New-Old Grand Challenge: Parallel Computing

Uzi Vishkin *

The generic "Grand Challenge" problem proposed is that of parallel computing. The challenge is: For general-purpose computing, develop a cost effective architecture for improving single task completion time by exploiting parallelism. In other words, the following concerns need to be addressed: programmability and algorithmic theory, applications (e.g., for improved productivity), performance, buildability, power/energy, cost.

As is presented above, the grand challenge is nearly identical to the classic parallel computing one. However, in spite of a very extensive effort over at least two decades, general-purpose parallel computing is in a state of limbo. On one hand, parallel machines have not been cost effective when it came to their programming. On the other hand, easy-to-program programming and algorithms models (such as the PRAM) that provided great virtual parallel scalability have been developed; but, unfortunately, it was not possible in the 1990s to build machines that efficiently support these models; the overheads for managing the parallelism provided by these models were simply too high.

The challenge of parallel computing is worth an out-of-the-box revisit because of the following:

- An increase by factors exceeding ten thousand in the number of transistors per chip since the Intel 8086 CPU of the early 1980s. In other words, even a single chip can become the core of a powerful parallel machine. Within a chip, much higher bandwidth and much lower latencies can be achieved. This could reduce drastically the overheads for managing parallelism–the main predicament for bringing the scalability of algorithms to the hardware in the past.
- Known architecture technique for using the increased amounts of hardware have been mined out.
- The basic serial 1946 "von-Neumann model" for general-purpose computers has not changed in any fundamental way (possibly contributing to the "incremental" image of computer science and engineering, as per the call for this CRA conference).
- The importance attached to SPEC-type benchmarks in evaluating future designs certainly suggests that they are very important. The current comment is also not meant to play down the tremendous amount of thinking behind these benchmarks. However, the importance attached to them sometimes appears to stand in the way of some out-of-the-box approaches. The problem is that code optimized for past architectures ends up being used to evaluate future ones.
- In the coming era of "deep-submicron" VLSI wire delays start dominating switching times. This gives greater advantage to a more distributed, or pipelined, engineering of chips, and, in turn, to greater program parallelism.
- Power and energy considerations become more important.

*University of Maryland. E-mail: vishkin@umiacs.umd.edu

Some key questions are:

- What would be the programming model? the algorithms model? Which algorithmic theory matches the serial algorithmic theory being challenged?
- What kind of applications will benefit?
- Can more computing power help new non-programming APIs, to increase productivity of users who have no knowledge of programming (e.g., for computer games and graphics simulations)?
- How to obtain serial compatibility? and last but not least,
- How should the architecture look?

From the ACM-SPAA (The Annual Symposium on Parallel Algorithms and Architectures) point of view, the biggest open problem appears to be as follows. The PRAM algorithmic theory is the only serious contender ever proposed as a true alternative to the serial algorithmic theory; that is, no other algorithmic approach came close in terms of the magnitude of the knowledge base it provided. Under "Potential Breakthroughs" in the Culler-Singh 1999 book Parallel Computer Architecture, the following goal is stated: "somehow design machines in a cost-effective way that makes it much less important for a programmer to worry about the data locality and communication; that is, to truly design a machine that can look to the programmer like a PRAM". **Concrete grand challenge problem**: Can such a breakthrough be achieved?

One extension of the generic parallel computing challenge is in the direction of "Massively Parallel Computing" (in the spirit of the IBM Blue Gene effort). Molecular simulations for applications such as drug design, or protein folding would be the main target. Unlike some approaches to parallel computing an approach that allows "sequence scalability" is needed. That is, for some molecular simulations the number of steps that need to be simulated within a reasonable time appears to be much higher than currently available. Of course, the total number of concurrent operations that need to be simulated is also large. For example, suppose that 10^{14} steps need to be simulated. If a rate of a step per nanosecond becomes possible, the whole simulation takes around one day.

Let me note a major policy/lobbying challenge. Life science is now at the center of attention in Washington. Life science does not provide the same clear mapping from computing power to applications as some physical sciences. Physicists were used to campaign for funding research towards increasing computing power till the late 1980s. So far, life scientists have not done that in a big way. Can we recruit some for a campaign?

To test the water, I made contacts with several super senior life scientists, and found out that they are very open to the idea of having them campaign for us, if we do the leg work. For example, one just wrote back to me: *In answer to your question: "Who needs to simulate this many steps?", my answer is simple: Biology. If we are ever to make Biology into an exact predictive science, we need to couple physical simulation to the biological information. These are large systems with 1,000's to 1,000,000's of particles. We need to simulate 1,000,000,000,000 time steps to get to slow phenomenon or get adequate sampling.* If invited by the CRA to help with trying to put together an effective campaign of endorsement from life scientists, I will be glad to do it.

Bio

Uzi Vishkin is Professor, the University of Maryland Institute for Advanced Computer Studies, and Electrical and Computer Engineering since 1988. He was Professor of Computer Science at the Technion in Israel in 2000-2001. Previously, he was Professor of Computer Science at Tel Aviv University, Israel, where he was Chair of the Computer Science Department in 1987-8 and Professor since 1988. He was with the Courant Institute, New York University, from 1982 till 1988, after being a Postdoctoral Fellow at IBM T.J. Watson Research. He received the B.Sc. and the M.Sc. degrees in Mathematics from the Hebrew University 1974 and 1975, respectively, and the D.Sc. degree in Computer Science from the Technion in 1981.

Motivated, since 1979, by the grand, but so far mostly elusive, challenge of reducing the completion time of a single computing task by way of parallelism the main concrete motivating question guiding Vishkin's research has been "how to think in parallel?". New paradigms and methodologies for the development of parallel algorithms that he has been a participant in developing are represented in textbooks on the design of algorithms. In the most cited review publications on parallel algorithms the percentage of references to works that he co-authored is %10 to %20. He also made contributions to the development and understanding of underlying principles for the evolving generation of parallel computer systems. An international speaker, he has authored, or co-authored, over 60 papers in archival journals. He is Fellow of the ACM. He is on the editorial board of J. Algorithms, IEEE Transaction on Computers, and Parallel Processing Letters and on the Steering Committee for ACM-SPAA. Vishkin was recently recognized as one of the 100 most cited researchers in Computer Science (by ISI - Thompson Scientific, the publishers of the Science Citation Index); the list will be featured in ISIHighlyCited.com, once extended to Computer Science.

Some relevant points of reference in his professional career include:

- He advocated seeking linear speedup relative to the best serial algorithm as a goal for parallel algorithms design since 1979. That is, a good parallel algorithm should not perform many more operations than the best serial algorithm for the same problem. Until late 1984, the theory community favored NC-type algorithms (i.e., parallel algorithm whose run time is poly logarithmic in the length of the input using a polynomial number of processors). Only at about 1984 that community accepted the position that the "linear speedup" theory is the more relevant algorithmic theory (while the NC theory continued to play an important role, similar to the NP-completeness theory for serial algorithms).
- When it became clear around 1993 that the Federal High Performance Computing and Communication program does not have a clear strategy for impacting main stream computing, he put together an NSF-DARPA sponsored workshop in which leaders in the field were invited to suggest agendas. The following edited book includes these agendas: U. Vishkin (editor), Developing a Computer Science Agenda for High-Performance Computing, ACM Press, 1994, 158 pages.
- In several position papers, and presentations during the past two decades, He predicted that programmability will fail parallel computing in the 1990s. His concern was that most systems-centered projects at the time tended to ignore the insufficient algorithmic theory supporting them. He will try to represent this type of theory point of view in the current workshop.