The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling

Journal:	*IEEE Micro*
Manuscript ID:	MicroSI-2008-01-0004
Manuscript Type:	Special Issue May/June 08 Interaction of Computer Architecture and OS (submissions due 5 Jan 2008)
Date Submitted by the Author:	04-Jan-2008
Complete List of Authors:	Bower, Fred; Duke University, Computer Science Sorin, Daniel J.; Duke University, ECE Cox, Landon; Duke University, Computer Science
Keywords:	C.0.b Hardware/software interfaces < C.0 General < C Computer Systems Organization, C.0.a Emerging technologies < C.0 General < C Computer Systems Organization, C.1.2 Multiple Data Stream Architectures (Multiprocessors) < C.1 Processor Architectures < C Computer Systems Organization, C.1.4.e Multi-core/single-chip multiprocessors < C.1.4 Parallel Architectures < C.1 Processor Architectures < C Computer Systems Organization

The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling

Fred A. Bower[1,2], Daniel J. Sorin[3], and Landon P. Cox[2]

bowerf@us.ibm.com, sorin@ee.duke.edu, lpcox@cs.duke.edu

[1]IBM Systems and Technology Group, System x Development

[2]Duke University, Department of Computer Science

[3]Duke University, Department of Electrical and Computer Engineering

Contact author: Daniel J. Sorin
PO Box 90291
Durham, NC 27708
phone: 919-660-5439
fax: 919-660-5293
sorin@ee.duke.edu

Abstract

The computer industry has turned to multicore processors as a power-efficient way to use the vast number of transistors on a chip. Most current multicore processors are homogeneous (i.e., all the cores are identical), and scheduling them is similar, but not identical, to what operating systems have been doing for years. However, microarchitects are proposing heterogeneous core implementations, including systems in which heterogeneity is introduced at runtime. These processors will require schedulers that can adapt to heterogeneity.

In this position paper, we discuss the trends that are leading to dynamic heterogeneity, and we show that schedulers must consider dynamic heterogeneity or suffer significant power-efficiency and performance losses. We present the challenges posed by dynamic heterogeneity, and we argue for research into hardware and software support for efficiently scheduling such systems.

1 Introduction

Moore's Law provides computer architects with more transistors than they can effectively use to extract instruction level parallelism in a single core. Thus, all current and future high-performance processor chips are *multicore processors* (also known as *chip multiprocessors* or *CMPs*). These multicore processors include the Cell Broadband Engine [11], Intel's CoreDuo and Quad-Core Xeon, AMD's Dual-Core Opteron, Sun Microsystems' Niagara [13], and IBM's Power5 [12]. These processors have between two and eight cores in a single chip package, with the expectation of greater numbers of cores in future generations.

At first glance, scheduling a multicore processor may not appear to present a substantially new problem for operating systems. There is a long history of OS scheduling for multithreaded microprocessors and traditional multi-chip multiprocessors. There has even been recent research [9, 8] into one aspect of scheduling that is unique to multicores, which is that processors often share L2 caches. Except for this issue of cache sharing, it might at first appear that scheduling of multicore processors would be a straightforward extension of existing scheduling techniques, *except future multicore processors are unlikely to be composed of homogeneous cores* [15, 1]. Due to core specialization and runtime fault handling and power management, it is likely that multicore processors will feature heterogeneous cores. Furthermore, this heterogeneity is likely to be both static (as an intentional design feature that does not change) and dynamic (as a response to run-time events such as physical faults or power management). In this position paper, we focus on dynamically heterogeneous multicore processors (DHMPs), because they will present a greater challenge to future operating systems.

In a DHMP, the OS scheduler has a significant impact on power efficiency and performance. The scheduler must decide which threads (or portions of threads) should run on which cores. A good schedule will match each thread with a core that can provide it with sufficient performance at an acceptable power cost. A poor schedule will match each thread with a core that either cannot run it at an acceptable performance (e.g., due to faults in that core) or that needlessly wastes power running it. A poor schedule can lead to shorter battery life for a laptop, slower response time for a gaming console, greater power costs for small business computing, or less computational throughput for a server farm. Scheduling a DHMP is a fundamentally different and more difficult problem than scheduling homogeneous systems.

In the rest of this paper we further motivate and define the research challenges presented by DHMPs. Section 2 reviews multicore architecture to frame the issues related to scheduling these systems. In Section 3, we present the challenges we see for the efficient scheduling of DHMPs. In Section 4, we discuss the limited research that has been done in this area. We conclude in Section 5 with a call to action for scheduling research, outlining fruitful future areas of research.

2 Multicore Trends and Impact

With the increasing transistor budgets afforded by Moore's Law, architects have sought ways to use all of them in a power-efficient manner. Until recently, architects have dedicated their transistor budget to extracting instruction level parallelism (ILP) out of single-threaded code. However, dedicating current transistor budgets strictly to ILP is not power-efficient, and thus architects have sought to use transistors to also exploit thread level parallelism. An initial approach was simultaneous multithreading (SMT) [18], such as in the Pentium4, in which multiple threads share a single core's resources. Unfortunately, a single SMT processor is also limited in how many transistors it can use power-efficiently. Thus, the industry has begun placing multiple independent cores on each chip. Each of these cores may itself be multithreaded, providing a multiplicative number of schedulable contexts.

Homogeneous multicore processors are composed of identical cores that provide a consistent computing capability for each schedulable context. Homogeneity simplifies the job of the scheduler and allows us to use existing scheduling algorithms for multiprocessor systems. These algorithms may factor cache warmth and cache sharing into scheduling decisions, but they generally do not discriminate amongst cores, as they are all viewed as equally capable of performing computations.

2.1 Sources of Heterogeneity

The primary problem with homogeneous multicore processors is that naive replication of state-of-the-art single-core designs in a single package (or chip package) stress the power and cooling limits for the chip. There is a fixed amount of power that a chip can consume before we cannot cool it (with air cooling). Given this power budget, a chip cannot contain dozens of Pentium4-like processors, even if each one is itself power-efficient. Nevertheless, for high-priority tasks, we still want the single-threaded performance provided by current high-performance cores. Thus, architects believe that (statically) heterogeneous multicore designs, such as the Cell Processor [11], will be prevalent in coming generations [15, 1]. As illustrated in Figure 1, a multicore design might consist of a small number of high-power and high-performance cores, coupled with a