



Sponsored by

Seminar Series Supported by Jeffrey and Holly Ullman



Multi-Core Processors Day: Past & Future

14 May, 2008

10:30 Coffee and tagging

10:50 Hopscotch Hashing: a Hash Map for the Multicore Era
Nir Shavit, Tel-Aviv University

Abstract: We present a new resizable hash table algorithm directed at multicore machines. The algorithm is based on a novel hopscotch multi-phased probing and displacement technique that combines elements of cuckoo hashing and linear probing. The hopscotch technique avoids the double-hash and linear probe overheads of these former algorithms, providing a table with very low synchronization costs and high cache hit ratios.

In a series of benchmarks on a state-of-the-art 64-way Niagara II multicore machine as well as a 4-way Intel multicore, the new algorithm proves to be highly scalable, delivering about 3 times the throughput of today's most efficient concurrent hash algorithm, Lea's ConcurrentHashMap from java.concurr.util. It does so with a memory utilization of over 90%. Perhaps more interestingly, on a single core, a sequential version of Hopscotch outperforms all know sequential hash table algorithms, and by a significant factor as the table density increases beyond 50%.

Joint work with Maurice Herlihy (Brown University) & Moran Tzafrir (Tel Aviv)

11:30 Software Enablement for Multi-Core Architectures
Bilha Mendelson, IBM Haifa Research Lab

Abstract: A recent trend of transitioning to multi-core architectures in the mainstream market segments creates significant challenges for programming systems. The market need for creating portable multi-threaded applications that exploit high performance of chip multiprocessors is not easily supported by existing programming models and languages, compiler technology, performance analysis and testing tools.

In the talk I will overview various research directions in this area and present some partial results achieved for the Cell and PPC platforms.

12:10 Locality in Concurrent Data Structure
Hagit Attiya, Technion

Abstract: With multi-core and multiprocessing systems becoming widespread, most algorithms and data structures need to accommodate concurrent access; to reap the performance benefits of such systems, this should be done without effectively sequentializing competing operations. We explain the concept of locality and describe two methodologies for designing concurrent data structures so as to decrease interference among operations and increase concurrency and throughput. One methodology specifically addresses doubly-linked lists, while the other deals with generic multi-word operations.

12:50 Lunch

13:30 Industrial and Research Challenges in the Area of Multi-Core and Many Cores
Avi Mendelson, Intel Haifa and Technion

Abstract: Recently the computer architecture world is shifting toward multi-core and many core systems. This "new" trend already happened in the past with only partial success. As we discuss in the talk, a major part of the past failure was the inability to efficiently use parallel systems. In order to avoid repeating past mistakes, the research community needs to provide essential solutions to critical issues.

In my talk I will provide a short historical perspective on the past similar trends, and highlight a few critical directions that the research must address in order to make the new trend result in greater success.

14:10 Dynamic Asymmetric Management of Large-Scale Multi-Cores
Amnon Barak, Hebrew University of Jerusalem

Abstract: This presentation is about several operating system alternatives for dynamic management of resources in large scale, distributed memory multi-cores (cluster on a chip), with special emphasis on asymmetric systems that can make all the cores perform like a single computer with many processors.

14:50 Coffee

15:05 Timing-based Synchronization Algorithms
Gadi Tubenfeld, Interdisciplinary Center in Herzliya (IDC)

Abstract: Concurrent systems exhibit a significant degree of synchrony in practice, but few guarantee to do so. This synchrony can be exploited to achieve algorithms with properties that are not possible in completely asynchronous systems; however, in general we cannot afford to risk incorrect behavior in case our assumptions about this synchrony are occasionally violated. In this talk we will explore the possible middle ground of exploiting synchrony when it is available, but in any case guaranteeing correctness regardless of the timing behavior of the system.

That is, we will discuss the design of indulgent algorithms which are algorithms that work correctly and efficiently whenever the timing assumptions are satisfied, but never violate their safety properties when these assumptions are not satisfied by the underlying system. The appeal of such algorithms lies in the fact that when they are executed in an asynchronous system, they "lie in wait" for a short period of time during which certain timing constraints are met, and when this happens these algorithms take advantage of the situation and efficiently complete their mission.

15:45 Scheduling-Based Collision Avoidance and Resolution for Software Transactional Memory
Danny Hendler, Ben-Gurion University

Abstract: The emergence of multi-core architectures is adding impetus to the shift from single-threaded applications to concurrent, multi-threaded applications. Transactional memory is a concurrent programming abstraction that is viewed by many as having the potential of becoming a superior alternative to lock-based programming for these architectures.

We present CAR-STM, a scheduling-based mechanism for Collision Avoidance and Resolution that can be incorporated into existing software transactional memory (STM) implementations. In addition to proactive collision avoidance that is based on application-specific hints, CAR-STM's transaction scheduling supports novel and highly efficient contention managers that resolve transaction conflicts by serializing the execution of colliding transactions.

Joint work with Shlomi Dolev & Adi Suissa (both from BGU)

