10:45 Coffee & Tagging

11:05 No Need to Constrain Many-Core Parallel Programming
Uzi Vishkin, University of Maryland, MD, USA
Abstract: The transition in mainstream computer science from serial to parallel programming for many-core on-chip computing offers parallel computing research the wonderful opportunity it had sought all along. However, such transition is a potential trauma for programmers who need to change the basic ways in which they conduct their daily work. The long experience with multi-chip parallel machines only adds to the apprehension of today's programmers. Many people who tried (or deliberately tried) to program their multi-chip machines consider their programming “as intimidating and time consuming as programming in assembly language” (~1981 NSF Cyber-infrastructure Blue Ribbon Committee) and have literally walked away. Programmers simply did not want to deal with lock-free programming for locality in order to extract the performance that these machines promise. Consequently, their work fell far short of historical expectations. Now, with the emerging many-core computers, the foremost challenge is ensuring that mainstream computing is not retaliated against another major disappointment. Limiting many-core parallel programming to more or less the same programming approaches that dominated parallel machines could avoid: (i) simplifying programmers; (ii) reducing productivity of those programmers who hold on, getting the performance promise required high development time and leads to more error-prone code; (iii) raise by too much the minimal professional development stage for introducing programmers to parallel programming, reducing further the use of potential programmers; and overall (iv) fail to meet expectations regarding the use of parallel computing; only this time for many-cores. The talk will overview a hardware-based PRAM-On-Chip variant that seeks to rebuild parallel computing from the ground up. Grounded in the richest and most abstract theory of parallel algorithms, known as PRAM, where the programmer only needs to specify at each step what operations can be executed concurrently, an on-chip architecture that scales to thousands of processors on chip called XMT (explicit multi-threading) was introduced. Significant hardware and software prototyping of XMT will be reported, including a 64-processor FPGA-based machine and two ASIC chips fabricated using 90nm CMOS technology, as well as strong speedups on applications. By having XMT programming taught at various levels from rising 6th grade to graduate students, we developed evidence that the stage at which parallel programming can be taught is earlier than demonstrated by other approaches. For example, students in a freshman class were able to program 3 parallel sorting algorithms. Software release of the XMT environment can be downloaded to any standard PC platform along with extensive teaching materials, such as video-recorded lectures of a one-day tutorial to high school students and a full-semester graduate class, class notes and programming assignments. Preliminary thoughts on encapsulating XMT into a hardware-enhanced programmer’s workstation will also be presented and the prospects for incorporating it as an add-on into some other many-core designs be discussed.

11:55 Coffee Break

12:10 Control and Resource Management of Parallel Systems
Dror Feidelson, Hebrew University
Abstract: We are at the beginning of a period where parallel systems are becoming commonplace, and available degrees of parallelism are expected to grow exponentially. This is often considered a “good thing”, as it will enable ever increasing performance for applications. But it raises the question of how to control all this parallelism, how to best manage it, and how to tolerate faults. The talk will review several approaches with different degrees of regularity and synchronicity, including issues like functional partitioning and user control.

12:55 Lunch

13:55 Asynchronous Pattern Matching
Amihood Amir, Bar-Ilan and Johns Hopkins University
Abstract: The LCA (Lowest Common Ancestor) problem is to find the lowest node in a tree that has two specified nodes as its ancestors. For over 30 years, a number of algorithms for solving LCA have been proposed, including several O(n) algorithms. However, no known algorithm is optimal, or restricted to very limited classes of functions (e.g. linear or monotone).

In the talk we will review our previous work on this problem. Then, we will introduce a novel approach that generalizes previous results, and then apply a new perspective, and open the way to further improvements and research. A general optimization problem is defined that can be used for the compilation of local constraints, which, in general, define “safe zones”. Safe zones can be used to efficiently implement distributed threshold monitoring systems with very little communication overhead. The new approach uses geometric and probabilistic tools which, to the best of our knowledge, were not previously applied in monitoring systems nor in the analysis of distributed algorithms.

14:50 Safe Zones for Monitoring Distributed Data Streams
Assaf Schuster, Technion
Abstract: Communication typically affects the main bottleneck in monitoring distributed dynamic environments where the system is deal with massive incoming data streams at the nodes. Often, the monitoring problem consists of determining whether the value of a certain function, which depends on the combined data at all nodes, crossed a certain threshold, which may indicate a global phase change that calls for some action (such as sending an alert). Sending all the data to a central location is out of the question for several reasons (including overhead and privacy issues). Thus, there is a great deal of effort directed at reducing the monitoring of the global function’s value to the testing of local constraints, checked independently at each node. However, as far as results are obtained very sub-optimal, or restricted to very limited classes of functions (e.g. linear or monotone).

