Journal of
Information Technology & Software Engineering

Dechev, J Inform Tech Softw Eng 2013, 3:1
DOI: 10.4172/2165-7866.1000e115

Editorial Open Access

Making Exascale Computing Tractable by Enabling Effective Data Communication and Synchronization

Damian Dechev*

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, USA

Exascale systems, featuring the capability of executing quintillion 10^{18} operations per second, are expected to be deployed in 2018 and will bring significant advancements in a number of scientific fields of immediate global importance (such as medicine, biology, national security, and energy). Exascale platforms will be qualitatively different from current high-performance computing systems. The main driving force for the growth in computational power will be the increase of parallelism on-chip. It is expected that over the next decade the number of nodes will grow by a factor of 10× while on-chip parallelism will grow by a factor of 100×. Adapting software applications for exascale computing will be difficult as the architectural complexity of exascale systems will be extremely high in terms of their degree of concurrency and heterogeneity, sensitivity to communications and data movement, and requirements for locality.

The vision of this work is to allow for the support exascale application development by enabling advanced simulations of inter-node communication patterns and by engineering the tools for fast and effective intranode synchronization and resource sharing. To achieve this goal the we will:

- Design and implement automatic extraction of application skeletons for simulation analysis of internode communication using SST/macro [1]. Performance analysis of an application can be done in several ways with varying degrees of fidelity. One of the most cost-effective ways is to do a coarse-grained study of large-scale parallel applications through the use of program skeletons. The concept of a "skeleton" represents an abstracted program that is derived from a larger program where source code that is determined to be irrelevant is removed. In this work, we extend our prior work on using compiler-based program analysis with ROSE [2] to develop a semi-automatic approach for extracting program skeletons by employing the Program Dependance Graph (PDG) for our analysis.
- Introduce and apply a new methodology for large-scale simulation validation based on the execution's statistical characteristics. Validation is highly important in parallel application simulations with a large number of parameters, a process that can vary depending on the structure of the simulator and the granularity of the models used. Common practice involves calculating the percentage error between the projected and the real execution time of a benchmark program. This coarse-grained approach often suffers from a parameter insensitivity problem in regions of high-dimensional parameter space. We will develop a validation toolset that aims to capture fine-grained execution details. It consists of a trace analysis tool that decomposes execution time into finer granularity, a trace comparison tool that quantizes the disparity between correspondent metrics of two executions, and a visualization tool that renders the analysis and comparison results in graphs. The analysis process will take into account five groups of statistical data profiled from program traces: overall traffic and timing, per-node traffic and timing, MPI function histogram, collective synchronization and node-to-node communication.
- Develop the theory and practice for the implementation of data structures optimized for intra-node synchronization. We will invest in conducting fundamental research on: (a) obtaining deeper under-standing of the principles for design of multiprocessor data structures and (b) delivering the practical implementation, optimization, and tool support for a new generation of C++11 nonblocking data structures. In our preliminary work we demonstrated that our initial design of the first wait-free multiprocessor hash map consistently outperforms the alternative blocking designs by a factor of 5 or more. Our evaluation reveals that such data structures could bring immediate benefit to a variety of applications including programs relying on the common pipeline parallel processing model.
- Design and implement a fast and scalable approach for shared memory intra-node resource allocation. In this problem, processes compete for k shared resources where a process may request an arbitrary number $1 \leq h \leq k$ of resources at the same time. The challenge is for each process to acquire exclusive access to desired resources while preventing deadlock or livelock. Many existing approaches solve this problem in the context of distributed system, but the explicit message passing paradigm they adopt is not optimal for intra-node shared-memory. Other methods, like two-phase locking and resource hierarchy, suffer from performance degradation under heavy contention. We will develop a multi-resource lock based on a nonblocking queue that schedules processes to spin on previous conflicting resource requests. It guarantees the strongest FIFO fairness, which also implies starvation-freedom and deadlock-freedom. Our preliminary evaluation shows that as contention increases, our multi-resource lock obtains an average of 10× speedup with a worst case of 2× speedup over existing approaches.

Novelty and Assessment

This position paper represents a forward-looking and integrated approach that will lead to the discovery of principles for effective inter-node communication simulation and intra-node data synchronization for exascale application development. This work will uniquely combine and further advance the current knowledge and practice in large-scale application simulation and multiprocessor synchronization to enable a transformative parallelism extraction process that can be applied effectively to the complex exascale application space. This research will provide a new methodology for program skeleton generation from large-scale applications through the use of program analysis.

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

Received August 10, 2013; **Accepted** August 14, 2013; **Published** August 17, 2013

Citation: Dechev D (2013) Making Exascale Computing Tractable by Enabling Effective Data Communication and Synchronization. J Inform Tech Softw Eng 3: e115. doi:10.4172/2165-7866.1000e115

We will address the problem of parameter insensitivity in large-scale communication simulations by introducing five fine-grained accuracy evaluation metrics. The most intricate exascale challenges arise in the discovery of parallelization strategies within the node where most of the hardware disruptions will occur. This study will bring new knowledge about the principles for shared memory data structures design such as a novel wait-free software Multiple Compare-and-Swap (MCAS) primitive and a nonblocking contention resolution strategy. We will apply these principles for the discovery and implementation of 4 new high performing wait-free shared data structures (hash map, vector, queue, and a priority queue) and for finding a scalable solution to the fundamental h-out-of-k mutual exclusion problem for multiple resource allocation. The proposed program development approach will be evaluated through its application in a number of real-world codes and benchmark suites such as Mantevo [3].

Impact

This work will provide a valuable research vehicle necessary for understanding how to design applications and algorithms for the rapidly evolving massively parallel chip architectures. The success of this work will ensure the development of exascale software that is highly effective across science domains. The deliverables of this research will make exascale computing tractable and pragmatic thus opening new frontiers for scientific discovery and economic growth at the national level.

References

1. http://sst.sandia.gov/about_sstmacro.html
2. http://rosecompiler.org/
3. http://www.mantevo.org/