Current Challenges in Simulations of HPC Systems

Salvador Petit
Department of Computer Engineering
Universitat Politècnica de València
València, Spain
Email: spetit@disca.upv.es

EXTENDED ABSTRACT

Simulation is the common technique used to assess the behavior of future computing systems, ranging from embedded to *High Performance Computing* (HPC) systems, and including personal devices. Simulators are used to predict not only performance, but also energy consumption, power dissipation, reliability information, packaging costs, etc. However, with the increased complexity of multicore and many-core HPC systems, performing a detailed, cycle-accurate simulation of a next-generation computing system can take several months, negatively affecting time-to-market while also requiring an expensive simulation infrastructure.

The purpose of this paper is to provide a general overview of the recent research in software-based simulation frameworks allowing architects to increase simulation speed while retaining appropriate accuracy as well as providing a complete system model. Next sections discuss a subset of representative proposals regarding: i) accelerating simulation time through the use of the simulation host available parallelism, ii) lossy simulation techniques that trade-off simulation accuracy for speed, iii) advanced trace-driven simulation approaches, and iv) comprehensive simulation frameworks.

A. Parallel simulators

Simulation frameworks aimed at modeling tens of cores or more require from parallelization strategies to speedup simulation time. Parallel simulators often rely on well-known techniques such as *Parallel Discrete Event Simulation* (PDES) [1], although some simulators implement strategies to accelerate the simulation of time consuming components, which often provide shorter simulation times than conventional PDES.

P-GAS [2] is a cycle-accurate parallel implementation of the Godson-T Architecture Simulator (GAS) [3] based on PDES. Other interesting cycle-accurate parallel implementations not based on PDES can be found in [4] and [5]. In [4], a full-system simulator is extended to identify distinct guest execution threads that are mapped to the different host cores taking into account synchronization patterns among the guest threads. In [5], Almer et al. parallelize *Just In Time* (JIT) *Dynamic Binary Translation* (DBT) techniques to leverage the computing power of parallel hosts. Both techniques successfully allow simulating many-core systems exceeding one thousand cores at reasonable speeds. The microarchitecture of the simulated core, however, is simple (e.g., in-order execution, few pipeline stages), which is not appropriate for design spaces covering advanced core microarchitectures or the impact of realistic out-of-order execution on the memory subsystem.

Recently, two novel techniques have been proposed and implemented in the ZSim simulator to make thousand-core cycle-accurate simulation of out-of-order cores practical [6]. The first technique is using DBT techniques to avoid functional modeling and push most of the work usually performed by timing models onto the instrumentation phase, which is performed only once per instruction. The second technique relies on the fact that at small simulation scales (e.g. one thousand cycles), most concurrent accesses target unrelated cache lines, which provides opportunities to avoid synchronization among cores in the host.

On the other hand, parallelization techniques can also be applied to simulate specific subsystems such as the NoC. For instance, in [7], Lis et al. present DARSIM, a cycle-accurate parallel NoC simulator.

B. Lossy simulation techniques

Despite the recent advances that speedup cycle-accurate modeling of many-core systems, modeling a cycle accurate core could be either not necessary or significantly slowdown the simulation time. To deal with this fact, more scalable non-cycle-accurate or lossy alternatives have been proposed. Two main orthogonal approaches can be found in the recent literature: approximate component models and sampling.

1) Approximate component models: Approximate component models sacrifice cycle-accuracy to accelerate the simulation of individual components. Of course, these models can be used for those components whose simulation is most time-consuming and/or require a high amount of computing resources. It is also important to provide models that offer a good trade-off between accuracy and complexity.

An interesting alternative approach is taken by TaskSim [8]. TaskSim presents multiple simulation *modes* to model the execution of parallel applications with different levels of

abstraction. These levels range from a simple *burst* mode based on memory access and synchronization traces to a detailed *instr* mode allowing cycle accurate core modeling, including options to precisely model the cache hierarchy.

