PRAC: Systems Software for Scalable Applications

Pavan Balaji, William Gropp, Ewing Lusk, Rajeev Thakur, Antonio J. Peña

Final Report
June 2013

Abstract

This project aimed at improvements to MPI to enable various optimizations including topology-aware mapping of MPI processes on the Gemini network of Blue Waters, optimization of NAMD/Charm++ over MPI on Blue Waters, and integration of GPU and MPI related data movement for performance and programmer convenience.

Current HPC systems utilize a variety of interconnection networks, with varying features and communication characteristics. MPI normalizes these interconnects with a common interface used by most HPC applications, either directly or indirectly through libraries and runtime systems such as Charm++. However, network properties can have a significant impact on application performance. We explore the impact of the interconnect on application performance on irregular/anisotropic networks, such as the Cray Gemini network on Blue Waters, which provides twice the Y-dimension bandwidth in the X and Z dimensions.

On systems with GPUs, such as Blue Waters, current hybrid programming models require the user to explicitly manage the movement of data between host, GPU, and the network, which is both tedious and inefficient. We have developed a unified programming interface, MPI-ACC, that provides a convenient and optimized way of end-to-end data communication among CPUs and GPUs.

1 Introduction

A key component in high-performance computing (HPC) systems is the interconnection network, which integrates an array of computational resources into a single system and provides efficient data movement. Interconnection networks continue to be an area of high innovation, as they strive to keep pace with rapidly increasing levels of concurrency and greater demand for communication performance. As interconnects evolve, system topologies, link properties, and communication characteristics vary across systems.

Irregular and anisotropic topologies, such as the three-dimensional torus recent Cray XE and XK systems implement, pose a challenge for the efficient use of the network resources. The recent trend of employing accelerators —such as Graphics Processing Units (GPUs) or those based on the Intel Many Integrated Core (MIC) architecture— pose an additional level of irregularity on the data path, due to non-uniform memory access (NUMA) issues.

Most HPC applications use the Message-Passing Interface (MPI) [21] as a portable interface for expressing data movement, either directly or beyond higher-level libraries such as the runtime of the Charm++ programming model [15]. Our goal is to provide MPI with mechanisms to efficiently accomplish data transfers on a wide range of environments, including irregular/anisotropic networks and accelerator-equipped systems, in order to leverage highly-efficient communications at no cost to the wide community of MPI users.

The major contributions included in this document are:

1. A better adaptation of the Cray MPI library to the Charm++ runtime.
2. The study of the potential implications of the anisotropic behavior of Cray Gemini networks on applications.
3. The incorporation of efficient GPU awareness into MPI routines.

The rest of this document is organized as follows: Section 2 presents our work, Section 3 reviews some related work, followed by conclusions in Section 4.

2 Accomplishments

This section covers the work we carried out within the project. We first describe our research on the network's anisotropic behavior. Next, we introduce our work on assessing the performance difference between Cray's MPI and communications based on the interconnect's native application programming interfaces (APIs). Last, we introduce our work to include GPU awareness into the MPI library.

2.1 Analysis of Topology-Dependent MPI Performance on Blue Waters

We also investigated the implications of the anisotropic behavior of the Blue Waters' Gemini network on applications [19]. This network presents largely different behaviors in all three dimensions of the three-dimensional torus it forms. Despite both X and Z dimensions presenting the same number of links between routers, their performance behavior differs dramatically. In addition, the Y dimension features half of the number of links of any other dimension. Furthermore, every two nodes share a network application-specific integrated circuit (ASIC), i.e., a network topology coordinate, which contributes to the anisotropic behavior of this network.

After a characterization of this interconnect by means of point-to-point benchmarks, we evaluated the behavior of MPI collectives along the different dimensions and planes of the network topology of this system. We proved that when the nodes sharing a network coordinate are considered to be placed along the dimension featuring lesser network links, communication performance along that direction is higher than expected, thus maximizing the use of the network resources (see Figure 1). Hence, applications performing dimension-wise communications can highly benefit of an optimal process placement, as in Figure 2 for the case of planewise MPI `Allgather` collectives. Additionally, we demonstrated by means of a halo benchmark that including awareness of the network topology in the MPI library can outperform the heuristic-based rank ordering of this system, as shown in Figure 3.

