The Yin and Yang of Hardware Heterogeneity: Can Software Survive?

Kathryn S. McKinley
Microsoft Research

Abstract

Power and energy constraints are now the driving force in devices from smartphones to servers. Quantitative power, performance, and energy measurements suggest that hardware heterogeneity to match software diversity has the potential to deliver energy efficiency. However, programming heterogeneous hardware directly is a nightmare. We discuss some approaches and results that abstract, choose, and exploit hardware heterogeneity. New programming and system abstractions are essential for establishing a parallel heterogeneous ecosystem in the post-Dennard era.

Categories and Subject Descriptors C.1.0 [*Computer Systems Organization*]: Processor Architectures; D.3.0 [*Programming Languages*]:

Categories and Subject Descriptors Performance, Languages, Measurement

Keywords Power, Energy, Heterogeneous Hardware, Managed Languages, Virtual Machines

A New Computing Landscape

Over the past fifty years, computer software and hardware erected layers of abstraction to hide complexity and innovate rapidly. As processor speeds increased, the costs of abstraction reduced in each hardware generation. Unfortunately, physical limits in hardware are forcing a disruptive break in this virtuous cycle of innovation, in part by breaking a key abstraction: sequential instruction execution.

Hardware Moore's law predicts that technology advances will shrink transistors, increasing the number of transistors in the same silicon area. Shrinking transistors lowers their gate delay and until recently translated into faster clock frequencies, as predicted by Dennard scaling. The result was exponential single processor performance gains. Between 2000 and 2010, physical limits of silicon made shrinking transistors harder, in part due to power leakage, which Dennard scaling did not model. The result was the end of clock scaling. To continue to improve processor capabilities with smaller transistors, hardware vendors turned to parallelism and introduced chip multiprocessors.

Looking forward, physical limits on power, area, and wires will require even more radical solutions. Already emerging from research and industry are heterogeneous processors that improve capabilities using specialized processors and multiple types of general-purpose processors that offer a range of power and performance characteristics. Power constraints also limit memory scaling. Solutions to meeting the ever present demand for more and faster memories are likely to combine non-volatile storage and traditional DRAM. Parallel heterogeneous hardware requires parallel and differentiated software components, exposing software developers to ongoing hardware upheaval.

Software As in many good partnerships, software encourages, drives, and exploits improving hardware capabilities. Furthermore, hardware capabilities encourage, drive, and exploit substantial software innovations. This ecosystem heralded new ways to sell, deploy, and interact with hardware over the past two decades. The entire computing ecosystem changed to meet demands for portability, reliability, security, correctness, new features, and time-to-market.

These demands drove developers away from native compiled ahead-of-time languages. Developers instead increasingly choose high-level managed programming languages with safe pointer disciplines, garbage collection, extensive standard libraries, and dynamic just-in-time compilation for hardware portability. For example, modern web services deliver software as a service in the browser by combining managed languages, such as JavaScript on the client side and PHP on the server side. In markets as diverse as finance and mobile, Java and C# dominate. This ecosystem produced an explosion of capabilities and an expanding base of developers that continue to change how we live and learn. Improving hardware capabilities hid much of the cost of software abstractions, whereas software virtual machine technologies hid hardware dependence and complexity.

The result was a virtuous cycle with ever more capable and high-level computing systems. However, post-Dennard era processors break this virtuous cycle by changing abstractions and changing assumptions built into hardware and software abstraction layers, as depicted in Figure 1.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPLASH'13 October 26-31, 2013, Indianapolis, IN, USA
Copyright © ACM ACM 978-1-4503-1995-9/13/10. . . $10.00

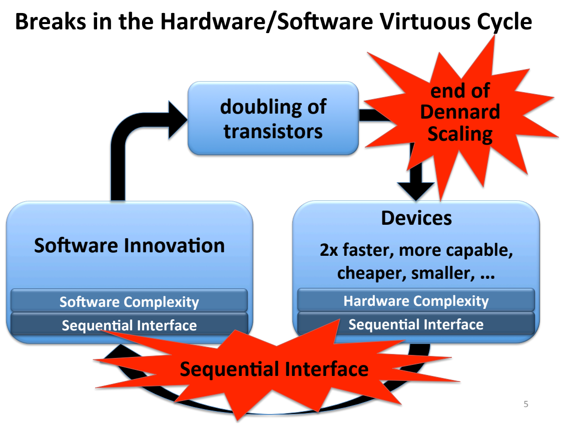

Figure 1. The virtuous cycle of hardware/software innovation is broken by the end of Dennard scaling and the introduction of a new hardware abstraction.

