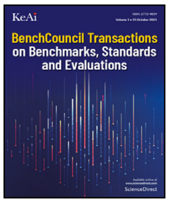
Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/



Research article

Enabling Reduced Simpoint Size Through LiveCache and Detail Warmup

Jose Renau [a], Fangping Liu [b], Hongzhang Shan [b], Sang Wook Stephen Do [b],*

[a] *Uncore LLC, Santa Cruz, CA, US*

[b] *IC LAB, Futurewei Technologies, Santa Clara, CA, US*

ARTICLE INFO

Keywords:
Architecture
Simpoint
LiveCache
Detail-warmup
Simulation sampling
Cycle-accurate simulation
Microarchitecture simulation

ABSTRACT

Simpoint technology (Sherwood et al., 2002) has been widely used by modern micro-architecture research community to significantly speedup the simulation time. However, the typical Simpoint size remains to be tens to hundreds of million instructions. At such sizes, the cycle-accurate simulators still need to run tens of hours or even days to finish the simulation, depending on the architecture complexity and workload characteristics. In this paper, we developed a new simulation framework by integrating LiveCache and Detail-warmups with Dromajo (https://chipyard.readthedocs.io/en/latest/Tools/Dromajo.html) and Kabylkas et al. (2005), enabling us to use much smaller Simpoint size (2 million instructions) without loss of accuracy. Our evaluation results showed that the average simulation time can be accelerated by 9.56 times over 50M size and most of the workload simulations can be finished in tens of minutes instead of hours.

1. Introduction

Modern computer architecture researches rely heavily on computer simulation to study new architectural features or estimate the performance, power, and area. A cycle-accurate simulator often takes hundreds of simulation hours, prohibiting its use in practice. To expedite the simulation, prior arts have explored various techniques. Sampling is one of the popular approaches, where a simulator runs sampled executions instead of the entire benchmark. The sampling could be based on either statistical sampling [1–3] or representative sampling [4].

Simpoint [4,5] is one of the most widely used sampling techniques, which could reduce the simulation time dramatically from months to days to hours. Running only the representative checkpoints of a program execution so-called 'Simpoints' generated by the Simpoint toolset enables computer architecture simulation to finish earlier than running the same program from the beginning to the end. However, the typical Simpoint size remains to be tens to hundreds of million instructions to maintain the accuracy. Depending on the architectural complexity, it still takes tens of hours to finish, still not fast enough for a quick turnaround.

In this paper, we developed a framework based on Dromajo [6,7] to enable us to use Simpoints with only 2 million instructions (2M). Compared with regular Simpoints with hundreds of million instructions or over, the simulation time could be greatly reduced from hours to minutes without loss of accuracy. Dromajo is a RISC-V RV64GC emulator, which enables executing an application under fast software simulation, generating checkpoints after a given number of instructions, and resuming such checkpoints in another slow, cycle-accurate simulator to generate micro-architecture simulation results.

In order to use smaller Simpoint size, one challenge needs to be addressed is the simulator needs to start from the up to date architectural status. Otherwise the simulation accuracy cannot be maintained. Large simpoint sizes may obviate this need. To fulfill this purpose, we integrate the LiveCache technique from [8,9] into Dromajo so that Dromajo can record the memory operations in timing order up to the Simpoint location in the checkpoint files. The number of memory operations to be recorded is a configuration parameter set accordingly with the cache size(s) of the simulated target micro-architecture. When the cycle-accurate simulator starts, by reading the checkpoint files, it can repeat these memory operations and bring the cache status up to date quickly. To bring the status of other architectural components up to date, such as a branch predictor, we resort to Detail-warmup, which allows us to run a specified number of instructions right before the Simpoint location, from which the simulator is dictated to start to measure the performance numbers. Correspondingly, the actual Simpoint locations will also be adjusted based on the number of instructions defined by Detail-warmup.

Putting all together, Simpoint execution is preceded by LiveCache-warmup first, followed by the Detail-warmup, then starts to collect the performance numbers thereafter. Compared with running only Simpoint itself, LiveCache and Detail-warmup enable us to bring the machine status up to date, preserving the simulation accuracy. As far as we know, this is the first framework combining both LiveCache and Detail-warmup together to generate 2 million Simpoints on RISC-V platforms.