The results of our study demonstrate the benefits of an MPI-network topology matching for common application scenarios. Since the current MPI implementation on Cray systems ignores the reorder parameter meant for this purpose, currently topology-sensitive applications need to be concerned about network placement, aided (one hopes) by external libraries. This situation poses an unnecessary overhead for application developers, who should be able to rely on the capabilities of the MPI library for this purpose. We concluded that MPI libraries would benefit from the incorporation of existing topology matching work.

2.2 Optimizing Charm++ over MPI on Blue Waters

We are currently evaluating MPI on the heavily irregular topology of the XE systems. As a first step, we have investigated the performance difference between MPI and native communication models on the

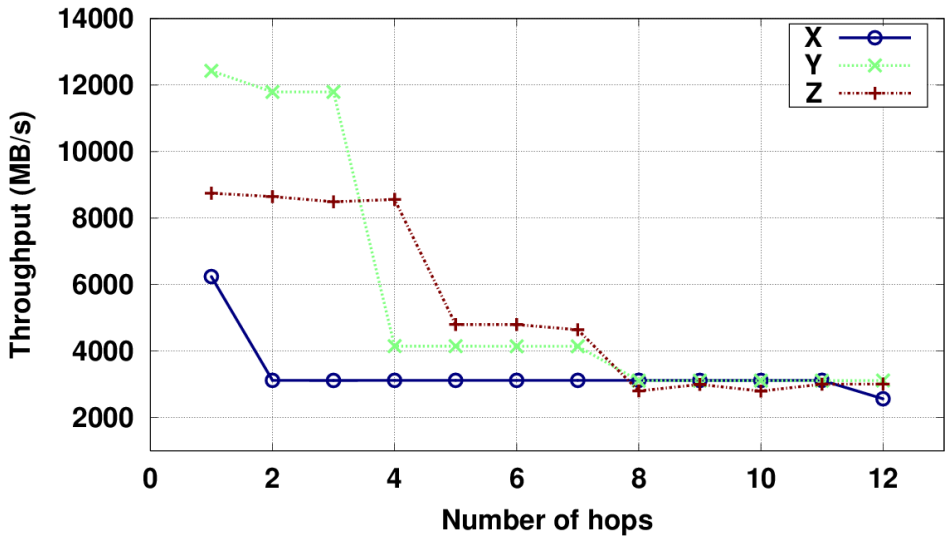

Figure 1: Internode aggregate transfer rate for 1 MB messages; 2 parallel paths concurrently transfer data.

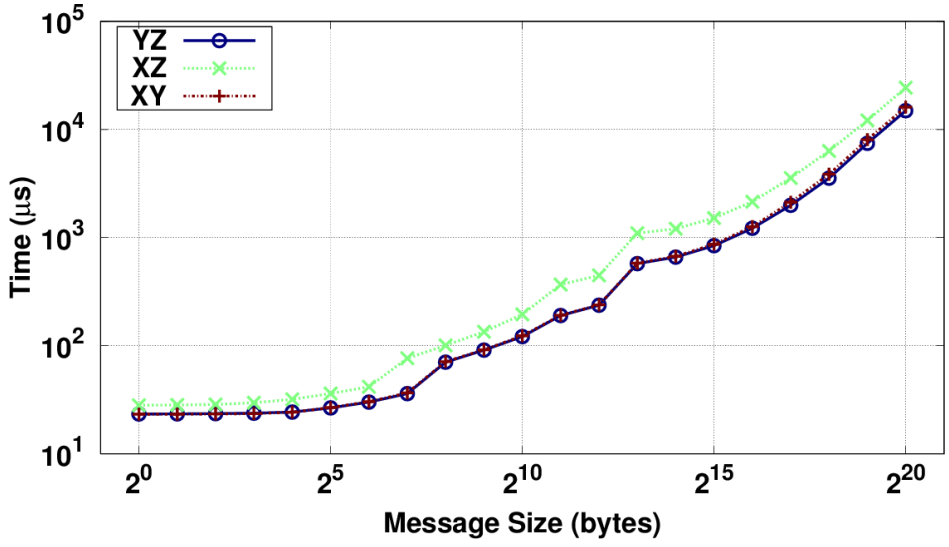

Figure 2: Planewise `MPI_Allgather` on a $7 \times 7 \times 7$ cube.

