Research Article Open Access

A Macroscale Simulator for Exascale Software/Hardware Co-Design

Damian Dechev[1,2] **and Gilbert Hendry**[2]

[1]Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL, USA
[2]Sandia National Laboratories, Livermore, CA, USA

Abstract

The next decade will see a rapid evolution of HPC node architectures as power and cooling constraints are limiting increases in microprocessor clock speeds and constraining data movement. Future and current HPC applications will have to change and adapt as node architectures evolve. The application of advanced exascale architecture simulators will play a crucial role for the design and optimization of future data intensive applications. In this paper, we present our imulation-based framework for analyzing the scalability and performance of massive interconnected networks.

Keywords: Exascale architecture simulator; Performance and scalability modeling; Software/Hardware co-design

Introduction

Developers of HPC software must navigate a challenging space of trade-offs with unforeseen effects on delivered application performance. A major challenge is understanding the impact of various design decisions before hardware, or even fully functional software, is available to evaluate the design. In this work we resent the Structured Simulation Toolkit's macroscale simulation components (SST/-macro) [1], our platform for studying and analyzing HPC application performance at scale. SST/macro helps multiprocessor programming in the following ways:

- Enhancing the effectiveness of architecture-specific optimizations by avoiding the hazards of communication and synchronization bottlenecks
- Adapting the applied programming model and motifs to better facilitate the computational and communication patterns of the application
- Evaluating the effectiveness and scalability of user applications before deploying them on advanced hardware platforms including heterogenous and exascale architectures and those based on chips with no builtin hardware cache coherence
- Understanding performance problems of multiprocessor programs with detailed information about the synchronization calls that affect the application's scalability and efficiency
- Validating the runtime behavior of multi-processor programs by matching the simulation data, the dynamic profiling and testing data, and the application's specifications.

Event-Driven Macroscale Simulation

Figure 1 provides an overview of the design of the SST/-macro simulator. The simulator makes use of lightweight application threads, allowing it to maintain simultaneous task counts ranging into the millions. SST/macro supports two execution modes: trace-driven simulation mode and skeleton model-driven execution.

Trace-driven simulation

In the trace-driven simulation execution, an application is executed and profiled in order to extract a wealth of information about its execution pattern. The trace-driven simulation can provide our infrastructure with the following information: average instruction mix, memory access patterns, communication mechanisms and bottlenecks, and the network utilization on a per-link basis. SST/macro supports the following two trace file formats, both of which record execution information by linking the target application with a library that uses the PMPI [2] interface to intercept MPI calls.

- **Open Trace Format (OTF):** OTF is a trace format designed for use with large-scale parallel platforms. OTF has three main targets: openness, flexibility, and performance [3].
- **DUMPI:** which we designed as a custom trace format distributed as a part of the SST/macro simulator. DUMPI's goal is to record more detailed information compared to OTF, including the full signature of all MPI-1 and MPI-2 calls [4,5]. In addition, DUMPI trace files store information regarding return values of MPI requests, which allows error checking and MPI operation matching. DUMPI files also provide hardware performance counter information using the Performance Application Programming interface (PAPI) [4], which allows information such as cache misses and floating point operations to be logged.

The main advantage of trace file driven simulation is accuracy, especially if the planned runtime system is known in detail. A main difficulty is the fact that it requires the execution of the actual application that could often be computationally intensive and have a long run time. Moreover, trace file simulation is not capable of

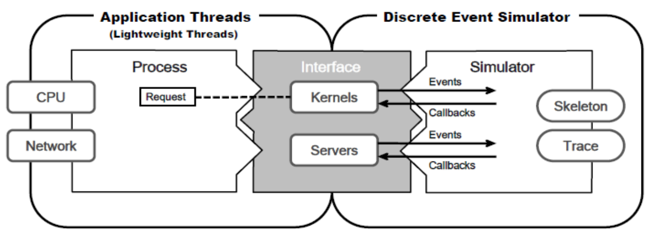

Figure 1: SST/macro Simulation Framework.

