Warm Time-Sampling: Fast and Accurate Cycle-Level Simulation of Cache Memory

Luis M. Jimeno Ochoa, Pablo E. Ibáñez, Víctor Viñals
Dpto. de Informática e Ingeniería de Sistemas. University of Zaragoza
email: ljimeno@mcps.unizar.es {imarin, victor}@posta.unizar.es

Abstract

This paper proposes a new technique for reducing cache memory simulation time when measuring CPI. We perform time-sampling simulation, but still use the parts of the trace that do not belong to the sample to update the state of the memory system in order to avoid cold-start problems at the beginning of the next simulated interval. In our simulation environment and using this "warm-up" technique we achieve a reduction by a factor of 3 in the elapsed simulation time with an error less than 0.25% in the CPI estimation.

1: Introduction

As parallel activity into both the processor and the memory system increases, any formula based on stack algorithms gives progressive inaccuracy in the CPI computation. Finer statistics can then be gathered to partially correct the results but finally, if a realistic CPI should be computed, it is necessary to perform a detailed cycle-by-cycle simulation. Its drawbacks are the (very) high computer time required for each experiment and the need to repeat it for each variant or sizing to test. It is mostly due to the need to test a realistic (high) number of references from a representative benchmark set [9, 1].

Two trace-sampling techniques have been proposed to reduce simulation time: *time sampling* [7], and *set sampling* [8]. They succeed in obtaining miss ratios with errors typically less than 10% at a simulation time cost less than 10% of that of a complete simulation [5]. In our knowledge, no method described so far addresses the specific target of reducing simulation time to compute CPI in processors with significant ILP coupled to arbitrarily complex cache systems.

Our technique to reduce the CPI simulation cost is an extension of the miss-ratio time-sampling simulation technique. We obtain several measurements of the CPI on some *intervals* of consecutive instructions. These *observations* constitute our *sample*. With it we estimate the true CPI of the whole trace. Our main contribution is to maintain all caches updated between intervals in order to minimize the cold-start problems. Thus, we will term this technique *warm time-sampling (WTS)*. It considerably reduces the time required for a detailed CPI simulation (typically, by a factor 3 for the simple processor/memory model used in this work), yet offering great accuracy (errors are typically less than 0.25%).

Next section presents our simulation environment. Section 3 describes the steps to implement WTS. In section 4 we present the factors which introduce errors in our CPI estimation and how to detect and reduce them. Section 5 shows some quantitative results, and finally we give some concluding remarks in section 6.

2: Simulation Environment

Our trace-driven memory system simulator is designed to measure the CPI of a program executing in a pipelined processor. Its main characteristics are:

i) It can be used as a reference simulator. It simulates in detail the actions that would take place in the real system, including the pipeline of the processor.

ii) It has been programmed using object-oriented techniques. These are particularly suitable for hardware simulation [6].

iii) It performs an event-driven simulation. Events are handled by an event manager which drives the simulation.

We use a program trace generated on the fly (Shadow, [2]). Our simulator allows reconfiguration to model different processors and memory hierarchies. At present we consider a SPARC first generation 5-stage pipelined processor, a first-level on-chip split cache (2KB+2KB, direct mapping), a second-level on-chip unified cache (8KB, 4-way) and a third level off-chip unified cache (128KB, 16-way). We assume: fetch-on-write, copy-back, exclusion contents management between the first two levels [4, 3], inclusion with the third one, write buffers and LRU replacement policy.

3: Implementing Warm Time-Sampling

The simulator we have just described performs a complete cycle-level simulation. We call this operation mode *time mode*, since it is the one which measures CPI. To achieve a good CPI estimate while keeping down the simulation time, a fast simulation mode which only updates cache directories is needed; it is also required to combine this mode with the *time mode* to estimate CPI from the resultant hybrid simulation scheme.

3.1: Update-only simulation mode

Our first task is to provide a simplified simulation mode to maintain the state of the caches updated by the reference stream. In this *update-only mode* no time measurement is required. We only need to pass the same reference stream produced by the cycle-level simulator to the objects which represent the caches. Each one will use its input sequence to update its contents and generate the corresponding sequence for the next level.

The order in which the references are received must be very close to that of the cycle-level simulation. However, since our main goal is to reduce the simulation time, two simplifications -which can slightly alter this referencing order- have been performed:

i) *Suppression of the event mechanism*
The events are used to measure the number of cycles spent by the simulated system to execute a program. As we do not need to count cycles there is no point in keeping the event mechanism with all its overhead. Instead, we use a *command mechanism* implemented by a function call from component which originates the command. Thus, all system activity is serialized.

ii) *Simplification of the CPU activity*
The model of the pipelined processor is replaced by a trivial pipeline in which no hazards are detected. No time is spent in simulating stalls.