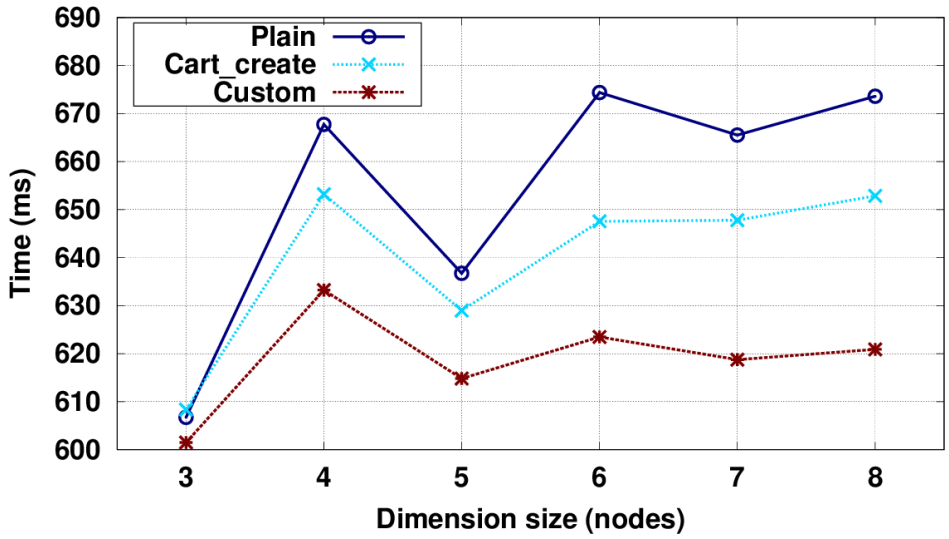

Figure 3: Three-dimensional Halo benchmark. *Plain*: user-distributed ranks along the user-defined application-level topology, based on their original ordering; *Cart_create*: ranks distributed by the MPI library; *Custom*: the MPI topology matches the network topology, considering the double nodes per ASIC to be placed along the Y dimension.

XE/XK system. Specifically, we used the NAMD molecular dynamics simulation [22] as a case study and compared the performance of its native implementation on the Blue Waters network with that of its MPI-based implementation [11].

A particular challenge in utilizing MPI as a runtime system for NAMD-like applications is that MPI is tuned for expected messaging semantics, where messages are explicitly "received" by the recipient. In contrast to this, the NAMD application and the Charm++ model of parallelism that it utilizes depend on unexpected message communication to trigger method invocations on shared objects.

We have investigated MPI implementation and specification aspects to resolve the impedance mismatch between these models, from unexpected message queue lengths to the frequency of calls to the MPI progress engine. We investigated the effects of the alternating computation versus communication phases of single-threaded Charm++. We also examined the trade-offs of utilizing eager versus rendezvous protocols in MPI. The most notable performance improvement came from modifying the MPI implementation to increase the eager/rendezvous threshold so that unexpected Charm++ messages could be transmitted using the eager protocol.

The results from this study are shown in Figure 4. We found that changing the default MPI eager threshold leads to a faster initialization phase in NAMD, as well as an improvement in timestep duration. In contrast to the rendezvous protocol, the eager protocol removes an initial synchronization step, and more closely resembles the asynchrony of the Charm++ model.

We have utilized this performance study to improve the MPI implementation to allow the library (such as Charm++) to perform per-communicator tuning of the MPI stack. These changes have already been incorporated into the MPICH stack and are being passed down to Cray to incorporate in their production MPI implementation.