* Corresponding author.
E-mail addresses: renau@uncore.io (J. Renau), julius.liu@futurewei.com (F. Liu), hshan@futurewei.com (H. Shan), sdo@futurewei.com (S.W.S. Do).

https://doi.org/10.1016/j.tbench.2022.100082
Received 23 September 2022; Received in revised form 12 December 2022; Accepted 12 December 2022
Available online 21 December 2022

In summary, we contribute the followings to the state of the art:

(1) Developed an open source framework that enables us to make use of smaller (2M) Simpoint size without loss of accuracy.
(2) Evaluated the 2M Simpoint size with SPEC 2006 CPU benchmark suite. Compared with 50M Simpoint size, the average simulation time has been improved more than 9 times.
(3) Quantitatively study the performance effects of LiveCache and Detail-warmup on simulation speed and accuracy.

Including the LiveCache technique, prior works such as [1–3,8,10–21] have proposed and discussed various micro-architecture warm-up techniques and effects, to which we plan to extend our work.

The rest of the paper is divided into the following sections. Section 2 describes the implementation. Section 3 presents the evaluation results with analysis. Lastly, Section 4 concludes the paper with comments on future directions.

2. Implementation

Our implementation is based on Dromajo [6,7], an open source RISC-V emulator. We extended Dromajo so that it is capable of generating Simpoints with user-configurable LiveCache and Detail-warmup.

2.1. Base Simpoint creation and execution

First, we modified the Dromajo source code to enable it to profile a benchmark based on Simpoint requirements. The profiling should follow the basic-block characterization described in the Simpoint papers [4,5]. We used the Dromajo's existing checkpointing option to generate two checkpoint files at each Simpoint location. The first file includes RISC-V instructions to be executed to restore the target machine's architectural state such as the contents of the physical register file and the control registers at the time of the corresponding Simpoint creation. The second file contains the memory image including the instruction and data areas with others necessary memory contents to run the benchmark. As far as we know, we have first used, designed and implemented Simpoint support on Dromajo since it was first discussed in [7].

To run Simpoints, we modified our target cycle-accurate simulator to copy the memory image into the target simulator's memory space and let the execution start from the first instruction in the first Simpoint (checkpoint) file. The RISC-V 'dret' instruction inserted by the Dromajo checkpointing option at the end should have execution jump to the desired Simpoint location. We also modified the target simulator to reset and start to (re)collect simulation statistics such as the number of cache misses and branch mispredictions, etc. right after the dret instruction execution.

2.2. LiveCache

For LiveCache, we implement similar mechanism described in [9], which adopts the MTR (Memory Timestamp Record) technique from [8], on top of the base Simpoint framework as described above to have Dromajo record memory operations up to the current Simpoint location and translate them into RISC-V 'load' instructions for clean cache lines or 'load' and 'store' instruction pairs for dirty cache lines using the memory addresses recorded. These load and store instructions are stored in the first Simpoint file and will be executed later by simulator to bring the cache status up to date. The total simulation time should increase accordingly because of the execution time of these additional load and store instructions.

To limit the number of the LiveCache load and store instructions, our framework takes 'Bootrom' size as an input parameter. For example, if the Bootrom size were 8 KB, then there are 1024 64-bit addresses recorded that corresponds to a maximum of 1024 loads or load and store pairs. The actual Bootrom size should be set based on the cache size(s) of the simulated target micro-architecture.

For the verification, we tested the following C language code snippet on our target cycle-accurate simulator. We made two checkpoints at the third loop iteration with LiveCache on and off. The results showed that with LiveCache on, the IPC was improved about eleven percent, confirming the effectiveness of the mechanism.

```
   //an array of 32k 32-bit words
   //occupying 2048 cache lines
int vec[0x8000];
register int sum = 0;
for (int i = 0; i < 5; ++i) {
 for (int j = 0; j < 0x8000; j += 16) {
  //do one load action
  //each cacheline or 64 bytes
  sum += vec[j];
 }
}
```

2.3. Detail-warmup

Detail-warmup aims at warming up micro-architecture components such as branch predictor and instruction and data caches by executing some instructions right before a Simpoint while LiveCache specifically aims at data caches. The goal is to restore the machine state of a target simulator as close as possible as if the target simulator has kept running up to the Simpoint location.

