Sniper: Scalable and Accurate Parallel Multi-Core Simulation

Wim Heirman[*,†,1], Trevor E. Carlson[*,†],
Lieven Eeckhout[*]

[*] *ELIS, Ghent University, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium*

[†] *Intel Exascience Lab, Kapeldreef 75, 3001 Leuven, Belgium*

ABSTRACT

Sniper is a next generation parallel, high-speed and accurate x86 simulator. This multi-core simulator is based on the interval core model and the Graphite simulation infrastructure, allowing for fast and accurate simulation and for trading off simulation speed for accuracy to allow a range of flexible simulation options when exploring different homogeneous and heterogeneous multi-core architectures.

The Sniper simulator allows one to perform timing simulations for both multi-programmed workloads and multi-threaded, shared-memory applications running on 10s to 100+ cores, at a high speed when compared to existing simulators. The main feature of the simulator is its core model which is based on interval simulation, a fast mechanistic core model. Interval simulation raises the level of abstraction in architectural simulation which allows for faster simulator development and evaluation times; it does so by 'jumping' between miss events, called intervals. Sniper has been validated against multi-socket Intel Core2 and Nehalem systems and provides average performance prediction errors within 25% at a simulation speed of up to several MIPS.

This simulator, and the interval core model, is useful for uncore and system-level studies that require more detail than the typical one-IPC models, but for which cycle-accurate simulators are too slow to allow workloads of meaningful sizes to be simulated. As an added benefit, the interval core model allows the generation of CPI stacks, which show the number of cycles lost due to different characteristics of the system, like the cache hierarchy or branch predictor, and lead to a better understanding of each component's effect on total system performance. This extends the use for Sniper to application characterization and hardware/software co-design. The Sniper simulator is available for download at `http://snipersim.org` and can be used freely for academic research.

KEYWORDS: Multicore; simulation; design-space exploration; heterogeous architectures

[1]E-mail: wim.heirman@elis.UGent.be

[2]This work is funded by Intel and by the IWT. We used the compute infrastructure provided by the VSC Flemish Supercomputer Center, the ExaScience Lab, Leuven, Belgium and the Intel HPC Lab, Swindon, UK.

1 Introduction

We observe two major trends in contemporary high-performance processors as a result of the continuous progress in chip technology through Moore's Law. First, processor manufacturers integrate multiple processor cores on a single chip — multi-core processors. Eight to twelve cores per chip are commercially available today (in, for example, Intel's E5-4600 Series, IBM's POWER7 and AMD's Opteron 6000 Series), and projections forecast tens to hundreds of cores per chip in the near future. One near-future architecture, the Intel Knights Corner Many Integrated Core is expected to contain more than 50 x86-compatible cores on a single chip. Additionally, power and energy constraints will require a differentiation in clock speed and complexity of cores within a single chip, such as in ARM's big.LITTLE concept.Second, we observe increasingly larger on-chip caches. Multi-megabyte caches are becoming commonplace, exemplified by the 30MB L3 cache in Intel's Xeon E7-8870.

These two trends pose significant challenges for the tools in the computer architect's toolbox. Current practice employs detailed cycle-accurate simulation even during the early stages of the design cycle. While this has been (and still is) a successful approach for designing individual processor cores as well as multi-core processors with a limited number of cores, cycle-accurate simulation is not a scalable approach for simulating large-scale multi-cores with tens or hundreds of cores, for two key reasons. First, current cycle-accurate simulation infrastructures are typically single-threaded. Given that clock frequency and single-core performance are plateauing while the number of cores increases, the simulation gap between the performance of the target system being simulated versus simulation speed is rapidly increasing. Second, the increasingly larger caches observed in today's processors imply that increasingly larger instruction counts need to be simulated in order to stress the target system in a meaningful way.

