RESEARCH STATEMENT

Alen Sabu

School of Computing, National University of Singapore, Singapore

The Context

Central processing units (CPUs or processors) have long been the cornerstone of computing, responsible for executing instructions and managing system resources. To ensure efficient resource allocation and meet the ever-growing computational demands, accurately estimating the performance of processors is essential. Computer architects typically rely on microarchitectural simulations to assess system performance metrics and compare design choices. The processor designs undergo a comprehensive evaluation of power consumption, performance capabilities, area requirements, and their trade-offs prior to fabrication.

With the end of Moore's law [45], computer architects have turned to alternative approaches to enhance computational capabilities. One prominent strategy involves a shift towards increasing the core count [9, 2] and embracing heterogeneity in architectures [16], complemented by the introduction of several software- and system-level optimizations aimed at improving performance and power efficiencies [41]. 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 performance prediction of 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 [8] and Sniper [10] are heavily used to estimate the performance of real-world workloads on a futuristic processor. 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 [8]) slower as compared to native execution. For example, the detailed simulation of SPEC CPU2017 benchmarks may take months to years, as shown in Figure 1. This challenge is further exacerbated by the increasing complexity of modern architectures, which necessitates the development of efficient performance evaluation techniques. The focus of my research is to address this critical gap by proposing novel workload sampling methodologies that enable fast and accurate performance evaluation of future systems.

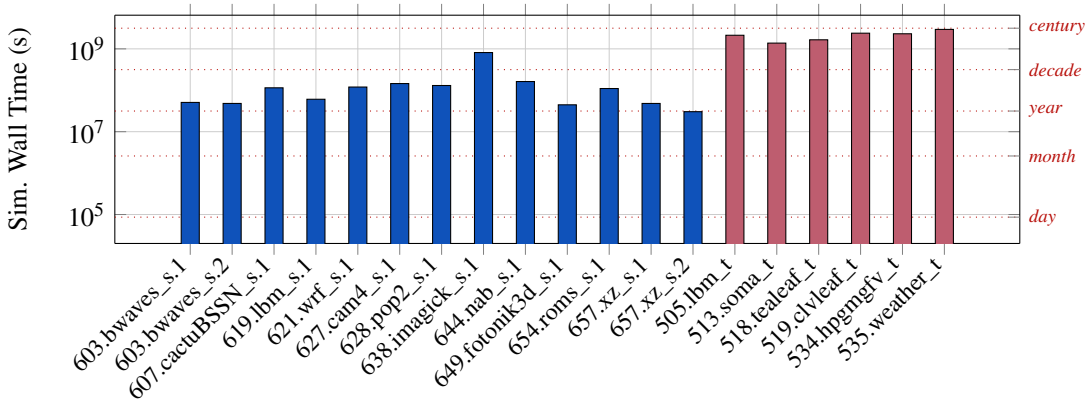

Figure 1: The estimated wall times (in seconds) for the full simulation of multi-threaded (eight OpenMP threads) SPEC CPU2017 benchmarks and SPEChpc 2021 Tiny benchmarks (rank=1) both using *Reference* inputs. The benchmarks were compiled with the Intel oneAPI toolchain. We assume the simulation speed of gem5 (CPU portion) and AccelSim (GPU portion) to estimate the wall times based on the instruction counts of the benchmarks.

Challenges Involved

For several decades, Moore's law, coupled with Dennard scaling [18], fueled exponential performance gains in single-core processors. However, as Dennard scaling reached its physical limits, the industry shifted its focus to multi-core and heterogeneous architectures. This effectively extended the performance gains predicted by Moore's Law but also necessitated the development of entirely new techniques and infrastructures to accurately evaluate system performance in the post-Dennard era.

