Invited Paper : Benchmarking and Optimizing Data Movement on Emerging Heterogeneous Architectures

Amanda Bienz
Department of Computer Science
University of New Mexico
Albuquerque, USA
Email: bienz@unm.edu

***Abstract*—As supercomputers evolve, nodes are continually increasing in complexity. As a result, each generation of parallel systems brings new performance challenges. For instance, on recent systems inter-node communication has outperformed inter-socket, resulting in poor performance of many node-aware communication optimizations. Communication optimizations are critical for the performance and scalability of parallel applications, but are dependent on the parallel architecture, which varies significantly among recent generations of supercomputers. This paper investigates the performance of various paths of data movement on recent generations of systems, and analyzes the increased complexity of communication, particularly on recent heterogeneous systems. The paper also introduces MPI Advance, a communication library that enables optimizations to be created based on benchmark analysis of each emerging system.**

I. Introduction

Optimizing communication presents unique challenges with the emergence of each new generation of parallel computers. Traditionally, data movement has been optimized based on the assumption that communication between all process sets incurs an equal cost. However, with the advent of symmetric multiprocessing (SMP) nodes, inter-process communication has become heterogeneous, resulting in significant performance differences between intra-socket, inter-socket, and inter-node messages. The use of accelerators in recent supercomputers, such as Summit and Sierra, has further amplified this heterogeneity. As a result, optimizing inter-process communication costs has become increasingly challenging due to the growing variety of paths for data movement.

Inter-process communication, from collective operations to irregular point-to-point messages, is typically the bottleneck in strongly scaled parallel applications. Consequently, the performance and scalability of codebases, such as simulations and numerical solvers, heavily depend on the cost of communication. Communication optimizations, however, are dependent on parallel architecture, with notable performance irregularities among recent generation systems. This paper presents methods for benchmarking the available paths of data movement on emerging architectures and optimizing communication for each generation of supercomputer. Additionally, the paper introduces MPI Advance, a lightweight optimization library that can be tuned based on analysis of these benchmarks.

The remainder of the paper is outlined as follows. Section II discusses recent trends in parallel architectures and corresponding algorithms, while benchmarks spanning three recent generations of systems are discussed in Section III. Communication optimizations and the library MPI Advance are discussed in Section IV. Finally, Section V provides concluding remarks.

II. Recent Trends in Parallelism

Over the past few decades, super computers have vastly increased in computational power with current generation systems reaching Exascale. However, rather than increasing node count, the nodes of emerging systems are becoming more complex. For instance, the Blue Gene/L computer at Lawrence Livermore National Laboratory (LLNL) achieved over 200 teraflops through $65\,536$ dual-processor nodes [1] Intrepid, a Blue Gene/P system, doubled that peak performance to over 500 teraflops with only $40\,960$ quad-core nodes [2], while Sequoia, a Blue Gene/Q system, was the first system to achieve 10 petaflops with nearly $100,000$ nodes, each containing 16 cores [3]. Increases in per-node core counts has resulted in large variations among communication costs, such as notable distinctions between intra-socket, inter-socket, and inter-node communication as well as measurable impacts of injection bandwidth limits [4], [5].

More recent systems have not only increased CPU cores per node, but have already increased heterogeneity through accelerators. Summit achieved 122 petaflops with only $4\,536$ nodes, with each node containing 44 cores and 6 GPUs split across two sockets [6]. Similarly, Frontier is currently emerging as the first Exascale computer with $9,408$ nodes, each with 64 cores and 4 dual-GPU chiplets [7], [8]. Accelerators have further increased variation in communication costs, with data movement often occurring between GPU memories. Not only can data be communicated directly between GPUs with GPUDirect, it can also be copied to CPU cores and communicated directly between CPUs.

Parallel communication has traditionally been implemented with the assumption that all data movement is equivalent. However, as architectures increase in heterogeneity, the costs associated with the possible paths of data movement vary greatly. Standard collective algorithms, such as recursive doubling [9], [10], the Bruck algorithms [11], and Rabenseifner's algorithm [12], minimize the message count for smaller collectives and total bytes communicated for collectives with larger data sizes. Topology-aware collective algorithms have shown large improvements by minimizing the hop count for messages that traverse the network [13], [14]. However, in recent years, a large number of optimizations have focused on the SMP architecture. Hierarchical collective algorithms reduce injection bandwidth limitations with only a small subset of processes per node participating in inter-node communication [15], [16] Furthermore, locality-aware algorithms minimize the number of inter-node messages, ideal for latency-bound algorithms [17], [18], while multi-lane collectives minimize inter-node sizes, optimizing bandwidth-bound messages [19].

