Accelerating the Evaluation of Large Workloads on Post-Dennard Systems using Sampling

Alen Sabu
School of Computing, National University of Singapore

The Context

With the end of Moore's law, computer architects have turned to alternative approaches to enhance computational capabilities. One prominent strategy involves a shift towards increasing the core count and embracing heterogeneity in architectures, complemented by the introduction of several software- and system-level optimizations aimed at improving performance and power efficiencies. As processors/systems continue to evolve in complexity and power, accurately assessing their performance characteristics becomes increasingly intricate. Understanding the workload for the analysis and prediction of performance for future systems is an extremely difficult task. Workloads may have extremely long run times and are fairly sophisticated with OS, library, and hardware requirements. Microarchitecture simulators like gem5 [7] or Sniper [8] are heavily used to estimate the performance of real-world workloads on a futuristic processor and for performance comparisons. The purpose of these simulations is to evaluate the performance of a proposed architecture, identify potential bottlenecks, and improve the efficiency of the hardware design before it is implemented in physical hardware. However, simulators are orders of magnitude (typically, $10,000\times$ or more [7]) slower as compared to native execution. The focus of my research is to develop methodologies to improve the accuracy and efficiency of the performance measurements of post-Dennard computing systems.

Challenges Involved

With the widening gap between simulator performance and the processors they model, running a cycle-accurate full-system simulation of large designs can be extremely time-consuming. Sampled simulation is one key solution to making these extremely long simulation times tractable. Sampled simulation begins with program analysis and determines representatives of an application to simulate by leveraging the correlation between executed code and program performance [30]. Generic multi-threaded sampled simulation has been a long-standing, challenging problem with the potential to help change how researchers study modern, complex computing systems. Yet, a practical solution for reducing complex multi-threaded applications into a representative sample has been elusive. While there are a number of solutions proposed for sampling single-threaded [45, 46, 56, 41, 47, 18, 38, 32, 31, 20], multi-program [51], and multi-process [54, 52] applications, these techniques are not deemed extensible for synchronizing multi-threaded applications [2]. It is challenging to accurately capture or represent the behavior of all threads in multi-threaded applications, as the exact timing and execution of each thread can vary greatly. Existing techniques either do not provide significant speedups to be practical (Time-based Sampling techniques [9, 5] can show less than $10\times$ as speedup compared to fully-detailed simulation) or apply only to particular synchronization types (BarrierPoint [10] for barrier-based workloads). In addition, workload-specific solutions can be rigid with respect to region selection, which can limit the overall simulation speedup when regions are large. A solution is needed that both supports generic multi-threaded applications, no matter the synchronization primitives used, as well as allows for ease of deployment and fast evaluation.

Simulation of Multi-core Systems

We aim to solve these challenges and propose a novel sampled simulation technique for multi-threaded applications, which is both agnostic to the type of synchronization primitives used and scales by the similarity exhibited by the application. We proposed *LoopPoint* [44], a generic multi-threaded sampled simulation methodology that utilizes application loops to represent the amount of work done by the threads. LoopPoint combines several vital features, including (a) repeatable, up-front application analysis, (b) a novel clustering approach to take into account run-time parallelism, and (c) the use of loop-based simulation markers to divide the work into measurable chunks, even in the presence of spin-loops. LoopPoint chooses representative regions within an application called looppoints that act as checkpoints that can be simulated in parallel. These checkpoints can be used to reproduce the performance of the original application and can significantly reduce simulation runtime compared to prior works.

With loop entries as region boundaries, the simulation regions can be demarcated by a $(PC, count)$ pair to represent the loop entry for each simulation region (PC is the address of the loop entry, and $count$ is the number of occurrences of the address). By monitoring the amount of work as represented by loops and not instructions or

barriers, we can isolate multi-threaded application representatives and understand the amount of global work completed. Our methodology enables synchronization-agnostic application sampling for multi-threaded workloads while still scaling the amount of work based on the representative nature of the application by taking into account several key factors like understanding (a) where to simulate, which requires a reproducible analysis technique, and using a precise clustering mechanism that partitions the regions to reduce the workload into its representative components. Moreover, the methodology presents (b) how to simulate the regions to allow the application to take advantage of the underlying hardware while not constraining execution to a deterministic path that might not exhibit true application behavior. The methodology has been adapted to widely used microarchitecture simulators like gem5, Sniper, etc., as well as in the industry. We released the representative checkpoints (as x86 executables or *ELFies* [39]) of a subset of SPEC CPU2017 benchmarks for the public to use.