The widening performance disparity between microarchitecture simulators and the systems they model necessitates exploring alternative simulation techniques. Cycle-accurate full-system simulation, while invaluable for design verification and performance analysis, becomes increasingly time-consuming for multi-core architectures and heterogeneous CPU-GPU architectures. GPUs, characterized by their numerous execution units with a large number of threads, can

lead to significant slowdowns when simulated on traditional CPUs [33, 42]. Sampled simulation is considered a sophisticated solution to making these extremely long simulation times tractable. This technique employs program analysis to determine representative regions of an application for detailed simulation. Sampled simulation methodologies exploit the well-established correlation between executed code and program performance, as shown in prior research [5, 36].

While there are a number of solutions proposed for sampling single-threaded [54, 55, 65, 51, 56, 24, 48, 38, 37, 26], multi-program [60], and multi-process [62, 61] applications to accelerate simulation, these techniques are not deemed extensible for multi-threaded and heterogeneous workloads. Multi-threaded applications tend to synchronize the threads at certain points during execution and shared memory accesses, presenting a unique challenge [3]. This challenge is particularly evident in heterogeneous systems, where diverse compute units are closely integrated. In such cases, representing the amount of work done by the threads or compute units in terms of instructions per cycle (IPC), as shown to work for single-core performance, may lead to inaccurate measurements. It is also challenging to accurately capture or represent the execution pattern of the compute units, as the exact timing of each compute core can vary greatly.

Existing techniques for the sampled simulation of multi-threaded applications either do not provide significant speedups to be practical (Time-Based Sampling techniques [11, 6] can show less than 10× speedup as compared to fully-detailed simulation) or apply only to particular synchronization types (BarrierPoint [12] for barrier-based workloads). A solution is needed that both supports generic multi-threaded applications, irrespective of the synchronization primitives used, as well as allows for fast evaluation.

Most of the prior research on sampled simulation assumes the system to be static. However, 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) [20, 29, 35], dynamic cache reconfiguration [44, 43, 4], TurboBoost [13], etc., have been developed to adjust the hardware state in response to executed instructions and active processes. Additionally, dynamic scheduling techniques [19] have been developed for multi-threaded applications. To quickly estimate the performance of multi-threaded 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.

The profound increase in the demand for high-performance computing (HPC) resources in recent years has driven the widespread adoption of heterogeneous architectures, such as CPU-GPU systems [16]. However, evaluating the performance of these systems poses a significant challenge due to the lengthy simulations involved. While some efforts have addressed these challenges for specific workload classes [23, 22], they are often rigid with respect to region selection and can limit the overall simulation speedup when regions are large. Existing sampled simulation techniques for GPU kernels [28, 7, 42] may not represent an accurate performance estimate of the entire system in such cases. This highlights the need for sampled simulation techniques specifically designed for heterogeneous applications.

Simulation of Multi-core Systems

We aim to solve the challenges related to multi-threaded applications and propose a novel sampled simulation technique, which is both agnostic to the type of synchronization primitives used and scales by the similarity exhibited by the application. We proposed *LoopPoint* [53], 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 a multi-threaded application that serve as checkpoints, allowing parallel simulation. These checkpoints can reproduce the performance of the original application and can significantly reduce simulation runtime compared to prior works.

LoopPoint considers loop-based regions demarcated by loop entries, allowing for repeatable regions. 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. LoopPoint enables synchronization-agnostic application sampling for multi-threaded workloads while still scaling the amount of work based on the representative nature of the application. 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* [49]) of a subset of SPEC CPU2017 benchmarks for the public to use.

While sampling techniques like BarrierPoint and LoopPoint improve the efficiency of microarchitectural simulations, the granularity of the identified regions may not be suitable for achieving comparable speedups at the RTL level. Recent works [66] attempted to adapt prior solutions like SimPoint [55] for RTL-level simulations on Verilator [58] 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 [32], 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.

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. 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 [39, 5, 50] 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. 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. We proposed *Viper* 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.

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. 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.

We proposed *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. Pac-Sim 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 [55] methodologies but without the need for upfront pre-processing, as seen in SMARTS-like [65] methodologies. Pac-Sim makes intelligent simulation decisions through online learning and implements 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.