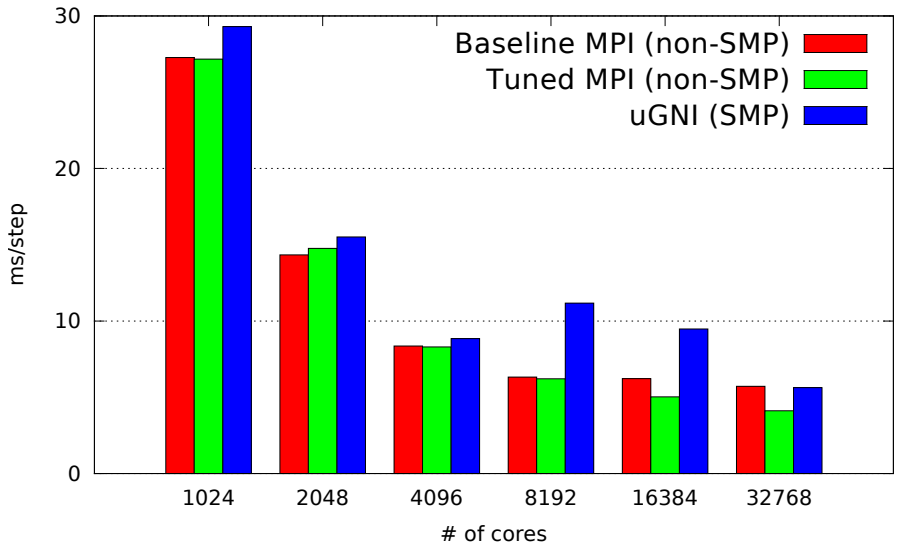

Figure 4: Comparison of NAMD over MPI versus the native Charm++ module.

2.3 Incorporating GPU awareness into MPI

We have also developed MPI-ACC [4], a system that enables users to call MPI functions directly on data stored in GPU memory, with MPI-ACC doing the work of efficiently moving data between GPU and host using various optimizations, as overlapping GPU and network data transfers, or efficiently driving intranode communications to avoid unnecessary memory copies [14] or assisted by direct memory access (DMA) engines [13]. This support brings users highly optimized data transfers involving GPU memory spaces while insulating them from its development complexity [5]. An example of the performance gains enabled by our implementation is shown in Figure 5. We have also developed techniques to efficiently handle movement between GPU and host of noncontiguous data described using MPI derived datatypes [12], and studied the efficiency of the different synchronization and ordering semantics which can be exposed to the users [3]. We are currently evaluating the performance of these optimizations on Blue Waters using applications such as epidemiology simulation and seismic modeling.

3 Related Work

Recently, Sun *et al.* [20] examined Charm++ implementation issues on current Cray Gemini-based systems, while Kumar *et al.* [16] studied the same on IBM Blue Gene/Q systems. These works look at optimizations to the native (non-MPI) networking layer built into Charm++. The techniques may serve as the basis for future Charm++-over-MPI optimizations to help mitigate the programming model impedance mismatch.

Performance evaluations of communications employing the MPI interface have been covered for the different supercomputers since MPI emerged two decades ago. Recent examples include studies of the performance behavior of MPI on the Blue Gene/P [7, 8, 9]. Performance evaluations on recent Cray XE/XK platforms have also been carried out. Early experience on the Titan system at Oak Ridge was described in [10]. Subsequent work includes an evaluation of the overhead of the Cray MPI implementation (as a justification for the design of a low-level benchmarking tool for internode, inter-GPU data transfers) [18]

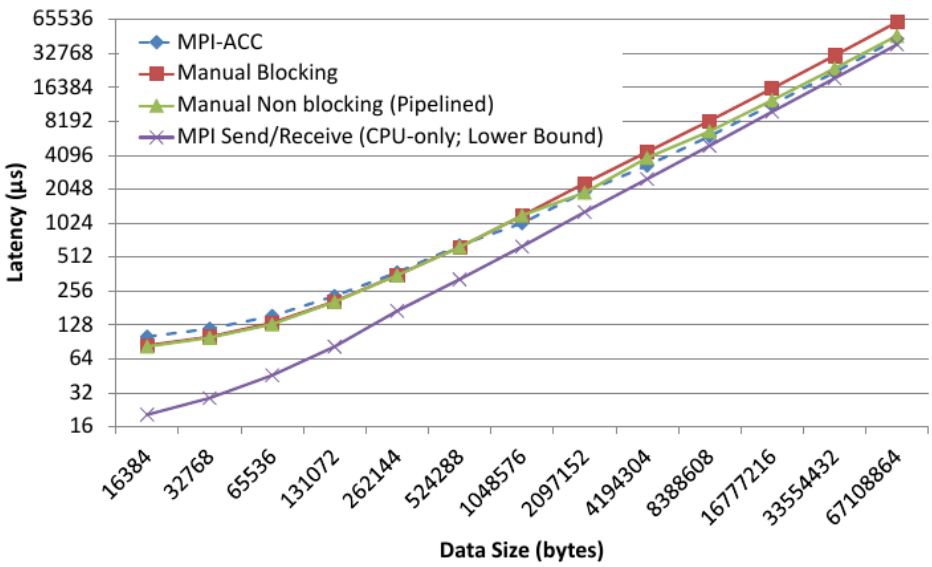

Figure 5: Internode communication latency for GPU-to-GPU (CUDA) data transfers. Similar performance is observed for OpenCL data transfers.