***Corresponding author:** Damian Dechev, Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL, USA, E-mail: dechev@eecs.ucf.edu

Received August 10, 2013; **Accepted** November 18, 2013; **Published** November 23, 2013

Citation: Dechev D, Hendry G (2013) A Macroscale Simulator for Exascale Software/Hardware Co-Design. J Inform Tech Softw Eng 3: 123. doi:10.4172/2165-7866.1000123

predicting performance on future hardware platforms, as the generated trace files are specific to some features of the execution environment.

Skeleton model-driven simulation

While trace driven simulation can run applications on the largest available machines and then analyze the collected traces, skeleton model-driven simulation is required to scale simulations and interpolate application behavior on future exascale machines to be designed. Here the simulation is driven by skeleton applications, which are simplified models of actual HPC programs with enough communication and computation information to mimic the application's behavior. One method of constructing a skeleton model is to manually replace portions of the code performing computations with system calls that instruct the simulator to account for the time implicitly. Since the performance models can be embedded in the skeleton application and intensive calculations are not performed, the simulator requires significantly less computational time than simulating the entire application. Skeleton application simulation can also evaluate efficiency and scalability at extremely different scales, which provides a powerful option for performance prediction of non-existing super-scalar systems.

Communication models

The purpose of SST/macro's communication component is to study the complex interaction of the various software components and the network. Recent growth of large-scale systems has made evaluation of communication loads across complex networks vital. SST/macro is capable of simulating and evaluating advanced network workload with diverse topology and routing. A simple processor model is added to provide timings for processor workload and data movement within each node. The simulator currently supports torus, fat-free, hypercube, Clos, and gamma topologies [6]. Moreover, general network frameworks can be evaluated with network parameters such as bandwidth and latency, and the modularity of the simulator makes defining new topologies and routing protocols easy. These components enable us to investigate interconnect design options such as choice and tuning of topologies (high-dimensional meshes, fit-trees as opposed to fat-trees); routing algorithms (wormhole vs dispersive routing, oblivious vs. adaptive routing); and system parameter choices (e.g, router buffer sizes, bandwidths and latencies). At the same time, we will be able to quantify the benefits of new algorithms and algorithmic paradigms such as alternatives to the infamous bulk-synchronous parallelism. In addition, we will study how to decompose the main problem into subproblems and how to map tasks to the processors.

Programming models

SST/macro is designed to support a variety of programming and synchronization models. The most recent release of SST/macro [1] provides full support for Pthreads and the Message Passing Interface (MPI) [7]. MPI is the most common message passing library interface specification for a distributed memory system and is widely applied in a large number of scientific codes. In SST/macro, lightweight application threads perform MPI operations (Figure 1). The simulator implements a complete MPI which skeleton applications can use to emulate node communication in a straightforward manner. SST/macro has been used to test the performance impact of proposed extensions to the MPI standard [8,4] and optimizations to mpiBLAST [9,10].

SST/macro Simulation Accuracy: We provide a brief overview of our simulator validation study [4]. Figure 2a shows the result of our validation studies, demonstrating that SST/macro's predictions are always within 10% of the observed runtimes. Figure 2b shows the runtime performance of our simulator, demonstrating that the simulator can easily simulate up to millions of processors, with its performance bounded by cache size. In this study we have executed a trivial MPI Ping-Pong test [10]. The MPI Ping- Pong is a communication test between two hosts used to determine whether a particular host is reachable across an IP network. Figure 2c shows how the simulator can be used to understand how architectural features impact application performance. A detailed discussion of these results is available in [4,5].