The implementation goal is to make a checkpoint at a location prior to a Simpoint by the number of instructions specified by the Detail-warmup size parameter. The base Simpoint creation does not consider these additional instructions, and we had to add an additional step to adjust the Simpoint location accordingly. We intervened the base Simpoint creation process right before the final step with our scripts to adjust the actual Simpoint location earlier by the Detail-warmup size. For a rare case where a base Simpoint needs to be created at the very beginning of execution, the whole Simpoint window needs to be adjusted to make room for Detail-warmup because a Simpoint location cannot be specified prior to the very beginning.

From the description, we can find that Detail-warmup is more powerful than LiveCache to update the architectural states. However, Detail-warmup is much more expensive. LiveCache can help to reduce the Detail-warmup size. It is the combination of LiveCache and Detail-warmup that enables smaller Simpoint size fast and accurate.

In summary, we applied the LiveCache and Detail-warmup modifications to related places in the Dromajo source code, where the original code adds the LiveCache memory instructions and captures the corresponding architectural snapshot in a checkpoint file respectively. We expect that one can port the modifications in a different tool set other than Dromajo.

3. Evaluation

3.1. Simulation setup

For the benchmark setup, we use the SPEC CPU 2006 benchmark suite [22]. We use Buildroot [23] to include a Linux kernel in Simpoint for the system call support.

For the target simulator setup, we use our in-house cycle-accurate simulator, which runs unmodified RISC-V instructions. The target simulator also implements hardware support to handle interrupts and exceptions based on the RISC-V specification to run the benchmark applications as intended. Table 1 shows the target simulator configuration.

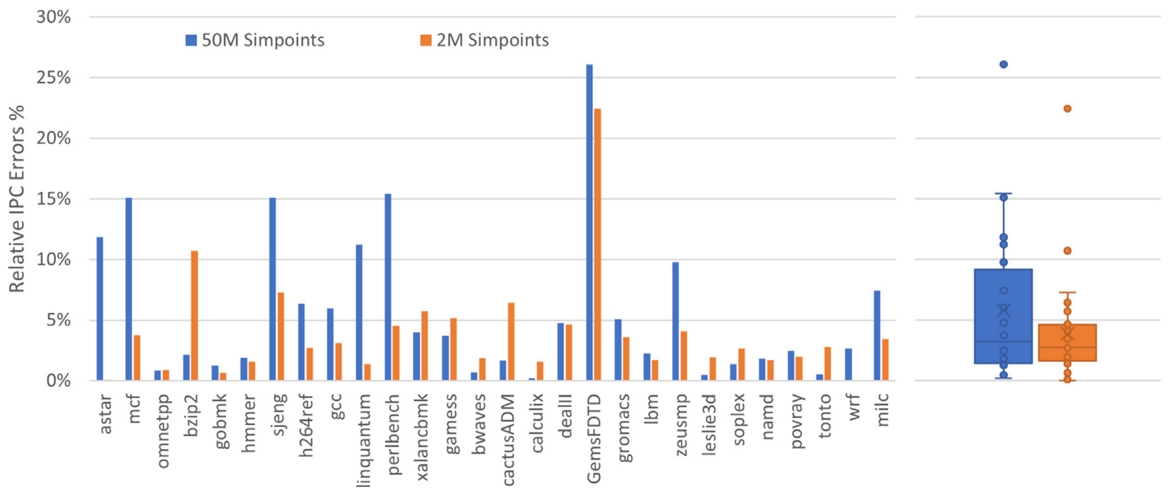


Fig. 1. The relative IPC errors for 2M Simpoints (with LiveCache and 4M Detail-warmup instructions) and 50M Simpoints (with 50M Detail-warmup). Using 2M Simpoints can reduce the IPC errors from average 5.46% to 3.89%.

Table 1
Simulator configuration.

