SimTrace: Capturing Over Time Program Phase Behavior

Steven Flolid Emily Shriver Zachary Susskind Benjamin Thorell Lizy K. John
University of Texas at Austin, Intel Labs
{stevenflolid, blthorell, zsusskind, ljohn}@utexas.edu, emliy.shriver@intel.com

***Abstract*—As computers and the workloads they run have grown in size and complexity, it has become difficult to test the performance and power of future products under design. These products are often designed on simulators that are orders of magnitude slower than the final product. For this reason, industry and academia have developed methodologies to reduce run times. However, in order to study runtime adaptive techniques for performance and power/energy management, it is important to capture the over time phase behavior of workloads. One technique, SimPoint, has been demonstrated to capture average behavior accurately, but it is not known how well a sequence of SimPoints can capture over time program phase behavior. To explore this, we replay the sequence of SimPoints and evaluate the sequence's accuracy. Using SPEC CPU 2017 benchmarks as a case study, we discover good accuracy for the replayed sequence: with less than 5% performance error (Instructions Per Cycle) for four time-series metrics.**

***Keywords*—Simulation Point Replay; SimPoint; Representative Region**

I. INTRODUCTION

The behavior of an application during execution is known to vary over time. This over time phase behavior is of interest to both hardware and OS designers in making trade-offs during design and in making scheduling decisions. How rapidly or slowly the phases are transitioning, the magnitudes, and bursts of behavior are all examples of over time phase behavior. Architectures can use time sensitive information to improve performance and energy efficiency through techniques such as Dynamic Voltage Frequency Scaling (DVFS) which can achieve better energy efficiency in longer program phases [1] [2]. Thermal design also benefits from knowing the long program phases, as variations in thermals require longer time constants [3].

Despite the importance of over time behavior, existing techniques [4] [5] [6] [7] only focus on average workload characteristics. However, many different over time behaviors can result in the same average behavior. The simplest comparison is between a program with a constant rate of cache misses while a second program shifts from half the miss rate to double the miss rate during execution. Both programs have identical averages but will affect a computer in different ways. This is especially true for memory where increases in traffic can have exponential effects on performance. Thus, capturing average behavior is insufficient to reproduce similar over time behavior.

In order to capture the over time phase behavior of the entire application, we propose using approaches developed in SimPoint [4] [5] [8] to identify the important phases of behavior and then replay these phases in order. We call this replay a **SimTrace**. To examine the feasibility of SimTrace, we conduct experiments on CPU2017 benchmarks. We use the best-known similarity metrics for time series [9] [10] to compare the accuracy of the phase behaviors in the SimTrace with those in the original application. We compare the similarity on performance metrics of interest to computer and software architects such as IPC (instructions per cycle), cache misses, and branch mispredictions.

II. BACKGROUND

SimPoint is a powerful technique used to create representative regions to approximate a program's behavior [11]. The technique aims to identify a small set of regions of interest that represent the performance of the original workload. The representative simulation points are chosen based on architecture independent characteristics of the original program, specifically a basic block vector (BBV).

III. METHODOLOGY

This section covers in more detail how we apply the SimPoint technique, which works well for average behavior, to an over time context. To do this, we extend SimPoint by replaying over time either the performance metrics of each cluster or by replaying the clusters' representative regions in their entirety. We call this modified technique SimTrace to reference both the SimPoint technique and the fact that we are using a Trace of the clusters. SimTrace seems to be the first technique to apply SimPoint in such a way.

