



Seminar Series Supported by Jeffrey and Holly Ullman

Parallel Computing Day

20 October, 2009

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 impact 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 deliberated trying) to program these multi-chip machines consider their programming "as intimidating and time consuming as programming in assembly language" (~2003 NSF Cyber-infrastructure Blue Ribbon Committee), and have literally walked away. Programmers simply did not want to deal with the constraint of programming for locality in order to extract the performance that these machines promise. Consequently, their use fell far short of historical expectations. Now, with the emerging many-core computers, the foremost challenge is ensuring that mainstream computing is not railroaded into another major disappointment. Limiting many-core parallel programming to more or less the same programming approaches that dominated parallel machines could again: (i) repel programmers; (ii) reduce productivity of those programmers who hold on; getting the performance promise requires 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 pool 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 vision that seeks to rebuild parallel computing from the ground up. Grounded in the richest and easiest known theory of parallel algorithms, known as PRAM, where the programmer only needs to identify at each step operations that can be executed concurrently, an on-chip architecture that scales to thousands of processors on chip called XMT (for 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 graders 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 workflow 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 Feitelson, 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 rigidity and synchronism, 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: Traditional Approximate Pattern Matching (e.g. Hamming Distance errors, edit distance errors) assumes that various types of errors may occur in the data, but an implicit assumption is that the order of the data remains unchanged.

Over the years some applications identified types of "errors" where the data remains correct but its order is compromised. The earliest example is the "swap" error motivated by a common typing error. Other widely known examples such as transpositions, reversals, and interchanges, are motivated by Biology.

We propose a model that formally separates the concepts of "errors in the data" and "errors in the address" since they present different algorithmic challenges solved by different techniques. The "error in address" model, which we call asynchronous pattern matching, since the data is not assumed to arrive in a synchronous sequential manner, is rich in problems not addressed hitherto.

We will consider some reasonable metrics for asynchronous pattern matching and show some efficient algorithms for these problems. As expected, the techniques needed to solve these problems are not taken from the standard pattern matching "toolkit".

14:40 Coffee

14:50 Safe Zones for Monitoring Distributed Data Streams

Assaf Schuster, Technion

Abstract: Communication typically poses the main bottleneck in monitoring distributed dynamic environments when the system has to 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, so far results are either very sub-optimal, or restricted to very limited classes of functions (e.g. linear or monotonic).

In the talk we first review our previous work on this problem. Then, we introduce a novel approach that generalizes previous results, puts them in a new perspective, and opens the way for further improvements and research. A general optimization problem is defined that can be used for the compilation of local constraints, which, in turn, 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 preprocess a rooted tree for subsequent queries asking for the common ancestor of two nodes that is located farthest from the root. RMQ and LCA have numerous applications in string processing and computational biology and were shown by Gabow et al. to be equivalent in the sense that either one can be reduced in linear time to the other. Harel and Tarjan were the first to show that the LCA problem can be optimally solved with linear-time preprocessing and constant-time queries by relying on word-level parallelism. Their data structure was later simplified by Schieber and Vishkin, Berkman and Vishkin, and Bender et al. These results presented further simplifications, removed the need for word-level parallelism, and showed how to make the data structure efficient in a parallel computing environment. The key idea for all these algorithms is the connection between LCA and RMQ captured by what is known as the Cartesian tree.

In this talk I will overview the ideas and techniques that were developed over the years for the LCA and RMQ problems. I will then describe some generalizations and extensions of the RMQ problem. Namely, I will introduce an optimal cache-oblivious RMQ solution that minimizes the number of block transfers between main memory and disk. I will also describe a new Cartesian tree for solving a generalization of RMQ from arrays to trees and undirected graphs (the Bottleneck Edge Query problem). Finally, I will discuss the two-dimensional version of RMQ. I will describe a proof showing that no Cartesian tree exists for 2D-RMQ and briefly describe how to overcome this and obtain an optimal 2D-RMQ data structure.

16:10 Coffee Break

16:25 Buffer Management for Colored Packets with Deadlines

Yossi Azar, Tel-Aviv University

Abstract: We consider buffer management of unit packets with deadlines for a multi-port device with reconfiguration overhead. The goal is to maximize the throughput of the device, i.e., the number of packets delivered by their deadline. 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 1-competitive when there is a reconfiguration overhead.

We design an online algorithm that achieves a competitive ratio of $1 - o(1)$ when the ratio between the minimum laxity of the packets and the number of ports tends to infinity. This is one of the rare cases where one can design an almost 1-competitive algorithm. One ingredient of our analysis, which may be interesting on its own right, is a perturbation theorem on EDF for the classical packets scheduling problem. Specifically, we show that a small perturbation in the release and deadline times cannot significantly degrade the optimal throughput. This implies that EDF is robust in the sense that its throughput is close to the optimum even when the deadlines are not precisely known.

Joint with with U. Feige, I. Gamzu, T. Moscibroda and P. Raghavendra

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 robust, 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 as transactions. At runtime, these transactions may be executed concurrently based on 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, a TM contention manager decides how it should be resolved. Up until recently, TM contention managers had little control of transaction threads, which remained under the supervision of the operating-system's transaction-ignorant scheduler. TM schedulers, introduced for the first time last year, allow TM implementations to apply transaction scheduling policies and have been shown to significantly boost TM performance under high contention. My talk will survey state-of-the-art work on user-level, kernel-level, and adaptive TM scheduling.