Moving Forward The economics of commodity hardware and the potential for efficiency through specialization create a tension for the designers of future systems. The more specialized the hardware, the higher the potential, but the more likely only a small set of applications will benefit. The less specialized the hardware is, the less the potential improvement, but the more applications that may benefit. At the limits, commodity processors execute every appliation, but are inefficient, whereas an application specific ASIC executes a single application very efficiently. These practical concerns motivate heterogeneous processors that use the same instruction set architectures (ISAs) but have different power and performance characteristics. For example, combining a few big cores with numerous little cores to meet power constraints, where big cores deliver the highest possible performance but are power hungry, and multiple little cores deliver throughput using substantially less power but with less performance. Vendors, such as NVIDIA, TI, and Qualcomm, already are shipping big/little heterogeneous systems.

To create a new virtuous cycle for modern systems ranging from smartphones to data centers requires new abstractions. The best abstractions meet current design goals and capabilities, but ease, encourage, and even create the potential for future innovations. Whereas in the Dennard era, performance was the primary design goal, the post-Dennard era adds power and energy to every market segment, not just mobile and embeded. Optimizing for performance, power, and energy (energy = power $\times$ time) is substantially more challenging than optimizing solely for performance.

Creating a new energy-efficient ecosystem requires deep understanding of the power, performance, and energy characteristics of software and optimizing for the hardware transparently. Whereas the programming language, system, and architecture communities understand how to measure and optimize performance, systematic power and energy measurements are not yet common. We show how power measurements lead to new insights about energy efficiency on modern hardware, and the differences between native, managed, parallel, and sequential workloads. With deeper understanding of capabilities in hand, we explore potential ways to optimize, matching software diversity to heterogeneous hardware using existing and new abstractions.

Heterogeneous hardware requires software with differentiated characteristics, parallelism (yes, still!), and ubiquity. We identify and exploit these characteristics to improve energy efficiency in case studies ranging from managed applications to smartphones to interactive services in data centers. For managed applications, we hide overheads from virtual machine services (such as garbage collection and just-in-time compilation), and optimize mapping threads to cores by learning properties such as parallelism, criticality, and sensitivity to core choice. For smartphones, we learn and optimize the tradeoff of responsiveness and battery drain. For interactive cloud services, such as search, we learn and optimize the quality and latency tradeoff, improving performance and energy efficiency. These results offer several insights for optimizing future systems. (1) Dynamically monitoring and adapting to the application is critical to energy efficiency. (2) Although software performance improvements on a particular hardware platform almost always lead to energy efficiency improvements, exploiting tradeoffs in latency and quality of service inherent in many emerging applications, both at design time and on deployed hardware, has the potential to improve energy efficiency more substantially.

These results offer hope that software may perhaps thrive as heterogeneous hardware evolves in the post-Denard era.

Acknowledgments and References My thanks to my collaborators at Microsoft, The University of Texas at Austin, and the Australian National University; and in particular to Steve Blackburn (ANU) and Yuxiong He (Microsoft). For our publications, please see `http://research.microsoft.com/en-US/people/mckinley/papers.aspx`.

Kathryn S. McKinley is a Principal Researcher at Microsoft Research and served on the faculty of The University of Texas at Austin (2001-2013). She likes to solve programming language implementation problems and make systems programmable, fast, reliable, secure, and energy efficient. Dr. McKinley and her collaborators produced the widely used DaCapo Java Benchmarks, TRIPS Compiler, Hoard memory manager, MMTk garbage collector toolkit, and Immix garbage collector. She is co-Chair of the CRA Committee on the Status of Women in Computing Research, and has testified on how to sustain the information technology innovation ecosystem to the U.S. House Science Committee. Dr. McKinley has graduated 17 PhD students. She is married to artist Scotty Strahan and they have three sons. She is an IEEE Fellow and an ACM Fellow.