J. Parallel Distrib. Comput. 118 (2018) 128–139

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Designing lab sessions focusing on real processors for computer architecture courses: A practical perspective

Josué Feliu *, Julio Sahuquillo, Salvador Petit

Department of Computer Architecture (DISCA), Universitat Politècnica de València, Camí de Vera s/n, 46022, València, Spain

HIGHLIGHTS

- We present a new approach for computer architecture labs based on real processors.
- We discuss the methodology and scheduling framework to support the labs.
- Five lab examples studying different topics illustrate the scope of the approach.

ARTICLE INFO

Article history:
Received 15 June 2017
Received in revised form 2 February 2018
Accepted 26 February 2018
Available online 8 March 2018

Keywords:
Lab sessions
Computer architecture
Real processors
Processor complexity
Scheduling framework

ABSTRACT

Computer architecture courses typically include lab sessions to reinforce, from a practical perspective, concepts and architectural mechanisms studied in lectures. Lab sessions are mainly based on simulation frameworks because they benefit learning. Reading the source code that models certain processor mechanisms allows students to acquire a sound knowledge of how hardware works. Unfortunately, simulators that model current multicore processors are getting more and more complex, which lengthens the learning phase and complicates their use in time-bounded lab sessions.

In this paper, we propose a new approach that complements the use of simulation frameworks in lab sessions of computer architecture courses. This approach is based on performing experiments on current commercial processors, where multiple hardware events related to the performance of the computer components under study are monitored. Then, students analyze the measured events and how they impact the overall performance. Such analysis motivates students and, not only helps reinforcing the theoretical concepts, but also increases their analysis skills. In this paper we present the methodology and scheduling framework that support the proposed approach and discuss five lab sessions, which can be applied in different courses, covering multiple computer architecture topics.

1. Introduction

Most electrical and engineering schools around the world offer two or three courses on computer organization and computer architecture topics. These courses usually comprise both conventional lectures at classroom and practical sessions at laboratories. Computer architecture courses are typically considered as difficult courses by students mainly due to the wide range of topics that are covered as well as the intrinsic difficulty of some of them. In addition, topics are usually studied from a theoretical perspective, which discourages many students from continuing their education in computer architecture.

Lab sessions are an excellent way to reinforce the theoretical concepts taught at conventional lectures. They provide a clear understanding about how the computer mechanisms studied at lectures work, which helps correcting any possible misunderstanding and motivates these students that might feel discouraged at classrooms. To this end, labs use computer simulation frameworks like Multi2sim [21] or Snipper [6], which model complex processors and their structures in detail. Working on small fragments of the simulator source code helps students to appreciate the details about how specific processor mechanisms work, allowing them to acquire a sound knowledge about internal architectural mechanisms from a practical perspective. Therefore, simulators play an important role in post-graduate courses, especially when a major goal of the course is to provide research skills to students.

While simulators are valuable tools to study the details of hardware, they fail to provide an overview of how the distinct components of the machine interact to each other. For example, how last level cache (LLC) misses impact processor performance. One reason that explains this drawback is that simulating a current multicore with complex cores and a huge cache hierarchy is time

* Corresponding author.
E-mail address: jofepre@gap.upv.com (J. Feliu).

https://doi.org/10.1016/j.jpdc.2018.02.026
0743-7315/

consuming. In fact, running just a millisecond of execution on real hardware can take several hours with the simulator.

To deal with such problems, we propose lab sessions where students work on real hardware. The proposed labs make use of the hardware performance counters implemented in most recent processors from the major manufacturers Intel [14], AMD [9], IBM [18] or ARM [2]. Performance counters consist of a set of special purpose registers that allow tracking advanced processor events such as committed instructions, run cycles, memory accesses, or branch misses, among many others.

In summary, this paper presents a new approach to study computer architecture topics at lab sessions focusing on real hardware and makes two main contributions.

