Ten Ways to Waste a Parallel Computer

Katherine Yelick
Electrical Engineering and Computer Sciences Department, UC Berkeley
NERSC, Lawrence Berkeley National Laboratory

Abstract

As clock speed increases taper off and hardware designers struggle to scale parallelism within a chip, software developers and researchers must face the challenge of writing portable software with no clear architectural target. On the hardware side, energy considerations will dominate many of the design decisions, and will ultimately limit what systems and applications can be built. This is especially true at the high end, where the next major milestone of exascale computing will be unattainable without major improvements in efficiency.

Although hardware designers have long worried about the efficiency of their designs, especially for battery-operated devices, software developers in general have not. To illustrate this point, I will describe some of the top ways to waste time and therefore energy waiting for communication, synchronization, or interactions with users or other systems. Data movement, rather than computation, is the big consumer of energy, yet software often moves data up and down the memory hierarchy or across a network multiple times. At the same time, hardware designers need to take into account the constraints of the computational problems that will run on their systems, as a design that is poorly matched to the computational requirements will end up being inefficient. Drawing on my own experience in scientific computing, I will give examples of how to make the combination of hardware, algorithms and software more efficient, but also describe some of the challenges that are inherent in the application problems we want to solve. The community needs to take an integrated approach to the problem, and consider how much business or science can be done per Joule, rather than optimizing a particular component of the system in isolation. This will require rethinking the algorithms, programming models, and hardware in concert, and therefore an unprecedented level of collaboration and cooperation between hardware and software designers.

Categories & Subject Descriptors: C.1.2 [Computer Systems Organization]

General Terms: Design

Bio

Katherine Yelick is the Director of the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory and a Professor of Electrical Engineering and Computer Sciences at the University of California at Berkeley. She is the author or co-author of two books and more than 85 refereed technical papers on parallel languages, compilers, algorithms, libraries, architecture, and storage. She co-invented the UPC and Titanium languages and demonstrated their applicability across architectures through the use of novel runtime and compilation methods. She also developed techniques for self-tuning numerical libraries, such as the OSKI sparse matrix library, which automatically adapt the code to machine properties. Her work includes performance analysis and modeling as well as optimization techniques for memory hierarchies, multicore processors, communication libraries, and processor accelerators. She has worked with interdisciplinary teams on application scaling, and her own applications work includes parallelization of a model for blood flow in the heart. She earned her Ph.D. in Electrical Engineering and Computer Science from MIT and has been a professor of Electrical Engineering and Computer Sciences at UC Berkeley since 1991 with a joint research appointment at Berkeley Lab since 1996. She has received multiple research awards, as well as teaching awards from both UC Berkeley and MIT.

ISCA'09, June 20-24, 2009, Austin, Texas, USA.
ACM 978-1-60558-526-0/09/06.