INTERACTION OF MANY-CORE COMPUTER ARCHITECTURE AND OPERATING SYSTEMS

Sangyeun Cho
University of Pittsburgh

Tao Li
University of Florida

Onur Mutlu
Microsoft Research

Practices and conventions in designing a microprocessor are undergoing dramatic changes. Squeezing more performance from a single, complex, and power-hungry core has become a difficult proposition, whereas incorporating multiple processor cores on a silicon die is relatively simple due to the continuing advances in process technology. Major microprocessor vendors are currently marketing multicore processors, carrying two to eight cores and supporting dozens of threads simultaneously. Increasing the number of cores in a processor chip with each process technology generation is now commonplace, and the era of "many-core" processors is around the corner.

Efficiently harnessing a many-core processor's raw horsepower and intelligently managing many on-chip resources is a difficult challenge. To achieve high performance-power and performance-area ratios, for instance, multicore processors are beginning to employ heterogeneous processor cores and specialized function blocks. However, software-based management of such heterogeneous chips is very complicated. On-chip network and distributed shared resources such as L2 cache banks and DRAM memory cause considerable performance asymmetry and lack-of-performance guarantees for coscheduled software threads. Various manufacture-time and runtime variability and technology-induced reliability issues pose further complications to operating next-generation many-core processors robustly and efficiently via the existing system software. In addition, the trend toward incorporating virtualization into the software-hardware stack will require new approaches for the efficient division of labor between hardware and software.

The operating system (OS) or system software in general, including hypervisors or virtual machine monitors, is the centerpiece of a computer system managing various platform hardware resources. The rapid changes in the platform hardware resources with the evolution of many-core architectures will require a fundamental reexamination of mainstream system-software design decisions to support multiple cores and to efficiently manage on-chip hardware resources shared among the multiple cores. In turn, the evolution of many-core processor architectures will be successfully sustained by the new capabilities and features added to the system software, perhaps while requiring substantial support from hardware. Many important processor resource management issues—including support for quality of service (QoS) and differentiated services, fault tolerance and faulty resource isolation, power and thermal control, and efficient virtual machine support—could be best dealt with by a combination of or cooperation between novel and well-con-

Published by the IEEE Computer Society

ceived system software and hardware architecture techniques. Synergistic and efficient interaction between the system software and new many-core architectures will be essential to the successful design of efficient and robust computer systems of the future.

The purpose of this special issue of *IEEE Micro* is to bring to the readers the latest advances in the interface of system software and computer architecture, with a focus on how architecture design affects system software and vice versa, to achieve increasingly involved platform and system design goals. We are pleased to introduce a collection of five articles that highlight some of the important aspects of the interaction between computer architecture and system software. These articles span topics including hardware-software cooperative management of shared on-chip resources, implications of dynamically heterogeneous multicore processors for thread scheduling, the use of asymmetric multicore processors to save energy in system software, system performance metrics, and operating system improvements to achieve higher throughput in existing multicore systems. Our hope is that the problems and solutions described in these articles will attract more researchers to rethink the interface between computer architecture and system software.

Editorial preview

The first article presents an interface between software applications and multicore hardware: virtual private machines. VPMs virtualize a system's shared physical resources. In "Multicore Resource Management," Nesbit et al. describe two orthogonal methods for virtualization—spatial and temporal—and discuss how system resources are allocated to applications based on application demands. The authors describe the components of the VPM abstraction, which enables the separation of application and system policies from underlying mechanisms. The article concludes that VPMs are a way to bridge the gap between real-time application demands and the underlying shared hardware resources.

The second article describes the notion of "dynamic heterogeneity," the performance or functionality asymmetry of cores introduced by runtime events such as dynamic changes in voltage and frequency or hardware faults. In "The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling," Bower, Sorin, and Cox describe the challenges posed by dynamically heterogeneous multicore processors to operating system designers. In particular, the article focuses on how performance can be improved by making thread scheduling aware of dynamic heterogeneity. The authors describe open research problems in efficiently dealing with dynamic heterogeneity and issue a call for action to the research community to solve these problems by rethinking the OS-architecture interface.

The next article also deals with heterogeneity in multicore architectures and its application to saving energy in system software. "Using Asymmetric Single-ISA CMPs to Save Energy on Operating Systems," by Mogul et al., proposes the use of performance-asymmetric multicore systems in which some cores are specialized to execute operating system code. The authors observe that operating systems do not use many of the power-consuming features intended to improve application performance, so that designing specialized, less powerful, "OS-friendly" cores can improve energy efficiency. The article examines how the operating system can use OS-friendly cores and proposes a way to switch execution to these cores when the OS kernel is called. The authors also elaborate on the design trade-offs involved in designing such OS-friendly cores. Their preliminary results show that OS-friendly cores make possible a significant energy savings in operating system code.

