

Algorithms-based extension of serial computing education to parallelism

Uzi Vishkin, University of Maryland Institute for Advanced Computer Studies (vishkin@umd.edu)

Parallel computing provides now the only avenue for continued improvement in performance in general-purpose computing. The basic transition to parallelism appears to be robust: the mainstream hardware and software computing industries have been forced to bet their future on parallelism. But, how the commodity parallel general-purpose computer of the future will be built and programmed for performance remains unclear. Points of reference include:

- The programmer of today's parallel machines must overcome several 'productivity busters' beyond just identifying operations that can be executed in parallel:
 - impose the [CS99] 4-step programming-for-locality recipe: decomposition, assignment, orchestration, and mapping, which is often difficult;
 - reason about concurrency, including race conditions, in threads;
 - for machines such as GPU, that fall behind on serial (or low-parallelism) code, whole programs must be highly parallel.
- The National Research Council report [FM10] points out that while heroic programmers can exploit today vast amounts of parallelism, whole new computing "stacks" are required to allow expert and typical programmers to do that easily.
- None of 40+ students in a fall 2010 joint UIUC/UMD course got any speedups using OpenMP programming on simple irregular problems using an 8-processor SMP, but they got 8X-25X speedups on XMT.
- SMPs do not scale beyond 8-16 processors and it is not clear how well other cache coherence solutions work due to their high overheads.
- Ease-of-teaching comparison by DARPA-HPCS-funded software engineering experts Basili and Hochstein (UMD) [HBVG08] showed XMTC/PRAM development time is half of MPI.

Beyond introductory programming, standard CS curriculum emphasizes data-structure and algorithms courses over programming ones. This fact and the noted lack of clarity about the future led me to devote class time to algorithms over programming and favor teaching the simplest common denominator of current approaches.

The education platform I developed is based on the following elements:

1. Identify 'thinking in parallel' with the basic abstraction behind the [SV82b] work-depth framework. This framework was previously adopted as the presentation framework in 2 PRAM algorithms texts: [J92, KKT01].
2. Teach as much PRAM algorithms as timing constraints and developmental stage of the students permit; extensive 'dry' theory homework is required from graduate students, but little from high-school students.
3. Students self-study programming in XMTC (standard C plus 2 commands, spawn and prefix-sum) and do demanding programming assignments.
4. Provide a programmer's workflow that links the simple PRAM abstraction with XMTC programming. The synchronous PRAM provides ease of algorithm design and reasoning about correctness and complexity. Multi-threaded programming relaxes this synchrony for implementation. Since reasoning directly about soundness and performance of multi-threaded

code is known to be error prone, the workflow assigns a much simpler task to the programmer: establish that the multi-threaded program behavior matches the PRAM-like algorithm it implements.

5. Unlike the PRAM theory, XMTC is far from ignoring locality. Unlike today's common approaches, XMTC preempts the harmful effect locality has on programmer's productivity.

6. If the XMT architecture is presented, it is done only at the end of the course; students don't learn serial architecture prior to learning serial programming, so why should they learn parallel architecture?! (However, parallel architecture relevant to OpenMP and MPI had to be taught in the UIUC/UMD course.)

Experience:

K-12 Since 2007, various snippets of the approach were taught mostly by two high school teachers to more than 100 middle school and high school students of a wide range of backgrounds. Among them were students from Montgomery Blair, Maryland and Thomas Jefferson, VA, magnet high schools, Baltimore Polytechnic high school, whose student body is 70% African-American, and a Montgomery County, MD Public Schools middle-school summer workshop for children from underrepresented groups. Teacher S. Torbert (TJHS) self-taught himself from publicly available material. Teacher D. Ellison (Math Ed, PhD student, U. Indiana) who taught at the Baltimore high school and the middle-school workshop was advised by Math Ed professor R. Tzur (Purdue/U. Colorado), an expert in learning as understood in education.