3.2: Warming gaps

How can we estimate the CPI with a simulation time close to that of the *update-only mode*? Our proposal is to implement WTS by switching between *time* and *update-only* modes. In WTS each set of simulated (consecutive) instructions in *time mode* is termed *interval*. Each set of (consecutive) instructions between intervals is a *warming gap* (Fig. 1). A sample of the trace is formed by collecting the CPI of each interval, and computing with them an estimation of the true CPI of the whole trace.

In a conventional *time-sampling* approach, the gaps are simply ignored. As a result, important cold-start problems appear as each interval begins, since the contents of the caches are out of date. Although several strategies have been proposed to reduce errors, none of them achieves good accuracy [5]. In contrast, we suggest to carry out an *update-only* simulation during the gap, making it an interval *warming gap*.

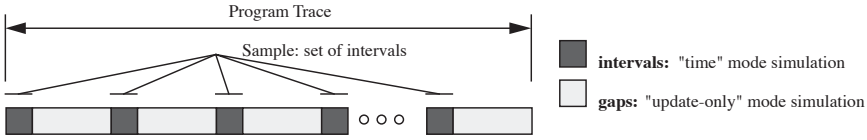

Figure 1. *Warm Time-Sampling simulation of the program trace*

Clearly, simulation time raises with respect to a conventional *time-sampling* simulation, since gaps must be still processed. But the accuracy is much better, and the simulation time is significantly reduced compared to a full CPI simulation.

3.3: Mode switching

When a warming gap is over and an interval must begin, the simulator enters a transient simulation mode called *start mode*. First, the CPU pipeline, which contains instructions to be processed using commands, is emptied by giving up inserting new instructions. Then, the following instructions will be inserted and they will be allowed to generate events. However, neither these instructions nor the cycles they consume will be counted. Only when the amount of work that has been placed into the system could be typical of the steady state, the count of instructions and cycles is initiated for the new interval. At that moment, the *start mode* is over and the simulation continues in *time mode*.

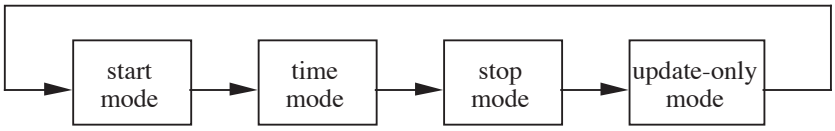

Figure 2. *Mode switching in WTS simulation. CPI is measured only in time-mode.*

When all the instructions of the interval have been entered, its CPI is computed. Due to the events in the event manager waiting to be issued, the switching to *update-only mode* cannot be done immediately. Instead, another transient simulation mode named *stop mode* is initiated. In it, no instructions are placed into the pipeline, and the simulation will process the pending events until the activity of the system eventually concludes. Finally, the *update-only mode* process the warming gap until the *start mode* begins again.

4: Possible error sources

Our CPI estimation for the whole trace is not exactly the true CPI (full *time-mode* simulation). These three sources of deviation can be considered:

Statistical errors To estimate the CPI of the trace we have collected the CPIs measured in each interval. We can use the sample mean as an unbiased estimate of the population mean, that is, the true CPI of the sample. However, if the number of observations is too small the sample mean can be quite far from the population mean. For this reason we take into account the confidence intervals for the population mean using the sample mean and variance. These intervals have been calculated under two assumptions: gaussian distribution and random sampling. Though our sampling is not random but systematic, this assumption results in a conservative estimate and allows much easier calculations [5].

Address-ordering errors In theory, WTS simulation avoids cold-start errors by keeping up to date cache directories at any time. In practice, we have introduced the two simplifications mentioned in subsection 3.1. Therefore, at the beginning of each interval the contents of the caches are not exactly the same as if the complete simulation would have never stopped.

Inaccuracies when measuring the CPI of each interval Prior to begin to count an interval after switching it is necessary to wait until the system has received enough work to do from the instructions inserted during the *start mode*. If the preloaded amount of work is less than that discarded at the end of the interval, an underestimating error will occur each time we compute the CPI of an interval. The minimum number of instructions of the *start mode* depends on the complexity of the memory hierarchy, the complexity of the processor and the program being traced.

5: Results

In this section we present the results obtained using the techniques described in this paper. We show the simulation time saved using WTS and the errors caused when measuring CPI. A typical symbolic-code benchmark (Gcc) and a floating-point intensive one (Spice) are used. Similar experiments have been carried out using other SPEC92 benchmarks leading to similar results. In all cases at least 200 million instructions have been simulated.