SimTrace extends on the SimPoint Technique in the following way. SimPoint maps every region to a specific cluster. This creates what is essentially a cluster trace for the entire program. SimPoint, however, does nothing with this trace and only counts the total number of regions mapped to

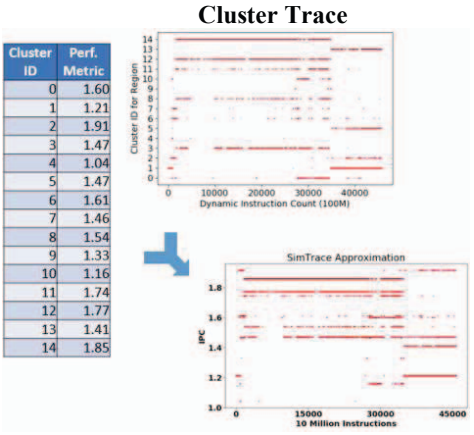

Fig. 1. SimTrace Generation Process

DOI 10.1109/ISPASS48437.2020.00041

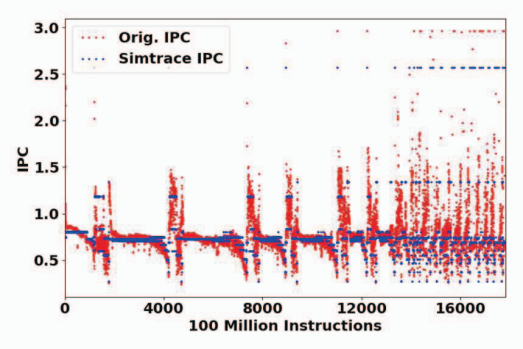

Figure 2: IPC of the SimTrace of MCF

each cluster. This makes sense as SimPoint intends to capture only the average behavior of a workload without considering the program's over time behavior.

Instead, SimTrace uses this cluster trace to capture the program's over time behavior. The essential step is using a tuple of a cluster ID and an associated metric as outlined in Figure 1. This metric can be the cluster's IPC, cache misses, branch behavior, etc. We create a SimTrace by replacing every cluster ID with the associated performance metric. This results in a time series of performance values that can be used to approximate a program's over time behavior. An example final output for SimTrace can be seen in Figure 2.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

We experimented with SimTrace to analyze its over time accuracy as compared to the original workloads using best known similarity metrics from [9] and [10]. We gathered performance counters for IPC, L1 cache MPKI (misses per thousand instructions), L2 cache MPKI, Branch Misprediction MPKI, and TLB MPKI. We created a SimTrace for each workload based on these measurements and compared them with the original program's performance counter values. We used the Performance Counter API (PAPI) to gather the data and the PinPoint [8] tool to generate the clustering for our experiments.

The experiments were conducted on a Dell PowerEdge R320 server equipped Xeon E5-2430 v2 processor (codenamed Ivy Bridge), and 64 GB DDR3 memory. For our experiments, we used the single-threaded speed versions of the SPEC CPU 2017 integer benchmark suite. Each single threaded program was run on a single core of this system with minimal overhead from the operating system due to core isolation. With this set up, SimTrace captures the over time phase behavior while using 32 or fewer clusters.

B. SimTrace Similarity Results

This section presents and discusses the results of applying four similarity techniques for the CPU2017 workloads. We present the results for IPC while omitting branch and memory related metrics due to space constraints.

The results for IPC similarity are illustrated in Table 1. The similarity metrics of mean percent error (MPE), mean absolute percent error (MAPE), normalized root mean square error (NRMS), and dynamic time warp (DTW) error all have error <5% averaged across workloads. This low error indicates that a small number of simulation points, sequenced in order, can capture the over time large scale phase behavior if IPC is the metric of interest. Observe that the MPE is extremely low for SimTrace compared to the other metrics. This is expected as MPE allows both positive and negative errors which have a canceling effect when summed together, producing consistently lower errors.