III. Benchmarking Across Generations

Blue Waters, a petascale computer comprised, mostly, of 16-core SMP nodes, was in operation from 2012 through 2021. Each node contained two 8-core sockets, with all cores per socket sharing L2 cache, while sockets shared only main memory. Figure 1 displays the performance of intra-socket,

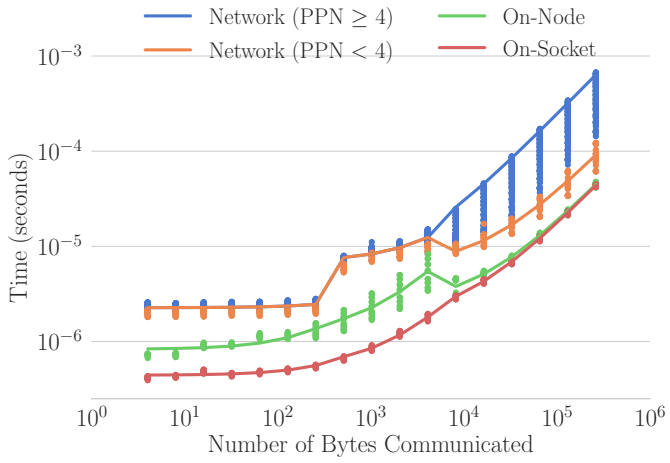

Figure 1. Blue Waters Data Movement Performance

inter-socket, and inter-node messages on Blue Waters using CrayMPI. When message size is small enough to fit in L2 cache, intra-socket communication greatly outperforms inter-socket as the data is sent through cache. However, all types of intra-node communication outperform inter-node, regardless of message size. Finally, the cost of large inter-node messages increases with the number of active processes per node, as injection bandwidth limits performance.

In 2018, LLNL began operation of Lassen, the unclassified Sierra architecture. As such, Lassen is comprised of IBM Power9 CPUs and NVIDIA Volta GPUs. Each node has two 22-core CPUs and 4 GPUs. There are direct connections between each GPU and the network interface card, allowing for data to be communicated directly from GPU memory to the network without passing through the CPU. Each CPU is directly connected to two of the four on-node GPUs, and these two GPUs are also directly connected to on another. Figure 2 shows the cost of communicating

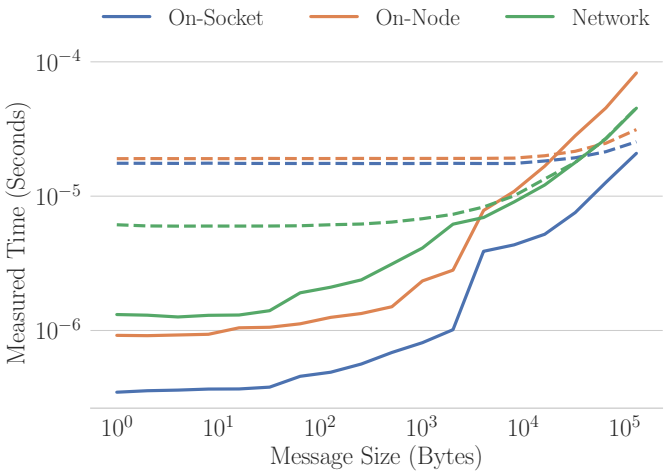

Figure 2. Lassen Data Movement Performance

data between CPU cores (solid lines) as well as directly between GPU memories (dotted lines) on Lassen with SpectrumMPI. When communicating between CPU cores, intra-socket greatly outperforms inter-node, as on previous generation SMP architectures. However, as the interconnect has higher bandwidth than the shared memory system, large inter-socket messages are more than twice as expensive as inter-node. As a result, it is cheaper to send data to another node and for them to send the data back to the correct process than to communicate directly across the NUMA region.

For small inter-GPU messages, it is significantly cheaper to send inter-node than intra-node, including intra-socket messages which are sent across the link directly connecting the two GPUs. However, for larger messages, inter-GPU communication performs as expected with intra-socket outperforming inter-socket, and both achieving better performance than inter-node.

While inter-CPU communication outperforms inter-GPU, most applications require data to be communicated between GPUs. This can happen through GPUDirect, as previously shown, or by manually copying data from GPU to CPU before inter-CPU communication. Due to the high latency associated with the GPU to CPU copy, GPUDirect communication is often cheaper when sending a single message.

However, when sending many messages, all data can be copied to the CPU at once. As a result, when sending multiple messages on Lassen, it is typically cheaper to first copy to the CPU.