Applications

Exploring HPC Codes: Here we briefly mention three representative HPC applications to demonstrate our performance modeling and simulation process [5] and also outline a range of important application requirements that will motivate future system designs. Our target applications include: the Gyrokinetic Toroidal Code (GTC) [11,12], an application developed for fusion simulations, including efficient ITER designs; the Global Cloud Resolving Model (GCRM) [13], designed for climate simulations at unprecedented resolutions; and the Materials Science LS3DF [14] computation, which is applied to numerous nanoscience simulations including next-generation solar cell design. Full applications, especially those that are capable of testing the system at very high concurrency and scale, have complicated interactions and access patterns that are difficult to predict and understand without the concrete evidence of a full application. Each application mentioned here employs a different mathematical foundation for the solution of the underlying physical process.

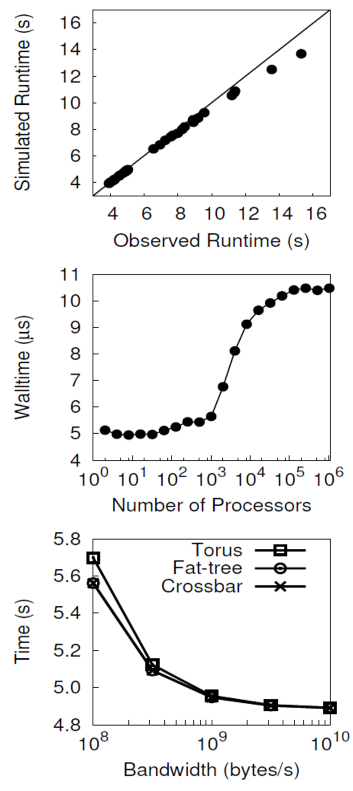

Figure 2: SST/macro simulator results showing (a) validation data with simulated runtime plotted against observed runtime, (b) real time required to simulate an MPI ping pong round trip, and (c) use of the simulator to study machine topology and bandwidth affects on application runtime.

Compact applications

The Mantevo project [15] is an effort to provide open source software packages for the analysis, prediction and improvement of high performance computing applications. These packages consist of mini-applications, mini-drivers, and application proxies. The mini-applications are small, self-contained programs that embody essential performance characteristics of certain key applications. The mini-drivers are small programs that utilize performance impacting Trilinos [16] packages. Application proxies are parameterizable programs that can be calibrated to mimic the performance of a large-scale application. The application space covered by the Mantevo software includes implicit unstructured PDE applications, explicit dynamics (contact), molecular dynamics, and circuit simulation. These applications provide wide coverage in the HPC software design space. In this project we rely on Mantevo to complement to the full applications and the computational motifs. Because the Mantevo software packages are relatively small compared to a full application (ranging in size from about 1000 to 10000 lines of code), it is simple to port the code to our SST/macro-based simulation toolchain and experiment with code alterations [5]. Computational Motifs and Auto-Tuned Kernels: In addition to full and compact application codes, we are also considering computational motifs that capture computation and communication patterns for a broad range of scientific methods. For these motifs we leverage the work on succinct set of well-defined algorithms and numerical methods [17] at a high level of abstraction. These benchmark codes serve to broaden the space of architectural drivers to include virtually all of the important access patterns and communication topologies encountered in problems of interest to high end computing.

Conclusions and Future Work

The application of hardware/software co-design has been a feature of embedded system designs for a long time. So far, hardware/software co-design techniques have found little application in the field of high-performance computing. The multi-core paradigm shift has left both software engineers and computer architects with a lot of challenging dilemmas. The application of hardware/software co-design for HPC systems will allow for a bi-directional optimization of design parameters where software specifications and behavior drive hardware design decisions and hardware constraints are better understood and accounted for in the implementation of effective application software. The use of discrete event simulation tools provides the data and insights to estimate the performance impact on an HPC applications when it is subjected to certain architectural constraints. In this work we discussed the design of a newly developed open-source macroscale simulator (SST/macro). SST/macro provides the simulation abilities which can play a crucial role for the effective design and implementation of large-scale data intensive applications to be executed on the future multicore hardware platforms. Such platforms could include a wide variety of features including a heterogenous design of CPUs, GPUs, and even FPGAs. In our future work we plan to implement components for supporting additional programming styles such as the partitioned global address space (PGAS) programming model [18], and non-blocking synchronization [19]. Such extensions will address the needs of applications and algorithms that increasingly rely on finegrained parallelism such as lock-free synchronization [19,20] and strong scaling while supporting fault resilience [21] to accommodate the massive growth of explicit on-chip parallelism and constrained bandwidth anticipated of future chip architectures.