Simulation of Heterogeneous Systems

The prevalence of CPU-GPU architectures in heterogeneous computing arises from their ability to address the evolving demands of modern workloads, coupled with their well-established programming models. GPUs have emerged as the most widely used general-purpose accelerators in modern data centers [27] and supercomputers [57] that accelerate massively parallel big data analysis [14, 30] and machine learning [1, 15] workloads. While previous works have investigated characterizing workloads that consist of CPU components [55, 65, 61, 11, 12] and GPU [28, 31, 7, 46] components independently, as well as their comparative analyses [40], 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 proposed *XPU-Point*, a unified sampling methodology for heterogeneous workloads that can accurately (a) understand the workloads running on heterogeneous systems to (b) build a representative sample for the fast and accurate performance analysis of the workloads. We also (c) estimate the accuracy of the proposed sampling methodology. XPU-Point proposes a comprehensive methodology for the sampled performance evaluation across a broad spectrum of real-world workloads, from scientific simulations to artificial intelligence on heterogeneous CPU-GPU architectures. This enables computer architects and performance researchers to quickly estimate the performance of long-running, heterogeneous workloads using sampled simulation, which was not possible before. While the primary focus of XPU-Point is to enable sampled microarchitecture simulation, its methodology can be adapted to broader performance analysis and characterization of several classes of heterogeneous workloads.

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 [34, 64, 25] of the ROIs is to measure the sampling error, which is the difference in the performance of the full workload and the extrapolated performance using the ROIs. The performance is typically obtained through detailed simulations [59]. However, the simulation of long-running workloads is infeasible, taking months to years.

We propose *ROIperf*, a framework that validates the ROIs selected using workload sampling methodologies. ROIperf leverages the native hardware performance counters [52] by evaluating both the full workload and its representative regions on real hardware systems. This approach ensures the validation of ROIs through the performance measurement on real hardware instead of simulation. The methodology is particularly beneficial for long-running programs for which the prevailing simulation-based validation techniques are infeasible. While this technique does not allow for the performance estimation of future hardware (where timing simulation is needed), this path enables one to evaluate if the selected ROIs 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 achieves a significant speedup in validating regions selected for simulation.

Future Research

Architects tend to integrate computing devices like CPUs and GPUs to share L3 cache, memory, etc., for higher throughput. For instance, NVIDIA Grace-Hopper [47] utilizes a CPU-GPU coherent memory model, while Apple's M-series processors deliver CPUs and GPUs on the same die that share memory. The trend towards tightly coupled CPU-GPU computing will continue, especially with the advancement in chiplet-based [17] IC packaging [63] technologies, enabling the tight integration of CPUs and GPUs within a single package, as seen in Intel's Meteor Lake architecture [21]. I intend to build efficient and scalable tools and methodologies to evaluate these systems and 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.

While workload sampling offers a solution by focusing on representative subsets of the workload, it represents just one direction for further research. Future investigations to explore complementary techniques for faster simulation include hardware emulation techniques, GPU-based simulations, analytical models, etc. We outline a few such directions below:

1. The characterization of emerging real-world workloads presents a significant challenge. With the increasing complexity of these workloads, like those found in mobile and data center environments, the methodologies proposed here may be inadequate. Current approaches to GPU and heterogeneous CPU-GPU sampling, while effective in controlled settings, exhibit limitations when dealing with real-world scenarios, such as over-subscribed GPUs, dynamic kernels, and multi-GPU configurations. Leveraging neural networks for program phase identification presents an alternative for BBVs or other signature vectors. Their ability to learn from program structure and runtime states allows for the effective characterization of application regions independent of the underlying ISA.

2. Accelerating cycle-accurate simulations remains a crucial area of research. The traditional simulation takes an extremely long time by simulating the entire workload on the CPU alone. One approach to speed up simulation involves leveraging profiling tools to understand the control flow of the workload, which can be used to optimize the datapath and parallelize the simulation of independent components across GPUs. In a similar direction, simulating CPU-GPU workloads can be accelerated by offloading the GPU kernel simulation onto actual GPUs. However, this approach may necessitate frequent CPU-GPU synchronization. Additionally, combining simulators with analytical models of traditional microarchitectural structures offers another potential for simulation speedups.

