Evaluation of SimPoint for Specific Architectural Studies

Veynu Narasiman
narasima@ece.utexas.edu

Aater Suleman
suleman@ece.utexas.edu

1 Introduction

As microprocessor complexity increases, the time required to complete a thorough simulation of an entire benchmark will become a bigger and bigger challenge that computer architects have to face. This simulation time is orders of magnitude greater than the time required to run the benchmark on real hardware, thus, it is no longer feasible to simulate long benchmarks in their entirety. Given such a scenario, there is a definite need to reduce this simulation time so that computer architecture research can be performed efficiently in the future. SimPoint [1] is a popular tool used to select *representative* samples of a benchmark. These samples are then used for simulation, instead of the entire benchmark, resulting in a significantly shorter simulation time.

Since these samples will be used by computer architects to determine performance improvements, it is imperative that there should be little compromise in accuracy. Otherwise, the results obtained when using SimPoint could not be trusted. The goal of our project is to evaluate the effectiveness of using SimPoint in determining the relative performance improvement of a particular architectural enhancement. The results will either validate or invalidate the effectiveness of using SimPoint for particular architecture studies.

2 Background and Motivation

The aim of most computer architecture research is to evaluate the impact of some enhancement on the overall performance. Therefore, the most important requirement of SimPoint or any other technique that reduces the size of benchmark code is that it should successfully capture the relative performance improvement obtained from the enhancement. Several different attempts have been made to evaluate the effectiveness of sampling tools (such as SimPoint). Most of them simply compared metrics (such as IPC, cache miss rate, and branch prediction accuracy) obtained using SimPoint to those obtained from simulating the entire benchmark but did not measure SimPoint's ability to successfully capture relative performance improvements.

For example, [4], is a paper from Calder *et al.* that describes the methodologies SimPoint uses to pick the representative samples of a benchmark. It concludes with a comparison of the IPC calculated from simulating an entire application to the IPC calculated from only simulating the samples SimPoint chose. The results obtained showed that using the samples from SimPoint produced an IPC that was within three percent of the actual IPC obtained from simulation of the entire benchmark. This particular paper only compares the IPC, citing it as the most important metric for evaluation.

Another paper, [3], also from Calder *et al.*, describes using Basic Block Distribution Analysis as a technique for finding simulation points. This paper uses multiple metrics to determine the accuracy of the proposed sampling technique. The metrics they use are IPC, Register Update Unit (RUU) occupancy, cache miss rate, branch prediction miss rate, address prediction miss rate, and value prediction miss rate. Once again, the value for the metric obtained using SimPoint was compared to the actual value obtained from simulation of the entire benchmark. For certain metrics such as IPC, the error was always within five percent, which is tolerable. However, for data cache miss rate and address prediction miss rate, the error could rise to as much as 20%. Such results suggest that SimPoint may not be appropriate for certain architectural studies. In addition, this paper did not target a specific architectural enhancement which is what we plan to do for our project.

A recent paper from Hawkins *et al.*, [5], evaluates the effectiveness of various sampling techniques such as SMARTS and SimPoint. This paper uses five different criteria to compare these sampling techniques. The important result here is that sampling the benchmark is the best technique to reduce the length of a benchmark for detailed execution. It is noteworthy to point out that one of the metrics used was architectural configuration dependence. It is important to computer architects that architectural configuration dependence remains unchanged after sampling. Only then can the results obtained be accepted with confidence.

Although most of these papers suggest that SimPoint is a powerful tool, according to [5], computer architects are hesitant to use SimPoint to determine the relative performance improvement for their study. In our project, we plan to evaluate the effectiveness of SimPoint targeted to specific architectural studies. A few of these particular studies are discussed below.

One of the architectural enhancements we plan on implementing is data prefetching. We want to see if the relative performance improvement obtained using SimPoint is comparable to the actual performance improvement from prefetching. We have not encountered any previous papers that evaluate SimPoint's ability to accurately capture the relative performance improvement obtained from prefetching. Our results will either validate or invalidate the use of SimPoint for architects doing research in the area of prefetching.

We also want to test SimPoint's effectiveness for studies related to branch characterization. One idea is to measure SimPoint's ability to accurately capture the variation in the predictability of a particular branch. Previous papers that have evaluated SimPoint looked at the overall branch prediction miss rate, but they do not focus on individual branches within a program. Validating SimPoint's ability to capture this information would allow computer architects performing research in this field to use SimPoint confidentally.

3 Research Plan

In order to conduct our research, we need to compare the true results obtained from complete simulation to those obtained using the SimPoint tool. We decided to use the SPEC 2000 benchmark suite for our comparisons since it is a widely accepted benchmark suite used by computer architects for their research. We plan to use the SPEC benchmarks with the reference input set because they are the most representative of real-world applications. However, the drawback of using the reference input set is that the simulation can be very time consuming. Thus, instead of using a simulator, we decided to use an instrumentation tool, PIN [2], for our experiments. We are going to develop tools implementing caches, prefetchers, and branch predictors that can be used with PIN to compute relevant statistics such as cache miss rates and branch prediction miss rates. We will collect this data for not only the entire benchmark, but also for the individual intervals. We will then use SimPoint to select the representative intervals and use the statistics from those intervals only to estimate the overall statstics. We will then compare this estimated value to the actual value recorded when simulating the entire benchmark.

The tools we want to develop for PIN will closely approximate cycle-accurate simulation. We will validate these tools by comparing the results to a previously done cycle-accurate simulation. We also need to program in the ability to collect statistics for each interval as well as for the entire application.

4 Conclusion

As an outcome of this project, we expect to either validate or invalidate the use of SimPoint for particular architectural studies. We will base our conclusion on the data collected using the aforementioned research plan. Specifically, we will evaluate SimPoint's ability to accurately portray the performance improvement obtained with prefetching, as well as its ability to accurately identify branch predictability. The results obtained from our prefecthing experiment could possibly validate the use of SimPoint for architects doing work in that field. Likewise, those involved in branch prediction, especially those interested in identifying hard to predict branches, could also use SimPoint more confidentally.

References

[1] Simpoint. http://www.cs.ucsd.edu/calder/simpoint/, March 2005.

[2] Vijay Janapa Reddi, Alex Settle, Daniel A. Connors, and Robert S. Cohn. Pin: A binary instrumentation tool for computer architecture research and education. In *Proceedings of the Workshop on Computer Architecture Education*, June 2004.

[3] Timothy Sherwood, Erez Perelman, and Brad Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *PACT '01: Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.

[4] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. Automatically characterizing large scale program behavior. In *ASPLOS*, pages 45–57, 2002.

[5] Joshua J. Yi, Sreekumar V. Kodakara, Resit Sendag, David J. Lilja, and Douglas M. Hawkins. Characterizing and comparing prevailing simulation techniques. Laboratory for Advanced Research in Computing Technology and Compilers, February 2005.