References

1. Sandia Nationional Labs, 2011.
2. Mintchev S, Getov V (1997) PMPI: High-Level Message Passing in Fortran 77 and C. High-Performance Computing and Networking 1225: 601-614.
3. Kn¨upfer A, Brendel R, Brunst H, Mix H, Nagel W (2006) Introducing the Open Trace Format (OTF). Lecture Notes in Computer Science 3992: 526-533.
4. Janssen C, Adalsteinsson H, Cranford S, Kenny J, Pinar A, et al. (2010) A Simulator for Large-Scale Parallel Computer Architectures. Inter Jour of Distributed Systems and Technologies 1: 57-73.
5. Janssen CL, Adalsteinsson H, Cranford S, Dechev D, Kenny JP, et al. (2010) Exascale Co-design with Sandia's Structural Simulation Toolkit (SST). Proceedings of 1st International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computing Systems (PMBS 2010), Supercomputing (SC 2010).
6. Dally W, Towles B (2003) Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc San Francisco, CA, USA.
7. MPI (2009) MPI (Message Passing Interface) standards documents, errata, and archives of the MPI Forum.
8. Dakshinamurthy A, Dechev D (2011) SRC: Automatic Extraction of SST/macro Skeleton Models. Proceedings of the 25th ACM International Conference on Supercomputing.
9. Altschul S, Gish W, Miller W, Myers E, Lipman D (1990) Basic local alignment search tool. Journal of Molecular Biology 215: 403-410.
10. Ahn TH, Dechev D, Lin H, Adalsteinsson H, Janssen C (2011) Evaluating Performance Optimizations of Large- Scale Genomic Sequence Search Applications Using SST/- macro. Proceedings of the 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications.
11. Lin Z, Hahm T, Lee W, Tang W, White R (1998) Turbulent transport reduction by zonal flows: Massively parallel simulations. Science 281: 1835-1837.
12. Ethier S, Tang W, Lin Z (2005) Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms. Journal of Physics: Conference Series 16: 1-15.
13. Randall DA, Ringler TD, Heikes RP, Jones P, Baumgardner J (2002) Climate Modeling with Spherical Geodesic grids. Computing in Science & Engineering.
14. Wang W, Lee B, Shan H, Zhao Z, Meza J, et al. (2009) Linear scaling 3D fragment method for large scale electronic structure calculations. Proc. ACM/IEEE Conf. on Supercomputing (SC), Portland, OR, USA.
15. Sandia Nationional Labs. Mantevo, 2011.
16. Sandia Nationional Labs. Trilinos, 2011.
17. Asanovic K, Bodik R, Catanzaro BC, Gebis JJ, Husbands P, et al. (2006) The Landscape of Parallel Computing Research: A View from Berkeley. EECS Department, University of California, USA.
18. Pirkelbauer P, Liao C, Panas T, Quinlan D (2011) Runtime Detection of C-Style Errors in UPC Code. Proceedings of the Fifth Conference on Partitioned Global Address Space Programming Models.
19. Dechev D, Stroustrup B (2009) Scalable Nonblocking Concurrent Objects for Mission Critical Code. Proceedings of the ACM SIGPLAN conference on Objectoriented programing, systems, languages, and applications.
20. Dechev D, Pirkelbauer P, Stroustrup B (2006) Lock-Free Dynamically Resizable Arrays. Lecture Notes in Computer Science 4305: 142-156.
21. Minnich RG, Janssen CL, Krishnamoorthy S, Marquez A, Gokhale M, et al (2011) Fault oblivious exascale whitepaper. Proceedings of the 1st International Workshop on Runtime and Operating Systems for Supercomputers, ROSS '11, New York, NY, USA.