Core	Single-Core, 96 Inst. Q entries dispatch width - 8 Int, 4 Fp instructions 3-ALU, DIV, MUL, FP 48-bit VA, 40-bit MAX PA
Instruction fetch	64 KB, 4-way, 64B-line Fully pipelined 48-entry TLB, 32-entry RAS BTB, TAGE predictor
L1 data	64 KB, 4-way, 64B-line LRU, 4-cycle latency
L2	1 MB, 8-way, 64B-line LRU, 9-cycle latency
L3	4 MB, 16-way, 64B-line LRU, 13-cycle latency
Memory	167-cycle latency

Table 2
Simpoint configuration.

Setting	Description
Execution window size	10B instructions
Simpoint size	2M instructions
Bootrom size	256 KB
LiveCache-warmup	On or Off
Detail-warmup	0, 2, or 4M instructions

For the Simpoint setup, we choose a 10 billion execution window size to avoid very long simulation time, which still allow us to conduct a fair evaluation. We also have Dromajo move this 10 billion instruction window by 100 million instructions to allow Detail-warmup for the case where a Simpoint is created at the very beginning of the execution.

Table 2 shows the Simpoint configuration used for evaluation.

3.2. *Simulation results*

This section focuses on the evaluation of accuracy and speed of our Simpoint approach. We compare our 2M Simpoint results with the popular 50M Simpoint ones using SPEC CPU2006 benchmark suite as our driving applications, which includes 28 integer and floating-point applications. Sphinx3 is not included currently due to that our RISC-V simulator could not handle its input data set correctly.

3.2.1. *IPC accuracy*

To compare the accuracy, we compute the IPC (instructions per cycle) errors relative to the 10 billion instruction reference mentioned before. Fig. 1 shows the relative errors of our 2M Simpoints (with LiveCache on and 4M Detail-warmup) and the 50M Simpoints (with 50M Detail-warmup, a similar approach to [24,25]) with both bar chart (left side, for individual results) and whisker box (right side, for overall results) for all 28 individual applications. The whisker boxes illustrate that the 2M Simpoints incur lower mean error (3.89%) than the popular 50M Simpoints (5.80%). Also, the 2M Simpoints have lower maximum error value and tighter bound ranges. The minimum error values are similar, all close to zero. Clearly, 2M Simpoints exhibit a more concentrated error distribution with higher IPC accuracy.

There is one outlier in the whisker box, *GemsFDTD* which solves the Maxwell equations in 3D in the time domain using the finite-difference time-domain method. Neither 2M size nor 50M size works well with this application. Both sizes produce much higher IPC results than the reference. The full IPC trace has a high IPC with short low IPC spikes that Simpoint does not capture correctly. We believe this should be related with the statistical approach used by Simpoint technology. Further study is out of the scope of this short paper.

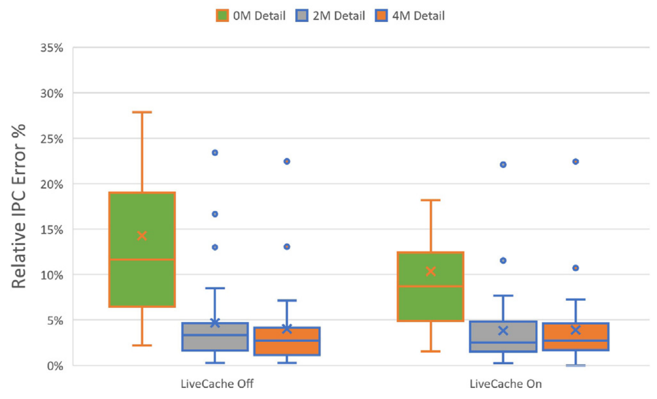


Fig. 2. The relative IPC errors for cases of LiveCache on/off and 0M/2M/4M for Detail-warmup. The best result is obtained with 4M Detail-warmup and LiveCache together while the worst result is from case running 2M Simpoints without LiveCache nor Detail-warmup (LiveCache off/0M Detail-warmup).

