

Challenges and Opportunities at All Levels

Interactions Among Operating Systems, Compilers, and Multicore Processors

Kim Hazelwood
University of Virginia
hazelwood@acm.org

Mohamed Zahran
City University of New York
mzahran@acm.org

1. Introduction

The tremendous advances in process technology have opened the doors to many opportunities, but have also introduced a wealth of challenges. On one hand, microprocessor designers can pack several cores, whether homogeneous or heterogeneous, on a single chip, enabling more parallelism. On the other hand, programmers must focus on producing parallel programs since automatic parallelization is still in its infancy, especially for non-scientific code. System designers must also focus on managing resources to optimize performance and minimize the impact on power and reliability. The increase in on-chip transistors results in increased power dissipation, both static and dynamic, and gives rise to other challenges related to parameter variations.

The main question facing the entire computing community is how to make use of all the available cores in the most efficient and cost-effective way. This challenge can be addressed by changing the way multicore processors, compilers, and operating systems interact. This interaction is the main topic of this special issue.

2. Open Challenges Today

In the multicore era, many challenging research questions exist. Some are tackled by the papers in this issue, some are open challenges waiting to be solved. Still other challenges have yet to surface! Some of the open questions at this juncture include the following:

- Is automatic parallelization of general-purpose applications possible with the current programming models?
- What does the compiler need to know about the underlying hardware to do the best job at optimizing programs?
- What does the hardware need to know from the compiler for best performance?
- What is the role of virtualization for multicore processors?
- How can the operating system juggle various, often conflicting goals, such as performance, power, temperature, and reliability?
- Should we parallelize the operating system kernel itself?

- How should the memory hierarchy be designed in a manycore architecture?
- Will bandwidth be our next wall?
- Can we effectively leverage 1000 on-chip cores, or should we stop now?

We formed this special issue to serve two main purposes. First, we wanted to highlight the latest advances in cross-layer integration, since much of the work appearing in this issue tends to be distributed across several communities. Second, we hope that this special issue will spark the imagination of our readers.

We begin the issue with two invited papers. The first paper, by Philip M. Wells, Koushik Chakraborty, and Gurindar S. Sohi, proposes multicore virtualization. This technique allows the hardware designer to abstract the low-level details of the cores to alleviate software from the burden of managing these resources directly. This idea decouples the tasks of low-level core management and high-level concurrency extraction, and hence facilitates the evolution of manycore chips. The second invited paper, by Vijay Nagarajan and Rajiv Gupta, proposes OASES: OS and Architectural Support for Efficient Shadow memory implementation for multicore processors. OASES reduces the overheads of runtime monitoring tasks by exposing cache events to the software. Runtime monitoring will be critical for software to adapt to changing resource requirements and contention.

Following the invited papers are a collection of seven regular papers presenting bleeding-edge advances in software and hardware. We partition the papers into two groups – hardware solutions that have support from various software layers, and software solutions that have support from hardware.

3. Software-Assisted Hardware Solutions

In the first regular paper, titled *Supporting MapReduce on Large-Scale Asymmetric Multi-Core Clusters*, Rafique, Rose, Butt, and Nikolopoulos explore the design space of hybrid clusters. These clusters are built with general purpose processors (GPP) and asymmetric core processors (AMP). AMP is a multicore design built from heterogeneous cores

and geared toward high-end computing. Managing GPPs and AMPs in a single cluster is a very challenging task. The authors explore the hardware design as well as the software stack, and use this design space exploration to implement the MapReduce programming model.

In the second paper, Strong, Mudigonda, Mogul, Binkert, and Tullsen address the software cost of thread migration in their paper *Fast Switching of Threads Between Cores*. They discuss several techniques that can decrease switching cost, and they indeed cut Linux core-switching latencies in half, which has many benefits such as load balancing and thermal management.

4. Hardware-Assisted Software Solutions

Performance monitoring is crucial if we want to enhance the performance of hardware systems, optimize software systems, or match software behavior to hardware capability. As a result, performance monitoring has been featured in many studies both in academia and industry. Bratanov, Belenov, and Manovich present an interesting study about performance monitoring in virtual machines. Their paper, *Virtual Machines: A Whole New World For Performance Analysis*, collects hardware performance data while simultaneously tracking software states inside a virtual machine. Their methodology allows any commercially available hosted virtual machine to be used for software analysis and optimization.

Also on the topic of performance monitoring, Azimi, Tam, Soares, and Stumm show how hardware performance monitors can be used to provide a fine-grained feedback loop to dynamic optimizations by a multicore-aware operating system. Their paper *Enhancing Operating System Support for Multicore Processors by Using Hardware Performance Monitoring* proposes a scheme that results in several benefits: an increase in IPC, a decrease in cache misses, and improvements at the application level.

The next paper in the collection is *HASS: A Scheduler for Heterogeneous Multicore Systems*. HASS makes the operating system aware of any single-ISA heterogeneity present in the underlying multicore processor. Using their corresponding scheduling algorithm, the authors achieve up to 13% speedup.

As we can see from the above papers, the role of the operating system is crucial in the multicore era. As such, Wentzlaff and Agarwal propose a factored operating system (fos) in their paper, *Factored Operating Systems (fos): The Case for a Scalable Operating System for Multicores*. After

documenting the scalability problems of current operating systems, the authors propose a new operating system that can scale to large number of cores.

FlexDCP: a QoS framework for CMP architectures is another paper focusing on operating systems. It deals with the very important aspect of quality of service (QoS) as provided by the OS to the applications. The authors propose FlexDCP, a framework that allows the operating system to guarantee a QoS for each application running in a chip multiprocessor. The main idea is to allow the OS to convert QoS requirements into resource assignments. In an eight-core architecture, FlexDCP obtains a fairness improvement of 10% over the best policy in the literature that optimizes fairness.

5. A Glimpse into our Future

In addition to the invited and regular papers, we include a collection of five position papers, each of which represents a two-page forward-looking vision. One paper presents a scheme to auto-tune general purpose applications to make the best use of the available multicore machine. Another paper proposes a hardware transactional memory for multicore environment. A third paper discusses a run-time system that can automatically adapt programs to the architecture and usage environment. The fourth paper presents an infrastructure that can autonomously tune the efficiency of a data center or satisfy the end-users' need when running network-based applications. The fifth paper argues for the usefulness of off-loading some fraction of system calls and other OS functionality to a separate special-purpose core in a multicore architecture. Each of these position papers provides a potential glimpse into the future of multicore design solutions.

6. What's Next?

The papers in this special issue give us a unique opportunity to look into a crystal ball and try to imagine future systems. There still remain many open problems that need the creativity of researchers in computer architecture, compilation technology, and operating systems. Technology may give us hundreds of on-chip cores, but can the operating systems manage them? Can we write parallel programs to achieve the best performance on different multicore chips? Which layers need adaptation and reconfigurability: the hardware, the operating system, or the applications? The papers presented here are taking the first step. The complete solution requires additional creativity from researchers in computer architecture, compilation techniques, and operating systems.