Current sampling solutions are primarily targeted for microarchitecture-level simulations. Recent works [58] attempted to adapt prior solutions like SimPoint [46] for RTL-level simulations on Verilator [48] using smaller region sizes aiming to improve simulation efficiency, which, however, resulted in accuracy that is typically not acceptable. The result is that it is currently infeasible to evaluate the performance of large workloads on the RTL level. While FPGA simulation infrastructures, such as Firesim [26], offer a faster alternative for simulation, FPGAs are specialized devices with inherent limitations in terms of memory capacity and processing units. Therefore it is often not possible to fit large, realistic processor models on FPGAs. Previously proposed workload sampling methodologies typically rely on fixed-length intervals for analysis, which can often be out of sync with the periodicity of program execution. Since an application's phase behavior [33, 4, 40] is strongly correlated to the code it executes, it can exhibit a hierarchy of phase behaviors that can be observed at various interval lengths, rendering conventional sampling techniques inadequate.

This highlights the need for developing specialized workload sampling methodologies that can be flexibly applied to both microarchitecture-level and RTL-level simulations. These methodologies should support finer region granularities that align with the dynamic phase behavior exhibited by the application. By tailoring the sampling approach to capture the specific characteristics and phases of the workload, more accurate and efficient sampled simulations can be performed at both the microarchitecture and RTL levels. In follow-up work, *Viper*, we improved LoopPoint to determine the simulation regions more systematically, which resulted in shorter simulation regions better suited for RTL simulations. Utilizing the innate program structures instead of fixed-length intervals allows for flexible region sizes that are more likely to be aligned with the application periodicity, thereby reducing the possibility of aliasing. We evaluate Viper using NEMU-based RISC-V checkpoints for RTL simulations. We also show that Viper performs better than SimPoint [46] for single-threaded applications.

High-performance, multi-core processors are the key to accelerating workloads in several application domains. To continue to scale performance at the limit of Moore's Law and Dennard scaling, software and hardware designers have turned to dynamic solutions that adapt to the needs of applications in a transparent, automatic way. For example, modern hardware improves its performance and power efficiency by changing the hardware configuration, like the frequency and voltage of cores, according to a number of parameters such as the technology used, the workload running, etc. Techniques such as dynamic voltage and frequency scaling (DVFS) [17, 23, 29], dynamic cache reconfiguration [35, 34, 3], TurboBoost [11], etc., have been developed to adjust the hardware state in response to executed instructions and active processes. Additionally, dynamic scheduling techniques [15] have been developed for multi-threaded applications. In such cases, profile-driven sampling methodologies may result in different performances for each execution. With this level of dynamism, it is essential to simulate next-generation multi-core processors in a way that can both respond to system changes and accurately determine system performance metrics. Currently, no sampled simulation platform can achieve these goals of dynamic, fast, and accurate simulation of multi-threaded workloads. To quickly estimate the performance of multithreaded applications running on next-generation dynamic hardware and software, a sampled simulation methodology is needed that can dynamically adapt to changes in the system at runtime while accurately determining relevant performance metrics.

We propose *Pac-Sim*, which is designed for fast and efficient simulation of multi-threaded applications without the need for any up-front application analysis and allows for the simulation of dynamically scheduled multi-threaded applications even in the presence of runtime hardware events – this was not possible with previously proposed sampled simulation methodologies. This proposed methodology includes an online sampling and decision-making phase based on predictions that rely on previously executed code, thereby completely eliminating the need for offline profiling. We incorporate application analysis to guide sampled simulations, similar to SimPoint-like [46] methodologies but without the need for upfront pre-processing, as seen in SMARTS-like [57] methodologies. We make intelligent simulation decisions through online learning and implement lightweight online profiling, clustering, and warmup techniques for optimal performance. Moreover, the proposed methodology can accommodate hardware state changes, software features, and other factors that affect simulation results.

Validation of Selected Sample

Workload sampling can significantly speed up the simulation performance, assuming the regions of interest (ROIs) or the representative sample found can be proven to accurately represent the behavior of the full workload. One standard way to validate the representativeness [27, 55, 19] of the regions of interest is to measure the prediction error, which is the difference in the performance of the full workload and the predicted performance obtained using the representative regions, and the performance is typically obtained through simulation [49]. However, the simulation of long-running workloads is infeasible, taking months to years.