3. While sampled simulation addresses the problem of long simulation times, this introduces the challenge of warming up the microarchitectural state. Traditional CPU warmup techniques, such as statistical warming and checkpointing, are unavailable for GPU systems and heterogeneous CPU-GPU systems, whereas functional warming would incur a significant amount of time to be spent on simulations just for microarchitectural state reconstruction. Leveraging machine learning techniques on statistical profiles of relevant memory access patterns, branch predictor behavior, and prefetcher accesses could enable the development of sophisticated microarchitectural warmup methods.

4. While application checkpointing and deterministic replay have been established for CPU workloads, extending these techniques to heterogeneous CPU-GPU environments presents a significant research opportunity. For CPU workloads, ELFies offer a widely adopted solution for capturing application state through executable checkpoints. However, ELFies do not capture the system state, and system-level checkpointing techniques using QEMU are essential to capture the entire system state for accurate full-system simulation.

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] Anant Agarwal and Markus Levy. The kill rule for multicore. In *Proceedings of the 44th annual Design Automation Conference*, pages 750–753, 2007.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual design automation conference*, pages 746–749, 2007.

[10] 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.

[11] 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.

[12] Trevor E Carlson, Wim Heirman, Kenzo Van Craeynest, and Lieven Eeckhout. BarrierPoint: Sampled simulation of multi-threaded applications. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 2–12, 2014.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] DARPA. Common heterogeneous integration and ip reuse strategies (chips). `https://www.darpa.mil/program/common-heterogeneous-integration-and-ip-reuse-strategies`, 2024.

[18] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 1974.

[19] 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.

[20] 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.

[21] Wilfred Gomes, Slade Morgan, Boyd Phelps, Tim Wilson, and Erik Hallnor. Meteor lake and arrow lake intel next-gen 3d client architecture platform with foveros. In *IEEE Hot Chips 34 Symposium (HCS)*, pages 1–40. IEEE Computer Society, 2022.

[22] T. Grass, T. E. Carlson, A. Rico, G. Ceballos, E. Ayguadé, M. Casas, and M. Moreto. Sampled simulation of task-based programs. *Transactions on Computers (TC)*, 68(2):255–269, 2019.

[23] T. Grass, A. Rico, M. Casas, M. Moreto, and E. Ayguadé. TaskPoint: Sampled simulation of task-based programs. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 296–306, April 2016.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] Mahmoud Khairy, Zhesheng Shen, Tor M Aamodt, and Timothy G Rogers. Accel-sim: An extensible simulation framework for validated gpu modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 473–486. IEEE, 2020.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] Victor W Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 451–460, 2010.

[41] Charles E Leiserson, Neil C Thompson, Joel S Emer, Bradley C Kuszmaul, Butler W Lampson, Daniel Sanchez, and Tao B Schardl. There's plenty of room at the top: What will drive computer performance after moore's law? *Science*, 368(6495):eaam9744, 2020.

[42] Changxi Liu, Yifan Sun, and Trevor E. Carlson. Photon: A fine-grained sampled simulation methodology for gpu workloads. In *International Symposium on Microarchitecture (MICRO)*, page 1227–1241, New York, NY, USA, 2023. Association for Computing Machinery.

[43] 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.

[44] 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.

[45] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 2006.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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`.

[53] 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.

[54] 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.

[55] 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.

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

[57] Supercomputer Sites. Top500 supercomputer sites. `https://www.top500.org/`, 2022.

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

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] CP Wong and Michelle M Wong. Recent advances in plastic packaging of flip-chip and multichip modules (mcm) of microelectronics. *IEEE Transactions on Components and Packaging Technologies*, 22(1):21–25, 1999.

[64] 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.

[65] 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.

[66] 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.