# A Multi-resolution Emulation + Simulation Methodology

Laxmikant V. Kale and Nikhil Jain‡
Akhil Langer, Esteban Meneses, Phil Miller, Osman Sarood and Ehsan Totoni
Department of Computer Science, University of Illinois at Urbana-Champaign
‡{kale,nikhil}@illinois.edu

## 1. INTRODUCTION

As we design exascale applications and machines, it becomes important to be able to analyze and experiment with alternate designs of both machines and applications. These experiments have to be done before the machines are built since it will be too expensive to build a large number of alternate designs. One of the challenges in this process is how to represent application behavior in such machines. For analyzing network performance via simulations, for example, one can use pre-designed injection patterns, but they do not capture the feedback that occurs naturally in applications: if an incoming message is late, the ordering of events may change, and outgoing message injection will also change. To achieve a high fidelity simulation is therefore challenging.

One method that has shown promise is that of *emulation-followed-by-simulation*: one carries out a full-scale emulation of the application with the correct number of nodes and control threads, facilitated by some overdecomposition based system such as Charm++ [1], FG-MPI[2], or AMPI [3]. The emulation captures dependencies between sequential computations and remote data in traces. A helpful point here is the relative constancy provided by application steps, which makes the emulation feasible. In biomolecular simulations, for example, application behavior evolves slowly and is sufficiently self-similar throughout execution; In astronomy simulations or in rocket simulations, or SAMR-based applications, for example, the simulation changes over time, but one can sample timesteps in different phases of the simulation, by sampling from coarse-grained simulations. The emulation traces therefore provide a powerful method for capturing application behavior, especially when performance counter information is captured during emulations characterizing sequential execution blocks of the underlying application.

The traces generated by emulation can then be fed to a multi-component simulator, where *a variable resolution simulation* can be carried out to predict performance and other attributes. We advocate this methodology and elaborate on research challenges involved in following it in exascale design. At exascale, we expect the components, which are pluggable entities similar to those used in existing frameworks such as BigSim [4, 5], SST [6], to simulate network, resilience support, power management, thermal constraints, operating system and file system. In addition, the adaptive runtime system, essential for scalable execution at exascale, needs to be (and can be) simulated in detail, with realistic code and strategies, in order to attain high fidelity.

### 1.1 Scalable Backend

In order to simulate at scale, development of customized *scalable parallel discrete event simulators* for HPC use will be needed. Bauer Jr. et al [7] have shown good scalability on modern day supercomputers using optimistic PDES. Further evolution of such software with use of efficient low level libraries such as PAMI, GNI, VERBS etc will need to be done. More importantly, co-design of such simulators and adaptive runtime systems, resulting in software stack as shown in Figure 1, will be fruitful. The runtime system will not only provide key backend support such as load balancing, data management and communication to the PDES, it will also be useful in making intelligent decisions during simulation as a pluggable module. For example, using relatively simple modeling, the runtime module can categorize events based on their importance, and provide feedback to the PDES on the level of detail a event should be simulated at.

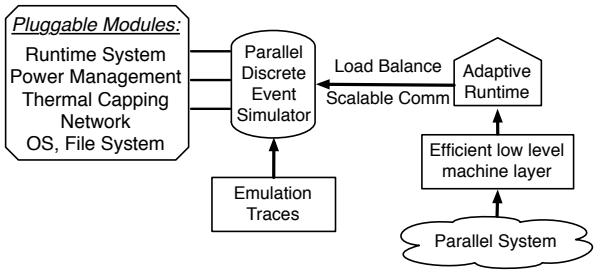


**Figure 1: Software stack for scalable simulation of performance and non-performance application characteristics.**

In the following section, we will discuss ideas for designing modules for simulating various performance and non-performance aspects of an application execution.

## 2. MODULES

**Network:** Simulation of network behavior for communication patterns exhibited by applications poses a challenging task as we move towards exascale. In addition to the increased size of the networks, the problem has been made more difficult by modern day applications whose communication patterns are dynamic and complex. As such, synergistic development and use of heuristics will be important.

Firstly, methodologies adopted to predict network behavior will need to be evolved. The prospect of raising granularity without loss of accuracy is worth considering given the significnt success achieved by replacing flit-level simulation by packet-level simulation [8, 5]. Further raising the granularity to k-packets may be useful in reducing the cost without significant loss of accuracy. A better understanding of modern networks with focus on identifying components

that are critical to determining network behavior will also be required. Currently, almost all the network components from buffers to switches to channels are simulated in detail. It may be feasible to increase the abstraction level for some components, omit some, and focus on the rest, and yet perform accurate predictions. Finally, faster simulations for phases of applications without contention can be performed by omitting complex network simulation.

