Keynote Address

Software Knows Best: Portable Parallelism Requires Standardized Measurements of Transparent Hardware

David A. Patterson
University of California, Berkeley

ABSTRACT

The hardware trend of the last 15 years of dynamically trying to improve performance with little software visibility is not only irrelevant today, its counterproductive; adaptivity must be at the software level if parallel software is going to be portable, fast, and energy-efficient. A portable parallel program is an oxymoron today; there is no reason to be parallel if it's slow, and parallel can't be fast if it's portable. Hence, portable parallel programs of the future must be able to understand and measure /any/ computer on which it runs so that it can adapt effectively, which suggests that hardware measurement should be standardized and processor performance and energy consumption should become transparent.

In addition to software-controlled adaptivity for execution efficiency by using techniques like autotuning and dynamic scheduling, modern software environments adapt to improve /programmer/ efficiency [1]. Classic examples include dynamic linking, dynamic memory allocation, garbage collection, interpreters, just-in-time compilers, and debugger support. Examples that are more recent are selective embedded just in time specialization (SEJITS) [2] for highly productive languages like Python and Ruby. Thus, the future of programming is likely to involve program generators at many levels of the hierarchy tailoring the application to the machine. These productivity advances via adaptivity should be reflected in modern benchmarks: virtually no one writes the statically linked, highest-level-optimized C programs that are the foundation of most benchmark suites.

The dream is to improve productivity without sacrificing too much performance. Indeed, how often have you heard the claim that a new productive environment is now "almost as fast as C" or "almost as fast as Java?" The implication of the necessary tie between productivity and performance in the manycore era is that these modern environments must be able to utilize manycore well, or the gap between highly efficient code and highly productive code will grow with the number of cores.

For industry's bet on manycore to win, therefore, both very high level and very low level programming environments will need to be able to understand and measure their underlying hardware and adapt their execution so as to be portable, relatively fast, and energy-efficient.

Hence, we argue that a standard of accurate hardware operation trackers (SHOT) would have a huge positive impact on making parallel software portable with good performance and energy efficiency, similar to the impact of the IEEE-754 standard had on portability of numerical software. In particular, we believe SHOT will lead to much larger improvements in portability, performance, energy efficiency of parallel codes than recent architectural fads like opportunistic "turbo modes," transactional memory, or reconfigurable computing.

Categories and Subject Descriptors: B.8.2 [Hardware]: Performance Analysis and Design Aids; D.1.3 [Software]: Concurrent Programming.

General Terms: Performance, Measurement.

Keywords: Measurements, Adaptivity, Parallelism.

1. REFERENCES

[1] Krste Asanović, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, Katherine Yelick. A View of the Parallel Computing Landscape. *Communications of the ACM*, 52(10), October 2009.

[2] B. Catanzaro, S. Kamil, Y. Lee, K. Asanovic, J. Demmel, K. Keutzer, J. Shalf, K. Yelick, A. Fox. SEJITS: Getting Productivity AND Performance With Selective Embedded JIT Specialization. *First Workshop on Programmable Models for Emerging Architecture (PMEA) at the 18th International Conference on Parallel Architectures and Compilation Techniques (PACT'09)*, Raleigh, North Carolina, September 2009.

WOSP/SIPEW'10, January 28–30, 2010, San Jose, California, USA.
ACM 978-1-60558-563-5/10/01.