Bias Scheduling in Heterogeneous Multi-core Architectures

David Koufaty Dheeraj Reddy Scott Hahn
Intel Labs
{david.a.koufaty, dheeraj.reddy, scott.hahn}@intel.com

Abstract

Heterogeneous architectures that integrate a mix of big and small cores are very attractive because they can achieve high single-threaded performance while enabling high performance thread-level parallelism with lower energy costs. Despite their benefits, they pose significant challenges to the operating system software. Thread scheduling is one of the most critical challenges.

In this paper we propose bias scheduling for heterogeneous systems with cores that have different microarchitectures and performance. We identify key metrics that characterize an application bias, namely the core type that best suits its resource needs. By dynamically monitoring application bias, the operating system is able to match threads to the core type that can maximize system throughput. Bias scheduling takes advantage of this by influencing the existing scheduler to select the core type that bests suits the application when performing load balancing operations.

Bias scheduling can be implemented on top of most existing schedulers since its impact is limited to changes in the load balancing code. In particular, we implemented it over the Linux scheduler on a real system that models microarchitectural differences accurately and found that it can improve system performance significantly, and in proportion to the application bias diversity present in the workload. Unlike previous work, bias scheduling does not require sampling of CPI on all core types or offline profiling. We also expose the limits of dynamic voltage/frequency scaling as an evaluation vehicle for heterogeneous systems.

Categories and Subject Descriptors D.4.1 [*Operating Systems*]: Process Management - Scheduling; C.1.3 [*Processor Architectures*]: Other Architecture Styles - Heterogeneous (hybrid) systems

EuroSys'10, April 13–16, 2010, Paris, France.

General Terms Algorithms, Performance

1. Introduction

Advances in semiconductor technology have enabled processor manufacturers to integrate more and more cores on a chip. Most multi-core processors consist of identical cores, where each core implements sophisticated microarchitecture techniques, such as superscalar and out-of-order execution, to achieve high single-thread performance. This approach can incur in high energy costs due to the power inefficiency of these techniques. Alternatively, a processor can contain many simple, low-power cores, possibly with in-order execution. This approach, however, sacrifices single-thread performance and benefits only applications with thread-level parallelism.

A heterogeneous system integrates a mix of *big* and *small* cores, and thus can potentially achieve the benefits of both [1, 2, 7–9, 12, 13, 20]. Despite their significant benefits in power and performance, heterogeneous architectures pose significant challenges to operating system design [4], which has traditionally assumed homogeneous hardware. Key among these challenges is scheduling. Homogeneous systems simplify the scheduler design by providing a uniform computing capability on each core.

Heterogeneous architectures can be classified into two types: *functional asymmetry* and *performance asymmetry*. Functional asymmetry refers to architectures where cores have different or overlapping instructions sets. For example, some cores may be general-purpose while others perform fixed functions such as encryption and decryption. Performance asymmetry refers to architectures where cores differ in performance (and power) due to differences in microarchitecture or frequency.

In this paper, we propose bias scheduling for performance asymmetric heterogeneous systems with cores that have different microarchitectures. We identify key metrics that characterize the potential benefits of scheduling an application on a big core over a small core, and the core type that best suits the resource needs of the application, namely its bias. By dynamically monitoring application bias, the scheduler is able to match threads to the core type that maximizes sys-