Award ER25844: Minimizing System Noise Effects for Extreme-Scale Scientific Simulation Through
Function Delegation
Indiana University
Investigator: Andrew Lumsdaine

Executive Summary

In software running on distributed computing clusters, time spent on communication between nodes in the cluster can be a significant portion of the overall computation time; background operating system tasks and other computational “noise” on the nodes of the system can have a significant impact on the amount of time this communication takes, especially on large systems. The research completed in this period has improved understanding of when such noise will have a significant impact. Specifically, it was demonstrated that not just noise on the nodes, but also noise on the network between nodes can have a significant impact on computation time [3]. It was also demonstrated that noise patterns matter more than noise intensity: very regular noise can cause less disruption than lighter (on average) but less regular noise [9]. It was also demonstrated that the effect of noise is more prominent as the speed of the network between nodes is increased [9]. Furthermore, a tracing tool, Netgauge, was improved via our work, and a system simulator, LogGOPSim, was developed; they can be used by application developers to improve performance of their program and by system designers to mitigate the effects of noise by adjusting the noise characteristics of the operating system. Both have been made freely available as open source programs. In the course of developing these tools, we demonstrated weaknesses in existing methodologies for modeling communication [1], and we introduced a more detailed model, LogGOPS, for simulating systems.

Not only were the deleterious effects of noise explored but we have also offered solutions. Our studies of simulations of system noise have led to specific recommendations on tuning systems to mitigate noise. We have also improved existing approaches to mitigating noise. “Non-blocking collective communication” avoids the effects of noise by letting communication continue simultaneously with computation (thus being “non-blocking”), so that the delays in communication introduced by noise have a smaller impact on overall computation time. Potentially, noise can be reduced much further by “offloading” communication tasks to a separate processing element than the operating system is using. We have improved our library LibNBC, which provides an implementation of non-blocking collectives, via this work. During this research, our proposal to include non-blocking collectives (which used LibNBC as a reference implementation) in the upcoming MPI-3 standard was accepted. As MPI is a ubiquitous and important standard for communication in parallel computing, this demonstrates a certain acceptance of the practicality and desirability of non-blocking collectives. Now that non-blocking collectives are a part of the standard we can expect to see optimized platform-specific implementations of non-blocking collectives. Also as part of this work we have also developed a language GOAL (Global Operation Assembly Language) that can be used as a starting point for defining languages to express optimized communication algorithms.

Accomplishments

We proposed to delegate the execution of collective communications to separate processing units. LibNBC provides a critical component of such delegation by providing an interface for overlapping the

communication and computation, though a final implementation of a delegated communication library remains in the prototype stage. We aimed to make non-blocking collective operations more accessible by standardizing them, and our proposal (for which LibNBC served as a reference implementation) to standardize them as part of MPI-3 was accepted.

We proposed to develop a system simulator to predict the performance characteristics of delegated communication. LogGOPSim provides such a simulation, and has proved very useful in predicting the performance of systems under different network and noise configurations [4][9]. The simulator is designed to make it easy to model communication delegation.

We proposed developing as part of this project open source software to enable delegation. While full communication delegation is not implemented, LibNBC provides a quality, open source reference implementation of non-blocking collectives that can be used via hardware threads to provide such delegation (and with non-blocking collectives as part of MPI-3, other high-quality, open source implementations are or will be forthcoming). In the end our performance analysis tools have demonstrated themselves to be very useful, and they have also received open source releases. NetGauge, LogGOPSim and ORCS (Oblivious Routing Congestions Simulator), are all freely available as open source software.

We proposed examining VH-1 and the Parallel Ocean Program (POP) to see how they were affected at large scale by delegated communication. Although this examination was not completed, tracing and simulation was performed on POP to characterize the effects of noise on the program at scale (32,000 processes).

Activities

Our two main goals as originally set out were to implement the delegation of the execution of collective communications to separate processing units (specifically, separate microprocessor cores) and to provide a non-blocking interface to collective communications operations. We had proposed to approach the design, development, and analysis of our implementations by using a system simulator, microbenchmarks, and complete applications; and we had proposed to provide high-quality implementations of our approaches and to work to standardize non-blocking collectives.

