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

Alen Sabu
School of Computing, National University of Singapore

Introduction

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 designs 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 [2] or Sniper [3] 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) slower as compared to native execution. My current research focus is to develop methodologies to improve the accuracy and efficiency of the performance measurements of modern computing systems. In the future, my goal is to build tightly-coupled high-performance, energy-efficient heterogeneous systems and to develop tools and techniques to quickly evaluate the performance of such systems.

Simulating Multi-threaded Workloads

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 starts with program analysis and determines representatives of an application to simulate. While there are a number of solutions proposed for sampling single-threaded applications, these techniques are not deemed extensible for synchronizing multi-threaded applications [1]. 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. Previously proposed methodologies for multi-threaded applications are either too slow to be practical or are applicable only to a certain class of the workload. On this front, we proposed *LoopPoint* [9], a generic multi-threaded sampled simulation methodology that utilizes application loops to represent the amount of work done by the threads.

LoopPoint chooses representative regions within an application called looppoints that can act as checkpoints. 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 simulators like gem5, Sniper, etc., as well as in the industry. We released the representative checkpoints (as x86 executables or *ELFies* [7]) of a subset of SPEC CPU2017 benchmarks for the public to use.

Current sampling solutions are primarily targeted for microarchitecture-level simulations. Recent works [15] attempted to adapt these solutions for RTL-level simulations on Verilator [12] 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 [4], 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 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. The region sizes that SimPoint and LoopPoint identify are still large for RTL simulations. 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 chances of aliasing. We evaluate Viper using NEMU-based RISC-V checkpoints for RTL simulations. We also show that Viper performs better than SimPoint [10] for single-threaded applications.

Researchers have introduced several dynamic optimization techniques in hardware and software to achieve higher performance and reduce power consumption. 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) and cache reconfiguration have been developed to adjust the hardware state in response to executed instructions and active processes. Additionally, dynamic scheduling techniques 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.

While LoopPoint solves a long-standing problem, evaluating multi-threaded workloads in dynamically optimized scenarios requires a methodology that adapts during runtime. 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 profiling. We incorporate application analysis to guide sampled simulations, similar to SimPoint-like methodologies but without the need for upfront pre-processing, as seen in SMARTS-like [14] 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.

Simulating Heterogeneous Workloads

There has been a profound increase in the demand for high-performance computing resources, resulting in the emergence of domain-specific architectures and accelerators. Computation exists everywhere in this era, from data centers to low-power devices, mobile CPUs, etc. While previous works have investigated understanding the performance bottlenecks in workloads that consist of CPU and GPU applications, hybrid solutions that support analysis and workload reduction for multiple types of 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. The primary goal of X-Point is to develop comprehensive methodologies for evaluating the performance of CPU-GPU systems across a spectrum of applications. We utilize instrumentation tools developed using Pin [6] along with NVBit [13] and GTPin [11] to instrument heterogeneous workloads. Another goal of the project is to devise checkpointing mechanisms for such systems to help with simulation and debugging. 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 [5] and gem5-gpu [8]. 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. I intend to explore novel architectures that seamlessly 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. The outcomes of my research will contribute significantly to the field of computer architecture by providing architects with powerful performance evaluation tools to explore and optimize CPU-GPU systems. Moreover, my research will influence the development of programming models that align with the capabilities of emerging architectures.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Conference on Programming Language Design and Implementation (PLDI)*, pages 190–200, June 2005.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Alex Skaletsky, Konstantin Levit-Gurevich, Michael Berezalsky, Yulia Kuznetcova, and Hila Yakov. Flexible binary instrumentation framework to profile code running on intel gpus. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 109–120. IEEE, 2022.

[12] W Snyder. Verilator: the fast free verilog simulator. 2012.

[13] Oreste Villa, Mark Stephenson, David Nellans, and Stephen W Keckler. Nvbit: A dynamic binary instrumentation framework for nvidia gpus. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 372–383, 2019.

[14] 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.

[15] 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.