These observations impose at least two requirements for architectural simulation in the multi-core and many-core era. First, the simulation infrastructure itself needs to be a parallel application so that it can take advantage of the increasing core counts observed in current and future processor chips. Second, we need to raise the level of abstraction in architectural simulation. Detailed cycle-accurate simulation is too slow for multi-core systems with large core counts and large caches. Moreover, many practical design studies and research questions do not need cycle accuracy because these studies deal with system-level design issues for which cycle accuracy only gets in the way (i.e., cycle accuracy adds too much detail and is too slow, especially during the early stages of the design cycle).

The Sniper simulator [CHE11] is designed to address exactly these problems. To combine the opposing requirements of having sufficient simulation detail, while still achieving high simulation speed, Sniper employs the technique of interval simulation [GEE10]. This is a recently proposed high-abstraction simulation approach based on mechanistic analytical modeling. It's main simulation speedup comes from the fact that, even though an out-of-order processor is being modeled, instructions can be traversed in-order; and that periods when a core is idle due to memory stalls, branch mispredictions or other long-latency events, can be fast-forwarded. Also, by assuming that the architecture being modeled is balanced, only a few key architectural components need to be modeled to get an accurate simulation. This assumption holds both when simulating well-designed existing architectures, and when doing architectural

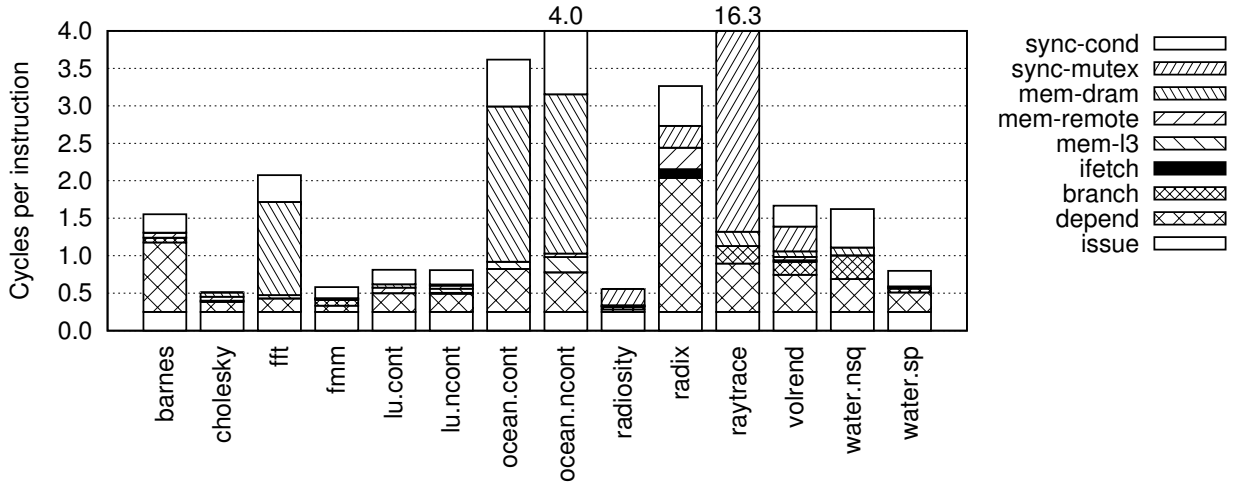

Figure 1: Detailed CPI stacks generated through interval simulation.

exploration — assuming this design phase is followed by a detailed design step which takes care of balancing out the architecture.

In addition to the interval core model, the Sniper simulator provides simulation models for a modern branch predictor, private and shared caches with optional prefetchers, and full support for DVFS and heterogeneous configurations. It also features a scriptable simulation control and analysis front-end that allows for on-line analysis of the simulation and interaction with the benchmark being simulated. This makes it possible gain a deep understanding of the application's use of hardware resources and to prototype new hardware/software interactions.