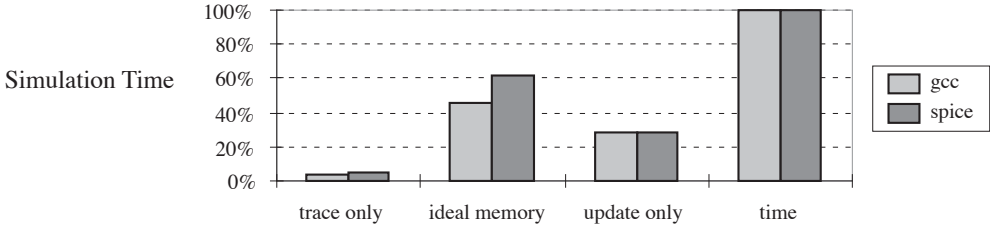

Figure 3. *Comparison of the simulation time for the basic simulation modes.*

Figure 3 shows the simulation time of the basic simulation modes. The important piece of data is the time spent in an *update-only* simulation. The value obtained for both used

benchmarks (close to 30%) can be considered as the lower limit of WTS simulation time. Further speedups would require to accelerate the *update-only mode*.

Figure 4 shows for each benchmark four series: they correspond to fixed interval lengths (*il*) of 100, 1000, 10000 and 100000 instructions respectively. The graphics have been completed with the simulation time of the *update-only mode* (0% in *time mode*) and the CPI and simulation time of the *time-mode* (100%). First we see that the simulation time varies linearly with the fraction of the trace simulated in *time mode* (*tm*). The difference between *il* series is quite small.

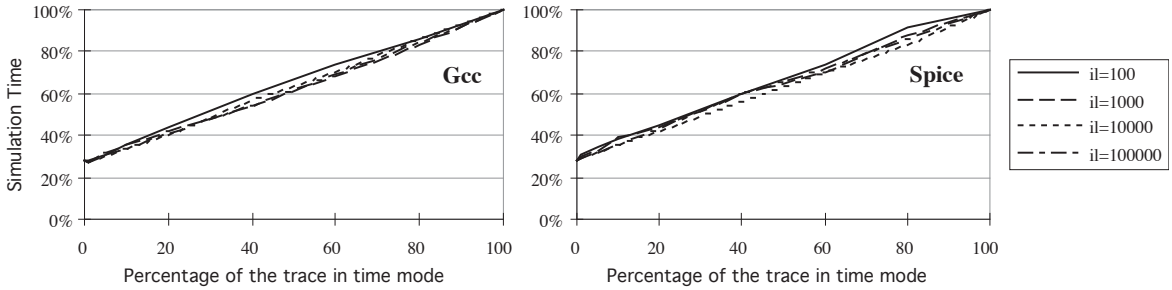

Figure 4. *WTS simulation time*

Figure 5 shows the relative CPI error. It can be seen that the shorter intervals series (*il*=100, 1000 instructions) present much smaller errors, since for a given *tm* the shorter the intervals the more observations we get. Thus, the sample can capture more CPI variation throughout the whole trace and is more representative of it. If we choose short enough intervals (less than 1000 instructions) we get the CPI of Gcc and Spice with an error which is about 0.21% and 0.05% respectively if we simulate at least 5% of the trace in *time mode*.

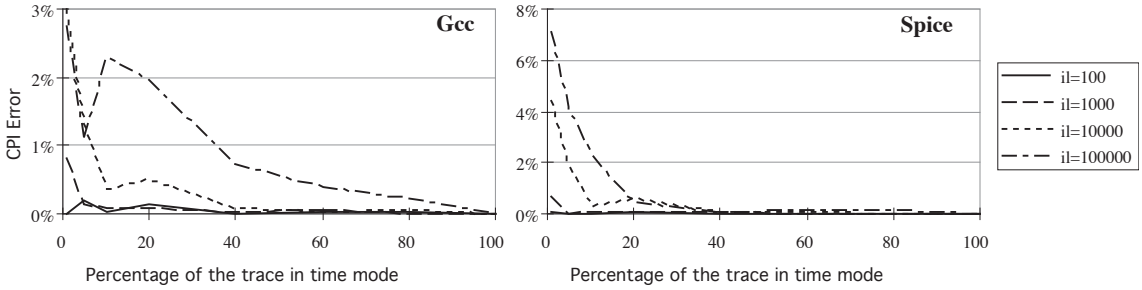

Figure 5. *CPI error in WTS simulation*

Finally figure 6 shows the upper and lower limits of the 90% and 99% confidence intervals for the estimated CPI. Inside the intervals the CPI actually measured is plotted. It can be seen that the confidence intervals offer a conservative estimate of the error. It might be due to our assumptions of systematic sampling and gaussian distribution.