TABLE 1. IPC Similarity Values (SimTrace vs Original)

Workload (avg. IPC orig.)	**MAPE**	**MPE**	**NRMS**	**DTW**
Leela (1.23)	1.18%	-0.15%	0.99%	0.80%
exch.2 (2.04)	1.60%	-0.06%	4.43%	0.99%
Gcc (1.61)	5.38%	-0.05%	5.52%	4.52%
Xz (1.56)	6.23%	-0.94%	4.88%	2.98%
Mcf (0.76)	10.43%	0.44%	5.29%	3.73%
Deepsjeng (1.71)	4.44%	1.81%	6.16%	2.01%
Omnetpp (0.78)	6.76%	-1.61%	6.20%	2.71%
Perlbench (1.29)	5.94%	-2.85%	4.73%	2.13%
x264 (2.32)	1.51%	0.66%	3.67%	1.59%
Average	**4.83%**	**-0.30%**	**4.65%**	**2.38%**

Compare this to MAPE and NRMS which do not have such cancellation, the errors increase to 5% across the workloads. Finally, DTW lies between the other three metrics as it is able to warp SimTrace to better match the trends of the original signal. Notably, the errors are small despite the wide variety of large scale phase behaviors present in the CPU2017 workloads.

V. CONCLUSION

Capturing over time variability and phase behavior of programs is important to create program models for performance estimation. This is particularly true when designing phase-dependent scheduling schemes for performance, power/energy, or thermal optimizations. The architecture community uses simulation regions as identified by tools such as SimPoint to reduce simulation time during pre-silicon explorations. In this paper, we present a phase-varying replay methodology (SimTrace) by sequencing multiple simulation points together. We use four similarity techniques for time-series (MAPE, MPE, NRMS, DTW) to evaluate the over time similarity between the original workloads and their approximation for multiple performance metrics. We observe that SimTrace produces an over time average error of less than 5% for a variety of performance metrics including IPC, L1/L2 cache misses, branch mispredictions, and TLB misses. The explored metrics indicate a high similarity. Hence, SimTrace can be used as a feasible technique for representing a program's large scale over time phase behavior.

ACKNOWLEDGEMENT

This research was supported in part by National Science Foundation under grant numbers 1725743, 1745813, 1763848, and an Intel Research Award. Authors would also like to acknowledge computational resources from Texas Advanced Computing Center (TACC). Any opinions, findings, conclusions or recommendations are those of the authors and not of the National Science Foundation or other sponsors.

REFERENCES

[1] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunities," in ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005., pp. 275–280, Aug 2005.

[2] W. L. Bircher and L. K. John, "Analysis of dynamic power management on multi-core processors," in Proceedings of the 22Nd Annual International Conference on Supercomputing, ICS '08, (New York, NY, USA), pp. 327–338, ACM, 2008.

[3] A. K. Coskun, T. S. Rosing, and K. C. Gross. "Utilizing predictors for efficient thermal management in multiprocessor SoCs." in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28, no. 10 (2009) (DAC2009): 1503-1516.

[4] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS X, (New York, NY, USA), pp. 45–57, ACM, 2002.

[5] T. Sherwood, E. Perelman, and B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in PACT, pp. 3–14, 2001.

[6] M. V. Biesbrouck, B. Calder, and L. Eeckhout, "Efficient sampling startup for SimPoint," IEEE Micro, vol. 26, pp. 32–42, July 2006.

[7] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling," in Proceedings of the 30th Annual International Symposium on Computer Architecture, ISCA '03, (New York, NY, USA), pp. 84–97, ACM, 2003.

[8] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi, "Pinpointing representative portions of large intel R itanium R programs with dynamic instrumentation," in 37th International Symposium on Microarchitecture (MICRO-37'04), pp. 81–92, Dec 2004.

[9] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. "Querying and mining of time series data: experimental comparison of representations and distance measures." Proceedings of the VLDB Endowment 1, no. 2 (2008): 1542-1552.

[10] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. "Experimental comparison of representation methods and distance measures for time series data." Data Mining and Knowledge Discovery26, no. 2 (2013): 275-309.

[11] E. Perelman, G. Hamerly, and B. Calder, "Picking statistically valid and early simulation points," in 2003 12th International Conference on Parallel Architectures and Compilation Techniques, pp. 244–255, Sept 2003.