3.2.2. *Detail-warmup and LiveCache effects*

The 2M Simpoint results shown in Fig. 1 is obtained with LiveCache and 4M Detail-warmup. To understand their individual effects on the IPC accuracy, we displayed the corresponding results using whisker plot

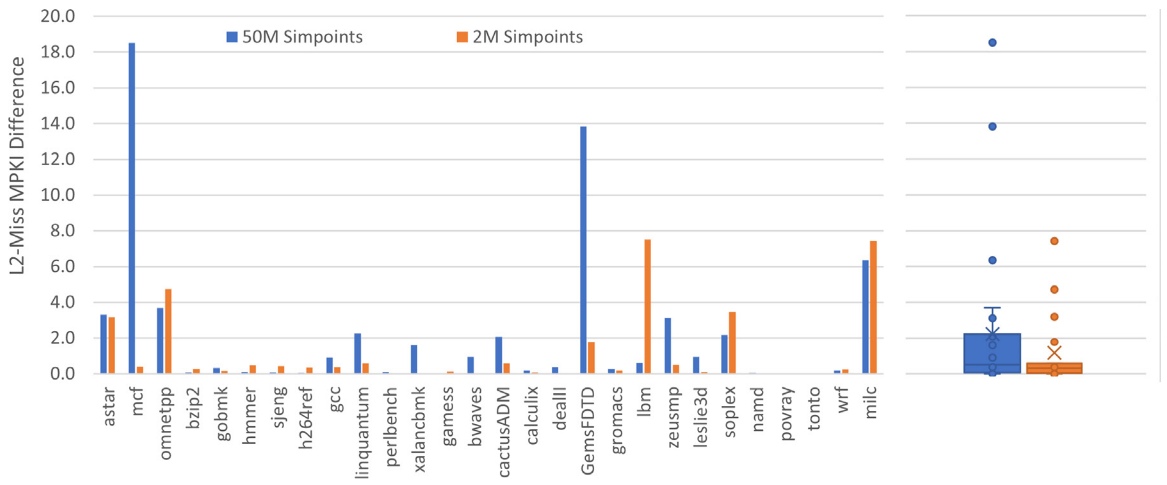


Fig. 3. The differences of L2 misses per thousand instructions (MPKI) with the base reference for both 50M simpoints (with 50M detail warmup) and 2M simpoints (with Livecache and 4M Detail-warmup).

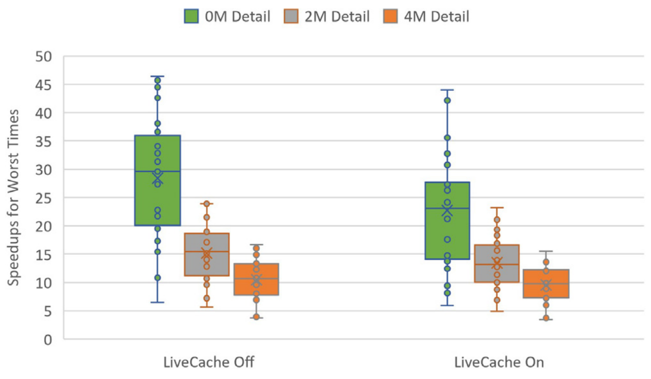


Fig. 4. The worst simulation time speedups of using 2M Simpoint size over 50M Simpoint. The speedup is about 10 times when running 2M Simpoints with LiveCache and 4M Detail-warmup.

in Fig. 2 for six cases: LiveCache on/off, and 0M/2M/4M for Detail-warmup. The leftmost box is for running 2M simpoints without Live-Cache nor Detail-warmups (LiveCache off/0M Detail). Clearly, it generates the highest errors. Enabling either LiveCache or Detail-warmups is essential to improve the accuracy.

First, we examine the performance impact of LiveCache. By comparing the cases with LiveCache on and off (left three whisker boxes vs. corresponding right three whisker boxes), we find that, the errors obtained with LiveCache on concentrated on narrower box ranges and all maximum, mean, and median error values are smaller. Such difference is more phenomenal when there is no Detail-warmup (0M instruction case). With the increase of the Detail-warmup size, LiveCache-warmup effect becomes less and less important. However, increasing Detail-warmup size will surely increases the simulation time. Using LiveCache allows us to shorten the Detail-warmup time so that we can avoid the problem of spending a large amount of simulation time on Detail-warmups [1].

Similarly, the IPC accuracy can be significantly improved when increase the Detail-warmup instructions from 0 to 2M. From 2M to 4M, the results can be still be improved. However, using 8M or larger sizes, the accuracy can no longer be further improved. Our best results are obtained when 4M Detail-warmup size are used.