6: Conclusions

In this paper we have described a new sampling technique, *Warm Time-Sampling*, to improve the results of other ways of sampling for detailed cycle-level cache memory simulation. It consists of keeping updated cache contents between intervals whose CPI is measured to

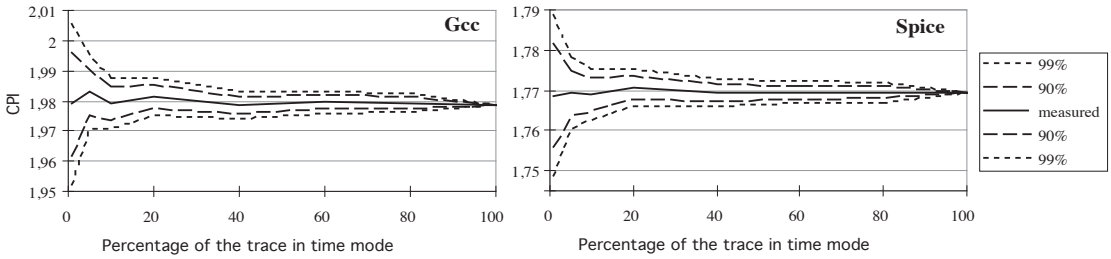

Figure 6. *Confidence intervals (*il*=100)*

estimate the true CPI of the whole sample. In order to update caches between intervals, an *update-only mode* has been implemented by removing all simulation overhead caused by event handling and time measurement. We think that other discrete-event simulation schemes, such as time-driven techniques, will also benefit from our approach.

When implementing a WTS based simulator, some precautions must be taken. First, the *update-only mode* to be used must drive the system to a final state quite close to that resulting from the detailed simulation. Special attention must be given to parallel activities which must be serialized, such as the behaviour of the write buffers. Second, switching between simulation modes must be carefully programmed to ensure that each interval CPI measured actually corresponds to an observation of the true CPI. Third, an adequate number of well-distributed over the trace intervals must be taken. And finally, statistical techniques shown in this paper can be used in the final stage of development of WTS to verify the absence of systematic or statistical (sampling) errors just by comparing a complete simulation with a few WTS results.

If we take about 5% of the whole program trace to form our sample we achieve a reduction by a factor 3 in the elapsed simulation time with an error which is typically less than 0.25% in the CPI estimation. As a final remark, we think that the gain can be even greater if processor/memory configurations with aggressive techniques to increase ILP are considered, due to the saved simulation overhead of a lot of parallel activities.

References

[1] A. Borg, R.E. Kessler, and D.W. Wall. Generation and analysis of very long address traces. In *Proc. 17th Ann. Int. Symp. on Computer Architecture*, pages 270–279, Jun 1990.

[2] P. Yan-Tek Hsu. Introduction to SHADOW. Technical report, Sun Microsystems Inc, July 1989. Revision A.

[3] P.E. Ibañez and V. Viñals. Performance assessment of contents management in multilevel on-chip caches. In *Proc. 22nd. Euromicro Conference*, Sept. 1996.

[4] Norman P. Jouppi and Steven J. E. Wilton. Tradeoffs in two-level on-chip caching. In *Proc. 21st Ann. Int. Symp. on Computer Architecture*, pages 34–45, April 18–21, 1994.

[5] R.E. Kessler, M.D. Hill, and D.A. Wood. A comparison of trace-sampling techniques for multi-megabyte caches. *IEEE Tansactions on Computers*, 43(6):664–675, Jun. 1994.

[6] S. Kumar, J.H. Aylor, B.W. Johnson, and Wm.A. Wulf. Object-oriented techniques in hardware design. *Computer*, 27(6):64–73, Jun. 1994.

[7] S. Laha, J.H. Patel, and R.K. Iyer. Accurate low-cost methods for performance evaluation of cache memory systems. *IEEE Transactions on Computers*, 37(11):1325–1336, Nov. 1988.

[8] T.R. Puzak. *Analysis of Cache Replacement Algorithms.* PhD thesis, Univ. Massachusetts, Amherst, Feb. 1985.

[9] A.J. Smith. Cache evaluation and the impact of workload choice. In *Proc. 12th Ann. Int. Symp. on Computer Architecture*, pages 64–75, Jun 1985.