Final Report for Enhancing the MPI Programming Model for PetaScale Systems

PI: William D. Gropp, University of Illinois at Urbana-Champaign
DOE #DE-FG02-08ER25835
Start Date: 4/1/08
Period of Report: 4/1/08-3/31/13

Abstract

This project performed research into enhancing the MPI programming model in two ways: developing improved algorithms and implementation strategies, tested and realized in the MPICH implementation, and exploring extensions to the MPI standard to better support PetaScale and ExaScale systems.

Results

Some of the most interesting results from this project address scaling issues in MPI. One early result on vastly more scalable algorithms for the MPI_Comm_split operation [25] inspired several papers at EuroMPI'11. We believe that our algorithm and work remains best for Exascale systems. In work that is critical for the use of MPI in upcoming trans-petascale and exascale systems, we developed efficient data structures used within the MPI implementation to represent quantities that are indexed by rank. It is sometimes alleged that the existence of such abstract data structures implies that MPI cannot be scalable, but this is not true – it isn't necessary to realize all of these elements [13]. By using sparse arrays and customized, efficient hash tables, we were able to significantly reduce the memory needs of the MPICH implementation with only a small performance penalty. This work, done jointly with the MPICH group at ANL, was reported at EuroMPI'11 [11]. We investigated communication overheads related to scalability on Blue Gene/P, in collaboration with the Argonne MPICH group. This work was described in [5, 4] and [8]. Related work investigated algorithms for irregular all-gather operations (MPI_Allgatherv) that demonstrated the value of pipelining in this irregular case to avoid delays caused by a mix of very long and very short messages. This work was presented in [32, 30]. Other work looked at hybrid collective algorithms; this is now part of the MPICH release [35]. A new set of collective algorithms that take into account network congestion were developed; this work was awarded best paper [24].

We continued to refine the scalable process management interface used in MPICH. Among the important features of this interface, known as PMI v2 (for process management interface, version 2), is better support for node "attributes." A conference paper was prepared and appeared on the PMIv2 API [1]. This interface is being implemented by Argonne as part of the Hydra process manager; we continue to support several other process managers (the whole point of the PM interface is to ensure that MPICH will work smoothly and scalably with native process managers).

We participated in the development of the MPI 2.1 standard as a chapter co-author and the MPI 3.0 standard as the overall editor and as chapter co-author for several key chapters, including the introduction and terms, and the communicator and groups chapter. Gropp co-led the MPI-3 RMA (Remote Memory Access) working group (with Rajeev Thakur of Argonne and Torsten Hoefler of Illinois). Supported by this grant, Gropp helped formulate and define the new RMA model that has become part of the MPI-3 standard; early work in this direction was published as [29, 26]. This functionality includes atomic remote memory operations, more dynamic access to remote process memory, and the option of memory ordering semantics, desired by some applications. The RMA model can be found in the MPI-3 Document, available at www.mpi-forum.org . A paper describing this model appeared at EuroMPI-'12 [17].

While testing the Graph500 Benchmark, we identified a number of shortcomings in the MPICH implementation of the RMA operations and designed and implemented new versions. These are now part of the MPICH release. Work in 2011 focused on developing more adaptive and dynamic algorithms that can provide good performance for both latency-dominated and bandwidth-dominated communication (current approaches are good at one or the other, but not both). A paper on this was presented at EuroMPI-'12 [34]. The new approach will be integrated into MPICH.

We continued to work on the development of lightweight thread support for MPICH. Work early in the project was performed in collaboration with both Argonne and the IBM Blue Gene team [10]. Work on fine grain multithreading support showed how to avoid excessive lock overhead in an MPI implementation [3, 2]. Recent work included a new algorithm for efficient allocation of context ids in MPI fixes a subtle race condition in the algorithm that had been used in MPICH; this new algorithm retains the efficient behavior for the expected case [9]. A previously submitted paper on test suites for thread safety in MPI was published [28].

We continued interaction with Argonne and Utah on the use of formal methods in MPI correctness. Some of these results appeared in [27, 33, 21]. A related project looked at performance correctness and was presented in [7]. A paper intended for a more general audience was accepted and appeared in the Communications of the ACM [12].

The use of performance modeling to better understand scaling issues has become an important part of the project; some issues, including ones relating to MPI, were discussed in [18], which was presented at SC'11. Previous results, for I/O [16] and for performance correctness [31, 19], have appeared, and a "Best Paper" at EuroMPI'10 on load balancing in hybrid MPI-OpenMP programs in [20].

We have collaborated on the enhancement of implementations of the MPI topology functions. These give the computational scientist a portable way to map an MPI program onto a large parallel machine in a way that provides for efficient (low contention) communication. Some of this work was reported on in [22]. Some related work looked at hot spots in two-level direct networks, such as those for the IBM PERCS system, which was originally to be installed at Illinois. This network has a number of advantages, and other supercomputer vendors are considering it as an alternative to mesh interconnects. This work was reported at SC'11 [6].