- First, we present a methodology that can be used as a guide in the preparation of computer architecture labs using performance counters. It is aimed at reducing the long time required to prepare and develop this kind of labs and considers both performance counters and common benchmark suites used in research. The methodology employs an adapted version of a research framework that has been successfully used in PhD thesis at our research group, and the proposed lab sessions are based on the authors' expertise acquired while doing research on computer architecture in commercial processors during the last decade. While performance counters have been widely used in current research, to the best of our knowledge this is the first time that their use is applied to computer architecture labs.
- Second, we discuss five lab examples covering different levels of difficulty, depending on the course level. The main novelty of these labs is that they study the different topics by measuring multiple hardware events related with the topics under study on current commercial processors. The presented labs are aimed at illustrating how lab sessions for computer architecture courses based on real machines can be designed. Instructors can use, if they consider appropriate, either a subset of the proposed labs or design their own labs. Nevertheless, we would like to remark that these labs do not intend to replace simulators, but both kind of labs are orthogonal and fit a different range of learning goals.

The remainder of this work is organized as follows. Section 2 motivates the use of real machines for the proposed labs. Section 3 discusses how computer architecture courses are typically organized at universities. Section 4 presents the proposed methodology. Section 5 introduces the scheduling framework used at labs. Section 6 discusses the proposed labs. Section 7 provides some evidences for evaluating the proposed methodology. Finally, Section 8 presents some concluding remarks.

2. Motivation

Simulation has been, and continues being, an extensive methodology widely used across computer architects for research purposes. Detailed cycle-by-cycle simulators model what happens at the different processor stages every processor clock, which helps researchers to precisely understand how the processor and its internal mechanisms work. Because of this reason, computer architects either develop their own simulators or use other simulators widely spread across the scientific community such as [6,21,16].

The use of simulators has probably been the best way, if not the only one, to go deeper into the study of computer architecture topics, either for research or teaching purposes, in the last two decades. Thus, simulators have been widely used by professors and instructors both in the academia and the industry. To avoid the difficulties of complex simulation frameworks in lab sessions, a wide range of *in house* simulators have also been developed by instructors at universities such as DLX [5]. These simulators provide different complexity levels depending on the learning requirements, and usually provide some kind of graphical representation (e.g. display of simple pipelines) to ease the understanding of computer architecture topics.

However, as processors become more advanced and computer architecture courses need to cover the latests features of recent processors, simulators inevitably get more and more complex (e.g. they model hardware prefetchers, memory controllers, cache coherence protocols, etc.). Hence, simulation frameworks that model current advanced processors present two important disadvantages: (i) their complexity translates into a very long learning phase that is not adequate for undergraduates, and (ii) due to the fast evolution of current processors they often fail to model the newest advances of all the system components (e.g. the main memory or the network on chip).

In addition, writing a simulator that mimics the behavior of a specific machine is almost an impossible task. A major drawback that must be overcome is that many hardware details are not publicly available. Thus, the simulator developer may implement hardware functionality incorrectly. Besides, many components that significantly affect system performance like the main memory, the LLC caches, or their replacement algorithms, along with their interactions, need to be modeled. Many times researchers join different simulators (e.g. for the core and for the main memory) with the aim of more accurately modeling the entire system, but even so they are only able to model a machine that is still quite different from the real hardware.

In summary, on the one hand simulators fail to precisely model all the machine components (e.g. the main memory, the interconnects, the cores, the LLC replacement algorithm, etc.) and even more their most advanced features. On the other hand, the high number of simulated components makes simulation impractical for studying current multicores in lab sessions with a limited length (e.g. around two hours). Given the previous rationales, we believe that students need to use real hardware to obtain accurate and precise results and understand how the different parts of the machine interact among them.

3. Computer architecture courses in computer engineering curricula

3.1. General overview

Most Computer Engineering curricula include an introductory course, e.g. Computer Organization, where the basics of computer systems (i.e. the arithmetic unit, the processor pipeline, the memory system, the I/O unit, etc.) are introduced to the students. Then, at least two courses (e.g. Computer Architecture I and Computer Architecture II) study computer architecture topics in depth. The former computer architecture course is usually a core course in the curricula, while the second one covers more advanced topics and can be offered either as core or elective course depending on the syllabus.

Computer Architecture I courses cover a wide range of computer architecture topics such as superscalar architectures, branch prediction, out-of-order execution, or the memory hierarchy, among others. They might also introduce more advanced topics such as multicore or multithreaded architectures and/or interconnection networks. Instructors usually teach students to identify the main components of the system (i.e. the processor, the caches, and the main memory) and how different designs for each of these components impact on performance.

Computer Architecture II courses typically focus on parallel architectures or very advanced topics. Hence, they go further in the study of multicore and multithreaded architectures, as well as,