One of the most well-known simulation frameworks that makes a comprehensive use of approximate models is Graphite [9]. In Graphite, approximate models are not only used for the core, but also for the uncore. For example, Graphite allows memory ordering violations to occur provided that this strategy does not incur in too high clock differences (i.e., too much slack) between distinct tiles of the modeled system. When the slack grows beyond to a few thousand cycles the threads that run ahead are suspended and tightly synchronized with PDES until the target slack is achieved. Although this strategy mitigates scalability problems of cycle-accurate PDES, contention modeling accuracy suffers because events at contention points are reordered.

Sniper [10] is a fork of Graphite that, among other extensions, provides more accurate core models. In particular, Sniper includes the *interval simulation* model [11]. The interval simulation model allows ROB stalls due to misses and mispredictions to be modeled without the complexity of cycle-accurate core pipeline simulation. Moreover, recently, the *Instruction-Window* (IW) -centric model [12] has been also added to Sniper. The IW-centric model improves the accuracy of interval simulation by modeling structural hazards inside the core and providing a more realistic interaction between the cores and the memory subsystem. This increase of accuracy is performed with a marginal increase in complexity with respect to interval simulation.

2) Sampling: Using well-known sampling techniques, architects simulate only a small percentage of a long-running workload and use the obtained simulation results to predict the performance of the complete execution. Sampling can be statistical [13], [14] or based on application phase detection [15].

Recent research on sampling show that for multithreaded applications, IPC and other metrics based on instruction counts cannot be used to predict runtime due to synchronization events and uncore contention. Thus, time-based sampling techniques become necessary [16]. In addition, application synchronization timing effects must be taken into account during fast forwarding phases between sampling periods. Otherwise, the accuracy of execution time estimates provided by sampling decreases [17].

C. Trace-driven simulation

Other option to deal with simulation complexity is the use of trace-driven methodologies. To obtain a realistic trace-driven simulation of multithreaded applications, however, a solution that captures timing-dependent thread execution interleaving is required. Moreover, capturing this timing information should be done in a way that allows the same trace to be used for simulating variants of the target architecture. In [18], Rico et al. propose a methodology for trace-driven simulation that augments classical traces with information about *parallelism-management* operations (parops) (e.g., synchronization operations). The extended trace drives a simulation tool that takes into account the modeled architecture and simulates parops with an execution-driven engine. In this way, authors combine the best from both trace and execution driven methodologies.

Other recent trace-driven approaches perform simulations in two phases. During the first phase, a fast functional simulation is performed to get the events that will be relevant to timing. Then, during the second phase the timing caused by these events is properly accounted. This technique can be applied to speedup simulation of complex out-of-order cores [19] as well as multicores [20].

D. Putting it all together: simulation frameworks

Current heterogeneous systems consist of multiple components whose simulation requires different levels of detail depending on the purpose of the design process. Thus, complex simulation frameworks [21], [22] have arisen trying to ease the replacement of component models. Nevertheless, when a simulation framework is not complete enough, it often becomes necessary to extend it in ways that were not expected to add new components (e.g., NoC, DRAM) and/or metrics (e.g, power dissipation, temperature, etc.). Thus, for researchers an important requirement is the capability to access and modify the source code of the framework.

E. Conclusion

Simulation of many-core HPC systems is nowadays an active and fruitful area of research. Recent and future proposals are driven by the need of a fast, efficient, and comprehensive simulation framework. This simulation framework should be complete in several ways. First, it should model a wide range of components and provide the mechanisms necessary to plug-in more components as needed. Second, it should allow the designer to focus on critical components while avoiding a large part of the simulation complexity. Each of these components should be able to be evaluated with multiple models with distinct detail levels, ranging from simply analytical models to detailed cycle-accurate simulations. Third, a complete simulation framework should provide a wide range of metrics of interest for the designer and the market. Finally, support for heterogeneous architectures combining CPU and GPU, as well as some degree of reconfigurability is surely required. Building such titanic framework is and will be a collaborative process between researchers around the globe and it is expected to be a hot research topic for the next years.

***Keywords*—HPC systems; simulation frameworks; parallel simulators; approximate models; sampling; trace-driven simulation**

ACKNOWLEDGMENT

This work has been supported by the Spanish *Ministerio de Economía y Competitividad* (MINECO), by FEDER funds through Grant TIN2012-38341-C04-01, and by the Intel Early Career Faculty Honor Program Award.

