 SAND2011-9465P

Proposal Title: Toward Reliable and Efficient Exascale Computing

Principal Investigators:

1. Rudolf Eigenmann (Lead PI)
   Purdue University
   765-494-1741
   eigenman@purdue.edu

2. Yung Ryn Choe
   Sandia National Labs
   925-294-6435
   yrchoe@sandia.gov

3. Seyong Lee
   Oak Ridge National Labs
   865-576-3869
   lees2@ornl.gov

4. Vijay S. Pai
   Purdue University
   765-496-6610
   vpai@purdue.edu

5. Olaf O. Storaasli
   Oak Ridge National Labs
   865-574-0494
   olaf@ornl.gov

6. Mithuna Thottethodi
   Purdue University
   765-496-6787
   mithuna@purdue.edu

7. T.N. Vijaykumar
   Purdue University
   765-494-0592
   vijay@purdue.edu

Estimated Cost: $945,000 (annual) $2,835,000 (total)

Toward Reliable and Efficient Exascale Computing

Overview

At a high-level, our proposal addresses foundational advances in checkpointing and reliability at scale, compiler-assisted automatic performance forecasting (which can be used for early forecasting/prediction of performance scaling) and elimination of key scalability bottlenecks in the runtime system. These solutions crosscut compilers, tools, and runtime systems for a comprehensive approach to reliable and efficient Exascale computing.

At large scale and long runtimes, reliability is a critical issue and one that is particularly exacerbated by the large number of possible failing components in an exa-scale system. Fundamentally, checkpointing strategies can be global or local (or some combination of the two). Each strategy has its own advantages and disadvantages. With globally coordinated scheduling, consistent checkpoints are guaranteed; however the co-ordination effort and time scales with system size. Unfortunately, long checkpoint coordination times interact poorly with the fact that large systems have lower mean-time-to-failure (MTTF). In the extreme, one may have a system where forward progress is jeopardized if MTTF is less than checkpoint coordination time. On the other hand, local uncoordinated checkpointing can avoid the latency of coordination; however, it offers weak guarantees of obtaining a consistent checkpoint. This can potentially lead to a domino effect where one is forced to revert to earlier checkpoints which, in turn, may be inconsistent, resulting in cascaded rollbacks and large wasted effort. The key innovation we propose is a fundamentally new approach to checkpointing that obviates the need for consistency in the checkpoints, while still guaranteeing reliable, consistent forward progress. Our technique employs timestamp-based local, uncoordinated checkpointing technique while avoiding the domino effect. The checkpointing will be automatically initiated by the runtime. Such a design offers the best of both worlds – fast, local, independent checkpoints and minimal wasted effort.

Another key challenge in Exascale computing is predicting performance scalability, as certain behaviors and even execution paths may only be observed at large scale. Our approach is to provide a compiler-assisted, automatic methodology for performance forecasting at scale (both in system size and dataset size). Such a tool/methodology will critically enable and automate detection and diagnosis of performance scaling early in the process. Our methodology leverages advanced symbolic program analysis and derives *performance expressions* for computation and communication. Several architecture-dependent parameters of the expressions are derived from profile runs on a given, mid-scale platform and training datasets, as well as from platform microbenchmarks that can be extended to scale. The performance expressions can be used to evaluate the program behavior on large scale system configurations and large datasets. Integrating this framework into the compiler allows the analysis of all possible execution paths, including those not observed in small-scale tests. Consequently, the compiler can provide feedback on likely performance scaling problems within certain execution paths even if those have not arisen during profile executions.

Another important source of performance scalability issues arises in the runtime system, and our contribution consists of novel algorithms that exploit common-case behavior to eliminate the bottlenecks that arise due to excessive conservatism. For example, we propose to address an important scalability bottleneck for MPI applications running on large HPC systems – the challenge of matching messages received from the network at a node to "receive"s posted by the MPI process running on the node. Real MPI benchmarks indicate that message matching is a potential bottleneck. Further, the performance of message matching degrades with system scale. This implies that message matching can become a scaling bottleneck for important, large-scale computations. We propose and explore novel matching algorithms in the runtime that reduce matching time. In addition to the above example, we propose to target other runtime system scaling bottlenecks including (but not limited to) collective communication and load-balance/hot-spot avoidance for both computation and communication.

Our proposal can be considered an integrated and cross-cutting approach to achieving scalability for Exascale computing. The first technique addresses scalability limitations that arise due to unreliable components and the software stack layers that enable reliability through checkpointing. The second technique addresses limitations from the structure of the applications that are being run. The third technique targets scalability bottlenecks in the runtime and how the application exercises the runtime. The combination of these techniques will yield substantial advances in reliability and efficiency as we march toward Exascale computing.