Mathematical Representation of the Hardware Round-Robin Scheduler Analytical Model for Single-ISA Heterogeneous Architectures

Daniel Nemirovsky†‡, Nikola Markovic†‡, Osman Unsal†, Mateo Valero†‡, Adrian Cristal†‡ł

Computer Architecture for Parallel Paradigms, Barcelona Supercomputing Center (BSC), Barcelona, Spain †
Department of Computer Architecture, Universitat Politecnica de Catalunya (UPC), Barcelona, Spain ‡
Artificial Intelligence Research Institute - Spanish National Research Council (IIIA-CSIC), Barcelona, Spain ł
(daniel.nemirovsky,nikola.markovic,osman.unsal,mateo.valero,adrian.cristal)@bsc.es

INTRODUCTION

Introduction Over the past few years, hardware chip manufacturers have been shifting strategies to overcome the well-known Power Wall which imposes the practical limitations to increasing CPU performance by raising the processor’s running frequency. The most recently applied schemes to exploit high single and multi-threaded performance at lower energy costs have been to increase the number of cores per chip as well as their diversity leading to what is commonly referred to as heterogeneous multi or many core architectures. In spite of the visible gains the architectures offer, heterogeneity poses significant challenges to the OS, one of the most critical being thread scheduling[1]. In this paper we provide contributions consisting of 1) a Hardware Round-Robin Scheduling policy motivated by [2] and derivation of a mathematical model for analyzing it, and 2) an analysis and comparison of the performance results obtained through our mathematical model with ones gathered via a simulator and real physical system. HRRS policy and mathematical model In order to effectively enhance the scheduling efficiency on a heterogeneous system within a realistic scope of implementation, we have sought a scheduling scheme that leaves the OS unmodified. To achieve this, we have provided the OS with an abstracted view of the heterogeneous physical hardware as being instead composed of four identical (homogeneous) logical cores. As a consequence, the OS scheduler will now map software threads onto the logical cores which enables the OS scheduling policies and implementation to be left unaltered. Meanwhile, as the OS scheduler maps threads to the logical cores, which may happen at every software-quantum or while handling interrupts, the Hardware Round-Robin Scheduler (HRRS), which is triggered into action at every hardware-quantum, maps the logical cores to the physical cores. In essence, the HRRS remaps the software threads that are abstractly assigned to a logical core by the OS to the physical cores of the underlying hardware which actually execute the threads.

$$T = \frac{W}{(1-C)*B+(N-2)*S+(1-C*\frac{B}{S})*S}*N*Q \quad (1)$$

Equation (1) represents the basic mathematical model for execution time of all workloads on all cores where N represents the total number of cores and hardware threads, Q is the scheduling quantum, W represents the workload per thread (equal per thread), B = CPI_large/Q , S = CPI_small/Q , and C<1 represents the fraction of the scheduling quantum spent in context switching. The overhead for migrating a workload from one core to another has three components: 1) a penalty for storing and restoring the architecture state (i.e., the registers which are at most a few kilobytes of state), 2) an overhead attributed to the time it takes to drain a core’s pipeline prior to migration, and 3) migration overhead due to cache compulsory misses and
sharing effects. The latter is by far the largest and most important component, being at least two order of magnitude larger in terms of latency cost than the first two components combined. In order to estimate the migration overhead we have compared the overhead on a real machine (Intel i7 quad-core Haswell micro-architecture) by applying the method described by Li C. et al. [3], with our simulation results which were obtained using the Sniper simulator [4] and SPEC2006 benchmark suite. Evaluation summary In order to be able to exploit time varying workload execution behavior while keeping migration overhead small, we set the time slice granularity (i.e., hardware scheduling quantum) to be 1ms, compared to the typical 4ms quantum used by the OS software scheduler. Based on evaluations from previous work as well as our own experimental results from running on a physical machine, we have assumed a fixed 3,000 cycle penalty for storing and restoring the architecture state. Our experiments were carried out on a real machine as well as with the Sniper simulator and used different scheduling quanta show that migration overheads ranges from 3,000 to 30,000 cycles for a workload less than 500KB and up to around 300,000 cycles for workloads greater than 500KB. Our simulations show average performance benefit of 11 percent for HRRS compared to the OS software scheduler. After including these migration overheads into our mathematical model we compare the model’s predicted performance of the HRRS scheme with results extracted using the simulator and got an error rate of less than 3 percent.

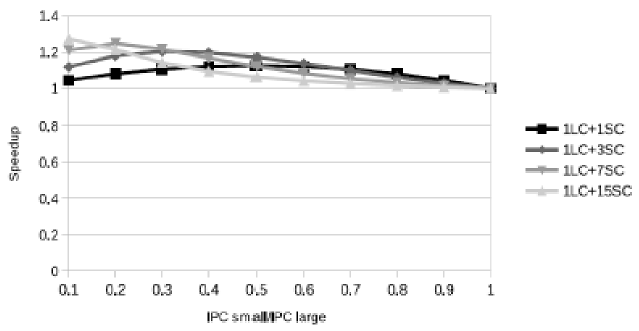

Figure 1. Analytical model comparing ideal HHRS performance with an ideal Linux OS scheduler on different heterogeneous system configurations

REFERENCES

[1] K. Van Craeynest et al. Scheduling heterogeneous multi-cores through performance impact estimation (pie). In Proc. ISCA, 2012.

[2] N. Markovic et al. Thread lock section-aware scheduling on asymmetric single-isa multi-core. In IEEE CAL, 2014.

[3] S. Li et al. Quantifying the cost of context switch. In Proc. Workshop Exp.Comput. Sci., 2007.

[4] T. Carlson et al. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In Proc. SC, 2011.