Recently, LLNL began operation of Tioga, a Cray system with similar architecture to Frontier. Each node of Tioga contains 64 AMD CPUs along with 4 dual-GPU chiplets. Similar to Power9 systems such as Lassen, each GPU is connected directly to the NIC, allowing for GPUDirect communication and all 4 dual-GPU chiplets are directly connected. Figure 3 displays the cost of inter-CPU (solid) and inter-GPU (dotted) communication on Tioga. As Tioga contains only a single CPU per node, communication is split only into intra-socket and inter-node. GPUDirect communication is vastly improved over Power9 systems such as Lassen, with minimal differences between inter-CPU and inter-GPU communication through the network. Inter-CPU communication continues to outperform inter-GPU on-socket, but the difference is less drastic.

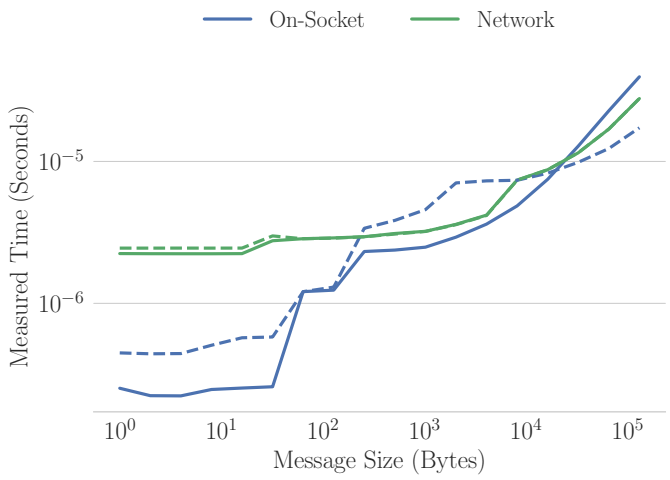

Figure 3. Tioga Data Movement Performance

While inter-CPU communication is typically comparable to inter-GPU, and intra-socket messages are in general cheaper than inter-node, there are some notable exceptions. When communicating large messages between two CPUs, on-socket communication is more expensive than inter-node. Furthermore, it is significantly cheaper to perform inter-GPU communication of large on-socket data than inter-CPU messages. Finally, when communicating eager messages, it is significantly more expensive to communicate between two GPUs on the same node than it is to perform inter-node communication.

IV. Optimizing Communication

As supercomputers increase in heterogeneity, the possible paths through which data can move increase. For instance, on systems such as Lassen and Tioga, inter-GPU communication can be performed through GPUDirect, by copying the data to a single GPU core, or by copying a portion of the data to each of the many available CPU cores per GPU. Furthermore, many of the optimizations for SMP architectures fail to achieve optimal communication on emerging systems due to unexpected performance associated with various paths of data movement. For example, communication optimizations for SMP architectures, such as locality-aware algorithms and multi-lane collectives, rely on intra-node communication outperforming inter-node, while the opposite performance is often achieved on emerging systems. Similarly, hierarchical collectives avoid injection bandwidth limitations, but these limits are often naturally avoided as the number of active CPU cores often matches the number of GPUs.

To achieve optimal performance on emerging systems, it is necessary to first create and analyze benchmarks in an effort to determine the ideal path of communication. Optimizations should then be reformulated to achieve performant communication on these systems. However, currently, the majority of communication optimizations reside within MPI implementations, and as such are optimized generally rather than specific to individual system benchmarks.

One solution to the need for optimizations based on benchmarks is MPI Advance [1], a library that provides numerous optimizations for MPI communication, including locality-aware aggregation as well as partitioned communication. The lightweight library sits on top of MPI, allowing for optimizations regardless of available versions of MPI. MPI Advance allows for researchers to test various communication optimizations indicated by benchmarks, such as reducing large inter-socket messages on emerging architectures. While optimizations within MPI Advance incur overhead from sitting on top of MPI, the ease of use allows for extensive testing of these optimizations, with hopes that they will eventually be added into production versions of MPI when applicable.

V. Conclusions and Future Work

As supercomputers continue to evolve, accurate benchmarking and performance modeling become increasingly important, due to the significant differences in costs associated with various data movement paths. Accurate benchmarks allow for communication optimizations, which are often counter-intuitive without these measurements, such as reducing communication costs by exchanging large inter-socket communication for additional inter-node messages. Portable communication optimizations, such as dynamically selected optimizations based on benchmark output, could have a large impact on the performance and scalability of applications on emerging systems.

[1] https://github.com/mpi-advance/

This material is based in part upon work supported by the Department of Energy under Award Number DE-NA0003966.