and an evaluation of the Partitioned Global Address Space runtime system of these platforms [23]. Prior work suggested that the placement of the nodes sharing the same Gemini ASIC in the Y direction was beneficial [6]. However, this work lacked a justificative base performance evaluation. Technical details supporting the anisotropy of this fabric were also missing in that work.

MVAPICH [1] is another implementation of MPI based on MPICH and is optimized for remote DMA (RDMA) networks such as InfiniBand. MVAPICH2-GPU, starting on release v1.8 of MVAPICH, includes support for transferring CUDA [17] memory regions across the network [24] (point-to-point, collective and one-sided communications). In order to use this, however, each participating system should have an NVIDIA GPU of compute capability 2.0 or higher and CUDA v4.0 or higher, because MVAPICH2-GPU leverages the unified virtual address (UVA) feature of CUDA. On the other hand, MPI-ACC takes a more portable approach: we support data transfers among CUDA, OpenCL [2], and CPU memory regions; and our design is independent of library version or device family. By including OpenCL support in MPI-ACC, we automatically enable data movement between a variety of devices, including GPUs from NVIDIA and AMD, CPUs from IBM and Intel, AMD Fusion, IBM's Cell Broadband Engine, and Intel MIC. Also, we make no assumptions about the availability of key hardware features (e.g., UVA) in our interface design, thus making MPI-ACC a truly generic framework for heterogeneous CPU-GPU systems.

4 Conclusions

MPI and Charm++ cater to different programming paradigms: the former favors BSP-styled codes, whereas the latter supports a more asynchronous design. While the former may be used as the networking engine for the latter, this comes at a small but perceptible cost. We investigated the reasons behind this overhead and tweaked the MPI library parameters to attain better performance for hybrid Charm++/MPI codes.

We also studied the anisotropic implications of the Cray Gemini network on MPI communications. The results of our study demonstrate the benefits of an MPI-network-topology matching for common application

scenarios. Additionally, we suggested that MPI libraries will benefit from the incorporation of existing topology matching work, which is planned for future work.

Finally, we developed MPI-ACC, an integrated and extensible framework that allows end-to-end data movement in accelerator-connected systems. We implemented several optimizations in MPI-ACC, such as pipelining, dynamic adjustment of pipeline parameters based on PCI Express affinity and NUMA effects, and efficient management of CUDA and OpenCL resource objects. We are currently working on its integration into the MPICH code.

Acknowledgments

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This work is also part of the "System Software for Scalable Applications" PRAC allocation support by the National Science Foundation (award number OCI-1036216). This work was supported in part by a NEIS-P2 subaward from NCSA/UIUC and in part by the U.S. Dept. of Energy under contract DE-AC02-06CH11307.

References

[1] MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE. `http://mvapich.cse.ohio-state.edu`.

[2] The OpenCL specification. `http://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf`, 2008.

[3] Ashwin M Aji, Pavan Balaji, James Dinan, Wu-chun Feng, and Rajeev Thakur. Synchronization and ordering semantics in hybrid MPI+GPU programming. 2013.

[4] Ashwin M. Aji, James Dinan, Darius Buntinas, Pavan Balaji, Wu-chun Feng, Keith R. Bisset, and Rajeev Thakur. MPI-ACC: An integrated and extensible approach to data movement in accelerator-based systems. In *14th IEEE International Conference on High Performance Computing and Communications*, Liverpool, UK, June 2012.

[5] Ashwin M. Aji, Lokendra S. Panwar, Feng Ji, Milind Chabbi, Karthik Murthy, Pavan Balaji, Keith R. Bisset, James Dinan, Wu-chun Feng, John Mellor-Crummey, Xiaosong Ma, and Rajeev Thakur. On the efficacy of GPU-integrated MPI for scientific applications. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, HPDC'13, pages 191–202, New York, NY, USA, 2013. ACM.