We propose *ROIperf*, a framework that assesses the quality of checkpoint-driven workload sampling methodologies. ROIperf leverages the native hardware performance counters [42] by evaluating both the full workload and its representative regions on real hardware systems. This approach ensures the validation of regions of interest through the measurement of performance on real hardware instead of simulation. The methodology is particularly beneficial for long-running programs for which the prevailing simulation-based validation technique is not feasible. While this technique does not allow for projection to future hardware (where timing simulation is needed), this path enables one to evaluate if the selected regions of interest are representative and, therefore, can be used to determine the overall performance characteristics of the workload accurately. We demonstrate the efficacy of ROIperf by evaluating various sample selection methodologies across a wide range of workloads. ROIperf provides a significant speedup in validating regions selected for simulation.

Simulation of Heterogeneous Systems

Computation exists everywhere in this era, spanning from large-scale systems to low-power devices, mobile CPUs, etc. There has been a profound increase in the demand for high-performance computing (HPC) resources in recent years. However, multi-core architectures cease to scale due to the associated power and thermal constraints (power wall), which limits their ability to deliver significant performance improvements [16]. This resulted in a shift to domain-specific architectures and accelerators. Embracing heterogeneity in architectures is the way forward for continued performance improvements [14] to meet the growing computational demands.

GPUs have emerged as the most widely used general-purpose accelerators in modern data centers [21] and supercomputers [50] that accelerate massively parallel big data analysis [12, 24] and machine learning [1, 13] workloads. While previous works have investigated understanding the performance of workloads that consist of CPU [46, 56, 53, 9, 10] and GPU [22, 25, 6, 36] applications independently, hybrid solutions that support analysis and workload reduction for multiple types of heterogeneous workloads, from CPUs, GPUs, and even custom hardware accelerators (like FPGAs), have not yet been investigated. Given the importance of these workloads, from HPC systems to data center use, simulation of heterogeneous workloads is key to understanding the interactions between compute components and how their interactions can affect overall runtime performance.

We gather our recent findings to sample heterogeneous CPU-GPU workloads and propose a unified sampling solution, *X-Point*, that can accurately (a) understand the workloads running on heterogeneous systems to (b) build a representative sample to more easily understand and interpret these results. We also aim to (c) extend ROIperf infrastructure to support the ROI validation of heterogeneous applications. The primary goal of X-Point is to develop comprehensive methodologies for evaluating the performance of CPU-GPU systems across a spectrum of applications. By investigating a wide range of workloads spanning from scientific simulations to artificial intelligence, we target to quickly evaluate CPU and GPU resource utilization, memory hierarchies, and system bottlenecks. The research involves studying real-world applications and designing workload characterization techniques to generate representative traces that are supported on heterogeneous simulators like MacSim [28] and gem5-gpu [43]. These traces will help validate simulation models and enable architects to make informed design decisions.

Future Work

The collaboration between CPUs and GPUs requires a holistic architectural approach. With the recent introduction of tightly-coupled heterogeneous systems like NVIDIA Grace Hopper Superchip [37] that are capable of addressing future computational demands, the need for efficient and scalable tools and methodologies to evaluate these systems has become increasingly important. I intend to explore novel architectures that integrate heterogeneous computing resources, enabling tighter coupling and improved data sharing between these components. Through in-depth analysis, I seek to identify workload traits that are amenable to parallelism, offloading, and data movement optimization, thereby enabling smarter task allocation and resource allocation strategies. I also intend to devise checkpointing mechanisms for such systems to help with simulation and debugging. The outcomes of this work will contribute significantly to the field of computer architecture by providing architects with useful performance evaluation tools to explore and optimize CPU-GPU systems. Moreover, this research will influence the development of programming models that align with the capabilities of emerging architectures.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[2] A.R. Alameldeen and D.A. Wood. IPC considered harmful for multiprocessor workloads. *IEEE Micro*, 26(4):8–17, 2006.

[3] D.H. Albonesi. Selective cache ways: on-demand cache resource allocation. In *MICRO-32. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, pages 248–259, 1999.

[4] Murali Annavaram, Ryan Rakvic, Marzia Polito, J-Y Bouguet, Richard Hankins, and Bob Davies. The fuzzy correlation between code and performance predictability. In *37th International Symposium on Microarchitecture (MICRO-37'04)*, pages 93–104. IEEE, 2004.

[5] E. K. Ardestani and J. Renau. ESESC: A fast multicore simulator using time-based sampling. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 448–459, February 2013.

[6] Cesar Avalos Baddouh, Mahmoud Khairy, Roland N Green, Mathias Payer, and Timothy G Rogers. Principal kernel analysis: A tractable methodology to simulate scaled gpu workloads. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 724–737, 2021.

[7] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Computer Architecture News*, 39(2):1–7, August 2011.