3.2.3. L2 misses

In addition to the IPC accuracy, we also compare the number of L2 misses, one import metric to measure the memory performance, to observe the direct effects on the cache, although those should be reflected in the IPC results. Fig. 3 shows the MPKI (misses per thousand instructions) differences with the 10 billion base reference for both the 50M simpoints with 50M Detail-warmups and 2M simpoints with LiveCache and 4M Detail-warmups. The left bar chart displays the results for individual benchmark (Lower value indicates the MPKI difference with the base reference is smaller) while the right whisker box illustrates the overall results.

Different from Fig. 1, the use of absolute differences instead of relative ratios in Fig. 3 is due to the fact that for some applications, the L2 MPKI is quite small. Using ratios may exaggerate the differences and lead to incorrect conclusions. For example, the L2 MPKI for *tonto* is only 0.03 for its base reference. For 50M simpoints and 2M simpoints, they are 0.01 and 0.03, respectively. Both are very close to the base case. If we use relative errors, the differences between 50M simpoints and 2M simpoints will be 67% ((0.03–0.01)/0.03) and 0% ((0.03–0.03)/0.03), respectively, which does not accurately reflect reality.

In summary, Fig. 3 shows that, similar to the IPC accuracy, using 2M simpoints not only significantly reduces the maximum error but also delivers much higher average accuracy. Also, the 2M simpoint results are obtained with both LiveCache and 4M Detail-warmups. Running only 2M simpoints itself will generate much higher errors and must be accompanied with LiveCache and Detail-warmup to maintain the accuracy.

3.2.4. Simulation speed

Using 2M Simpoint size instead of 50M, we expect the simulation time can be greatly accelerated, ideally 50 times ((50M + 50M)/2M). However, considering the LiveCache and Detail-warmup overhead, especially the Detail-warmup overhead, the actual speedups will be much lower. We compare the running times of 2M and 50M Simpoint sizes using the worst Simpoint simulation time. All the Simpoints of an application are launched at the same time in parallel until all Simpoints are finished. The benchmark running time is determined by the Simpoint with longest simulation time.

Fig. 4 shows the speedups of using 2M Simpoint size over 50M size in terms of worst running times. With LiveCache only without Detail-warmup, the maximum speedup is about 45 times, close to the ideal expectation of 50 times speedup. The average speedup is about 23. Turning the LiveCache on introduces a constant overhead of reading the data file and loading the data into caches. It takes about 2 min with our simulator. However, comparing with Detail-warmup, its effect on the simulation time is relatively small. With the increase of the Detail-warmup size, the average speedups decreases, falling to around 9 for 4M warmup size. The actual average running time for 50M size is about 450 min while for the 2M size, the average real running times are 22, 34, and 49 min for 0M, 2M, and 4M Detail-warmups, respectively.

4. Conclusion

In this paper, we propose a framework to create reduced size Simpoints for simulation sampling, further reducing simulation time of standard Simpoint simulation with slightly improved accuracy. We achieve the goal by incorporating well established LiveCache and Detail-warmup techniques into our base Simpoint framework. The framework shall benefit whoever relies on computer architecture simulation by significantly reducing simulation time with decent sampling error.

Future works include, but not limited to, studying the performance effects of LiveCache and Detail-warmups in detail, extending our framework to support multi-core multi-threaded benchmark applications; exploring and incorporating various other warmup techniques; and enhancing the checkpoint capability to an arbitrary location of interest.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Authors are currently employed by Futurewei Technologies.