2 Results

2.1 CPI stacks

A unique asset of interval simulation is that it enables building CPI stacks which summarize where time is spent. A CPI stack is a bar graph showing the different components contributing to overall performance. The base CPI component typically appears at the bottom and represents useful work being done. The other CPI components represent 'lost' cycle opportunities due to instruction interdependencies, and miss events such as branch mispredictions, cache misses, etc., as well as inter-thread synchronization time. A CPI stack is particularly useful for gaining insight in application performance. It enables a computer architect and software developer to focus on where to optimize in order to improve overall application performance. Figure 1 shows CPI stacks for the SPLASH-2 benchmarks running on a 16-core simulated architecture.

2.2 Hardware/software co-design

Here we look at how hardware/software co-design can be done using Sniper. The application being studied is a heat transfer simulation, where at each simulated time step the flow of heat over a 2-D grid is computed by applying a stencil to the data values on the grid. By

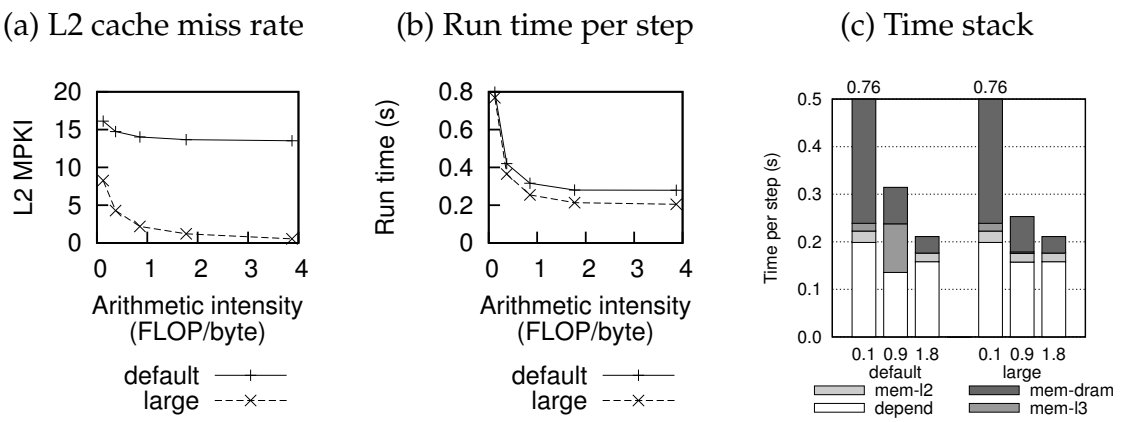

Figure 2: Co-design of hardware and software parameters for a heat transfer simulation.

traversing the grid in a tiled fashion, and by executing multiple time steps on each tile before moving on to the next one, the memory behavior of this application can be tuned. Changing the tile size directly affects the working set size; increasing the number of time steps improves re-use of cached data, this increases the arithmetic intensity (which is defined as the number of floating-point operations performed per byte read from main memory).

Figure 2 shows the results of this study. We keep the tile size constant, but simulate two hardware architectures where one has double the amount of cache size. We also vary the arithmetic intensity. From Figure 2(a), which plots the L2 cache miss rate, it is clear that the working set of this application is too big for the smaller cache sizes, as the miss rate is significant. On the machine with larger caches, miss rate goes down when arithmetic intenstity is increased — this indicates that data re-use improves, which should reduce both execution time and energy consumption. Figure 2(b) plots the per-iteration run time of the algorithm. Increasing arithmetic intenstity does indeed improve run time. The effect of larger caches is, however, much less distinct. Even though the L2 misses are radically lower, total execution time doesn't differ by that much. In Figure 2(c) the execution time is broken down by component. Whereas DRAM access latency was the dominating factor for low arithmetic intensities, this component is no longer dominant after the application has been optimized, and can — even on the architecture with the smaller caches — be mostly overlapped with out-of-order execution. Using this information, the algorithm and the hardware can be optimized simultaneously, leading to an optimal design point.

References

[CHE11] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, November 2011.

[GEE10] Davy Genbrugge, Stijn Eyerman, and Lieven Eeckhout. Interval simulation: Raising the level of abstraction in architectural simulation. In *Proceedings of the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 307–318, February 2010.