ATLAS: A Scalable Emulator for Transactional Parallel Systems

Christos Kozyrakis and Kunle Olukotun
Computer Systems Laboratory
Stanford University
http://tcc.stanford.edu

With uniprocessor systems running into instruction-level parallelism (ILP) limits and fundamental VLSI constraints, multiprocessor architectures provide a realistic path towards scalable performance. Nevertheless, the key factors limiting the potential of multiprocessor systems are the difficulty of parallel application development and the hardware complexity of large-scale systems. Several groups have recently proposed parallel software and hardware based on the concept of transactions as a novel approach for addressing the problems of multiprocessor systems [1-6]. Transactions have the potential to simplify parallel programming by eliminating the need for manual orchestration of parallel tasks using locks and messages [5]. Transactions have the potential to improve parallel hardware by allowing for speculative parallelism and increasing the granularity of the coherence and consistency protocols [4].

To realize the potential of transactional systems, researchers must address challenges that span across the fields of architecture, programming models, compilers, and operating systems. It is particularly important to prove that transactional software and hardware work well with large enterprise, scientific, and cognitive workloads, which rely on the performance potential of multiprocessor chips. While architectural studies of small-scale systems with reduced datasets are certainly possible through simulation, slowdowns of 5 orders of magnitude are typical when simulating large-scale parallel systems with large enterprise or scientific applications and modern operating systems [7]. The prohibitive slowdowns deter aggressive experiments and discourage the software community from participating in the development of new hardware and software models for large-scale parallel systems.

This talk will introduce ATLAS, a scalable emulator for transactional parallel systems under development at Stanford. The goal of ATLAS is to provide a fast emulation system for detailed architecture studies while supporting a broad set of software tools to serve as the base for programming model, compiler, and OS research. ATLAS is based on commodity FPGA boards with Xilinx Virtex II Pro devices. The current setup (ML310 boards with 2VP30 chips) allows for 2 PowerPC 405 cores, 300 Kbytes of dual-ported SRAM, 30K logic cells, and 8 3Gbps RocketIO transceivers per chip. Each board includes 256 Mbytes of SDRAM, 512 Mbytes of Compact Flash, a 10/100 Ethernet NIC, and IDE connections for hard disks.

ATLAS uses the PowerPC cores in the FPGA chips as the processors for the emulated parallel system. The dual-ported SRAMs and reconfigurable logic are used to construct the memory hierarchy with transactional execution support (speculative state buffers, violation detection and rollback logic etc.) [4]. Small-scale parallel systems with up to 8 boards (16 processors) are created using point-to-point connections through the RocketIO transceivers (full-connectivity). Larger systems require the use of additional boards or on-chip resources to implement a scalable interconnect network. Depending on the network configuration, ATLAS supports per processor network bandwidth of 4 to 30 bytes per cycle, which has been shown to be sufficient for transactional parallel systems [4].

The software environment for ATLAS relies on the rich software base for the ML310 board, which runs Linux out of the box [ref]. This allows us to incorporate several key open-source packages in our research infrastructure: Linux for OS research, gcc for C-based programming model and compiler optimizations, Jikes RVM for Java-based programming model and dynamic optimizations, Jboss, apache, and Berkeley DB for enterprise application studies. It is particularly important that the current sequential version of these applications run on ATLAS without porting. We can immediately investigate full-system issues and gradually port or adapt portions of these tools to the transaction-based environment as needed.

The main advantages of ATLAS are emulation speed, high modeling accuracy, and software flexibility. It allows fast emulation of transactional hardware with tens of processors at frequencies ranging from 10s to 100s of MHz, which is only one order of magnitude slower than an ASIC-based prototype. The emulation results can be highly accurate as ATLAS is based on real hardware (hard core for the processor, synthesized logic for the memory hierarchy). Moreover, ATLAS provides its hardware and software flexibility without significant up-front chip, board, or software design. The main shortcoming of ATLAS is its fixed and simple processor core. We believe that this is not a significant roadblock for future research. The processor microarchitecture is less critical than the memory hierarchy and interconnect in a parallel system. ATLAS trades off processor model complexity for emulation speed and for software flexibility, which are crucial for reaching our research goals.

We are currently developing the first version of the ATLAS emulator for the TCC transactional architecture [4-5]. Our goal is to build a system with at least 32 to 64 boards (64-128 processors), with each processor operating at 100MHz (based on current technology). The development includes emulator assembly, hardware synthesis, and transactional software development (programming models, API, compiler, applications). Given the commodity and low-cost nature of the ATLAS building block (ML310), we also intend to make the ATLAS design files and software available to the wider research community in order to disseminate the ideas and enable further research on transaction-based parallel hardware and software.

References

[1] M. Herlihy and J. Moss, "Transactional Memory: Architectural Support for Lock-Free Data Structures," in ISCA-20, 1993.
[2] R. Rajwar and J. Goodman, "Transactional Lock-Free Execution of Lock-Based Programs," in ASPLOS-X, 2002.
[3]T. Harris and K. Fraser, "Language Support for Lightweight Transactions," in OOPSLA, 2003.
[4]L. Hammond, et.al. "Transactional Memory Coherence and Consistency," in ISCA-31, 2004.
[5]L. Hammond, et.al. "Programming with Transactional Memory Coherence and Consistency," in ASPLOS-XI, 2004.
[6] S. Ananian, et.al. "Unbounded Transactional Memory", in HPCA-11, 2005.
[7] A. Alameldeen, et.al. "Simulating a $2M commercial server on a $2K PC", IEEE Computer, 2003.