**Runtime System:** System design, operating conditions, and application behavior will present dynamic variation in the exascale environment. This dynamic variation drives the need for dynamic runtime adaptation. Simulators for this environment must explicitly incorporate models of runtime system (RTS) behavior for two reasons. First, new applications are being written in a variety of paradigms beyond message-passing SPMD, such as work stealing, message-driven execution and migratable tasks. Therefore, a simulator must be usable to study performance under any of these models. Second, system-level responses to failures and power constraints will often be mediated through RTS mechanisms.

A key mechanism of runtime adaptation is the reassignment of some work or data from one processor to another. A prime example of this is load balancing, for which many strategies exist. When reassignment occurs, the simulator must adjust subsequent events connected to the remapped unit accordingly. Current systems only handle this through offline trace reprocessing, which limits studies. Instead, simulators can be built to directly model the runtime migration mechanisms with varying degrees of resolution (through oracular message redirection, explicit communication, or other means), possibly even by incorporating the RTS code itself.

A dynamic RTS presents many implementation choices, and simulations can help determine suitable decisions for various systems and applications. For instance, the work of a given processing element may be performed in many possible orders. The RTS could provide simple FIFO queueing, prioritization, or even online critical path detection. RTS simulation modules should be able to provide any such policies, with the simulator accounting for consequent variations in execution order (and thus communication timing, task dependencies, etc.). Finally, the RTS module may be useful to guide the simulator on deciding the resolution at which an event should be simulated.

**Resilience:** Resilience is one of the major challenges the HPC community will face at exascale [9, 10, 11]. It is expected that machines at that scale will have a failure every tens of minutes [9, 12, 13]. In addition, resilience's interplay with factors such as power management and performance considerations will also play a major role. Therefore, simulation of resilience at extreme scale requires to go beyond a simple model that only considers mean-time-between-failures (MTBF).

A more accurate simulation for resilience should consider the following aspects: 1) communication patterns and dependencies of the application, 2) characteristics of the machine's hardware, 3) different fault patterns (varying time between failures and the number of nodes failing), 4) various root causes of faults in hardware (e.g. nodes, network), software, environment etc. and 5) different types of failures: hard failures and soft failures. Moreover, this simulation task is particularly challenging, since one may need to simulate hours of application execution to be able to study the steady state of a fault tolerance protocol. In addition, millions of nodes and billions of threads need to be considered for exascale. Existing simulators can mostly simulate an HPC application for a few seconds by using a multi resolution approach. For instance, a simple model or a constant for execution time of local sequential functions of the application is used. This abstraction level needs to be raised even further using different techniques, to be able to scale from seconds to hours and days of simulation.

**Power**: Exascale systems will be power-constrained: one must ensure that the total power draw does not exceed a specified data-center maximum. Conservative enforcement of this constraint will create unacceptable performance loss. So, the system hardware and the runtime are expected to support dynamic management. To be able to do prediction of power and performance together, one needs to build accurate models of power draw of different components. The impact of power capping these subsystems on the performance of an application is difficult to predict [14].

Breaking the computation down into sequential execution blocks (SEBs) is helpful; e.g. an empirical model can be developed for predicting CPU and memory power while executing a specific SEB. In addition, the runtime response to power consumption must also be simulated, making it a mutually recursive endeavor. This will include a global power management system, which will specify frequencies or power-levels for each component, and the simulation can proceed with execution times predicted by the models. Interconnect power, which is usually constant today, can also be studied/simulated in a similar manner, with the RTS turning links off, or running them at lower speeds as needed. The research agenda here includes developing sub-scale simulations to help build fast models, and to incorporate prediction components of power/performance relationship for SEBs.

**Thermal:** Core temperature is a factor that can impact the performance, reliability and power consumption of a processor [15]. Processor temperature can depend on multiple factors like cooling environment, semiconductor process variations, and application characteristics. Preliminary work [16] has shown that it is possible, via adaptive runtime strategies, to control core temperatures tightly while mitigating resultant load imbalances. An exascale simulation must predict and incorporate the evolution of core temperature in response to application code execution, and runtime system decisions, vice versa. However, in order to do that, we need accurate and reliable models that can predict core temperatures depending on application characteristics and cooling environment. Past work highlights the difficulty here as core temperature behave differently for different applications [16]. However, most earlier experimental work is carried out on machines that are significantly smaller than an exascale machine. To study the impact of controlling core temperatures on larger machines requires significantly mature models on which simulations can be based. Such simulators will need to incorporate thermal models that take into account application characteristics, cooling infrastructure, and processor thermal design.