[8] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 52:1–52:12, November 2011.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout. Sampled simulation of multi-threaded applications. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 2–12, April 2013.

[10] T. E. Carlson, W. Heirman, K. Van Craeynest, and L. Eeckhout. BarrierPoint: Sampled simulation of multi-threaded applications. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 2–12, March 2014.

[11] James Charles, Preet Jassi, Narayan S Ananth, Abbas Sadat, and Alexandra Fedorova. Evaluation of the intel® core™ i7 turbo boost feature. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, pages 188–197. IEEE, 2009.

[12] Cen Chen, Kenli Li, Aijia Ouyang, Zhuo Tang, and Keqin Li. Gpu-accelerated parallel hierarchical extreme learning machine on flink for big data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(10):2740–2753, 2017.

[13] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. Tvm: An automated end-to-end optimizing compiler for deep learning. *arXiv preprint arXiv:1802.04799*, 2018.

[14] Eric S Chung, Peter A Milder, James C Hoe, and Ken Mai. Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus? In *2010 43rd annual IEEE/ACM international symposium on microarchitecture*, pages 225–236. IEEE, 2010.

[15] Alejandro Duran, Eduard Ayguadé, Rosa M Badia, Jesús Labarta, Luis Martinell, Xavier Martorell, and Judit Planas. Ompss: a proposal for programming heterogeneous multi-core architectures. *Parallel processing letters*, 21(02):173–193, 2011.

[16] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th annual international symposium on Computer architecture*, pages 365–376, 2011.

[17] Stijn Eyerman and Lieven Eeckhout. Fine-grained dvfs using on-chip regulators. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(1):1–24, 2011.

[18] Greg Hamerly, Erez Perelman, and Brad Calder. How to use SimPoint to pick simulation points. *ACM SIGMETRICS Performance Evaluation Review*, 31(4):25–30, March 2004.

[19] Haiyang Han and Nikos Hardavellas. Public release and validation of spec cpu2017 pinpoints. *arXiv preprint arXiv:2112.06981*, 2021.

[20] Sina Hassani, Gabriel Southern, and Jose Renau. LiveSim: Going live with microarchitecture simulation. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 606–617, March 2016.

[21] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[22] Jen-Cheng Huang, Lifeng Nai, Hyesoon Kim, and Hsien-Hsin S Lee. Tbpoint: Reducing simulation time for large-scale gpgpu kernels. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 437–446. IEEE, 2014.

[23] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 347–358. IEEE, 2006.

[24] Hai Jiang, Yi Chen, Zhi Qiao, Tien-Hsiung Weng, and Kuan-Ching Li. Scaling up mapreduce-based big data processing on multi-gpu systems. *Cluster Computing*, 18:369–383, 2015.

[25] Melanie Kambadur, Sunpyo Hong, Juan Cabral, Harish Patil, Chi-Keung Luk, Sohaib Sajid, and Martha A Kim. Fast computational gpu design with gt-pin. In *2015 IEEE International Symposium on Workload Characterization*, pages 76–86. IEEE, 2015.

[26] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee, N. Pemberton, E. Amaro, C. Schmidt, A. Chopra, Q. Huang, K. Kovacs, B. Nikolic, R. Katz, J. Bachrach, and K. Asanovic. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *International Symposium on Computer Architecture (ISCA)*, pages 29–42, June 2018.

[27] Humayun Khalid. Validating trace-driven microarchitectural simulations. *IEEE Micro*, 20(6):76–82, 2000.

[28] Hyesoon Kim, Jaekyu Lee, Nagesh B Lakshminarayana, Jaewoong Sim, Jieun Lim, and Tri Pho. Macsim: A cpu-gpu heterogeneous simulation framework user guide. *Georgia Institute of Technology*, 2012.

[29] Wonyoung Kim, Meeta S Gupta, Gu-Yeon Wei, and David Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134. IEEE, 2008.

[30] J. Lau, J. Sampson, E. Perelman, G. Hamerly, and B. Calder. The strong correlation between code signatures and performance. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2005.

[31] Jeremy Lau, Erez Perelman, and Brad Calder. Selecting software phase markers with code structure analysis. In *International Symposium on Code Generation and Optimization (CGO)*, pages 135–146, March 2006.

[32] Jeremy Lau, Erez Perelman, Greg Hamerly, Timothy Sherwood, and Brad Calder. Motivation for variable length intervals and hierarchical phase behavior. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 135–146, March 2005.

