Toward Integrated Multi-Resolution HPC Modeling for Rapid Performance Prediction (An Abstract)

Jason Liu
Florida International University

Stephan Eidenbenz
Los Alamos National Laboratory

High-performance computing architectures are changing constantly. Since the *de facto* death of Amdahl's Law (circa 2005), the field nevertheless has seen tremendous growth thanks to innovations, such as multi-core/many-core processors, specialized co-processing units (like GPU), and advanced memory/caching strategies. The constant flux in HPC architectures induces frequent changes to parallel applications.

To maintain the core capability in advanced computational physics, DOE has devoted significant resources in adapting its software to incorporate new system features for superior performance—it is said: *"no code shall be left behind."* Code adaptation, however, can be difficult as it may involve intricate changes to algorithms, and require new code analyses, programming structures, data layouts, and parallel strategies. The task usually requires good insight to sophisticated code from highly skilled software architects and domain scientists.

Modeling and simulation plays a significant role, in identifying potential performance issues, evaluating design alternatives, performing parameter tuning, and answering what-if questions. A large body of literature exists in HPC modeling and simulation, ranging from coarse-level full-scale models, to cycle-accurate simulations of individual components (processors, cache/memory, and interconnect), to analytical approaches. We note that none of the existing methods is capable of modeling full-scale HPC architectures and applications at finest granularity. It is both unrealistic and unnecessary.

Today's supercomputers are rapidly approaching exascale. That is, the processing speed gets to 10^{18} floating-point operations per second. We study the performance of parallel applications (in particular, computational physics code) on existing and future HPC systems. A cycle-accurate simulation may render good fidelity for a specific component (say, a multi-core processor) at a small time scale. Such models cannot be extended to deal with arbitrarily large systems or long time durations. Partially, this is due to the computational complexity of such models (spatial and temporal).

More importantly, no model would be able to capture entire system dynamics in detail. HPC applications written in specific programming languages interact with other software modules, libraries and operating systems, which in turn interact with underlying resources for processing, data access, and I/O. Any uncertainties involved with the aforementioned hardware and software components (e.g., compiler-specific libraries) can introduce modeling errors, far exceeding the fidelity achieved by cycle-accurate models for a specific component.

George Box, the statistician, once said: *"All models are wrong but some are useful."* In order to scale up to a full-system simulation, modelers are forced to raise the level of modeling abstractions (by reducing modelings details). It is true that, by choosing the "appropriate" modeling abstractions, we can draw useful conclusions by extrapolating results from high-level models. However, choosing appropriate modeling abstractions depends not only on the goal of the study, but also on how the components interact with one another.

We introduce "selective refinement codesign modeling". For codesign modeling, we start with both architecture and application models, possibly with various modeling abstractions. We want to find proper models for various components of the system, enough to answer the research questions. This would be an iterative process, based on the ability to locate performance bottlenecks in both hardware and software. A high-level process is as follows:

1) Start with coarse-level models
2) Run experiment and gather results
3) Identify potential performance bottlenecks
4) Replace components with more refined models
5) Go to step 2 until satisfied

We set out to design and develop a simulator, particularly for rapid assessment and performance prediction of large-scale scientific applications on current and future HPC architectures. There are four requirements. First, the simulator needs to easily integrate large-scale applications (computational physics code) and full-scale architecture models (processors, memory/cache, interconnect, etc.). Second, the simulator needs to combine selected models at different level of modeling abstractions. Third, the simulator needs to have a short development cycle, so that it can keep up with the fast refresh rate of HPC systems. Last, the simulator must be high performance and capable of handling extremely large models.

Our project *"Scalable Codesign Performance Prediction for Computational Physics"* at LANL aims to establish rapid assessment and performance prediction capabilities using the selective refinement codesign modeling method. So far, we have developed a minimalistic process-oriented parallel discrete-event simulator based on just-in-time compilation. The simulator is easy to use (consisted of only approx. 500 lines of code at its core); it has shown good performance, sometimes even outperforming some compiler-based simulators. On top of this simulator, we have developed models for multi-core processors, memory/cache, and interconnection networks (such as torus, dragonfly, fat-tree). We are currently focusing on computational physics applications, including benchmark applications (e.g., PolyBenchSim, ParboilSim) and production applications (e.g., SNAPSim, SPHSim, SpecTADSim).