## 3. ESTIMATED EFFORT

We estimate that an effort with 5 FTEs and 5 graduate research assistants, over a period of three years, and spread over multiple institutions including DOE labs, academia, and possibly separately funded industry participation will be needed to carry out the proposed research program. This will need to be followed by a consolidation of the prototype into usable and supported product.

# 4. REFERENCES


[1] Laxmikant Kale, Anshu Arya, Abhinav Bhatele, Abhishek Gupta, Nikhil Jain, Pritish Jetley, Jonathan Lifflander, Phil Miller, Yanhua Sun, Ramprasad Venkataraman, Lukasz Wesolowski, and Gengbin Zheng. Charm++ for productivity and performance: A submission to the 2011 HPC class II challenge. Technical Report 11-49, Parallel Programming Laboratory, November 2011.

[2] H. Kamal and A. Wagner. FG-MPI: Fine-grain MPI for multicore and clusters. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010*, pages 1 –8, april 2010.

[3] Orion Lawlor, Milind Bhandarkar, and Laxmikant V. Kalé. Adaptive mpi. Technical Report 02-05, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2002.

[4] Gengbin Zheng, Gunavardhan Kakulapati, and Laxmikant V. Kalé. Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In *18th International Parallel and Distributed Processing Symposium (IPDPS)*, page 78, Santa Fe, New Mexico, April 2004.

[5] Ehsan Totoni, Abhinav Bhatele, Eric Bohm, Nikhil Jain, Celso Mendes, Ryan Mokos, Gengbin Zheng, and Laxmikant Kale. Simulation-based performance analysis and tuning for a two-level directly connected system. In *Proceedings of the 17th IEEE International Conference on Parallel and Distributed Systems*, December 2011.

[6] K.D. Underwood, M. Levenhagen, and A. Rodrigues. Simulating red storm: Challenges and successes in building a system simulation. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 –10, 2007.

[7] David W. Bauer Jr., Christopher D. Carothers, and Akintayo Holder. Scalable time warp on blue gene supercomputers. In *Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, PADS '09, pages 35–44, Washington, DC, USA, 2009. IEEE Computer Society.

[8] Abhinav Bhatele, Nikhil Jain, William D. Gropp, and Laxmikant V. Kale. Avoiding hot-spots on two-level direct networks. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 76:1–76:11, New York, NY, USA, 2011. ACM.

[9] Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems, 2008.

[10] Marc Snir, William Gropp, and Peter Kogge. Exascale Research: Preparing for the Post Moore Era. `https://www.ideals.illinois.edu/bitstream/handle/2142/25468/Exascale%20Research.pdf`, 2011.

[11] Franck Cappello. Fault tolerance in petascale/ exascale systems: Current knowledge, challenges and research opportunities. *IJHPCA*, 23(3):212–226, 2009.

[12] Franck Cappello, Al Geist, Bill Gropp, Laxmikant Kale, Bill Kramer, and Marc Snir. Toward exascale resilience. *Int. J. High Perform. Comput. Appl.*, 23(4):374–388, November 2009.

[13] Kurt Ferreira, Jon Stearley, James H. Laros, III, Ron Oldfield, Kevin Pedretti, Ron Brightwell, Rolf Riesen, Patrick G. Bridges, and Dorian Arnold. Evaluating the viability of process replication reliability for exascale systems. In *Supercomputing*, pages 44:1–44:12, New York, NY, USA, 2011. ACM.

[14] Tapasya Patki, David K. Lowenthal, Barry Rountree, Martin Schulz, and Bronis R. de Supinski. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th international ACM conference on International conference on supercomputing*, ICS '13, pages 173–182, New York, NY, USA, 2013. ACM.

[15] Franciscoj. Mesa martÃ■nez Ehsan, K. Ardestani, and Jose Renau. Characterizing processor thermal behavior.

[16] Osman Sarood Phil Miller Ehsan Totoni and Laxmikant Kale. 'cool' load balancing for hpc data centers. *IEEE transactions on computers special issue on energy efficient computing*, 2012.