For the most part, we have followed these original goals. Throughout the project, decisions have been made to make best use of time and labor, and the most promising directions for research within the scope of this project have been pursued. Given resource constraints, we have been more successful in accomplishing some of our goals than other.

We have continued to work on the implementation of non-blocking collectives, improving LibNBC throughout the course of our work. Furthermore, we did work to standardize non-blocking collectives (for which our reference implementation was an important part) and were successful in doing so.

The development of a simulator capable of supporting complete systems simulation led to some results of interest in their own right that were not part of the original proposal (see, e.g. [9]). This has given us an opportunity to research fundamental questions about the effects of noise on systems, and to find

answers that will be of use not only to our own implementations but to others' as well.

The implementation of delegation was less successful. A partial prototype was developed, which could be developed more fully without much effort. But given time and labor constraints, more effort was invested in simulating this delegation to assess its effectiveness at mitigating noise. Our simulator was written to take into account delegation. However, it relies on actual execution traces from programs – programs that in the context of delegation would ideally use hybrid computation (that is, to compute on multiple cores per node and also use multiple nodes). Our tracing tools relied on techniques that were not able to handle this hybrid communication so we relied on more robust third-party tracing tools, but the complexity of these tools and the software being traced caused incompatibilities, and it was difficult to obtain the traces we needed for simulation. Given more time, work would have continued by adapting different (simpler but less accurate) tracing techniques to solve this problem.

Products

A. Software

a. LibNBC

The library LibNBC implements all MPI collective operations in a nonblocking way. It runs with inline (test-based) progression as well as threaded progression. Threaded progression enables the offload of the communication to a separate core. An OFED-optimized version is also available. A revised version of LibNBC (LibNBC2) based on the Group Operation Assembly Language (GOAL) [5] has been prototyped and shows good performance and overlap. Additional details about LibNBC and the source-code are available at <http://www.unixer.de/research/nbcoll/libnbc/>

b. Netgauge

The universal benchmarking tool NetGauge has been extended to support Operating System noise measurements. The new “noise” module supports the three methods “Fixed Work Quantum” (FWQ), “Fixed Time Quantum” (FTQ), and “Selfish Detour”. NetGauge supports the necessary high-resolution timers (often cycle-accurate) for the architectures: x86, x86-64, ia64, ppc, and mips and falls back to MPI Wtime on unsupported architectures. NetGauge now generates detailed noise traces that can be used for OS noise simulations. More details about NetGauge and the full source-code are available at <http://www.unixer.de/research/netgauge/>

c. ORCS, an Oblivious Routing Congestion Simulator

The ORCS simulator is able to model the effects of congestion and routing on different InfiniBand topologies. It is able to assess the “effective bisection bandwidth” of arbitrary InfiniBand topologies. This simulator is the first step (the networking side) towards full system simulation and optimized collective operations. More information about ORCS and the source-code are available in [8] and at <http://www.unixer.de/research/orcs/>. Based on ORCS simulation results, we analyzed the impact of Network Noise (due to background network congestion) on collective algorithms and applications [3].

d. LogGOPSim

LogGOPSim (previously LogGPSim) is a full LogGPS simulator. LogGPS is an established network model for parallel applications and algorithms. The simulator is capable of reading MPI traces from real MPI applications and replaying them with different LogGP parameters and injected system noise. The gathered noise traces (NetGauge output) can be used as input for the simulator to achieve accurate simulation results. The current version of the simulator supports full LogGPS, however we have extended LogGOPSim with the principles of ORCS and we can now simulate congestion on real networks and routings (together with the system noise). We have incorporated an efficient virtual memory paging approach to enable LogGOPSim to simulate much larger systems. LogGOPSim is publicly available and can read Netgauge noise traces and real application traces to simulate the influence of OS noise on parallel applications. Further details about LogGOPSim and the full source-code are available at <https://www.unixer.de/research/LogGOPSim/>

B. Standards

Our proposal of adding non-blocking collectives to the MPI-3 standard has been accepted, and is now part of the draft MPI-3 standard. Should our proposal survive the upcoming final vote it will be included in MPI-3.