REFERENCES

[1] J. Moreira, M. Brutman, J. Castaños, T. Engelsiepen, M. Giampapa, T. Gooding, R. Haskin, T. Inglett, D. Lieber, P. McCarthy, M. Mundy, J. Parker, and B. Wallenfelt, "Designing a highly-scalable operating system: The blue gene/l story," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 118–es. [Online]. Available: https://doi.org/10.1145/1188455.1188578

[2] A. Faraj, S. Kumar, B. Smith, A. Mamidala, J. Gunnels, and P. Heidelberger, "Mpi collective communications on the blue gene/p supercomputer: Algorithms and optimizations," in *Proceedings of the 23rd International Conference on Supercomputing*, ser. ICS '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 489–490. [Online]. Available: https://doi.org/10.1145/1542275.1542344

[3] R. Haring, M. Ohmacht, T. Fox, M. Gschwind, D. Satterfield, K. Sugavanam, P. Coteus, P. Heidelberger, M. Blumrich, R. Wisniewski, a. gara, G. Chiu, P. Boyle, N. Chist, and C. Kim, "The ibm blue gene/q compute chip," *IEEE Micro*, vol. 32, no. 2, pp. 48–60, 2012.

[4] W. Gropp, L. N. Olson, and P. Samfass, "Modeling MPI communication performance on SMP nodes: Is it time to retire the ping pong test," in *Proceedings of the 23rd European MPI Users' Group Meeting*, ser. EuroMPI 2016. New York, NY, USA: ACM, 2016, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/2966884.2966919

[5] A. Bienz, W. D. Gropp, and L. N. Olson, "Improving performance models for irregular point-to-point communication," in *Proceedings of the 25th European MPI Users' Group Meeting, Barcelona, Spain, September 23-26, 2018*, 2018, pp. 7:1–7:8. [Online]. Available: https://doi.org/10.1145/3236367.3236368

[6] J. Hines, "Stepping up to summit," *Computing in Science & Engineering*, vol. 20, no. 2, pp. 78–82, 2018.

[7] D. Schneider, "The exascale era is upon us: The frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second," *IEEE Spectrum*, vol. 59, no. 1, pp. 34–35, 2022.

[8] S. S. Pawlowski, "Exascale science: the next frontier in high performance computing," in *Proceedings of the 24th ACM International Conference on Supercomputing*, 2010, pp. 1–1.

[9] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *Int. J. High Perform. Comput. Appl.*, vol. 19, no. 1, pp. 49–66, Feb. 2005. [Online]. Available: http://dx.doi.org/10.1177/1094342005051521

[10] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 65:1–65:43, Aug. 2019. [Online]. Available: http://doi.acm.org/10.1145/3320060

[11] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 11, pp. 1143–1156, 1997.

[12] R. Rabenseifner, "Optimization of collective reduction operations," in *Computational Science - ICCS 2004*, M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–9.

[13] K. Kandalla, H. Subramoni, A. Vishnu, and D. K. Panda, "Designing topology-aware collective communication algorithms for large scale infiniband clusters: Case studies with scatter and gather," in *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, April 2010, pp. 1–8.

[14] T. Ma, G. Bosilca, A. Bouteiller, and J. J. Dongarra, "Kernel-assisted and topology-aware MPI collective communications on multicore/many-core platforms," *Journal of Parallel and Distributed Computing*, vol. 73, no. 7, pp. 1000 – 1010, 2013, best Papers: International Parallel and Distributed Processing Symposium (IPDPS) 2010, 2011 and 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731513000166

[15] H. Zhu, D. Goodell, W. Gropp, and R. Thakur, "Hierarchical collectives in MPICH2," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, M. Ropo, J. Westerholm, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 325–326.

[16] J. L. Träff and A. Rougier, "MPI collectives and datatypes for hierarchical all-to-all communication," in *Proceedings of the 21st European MPI Users' Group Meeting*, ser. EuroMPI/ASIA '14. New York, NY, USA: ACM, 2014, pp. 27:27–27:32. [Online]. Available: http://doi.acm.org/10.1145/2642769.2642770

[17] A. Bienz, L. Olson, and W. Gropp, "Node-aware improvements to allreduce," in *Proceedings of ExaMPI 2019*. United States: IEEE, Nov. 2019, pp. 19–28.

[18] A. Bienz, S. Gautam, and A. Kharel, "A locality-aware bruck allgather," in *Proceedings of the 29th European MPI Users' Group Meeting*, ser. EuroMPI/USA'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 18–26. [Online]. Available: https://doi.org/10.1145/3555819.3555825

[19] J. L. Träff and S. Hunold, "Decomposing mpi collectives for exploiting multi-lane communication," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 2020, pp. 270–280.