There is evidence that MPI datatypes can provide a significant benefit, especially for systems (such as all proposed Exascale system designs to date) where memory motion is relatively expensive. One effort looked at exporting the MPI datatype capabilities to other programming systems [23]. In collaboration with other researchers, we examined the implications of adding performance requirements on MPI implementations for the use of datatypes in [14]; this is important in ensuring that users of MPI can count on reasonable behavior from their MPI implementations. Following up on this work, we have begun developing datatype optimization approaches that extend "just in time" compilation techniques to MPICH.

A number of steps were undertaken to improve MPICH as both a research and production vehicle. For example, we enhanced the coverage analysis tool that was developed for MPICH. This has been increasingly important, as enhancements to MPICH for scalability sometimes create code paths that are not rigorously tested by the current test suites. The coverage tool helps us identify these issues. Our design is documented and available on the MPICH wiki. We also designed and implemented a low-overhead and flexible instrumentation interface. This interface provides basic support for the MPI-3 MPIT interface, and we used it to tune the MPICH RMA implementation, as described above.

Since one motivation for this work is to also look at models that might supplement or replace MPI, we participated in a workshop on PGAS collectives that also explored hybrid programming models that mixed MPI with UPC or CoArray Fortran. Some of these results were presented in [15].

Because this project is so closely coordinated with the Argonne MPICH group, this project shares the MPICH web site www.mpich.org , and uses this web site to communicate design discussions and results.

Publications and Talks

A number of publications were submitted that included some work supported under this project and have been detailed above. Each of these was presented at a conference or published in a journal.

References

[1] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, J. Krishna, E. Lusk and R. Thakur, *PMI: A Scalable Parallel Process-Management Interface for Extreme-Scale Systems*, in R. Keller, E. Gabriel, M. Resch and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2010, pp. 31-41.

[2] P. Balaji, D. Buntinas, D. Goodell, W. Gropp and R. Thakur, *Fine-Grained Multithreading Support for Hybrid Threaded MPI Programming*, International Journal of High Performance Computing Applications, 24 (2010), pp. 49-57.

[3] P. Balaji, D. Buntinas, D. Goodell, W. Gropp and R. Thakur, *Toward Efficient Support for Multithreaded MPI Communication*, *PVM/MPI*, Springer, 2008, pp. 120-129.

[4] P. Balaji, A. Chan, W. Gropp, R. Thakur and E. Lusk, *The Importance of Non-Data-Communication Overheads in MPI*, International Journal of High Performance Computing Applications, 24 (2010), pp. 5-15.

[5] P. Balaji, A. Chan, W. Gropp, R. Thakur and E. L. Lusk, *Non-data-communication Overheads in MPI: Analysis on Blue Gene/P*, *PVM/MPI*, Springer, 2008, pp. 13-22.

[6] A. Bhatele, N. Jain, W. D. Gropp and L. V. Kale, *Avoiding hot-spots on two-level direct networks*, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, Seattle, Washington, 2011.

[7] S. Byna, Y. Chen, W. D. Gropp, X.-H. Sun and R. Thakur, *Parallel I/O Prefetching Using MPI File Caching and I/O Signatures*, *Proceedings of SC08*, IEEE and ACM, 2008.

[8] A. Chan, P. Balaji, W. Gropp and R. Thakur, *Communication Analysis of Parallel 3D FFT for Flat Cartesian Meshes on Large Blue Gene Systems*, *15th IEEE International Conference on High Performance Computing*, 2008.

[9] J. Dinan, D. Goodell, W. Gropp, R. Thakur and P. Balaji, *Efficient Multithreaded Context ID Allocation in MPI*, *Recent Advances in the Message Passing Interface*, 2012, pp. 57-66.

[10] G. Dózsa, S. Kumar, P. Balaji, D. Buntinas, D. Goodell, W. Gropp, J. Ratterman and R. Thakur, *Enabling Concurrent Multithreaded MPI Communication on Multicore Petascale Systems*, in R. Keller, E. Gabriel, M. Resch and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2010, pp. 11-20.

[11] D. Goodell, W. Gropp, X. Zhao and R. Thakur, *Scalable Memory Use in MPI: A Case Study with MPICH2*, in Y. Cotronis, A. Danalis, D. Nikolopoulos and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2011, pp. 140-149.

[12] G. Gopalakrishnan, R. M. Kirby, S. Siegel, R. Thakur, W. Gropp, E. Lusk, B. R. D. Supinski, M. Schulz and G. Bronevetsky, *Formal analysis of MPI-based parallel programs*, Commun. ACM, 54 (2011), pp. 82-91.

[13] W. Gropp, *MPI at Exascale: Challenges for Data Structures and Algorithms*, in M. Ropo, J. Westerholm and J. Dongarra, eds., *Recent Advances in Parallel*

Virtual Machine and Message Passing Interface, Springer Berlin / Heidelberg, 2009, pp. 3-3.

[14] W. Gropp, T. Hoefler, R. Thakur and J. Traeff, *Performance Expectations and Guidelines for MPI Derived Datatypes*, in Y. Cotronis, A. Danalis, D. Nikolopoulos and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2011, pp. 150-159.

