2008.
- [9] Cliff Click, Gil Tene, and Michael Wolf. The pauseless gc algorithm. In *VEE '05: Proc. of the 1st Intl. Conference on Virtual Execution Environments*, pages 46–56, 2005.
- [10] Joseph Devietti, Brandon Lucia, Luis Ceze, and Mark Oskin. Dmp: Deterministic shared memory multiprocessing. In *Proc. of the 14th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, March 2009.

- [11] Sudheendra Hangal, Durgam Vahia, Chaityasit Manoit, Jun-Yeu Joseph Lu, and Shridhar Narayanan. Tsotool: A program for verifying memory systems using the memory consistency model. In *Proc. of the 31st Annual Intl. Symp. on Computer Architecture*, June 2004.
- [12] Maurice Herlihy and J. Eliot B. Moss. Transactional memory: Architectural support for lock-free data structures. In *Proc. of the 20th Annual Intl. Symp. on Computer Architecture*, pages 289–300, May 1993.
- [13] James R. Larus and Ravi Rajwar. *Transactional Memory*. Morgan & Claypool Publishers, 2007.
- [14] Shan Lu, Joseph Tucek, Feng Qin, and Yuanyuan Zhou. Avio: detecting atomicity violations via access interleaving invariants. In *Proc. of the 12th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 37–48, October 2006.
- [15] Milo M. K. Martin, Mark D. Hill, and David A. Wood. Token coherence: Decoupling performance and correctness. In *Proc. of the 30th Annual Intl. Symp. on Computer Architecture*, pages 182–193, June 2003.
- [16] Sun Microsystems. Opensparc: World’s first free 64-bit cmt microprocessors. <http://www.opensparc.net/opensparc-t2/index.html>.
- [17] David A. Moon. Architecture of the symbolics 3600. In *ISCA '85: Proceedings of the 12th annual international symposium on Computer architecture*, pages 76–83, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press.
- [18] Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill, and David A. Wood. Logtm: Log-based transactional memory. In *Proc. of the 12th IEEE Symp. on High-Performance Computer Architecture*, pages 258–269, February 2006.
- [19] C. Murphy, G. Kaiser, and M. Chu. Towards in vivo testing of software applications. Technical Report cucs-038-07, Columbia University, 2007.
- [20] Feng Qin, Shan Lu, and Yuanyuan Zhou. Safemem: Exploiting ECC-memory for detecting memory leaks and memory corruption during production runs. In *Proc. of the 11th IEEE Symp. on High-Performance Computer Architecture*, February 2005.
- [21] Burton J. Smith. Architecture and applications of the hep multiprocessor computer system. *Society of Photo-optical Instrumentation Engineers*, 298:241–248, August 1981.
- [22] Joseph Tucek, Weiwei Xiong, and Yuanyuan Zhou. Efficient online validation with delta execution. In *Proc. of the 14th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, March 2009.

- [23] G. Venkataramani, B. Roemer, Y. Solihin, and M Prvulovic. Memtracker: Efficient and programmable support for memory access monitoring and debugging. In *Proc. of the 13th IEEE Symp. on High-Performance Computer Architecture*, pages 273–284, February 2007.
- [24] Wisconsin Multifacet GEMS Simulator. <http://www.cs.wisc.edu/gems/>.
- [25] Greg Wright, Matthew L. Seidi, and Mario Wolczko. An object-aware memory architecture. Technical Report TR-2005-143, Sun Microsystems, February 2005.
- [26] Luke Yen, Jayaram Bobba, Michael R. Marty, Kevin E. Moore, Haris Volos, Mark D. Hill, Michael M. Swift, and David A. Wood. LogTM-SE: Decoupling hardware transactional memory from caches. In *Proc. of the 13th IEEE Symp. on High-Performance Computer Architecture*, pages 261–272, February 2007.
- [27] Pin Zhou, Feng Qin, Wei Liu, Yuanyuan Zhou, and Josep Torrellas. iwatcher: Efficient architectural support for software debugging. In *Proc. of the 31st Annual Intl. Symp. on Computer Architecture*, pages 224–237, June 2004.