In this talk we will review our previous work on this problem. Then, we will introduce a novel approach that generalizes previous results, and then apply a new perspective, and open the way to further improvements and research. A general optimization problem is defined that can be used for the compilation of local constraints, which, in general, define “safe zones”. Safe zones can be used to efficiently implement distributed threshold monitoring systems with very little communication overhead. The new approach uses geometric and probabilistic tools which, to the best of our knowledge, were not previously applied in monitoring systems nor in the analysis of distributed algorithms.

15:30 On Cartesian Trees, Lowest Common Ancestors, and Range Minimum Queries
Oren Weimann, University of Haifa
Abstract: The Range Minimum Query (RMQ) problem asks to preprocess an array such that the minimum element between two specified indices can be found efficiently. The Lowest Common Ancestor (LCA) problem is to compute the lowest node in the tree that has two specified nodes as its ancestors.

In this talk, we will overview the ideas and techniques that were developed over the years for the LCA and RMQ problems. We will then describe some generalizations and extensions of the RMQ problem. Namely, we will introduce an optimal cache-oblivious RMQ algorithm that minimizes the number of block transfers between main memory and disk. We will also discuss the differences between the Cuckoo-Carrot (aka “Cuckoo”) tree for solving a generalization of RMQ and the classic “Bottleneck Edge Query” problem. In the talk, we will also describe two important extensions of the “Bottleneck Edge Query” problem. Finally, we will discuss the two-dimensional version of RMQ. I will describe a proof (assuming that) its Cuckoo tree exists for 2D-RMQ.

16:10 Coffee Break

16:25 Buffer Management for Colored Packets with Deadlines
Yossi Azar, Tel-Avivy University
Abstract: We consider buffer management of unit packets with deadlines for a multi-port device with reconfiguration overhead. The goal is to maintain the throughput of the device, i.e., the number of packets delivered by their deadlines. For a single port or with free reconfiguration, the problem reduces to the well-known packets scheduling problem, where the celebrated earliest deadline first (EDF) strategy is optimal 1-competitive. However, EDF is not admissible in the presence of reconfiguration overhead.

We design an online algorithm that achieves a competitive ratio of at least when the ratio between the minimum deadline and the maximum deadline is at most We improve this result in the case where one can design an almost 1-competitive algorithm. Our idea is to use a technique which, to the best of our knowledge, was not previously applied in monitoring systems nor in the analysis of distributed algorithms.

In the talk we will overview the ideas and techniques that were developed over the years for the LCA and RMQ problems. We will then describe some generalizations and extensions of the RMQ problem. Namely, we will introduce an optimal cache-oblivious RMQ algorithm that minimizes the number of block transfers between main memory and disk. We will also discuss the differences between the Cuckoo-Carrot (aka “Cuckoo”) tree for solving a generalization of RMQ and the classic “Bottleneck Edge Query” problem. In the talk, we will also describe two important extensions of the “Bottleneck Edge Query” problem. Finally, we will discuss the two-dimensional version of RMQ. I will describe a proof (assuming that) its Cuckoo tree exists for 2D-RMQ.

17:05 Transactional Memory Scheduling
Danny Hendler, Ben-Gurion University
Abstract: With the emergence of multi-core architectures, we are facing the grand challenge of providing software developers with appropriate parallel programming abstractions. These abstractions must facilitate high-productivity development of the fast, efficient, and scalable concurrent applications. Transactional memory (TM) is a concurrent programming abstraction that is viewed by many as a promising alternative to lock-based synchronization. With transactional memory, critical sections are expressed as atomic blocks performed at transactions. At termination, these transactions may be executed concurrently, thereby optimizing the optimistic expectation that the set of locations accessed by one transaction will not overlap with the set of locations written by another concurrent transaction. If such a conflict does occur, TM contention manager decides how it should be resolved.

Up until recently, TM contention managers had little control of transaction semantics, which remained under the supervision of the operating-system’s transaction-processing mechanisms. TM schedulers, introduced for the first time last year, allow TM implementation to supply transactional scheduling policies and have been shown to significantly boost TM performance under high contention.

In this talk, we will survey state-of-the-art work on user-level, kernel-level, and adaptive TM scheduling.