(The MPI implementation MPICH already includes non-blocking collectives, and the implementation was informed by the LibNBC implementation.)

C. Research

1. Global Operation Assembly Language (GOAL)

We designed the Group Operation Assembly Language as a framework to define collective communications in [5]. We show the universality of this language and how it can be used to implement all existing collective operations. By design, it readily lends itself to blocking and nonblocking execution, as well as to off-loaded execution of complex group communication operations. We also define several offline and online optimizations (compiler transformations and scheduling decisions, respectively) to improve the overall performance of the operation. Performance results show that the overhead to express current collective operations is negligible in comparison to the potential gains in a highly optimized implementation.

2. LogGPS Analysis of Different Collective Communication Algorithms

In order to test LogGPSim, we conducted a series of experiments to simulate different collective operations with different real networking (LogGPS) parameters [1]. We use the gathered parameters to simulate LogGPS models of collective operations and demonstrate the errors in common benchmarking methods for collective operations. The simulations provide new insight into the nature of collective algorithms and their pipelining properties. We show that the error grows linearly with the system size.

3. Noise Simulation Study

We used experimentally measured noise characteristics for six common HPC architectures, and performed large-scale LogGPS simulations to characterize the performance of collective operations and

three ASC applications (Sweep3D, AMG, and POP) at scale and under different noise conditions. We analyze the particular circumstances under which noise is propagated or absorbed and show that the structure of the noise is significantly more important than its amplitude. For example, we show that the Jaguar system (with low noise amplitude) performs worse at scale than a Linux system (with 4 Jaguar's noise amplitude). We further describe how point-to-point optimization and co-scheduling techniques can significantly mitigate the effects of system noise.

4. Library for Delegation of Collective Communication to Separate Core

We implemented a partial prototype of a library that implements MPI and non-blocking collectives with all communication delegated to a separate core. After a redesign to take into account threaded progression of communication, it remains to integrate collectives and non-blocking collectives with the library, but that is a simple extension that could be completed in future work.

D. Publications

- [1] T. Hoefler, T. Schneider, and A. Lumsdaine. LogGP in Theory and Practice - An In-depth Analysis of Modern Interconnection Networks and Benchmarking Methods for Collective Operations. *Elsevier Journal of Simulation Modelling Practice and Theory*, Jun. 2009.
- [2] T. Hoefler, T. Schneider, and A. Lumsdaine. The Effect of Network Noise on Large-Scale Collective Communications. *Parallel Processing Letters (PPL)*, pages 573-593, Dec. 2009.
- [3] T. Hoefler, T. Schneider, and A. Lumsdaine. The Impact of Network Noise at Large-Scale Communication Performance. *Proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2009.
- [4] T. Hoefler, T. Schneider, and A. Lumsdaine. LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 597-604, Jun. 2010.
- [5] T. Hoefler, C. Siebert, and A. Lumsdaine. Group Operation Assembly Language - A Flexible Way to Express Collective Communication. *Proceedings of the 38th International Conference on Parallel Processing (ICPP-2009)*, pages 574-581, Sep. 2009.
- [6] T. Hoefler, C. Siebert, and A. Lumsdaine. Scalable Communication Protocols for Dynamic Sparse Data Exchange. *Proceedings of the 2010 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 159–168. ACM, Jan. 2010.
- [7] T. Hoefler on behalf of the MPI Forum. MPI: A Message-Passing Interface Standard – Working-Draft for Nonblocking Collective Operations. Technical report, MPI Forum, Apr. 2009.
- [8] T. Schneider, T. Hoefler, and A. Lumsdaine. ORCS: An Oblivious Routing Congestion Simulator. Technical Report 675, Indiana University, Feb. 2009.
- [9] T. Hoefler, T. Schneider and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. *Proceedings of the International Conference for High*

Performance Computing, Networking, Storage and Analysis (SC'10), Nov. 2010.