John (Jack) Sampson A05434795 jsampson@cs.ucsd.edu
Allen Chu allenchu@cs.ucsd.edu

A Comparison of the Effect of Warm-up Techniques on the Speed and Accuracy of Simulating Using Shorter SimPoints.

- Motivations

  At the heart of any approach to speed up simulation is the reduction of size of the code to be simulated in detail. However, as the length of any given run of fully simulated code decreases, the impact of stale or erroneous micro-architectural state on any results garnered from that section of the simulation can increase. When the SimPoint technique was first introduced, the expected contiguous block size for simulation was 100 million instructions, and the warm-up effects were thus not a large concern, as any effects would be overshadowed by execution. However, as contiguous block sizes in more recent applications of SimPoint based approaches drop closer to 10 million and to 1 million instructions, it becomes more important to consider what degree of warm-up of the micro-architectural state is necessary to avoid introducing error, and what sort of delay doing such a warm-up would impose. A thorough examination of tradeoffs between simulation speed and accuracy when using various warm-up techniques with SimPoint promises to be useful in determining which warm-up techniques are worth considering for use with SimPoint.

- Challenges

  The overall challenge in warming is to provide an accurate state for a simulation to begin or carry on a section of a trace. The more pressing challenge of warming is remaining timely. The purpose of simulators such as SimPoint and SMARTS is to decrease simulation time. If warming ends up dominating simulation time, the ultimate goal is not realized. Warming techniques built for both accuracy and speed are thus notably preferable to those designed for accuracy alone.

  A thorough side-by-side comparison of warm-up techniques does not appear to have been done in the context of SimPoint. This is unsurprising given that for the previous, and even current, granularity of SimPoint sizes, several well-known warm-up techniques have proved sufficient in practice. However, there is not as yet any formalism to the choice of warm-up method to use. Though there do not currently appear to be overarching complications due to error from lack of warm-up or excessive delays due to the warm-up mechanisms used, the lack of comparative data renders difficult any judgments as to whether this will continue to be true for smaller simulation chunks. We already know that, qualitatively, warming serves the smaller SimPoint chunks well; the challenge lies in coming up with a useful quantitative assessment across several different techniques and projected over different simulation chunk sizes.

John (Jack) Sampson A05434795 jsampson@cs.ucsd.edu
Allen Chu allenchu@cs.ucsd.edu

- Solution Space

  There are a number of solutions to the problem of warm-up that we plan on comparing. These solutions include the assume hit, stale state, calculated warm-up, and the continuously warm approach. Simulating no warm-up will be the baseline for speed comparison, and the known IPC and other metrics for the benchmarks we simulate will be the baselines for accuracy.

  The "assume hit" warm-up method avoids the notion of a cold start by adding a warm-up bit to each entry of caches, branch predictors, and other history dependent structures. This warm-up bit is set initially and is cleared once an entry in a cache or branch predictor is used. For that first use, a hit or correct prediction is assumed as the high accuracy of these structures makes the likelihood of a hit or correct prediction much greater than the opposite. To offset the somewhat unreal perfection while still avoiding a cold start, this approach can be augmented with a set miss rate percentage such that such that a miss or incorrect prediction is sometimes triggered for the first access.

  The "stale state" warm-up method is a bit less proactive then the "assume hit" method. "Stale state" does not attempt to mask the lack of warm-up by falsifying the first access to a structure's entries, but also does not reset any part of the micro-architectural state between the end of a simulation point through the beginning of the next simulation point. Essentially, a previous simulation section is used as a warm-up for a current simulation section.

  The "calculated warm-up" method calls for running a number of instructions immediately previous to the start of a simulation point to accomplish warming. A working set of architectural data such as branch addresses and accessed data is generated. Next, the number of instructions to use for warm-up before a simulation point is calculated, with the requirement being that the number of instructions is sufficient to capture the working set generated. The warm-up instructions are executed and then statistic-gathering tools are reset for the section to be simulated.

  The "continuously warm" method selectively keeps some structures, such as a the cache, warm while in fast-forward mode. The only concern of a "continuously warm" method is its effect on the fast-forward execution time. The purpose of the fast-forward is to skip over sections of instructions extremely quickly to save time in the simulation. Continuously warming structures should not affect the fast-forward to an extent that the overall simulation time is noticeably degraded.

  There are other possible warm-up techniques that may be compared as well. One of these is using Memory Reference Reuse Latency to do warming. Haskins and Skadron introduced a parallel to the "calculated warm-up" above by measuring the number of instructions that elapse between memory references in order to time when warm-up should begin. The closer memory references that are preceded by a sample will be more highly associated with the sample itself as opposed to the references

farther away. So those references should be included in the warm-up while the references farther away have no need to be included because they are irrelevant. Another method to consider is the dynamic warm-up mechanism from DiST. This method proposes using immediate instructions succeeding a simulated section. The number of instructions for warm-up used is dynamically determined by comparing simulation results with the next section to be simulated. When they are both similar enough, warm-up is considered complete.

- Opportunity

  In a somewhat circular fashion, it is because simulations are taking less time to run and skipping over long stretches of their data that quantifying the effectiveness of warm-up techniques is a worthwhile endeavor, and it is because simulations are becoming faster and skipping over long stretches of their data that we can hope to gather sufficient data, in a reasonable time frame, for meaningful comparison. Thus, the particular opportunity is that, as it is not yet a pressing concern for those using SimPoint to guide simulation, warm-up has not already been studied ad nauseum in the context of SimPoint, but it has been studied enough in other contexts that there exist interesting solutions which warrant the effort of a thorough comparison.

- Methodology

We plan on using SimpleScalar 3.0 to model our architecture for simulation.

Extensions are available for Skadron and Haskins's Memory Reference Reuse Latency approach, and we aim to use those.

We look to use work done on SMART to assist in implementing some of the warm-up techniques.

The following metrics will be gathered. Data gathered will stem from these metrics.

- Mean overall IPC
- Mean error
- Mean cache miss rates
- Mean branch misprediction rates
- Mean Per-Phase Variance on the above
- additional time needed for warming (measured in system time spent in simulator)

We plan on gathering data from running as much of the SPEC benchmarks as time will allow using SimPoints with chunk lengths of 1 million and 10 million instructions, over all of the warm-up techniques discussed.

Given the amount of CPU time needed to complete all these simulation runs, we aren't certain as to which machines we should be seek to be running our jobs on and welcome advice as to which resources we should look into scheduling time on.

John (Jack) Sampson A05434795 jsampson@cs.ucsd.edu
Allen Chu allenchu@cs.ucsd.edu

- Papers referenced for background material and motivation:

Greg Hamerly, Erez Perelman, and Brad Calder. How to Use SimPoint to Pick Simulation Points. ACM SIGMETRICS Performance Evaluation Review, 2004.

S. Girbal, G. Mouchard, A. Cohen, and O. Temam. DiST: A simple, reliable and scalable method to significantly reduce processor architecture simulation time. In *ACM Intl. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS'03)*, San Diego, California, June 2003.

Roland Wunderlich, Thomas Wenisch, Babak Falsafi and James Hoe. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. *In Proceedings of the 30th International Symposium on Computer Architecture*, June 2003

J.W. Haskins and K. Skadron. "Minimal Subset Evaluation: Rapid Warm-up for Simulated Hardware State." In *Proceedings of the 2001 International Conference on Computer Design,* pp. 32-39, Sept. 2001.

J.W. Haskins, Jr. and K. Skadron. "Memory Reference Reuse Latency: Accelerated Warmup for Sampled Microarchitecture Simulation." In *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 195-203, Mar. 2003.