References

[1] R.E. Wunderlich, T.F. Wenisch, B. Falsafi, J.C. Hoe, SMARTS: accelerating microarchitecture simulation via rigorous statistical sampling, in: Proceedings of the 30th Annual International Symposium on Computer Architecture, ISCA, 2003, pp. 84–95.
[2] T.M. Conte, M.A. Hirsch, K.N. Menezes, Reducing state loss for effective trace sampling of superscalar processors, in: Proceedings International Conference on Computer Design: VLSI in Computers and Processors, ICCD, 1996, pp. 468–477.
[3] T.M. Conte, M.A. Hirsch, W.W. Hwu, Combining trace sampling with single pass methods for efficient cache simulation, IEEE Trans. Comput. 47 (6) (1998) 714–720.
[4] T. Sherwood, E. Perelman, G. Hamerly, 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, 2002, pp. 45–57.
[5] E. Perelman, G. Hamerly, B. Calder, Picking statistically valid and early simulation points, in: Proceedings of the 12th International Conference on Parallel Architectures and Cimpilation Techniques, PACT, 2003, pp. 244–255.
[6] https://chipyard.readthedocs.io/en/latest/Tools/Dromajo.html.
[7] N. Kabylkas, T. Thorn, S. Srinath, P. Xekalakis, J. Renau, Effective processor verification with logic fuzzer enhanced co-simulation, in: Proceedings of the 54th International Symposium on Microarchitecture, MICRO, 2005, pp. 667–678.
[8] K.C. Barr, H. Pan, M. Zhang, K. Asanovic, Accelerating multiprocessor simulation with a memory timestamp record, in: Proceedings of 2005 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2005, pp. 66–77.
[9] S. Hassani, G. Southern, J. Renau, LiveSim: Going live with microarchitecture simulation, in: Proceedings of 2016 IEEE International Symposium on High Performance Computer Architecture, HPCA, 2016, pp. 606–617.
[10] A. Agarwal, J. Hennessy, M. Horowitz, Cache performance of operating system and multiprogramming workloads, ACM Trans. Comput. Syst. 6 (4) (1988) 393–431.
[11] S. Laha, J.A. Patel, R.K. Iyer, Accurate low-cost methods for performance evaluation of cache memory systems, IEEE Trans. Comput. 37 (11) (1988) 1325–1336.
[12] S.K. Reinhardt, M.D. Hill, J.R. Larus, A.R. Lebeck, J.C. Lewis, D.A. Wood, The wisconsin wind tunnel: Virtual prototyping of parallel computers, in: Proceedings of the 1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, SIGMETRICS, 1993, pp. 48–60.
[13] R.E. Kessler, M.D. Hill, D.A. Wood, A comparison of trace-sampling techniques for multi-megabyte caches, IEEE Trans. Comput. 43 (6) (1994) 664–675.
[14] J.W. Haskins, K. Skadron, Minimal subset evaluation: Rapid warm-up for simulated hardware state, in: Proceedings of 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors, ICCD, 2001, pp. 32–39.
[15] J.W. Haskins, K. Skadron, Memory reference reuse latency: Accelerated warmup for sampled microarchitecture simulation, in: Proceedings of 2003 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2003, pp. 195–203.
[16] Y. Luo, L. John, L. Eeckhout, Self-monitored adaptive cache warm-up for microprocessor simulation, in: Proceedings of 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2004, pp. 10–17.
[17] L. Eeckhout, Y. Luo, K.D. Bosschere, L.K. John, BLRL: Accurate and efficient warmup for sampled processor simulation, Comput. J. 48 (4) (2005) 451–459.
[18] T.F. Wenisch, R.E. Wunderlich, B. Falsafi, J.C. Hoe, TurboSMARTS: Accurate microarchitecture simulation sampling in minutes, in: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2005, pp. 408–409.
[19] T.F. Wenisch, R.E. Wunderlich, B. Falsafi, J.C. Hoe, Simulation sampling with live-points, in: Proceedings of 2006 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2006, pp. 2–12.
[20] N. Nikoleris, D. Eklov, E. Hagersten, Extending statistical cache models to support detailed pipeline simulators, in: Proceedings of 2014 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2014, pp. 86–95.
[21] N. Nikoleris, L. Eeckhout, E. Hagersten, T.E. Carlson, Directed statistical warming through time traveling, in: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2019, pp. 1037–1049.
[22] https://www.spec.org/cpu2006/.
[23] https://buildroot.org/.
[24] T.E. Carlson, W. Heirman, L. Eeckhout, Sampled simulation of multi-threaded applications, in: Proceedings of 2013 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2013, pp. 2–12.
[25] T. Grass, T.E. Carlson, A. Rico, G. Ceballos, E. Ayguade, M. Casas, M. Moreto, Sampled simulation of task-based programs, IEEE Trans. Comput. 68 (2) (2018) 255–269.