[15] W. D. Gropp, *MPI and Hybrid Programming Models for Petascale Computing*, *PVM/MPI*, Springer, 2008, pp. 6-7.

[16] W. D. Gropp, D. Kimpe, R. Ross, R. Thakur and J. L. Träff, *Self-consistent MPI-IO Performance Requirements and Expectations*, *PVM/MPI*, Springer, 2008, pp. 167-176.

[17] T. Hoefler, J. Dinan, D. Buntinas, P. Balaji, B. Barrett, R. Brightwell, W. Gropp, V. Kale and R. Thakur, *Leveraging MPI's One-Sided Communication Interface for Shared-Memory Programming*, *Recent Advances in the Message Passing Interface*, 2012, pp. 132-141.

[18] T. Hoefler, W. Gropp, W. Kramer and M. Snir, *Performance modeling for systematic performance tuning*, *State of the Practice Reports*, ACM, Seattle, Washington, 2011.

[19] T. Hoefler, W. Gropp, R. Thakur and J. Träff, *Toward Performance Models of MPI Implementations for Understanding Application Scaling Issues*, in R. Keller, E. Gabriel, M. Resch and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2010, pp. 21-30.

[20] V. Kale and W. Gropp, *Load Balancing for Regular Meshes on SMPs with MPI*, in R. Keller, E. Gabriel, M. Resch and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2010, pp. 229-238.

[21] S. Pervez, G. Gopalakrishnan, R. M. Kirby, R. Thakur and W. Gropp, *Formal methods applied to high-performance computing software design: a case study of MPI one-sided communication-based locking*, Software: Practice and Experience, 40 (2010), pp. 23-43.

[22] M. Rashti, J. Green, P. Balaji, A. Afsahi and W. Gropp, *Multi-core and Network Aware MPI Topology Functions*, in Y. Cotronis, A. Danalis, D. Nikolopoulos and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2011, pp. 50-60.

[23] R. Ross, R. Latham, W. Gropp, E. Lusk and R. Thakur, *Processing MPI Datatypes Outside MPI*, in M. Ropo, J. Westerholm and J. Dongarra, eds., *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer Berlin / Heidelberg, 2009, pp. 42-53.

[24] P. Sack and W. Gropp, *Faster topology-aware collective algorithms through non-minimal communication*, ACM, 2012, pp. 45-54.

[25] P. Sack and W. Gropp, *A Scalable MPI_Comm_split Algorithm for Exascale Computing*, in R. Keller, E. Gabriel, M. Resch and J. Dongarra, eds., *Recent Advances in the Message Passing Interface*, Springer Berlin / Heidelberg, 2010, pp. 1-10.

[26] G. Santhanaraman, *Natively Supporting True One-Sided Communication in*, in P. Balaji, K. Gopalakrishnan, R. Thakur, W. Gropp and D. K. Panda, eds., 2009, pp. 380-387.

[27] S. Sharma, S. S. Vakkalanka, G. Gopalakrishnan, R. M. Kirby, R. Thakur and W. Gropp, *A Formal Approach to Detect Functionally Irrelevant Barriers in MPI Programs*, *PVM/MPI*, Springer, 2008, pp. 265-273.

[28] R. Thakur and W. Gropp, *Test suite for evaluating performance of multithreaded MPI communication*, Parallel Computing, 35 (2009), pp. 608-617.

[29] V. Tipparaju, W. Gropp, H. Ritzdorf, R. Thakur and J. L. Traff, *Investigating High Performance RMA Interfaces for the MPI-3 Standard*, 2009, pp. 293-300.

[30] J. L. Traeff, A. Ripke, C. Siebert, P. Balaji, R. Thakur and W. Gropp, *A Pipelined Algorithm for Large, Irregular All-Gather Problems*, International Journal of High Performance Computing Applications, 24 (2010), pp. 58-68.

[31] J. L. Träff, W. Gropp and R. Thakur, *Self-consistent MPI Performance Requirements*, *PVM/MPI*, Springer, 2007, pp. 36-45.

[32] J. L. Träff, A. Ripke, C. Siebert, P. Balaji, R. Thakur and W. Gropp, *A Simple, Pipelined Algorithm for Large, Irregular All-gather Problems*, *PVM/MPI*, Springer, 2008, pp. 84-93.

[33] S. S. Vakkalanka, M. Delisi, G. Gopalakrishnan, R. M. Kirby, R. Thakur and W. Gropp, *Implementing Efficient Dynamic Formal Verification Methods for MPI Programs*, *PVM/MPI*, Springer, 2008, pp. 248-256.

[34] X. Zhao, G. Santhanaraman and W. Gropp, *Adaptive strategy for one-sided communication in MPICH2*, *Recent Advances in the Message Passing Interface*, 2012, pp. 16-26.

[35] H. Zhu, D. Goodell, W. Gropp and R. Thakur, *Hierarchical Collectives in MPICH2*, in M. Ropo, J. Westerholm and J. Dongarra, eds., *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer Berlin / Heidelberg, 2009, pp. 325-326.