College freshmen A class that included 3 sorting programming assignments and one for finding the median – a proper load for a freshmen serial programming course – was taken by 19 students, mostly non-CS majors. This overall K13 experience was reviewed in: (i) [VTETC09], a keynote at CS4HS'09@CMU and (ii) [TVTE10], a SIGCSE'10 paper that reports a decisive teaching advantage of XMT over CUDA, MPI and OpenMP at TJHS.

Graduate class included standard PRAM algorithms plus 6 XMTC programming assignments including the [SV82a] graph connectivity (done also by a couple of Blair 10th graders) and performance tuning.

Senior level course taught less theory. An inter-university senior-level course using teleconferencing with UIUC included 7 lectures on PRAM/XMTC, demonstrating a useful role for them in a course that included 3 joint OpenMP /XMTC programming assignments, as well as MPI.

Course on general algorithms Several PRAM classes were included.

Support available though the XMT homepage www.umiacs.umd.edu/users/vishkin/XMT/ includes: 1. Software release of the whole XMT environment (compiler and simulator [KTCBV11]) available for free download along with extensive documentation. However, many students programmed a 64-processor XMT FPGA machine. 2. Extensive teaching material including class notes, over 31 hours of video-recorded classes, and a day-long tutorial are also available on-line.

Though quite different, like-minded approaches include Cilk [CLRS09] and NESL [B96].

References

- [B96] G.E. Blelloch. Programming parallel algorithms. CACM 39,3, 1996.
- [CLRS09] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms, 3rd Ed. MIT Press, 2009. Chapter on Multithreaded Algorithms.
- [CS09] D. Culler and J. Singh. Parallel Computer Architecture: A Hardware/Software Approach. Morgan-Kaufmann, 1999.
- [FM10] S.H. Fuller and L.I. Millett (Eds.). The Future of Computing Performance: Game Over or Next level. Computer Science and Telecommunications Board, National Academies Press, December 2010.
- [KTCBV11] F. Keceli, A. Tzannes, G. Caragea, R. Barua and U. Vishkin. Toolchain for programming, simulating and studying the XMT many-core architecture. Proc. 16th Int. Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS), in conjunction with IPDPS, Anchorage, Alaska, May 20, 2011, to appear.
- [HBVG08] L. Hochstein, V. Basili, U. Vishkin and J. Gilbert. A pilot study to compare programming effort for two parallel programming models. Journal of Systems and Software, 2008. This paper compares the programming effort in MPI and XMTC for a similar programming assignment. The main finding is that with high confidence XMTC development time is nearly half of MP.
- [J92] J. JaJa. An Introduction to Parallel Algorithms. Addison-Wesley Publishing Company, 1992.
- [KKt01] J. Keller, C. Kessler, and J. Traeff. Practical PRAM Programming. Wiley-Interscience, 2001.
- [SV82a] Y. Shiloach and U. Vishkin. An $O(\log n)$ parallel connectivity algorithm. J. of Algorithms, 3:57–67, 1982.
- [SV82b] Y. Shiloach and U. Vishkin. An $O(n^2 \log n)$ parallel max-flow algorithm. J. Algorithms, 3:128–146, 1982. Introduced the work-depth framework.
- [TVTE10] S. Torbert, U. Vishkin, R. Tzur and D. Ellison. Is teaching parallel algorithmic thinking to high-school student possible? One teacher's experience. Proc. 41st ACM Technical Symposium on Computer Science Education (SIGCSE), Milwaukee, WI, March 10-13, 2010.
- [V11] U. Vishkin. Using simple abstraction to reinvent computing for parallelism. CACM 54,1 (January 2011), 75-85. This paper provides a good introduction to the PRAM abstraction and the XMTC programmer's workflow.
- [VTETC09] U. Vishkin, R. Tzur, D. Ellison, S. Torbert and C. Caragea. Programming for high schools, July 2009, keynote, The CS4HS Workshop at CMU. Download from <http://www.umiacs.umd.edu/~vishkin/XMT/CS4HSPATfinal.ppt>