[6] Carl Albing, Norm Troullier, Stephen Whalen, Ryan Olson, Joe Glenski, Howard Pritchard, and Hugo Mills. Scalable node allocation for improved performance in regular and anisotropic 3D torus supercomputers. In *Recent Advances in the Message Passing Interface*, volume 6960 of *LNCS*. 2011.

[7] Pavan Balaji, Anthony Chan, William Gropp, Rajeev Thakur, and Ewing Lusk. Non-data-communication overheads in MPI: Analysis on Blue Gene/P. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 13–22. Springer, 2008.

[8] Pavan Balaji, Anthony Chan, Rajeev Thakur, William Gropp, and Ewing Lusk. Toward message passing for a million processes: Characterizing MPI on a massive scale Blue Gene/P. *Computer Science-Research and Development*, 24(1-2):11–19, 2009.

[9] Pavan Balaji, Harish Naik, and Narayan Desai. Understanding network saturation behavior on large-scale Blue Gene/P systems. In *15th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 586–593. IEEE, 2009.

[10] Arthur S. Bland, Jack C. Wells, Bronson Messer, Oscar R. Hernandez, and James H. Rogers. Titan: Early experience with the Cray XK6 at Oak Ridge National Laboratory. In *CUG 2012*, May 2012.

[11] Ralf Gunter, David Goodell, James Dinan, and Pavan Balaji. Optimizing Charm++ over MPI. In *11th Annual Charm++ Workshop*, April 2013.

[12] John Jenkins, James Dinan, Pavan Balaji, Nagiza F. Samatova, and Rajeev Thakur. Enabling fast, non-contiguous GPU data movement in hybrid MPI+GPU environments. In *IEEE International Conference on Cluster Computing (Cluster)*, September 2012.

[13] F. Ji, A.M. Aji, J. Dinan, D. Buntinas, P. Balaji, R. Thakur, W. Feng, and X. Ma. DMA-assisted, intranode communication in GPU accelerated systems. In *14th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2012.

[14] Feng Ji, A.M. Aji, J. Dinan, D. Buntinas, P. Balaji, Wu chun Feng, and Xiaosong Ma. Efficient intranode communication in GPU-accelerated systems. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, pages 1838–1847, 2012.

[15] Laxmikant V. Kale and Sanjeev Krishnan. CHARM++: A portable concurrent object oriented system based on C++. *SIGPLAN Not.*, 28(10):91–108, October 1993.

[16] Sameer Kumar, Yanhua Sun, and L. V. Kale. Acceleration of an Asynchronous Message Driven Programming Paradigm on IBM Blue Gene/Q. In *Proceedings of 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Boston, USA, May 2013.

[17] NVIDIA. CUDA. `http://www.nvidia.com/cuda`.

[18] Antonio J. Peña and Sadaf R. Alam. Evaluation of inter- and intra-node data transfer efficiencies between GPU devices and their impact on scalable applications. In *The 13th International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2013.

[19] Antonio J. Peña, Ralf G. Correa Carvalho, James Dinan, Pavan Balaji, Rajeev Thakur, and William Gropp. Analysis of topology-dependent MPI performance on Gemini networks. In *Proceedings of EuroMPI 2013*, Madrid, Spain, September 2013.

[20] Yanhua Sun, Gengbin Zheng, Chao Mei, Eric J. Bohm, James C. Phillips, Laximant V. Kalé, and Terry R. Jones. Optimizing fine-grained communication in a biomolecular simulation application on Cray XK6. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 55:1–55:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[21] The MPI Forum. `http://www.mpi-forum.org`, 2013.

[22] University of Illinois at Urbana-Champaign. NAMD - scalable molecular dynamics. http://www.ks.uiuc.edu/Research/namd.

[23] Abhinav Vishnu, Monika Bruggencate, and Ryan Olson. Evaluating the potential of Cray Gemini interconnect for PGAS communication runtime systems. In *19th Annual Symposium on High Performance Interconnects (HOTI)*, 2011.

[24] Hao Wang, Sreeram Potluri, Miao Luo, Ashish Singh, Sayantan Sur, and Dhabaleswar Panda. MVAPICH2-GPU: optimized GPU to GPU communication for InfiniBand clusters. *Computer Science – Research and Development*, 26:257–266, 2011.