The fourth article revisits the definition of performance for multiprogram workloads and argues that multiprogram performance metrics should be derived in a top-down manner starting from system-level performance objectives such as program turnaround time and system throughput. Eyerman and Eeckhout further propose in "System-Level Performance Metrics for Multiprogram Workloads" two performance metrics: average normalized turnaround time (ANTT) as a user-oriented

performance metric and system throughput (STP) as a system-oriented performance metric. They present a case study that compares simultaneous multithreading (SMT) processor fetch policies to illustrate the general insights that can be gained by using the described performance metrics.

Our last article describes how system software can be modified to accommodate multicore systems. In ''Using OS Observations to Improve Performance in Multicore Systems,'' Knauerhase et al. describe how OS observations on task behavior can help the OS make effective scheduling decisions to achieve higher system throughput. Their ''observation subsystem,'' called OBS-M, inspects hardware performance counters and kernel data structures and gathers information on a per-thread basis. The authors studied a series of scheduling policies that exploit OS observations, with the purpose of mitigating performance variability caused by the shared last-level cache and the processors' functional asymmetry—for example, lack of floating-point units on specific cores. The authors' evaluations on multicore systems running modified Linux and Mac OS X demonstrate that relatively simple modifications to the existing scheduling policies can result in significant performance improvements.

We hope you enjoy these articles, and we very much welcome your feedback on this special issue.

Article selection process and reviewers

We received 14 high-quality articles for consideration in this special issue. We thank all the authors who took the time to submit their manuscripts to *IEEE Micro*. Each article was reviewed by at least three expert reviewers, and most articles received at least four expert reviews. Based on these reviews, we, the Guest Editors, made our recommendations to David Albonesi and Ruby Lee, *IEEE Micro*'s Editor in Chief and Associate Editor in Chief, who then made final decisions. This rigorous review process would not have been possible without the industrious efforts made by the following special issue reviewers, whom we gratefully acknowledge: Murali Annavaram (Intel), Krste Asanović (UC Berkeley), Brad Beckmann (AMD), Laxmi Bhuyan (UC Riverside), Martin Burtscher (UT Austin), Francisco Cazorla (Barcelona Supercomputing Center), Dan Connors (Colorado), Alexandra Fedorova (Simon Fraser), Mike Gschwind (IBM TJ Watson), Ravi Iyer (Intel), Hyesoon Kim (Georgia Tech.), Alvin Lebeck (Duke), Hsien-Hsin Lee (Georgia Tech.), Beng-Hong Lim (VMWare), Avi Mendelson (Intel), Chuck Moore (AMD), Nacho Navarro (UPC), Vijay Pai (Purdue), Moinuddin Qureshi (IBM TJ Watson), Rodric Rabbah (IBM TJ Watson), Steve Reinhardt (Reservoir Labs/Michigan), Scott Rixner (Rice), Mike Schlansker (HP Labs), Yan Solihin (NC State), Daniel Sorin (Duke), Karin Strauss (AMD and University of Washington), Edward Suh (Cornell), Mike Swift (Wisconsin), Rajeev Thakur (Argonne National Lab), Mithuna Thottethodi (Purdue), Manish Vachharajani (Colorado), Kushagra Vaid (Microsoft), Mateo Valero (UPC), Brad Waters (Microsoft), Emmett Witchel (UT Austin), and Jun Yang (Pittsburgh). MICRO

Acknowledgments

It has been a pleasure to put together this special *Micro* issue on the interaction of computer architecture and operating systems in the many-core era. The Guest Editors thank the Editor in Chief David Albonesi, Associate Editor in Chief Ruby Lee, and the *IEEE Micro* staff—especially Lindsey Buscher and Margaret Weatherford—for their support and guidance. This special issue would not have been possible without their help.

Sangyeun Cho is an assistant professor in the Computer Science Department at the University of Pittsburgh. His primary research interests are in computer architecture, system software design, and their synergy. His current projects investigate various ways to improve performance, reliability, and power/energy of next-generation many-core processors. He was a processor architect and designer at Samsung Semiconductor and participated in the commercially successful CalmRISC project. He has a BS in computer engineering from

Seoul National University, and an MS and PhD from the University of Minnesota, Minneapolis. He is a member of the ACM and the IEEE.

Tao Li is an assistant professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests include computer architecture, dependable computing systems, and the impacts of emerging techniques and applications on computer designs and operating systems. Li received 2006 and 2007 IBM Faculty Awards. He has a PhD in computer engineering from the University of Texas at Austin.

Onur Mutlu is a researcher at Microsoft Research. His research interests include computer architecture, especially the interaction and cooperation between programmers, languages, operating systems, compilers, and architectures. Mutlu has BS degrees in computer engineering and psychology from the University of Michigan, Ann Arbor, and MS and PhD degrees in electrical and computer engineering from the University of Texas at Austin. He was a recipient of the Intel PhD fellowship and the University of Texas George H. Mitchell Award for Excellence in Graduate Research. He is a member of the IEEE and the ACM.

Direct questions and comments about this special issue to Sangyeun Cho, 5407 Sennott Square, Pittsburgh, PA 15260; cho@cs.pitt.edu.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/csdl.