BIOGRAPHY

SALVADOR PETIT received the PhD degree in Computer Engineering from the Universitat Politècnica de València (UPV), Spain. Since 2009, he is an associate professor in the Computer Engineering Department at the UPV, where he has taught several courses on computer organization. He has published more than 100 refereed conference and journal papers. His research topics include multithreaded and multicore processors, memory hierarchy design, simulation techniques, task scheduling, as well as real-time systems. Prof. Petit is a member of the IEEE Computer Society.

REFERENCES

[1] R. M. Fujimoto, "Parallel discrete event simulation," *Communications of the ACM*, vol. 33, no. 10, pp. 30–53, 1990.

[2] H. Lv, Y. Cheng, L. Bai, M. Chen, D. Fan, and N. Sun, "P-gas: Parallelizing a cycle-accurate event-driven many-core processor simulator using parallel discrete event simulation," in *Principles of Advanced and Distributed Simulation (PADS), 2010 IEEE Workshop on*, May 2010, pp. 1–8.

[3] D. Fan, H. Zhang, D. Wang, X. Ye, F. Song, J. Zhang, and L. Fan, "High-efficient architecture of godson-t many-core processor," in *Proceedings of Hot Chips*, vol. 23, 2011.

[4] M. Monchiero, J. H. Ahn, A. Falcón, D. Ortega, and P. Faraboschi, "How to simulate 1000 cores," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 10–19, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1577129.1577133

[5] O. Almer, I. Bohm, T. von Koch, B. Franke, S. Kyle, V. Seeker, C. Thompson, and N. Topham, "Scalable multi-core simulation using parallel dynamic binary translation," in *Embedded Computer Systems (SAMOS), 2011 International Conference on*, July 2011, pp. 190–199.

[6] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 475–486. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485963

[7] M. Lis, K. S. Shim, M. H. Cho, P. Ren, O. Khan, and S. Devadas, "Darsim: a parallel cycle-level noc simulator," in *MoBS 2010-Sixth Annual Workshop on Modeling, Benchmarking and Simulation*, 2010.

[8] A. Rico, F. Cabarcas, C. Villavieja, M. Pavlovic, A. Vega, Y. Etsion, A. Ramirez, and M. Valero, "On the simulation of large-scale architectures using multiple application abstraction levels," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, p. 36, 2012.

[9] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*. IEEE, 2010, pp. 1–12.

[10] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 52.

[11] D. Genbrugge, S. Eyerman, and L. Eeckhout, "Interval simulation: Raising the level of abstraction in architectural simulation," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*. IEEE, 2010, pp. 1–12.

[12] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 11, no. 3, p. 28, 2014.

[13] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*. IEEE, 2003, pp. 84–95.

[14] 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 Special Issue on Computer Architecture Simulation and Modeling*, vol. 26, no. PARSA-ARTICLE-2007-001, pp. 19–31, 2006.

[15] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," *ACM SIGOPS Operating Systems Review*, vol. 36, no. 5, pp. 45–57, 2002.

[16] E. K. Ardestani and J. Renau, "Esesc: A fast multicore simulator using time-based sampling," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 448–459.

[17] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sampled simulation of multi-threaded applications," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 2–12.

[18] A. Rico, A. Duran, F. Cabarcas, Y. Etsion, A. Ramirez, and M. Valero, "Trace-driven simulation of multithreaded applications," in *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 87–96.

[19] K. Lee and S. Cho, "In-n-out: reproducing out-of-order superscalar processor behavior from reduced in-order traces," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*. IEEE, 2011, pp. 126–135.

[20] H. Lee, L. Jin, K. Lee, S. Demetriades, M. Moeng, and S. Cho, "Two-phase trace-driven simulation (tpts): a fast multicore processor architecture simulation approach," *Software: Practice and Experience*, vol. 40, no. 3, pp. 239–258, 2010.

[21] J. Wang, J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao *et al.*, "Manifold: A parallel simulation framework for multicore systems," in *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 106–115.

[22] E. Argollo, A. Falcón, P. Faraboschi, M. Monchiero, and D. Ortega, "Cotson: infrastructure for full system simulation," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 52–61, 2009.