[33] Jeremy Lau, Stefan Schoemackers, and Brad Calder. Structures for phase classification. In *IEEE International Symposium on-ISPASS Performance Analysis of Systems and Software, 2004*, pages 57–67. IEEE, 2004.

[34] Sparsh Mittal, Yanan Cao, and Zhao Zhang. Master: A multicore cache energy-saving technique using dynamic cache reconfiguration. *IEEE Transactions on very large scale integration (VLSI) systems*, 22(8):1653–1665, 2013.

[35] Sparsh Mittal, Zhao Zhang, and Jeffrey S Vetter. Flexiway: A cache energy saving technique using fine-grained cache reconfiguration. In *2013 IEEE 31st international conference on computer design (ICCD)*, pages 100–107. IEEE, 2013.

[36] Mahmood Naderan-Tahan, Hossein SeyyedAghaei, and Lieven Eeckhout. Sieve: Stratified gpu-compute workload sampling. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 224–234. IEEE, 2023.

[37] Nvidia gh200 grace hopper superchip architecture. `https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper/`, 2023.

[38] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. In *International Symposium on Microarchitecture (MICRO)*, pages 81–92, December 2004.

[39] Harish Patil, Alexander Isaev, Wim Heirman, Alen Sabu, Ali Hajiabadi, and Trevor E Carlson. ELFies: Executable region checkpoints for performance analysis and simulation. In *International Symposium on Code Generation and Optimization (CGO)*, pages 126–136, February/March 2021.

[40] E. Perelman, M. Polito, J.-Y. Bouguet, J. Sampson, B. Calder, and C. Dulong. Detecting phases in parallel applications on shared memory architectures. In *International Parallel Distributed Processing Symposium (IPDPS)*, April 2006.

[41] Erez Perelman, Greg Hamerly, and Brad Calder. Picking statistically valid and early simulation points. In *2003 12th International Conference on Parallel Architectures and Compilation Techniques*, pages 244–255. IEEE, 2003.

[42] Performance monitoring in the intel 64 and ia-32 architectures software developers manual, volume 3b. `https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.html`.

[43] Jason Power, Joel Hestness, Marc S Orr, Mark D Hill, and David A Wood. gem5-gpu: A heterogeneous cpu-gpu simulator. *IEEE Computer Architecture Letters*, 14(1):34–36, 2014.

[44] Alen Sabu, Harish Patil, Wim Heirman, and Trevor E. Carlson. LoopPoint: Checkpoint-driven sampled simulation for multi-threaded applications. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 604–618, 2022.

[45] Timothy Sherwood, Erez Perelman, and Brad Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proceedings 2001 International Conference on Parallel Architectures and Compilation Techniques*, pages 3–14. IEEE, 2001.

[46] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. Automatically characterizing large scale program behavior. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 45–57, October 2002.

[47] Timothy Sherwood, Suleyman Sair, and Brad Calder. Phase tracking and prediction. *ACM SIGARCH Computer Architecture News*, 31(2):336–349, 2003.

[48] W Snyder. Verilator: the fast free verilog simulator. *URL: http://www. veripool. org*, 2012.

[49] Rajat Todi. Speclite: using representative samples to reduce spec cpu2000 workload. In *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization (WWC-4)*, pages 15–23. IEEE, 2001.

[50] TOP500. Top500 supercomputer sites. `https://www.top500.org/`, 2022. Accessed on November 16, 2022.

[51] M. Van Biesbrouck, T. Sherwood, and B. Calder. A co-phase matrix to guide simultaneous multithreading simulation. In *IEEE International Symposium on - ISPASS Performance Analysis of Systems and Software, 2004*, March 2004.

[52] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. SimFlex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4):18–31, 2006.

[53] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. SimFlex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4):18–31, 2006.

[54] Thomas F. Wenisch, Roland E. Wunderlich, Babak Falsafi, and James 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*, SIGMETRICS '05. ACM, 2005.

[55] Qinzhe Wu, Steven Flolid, Shuang Song, Junyong Deng, and Lizy K John. Invited paper for the hot workloads special session hot regions in spec cpu2017. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 71–77. IEEE, 2018.

[56] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *International Symposium on Computer Architecture (ISCA)*, pages 84–97, June 2003.

[57] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *International Symposium on Computer Architecture (ISCA)*, pages 84–97, June 2003.

[58] Yinan Xu, Zihao Yu, Dan Tang, Guokai Chen, Lu Chen, Lingrui Gou, Yue Jin, Qianruo Li, Xin Li, Zuojun Li, et al. Towards developing high performance risc-v processors using agile methodology. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1178–1199. IEEE, 2022.