Evolving MPI to Address the Challenges of Exascale Systems

Progress Report (Sept 2012 – Mar 2013)

Rajeev Thakur (PI), Pavan Balaji, Jim Dinan, Dave Goodell, Rusty Lusk, Marc Snir
Mathematics and Computer Science Division
Argonne National Laboratory
{thakur, balaji, dinan, goodell, lusk, snir} @mcs.anl.gov

http://www.mpich.org
http://collab.mcs.anl.gov/display/mpiexascale

Project Goals

Most parallel scientific applications running in production today primarily follow a distributed-memory parallel programming model and use MPI as the standard interface for communication between processes. These application codes represent billions of dollars worth of investment and run on the largest systems in the world, which currently have as many as 1.6 million cores.

As we progress from petascale to exascale systems over the course of this decade, it is essential that MPI must evolve to run as efficiently as possible on future systems, so that applications can continue to gain the benefits of the performance offered by these systems. This requires that both the MPI standard as well as MPI implementations address the challenges posed by the architectural features, limitations, and constraints expected in future extreme-scale systems. This project aims to address these challenges based on our long-term expertise in all aspects related to MPI standardization and implementation. Specific goals of the project fall into three categories:

1. Continued enhancement of the MPI standard through the MPI Forum by leading several of its subcommittees to ensure that the standard evolves to meet the needs of future systems and also of applications, libraries, and higher-level languages.

2. Continued enhancement of the MPICH implementation of MPI to support the new features in future versions of the MPI standard (MPI-3 and beyond) and to address the specific challenges posed by exascale architectures, such as lower memory per core, higher thread concurrency, lower power consumption, scalability, and resilience.

3. Investigation of new programming approaches to be potentially included in future versions of the MPI standard, including generalized user-defined callbacks, lightweight tasking, and extensions for heterogeneous computing systems and accelerators.

Technical Progress

Below we describe our recent progress in meeting the goals of this project.

MPI Standardization We were active in the definition of the MPI-3 standard, which the MPI Forum successfully defined and released in September 2013 after three years of work. MPI-3 adds several new features to the MPI specification including significant extensions to the one-sided communication interface, nonblocking versions of all collective communication functions, neighborhood collectives, an interface for tools to access information internal to an MPI implementation, and bindings for Fortran 2008. Other major features, such as fault tolerance and improved support for hybrid programming, were not ready in time for inclusion in MPI-3 but are actively being discussed for inclusion in a future version of MPI.

Several members of this project played leading roles in the definition of MPI-3. Rajeev Thakur was co-chair of the RMA (remote memory access or one-sided communication) working group; Pavan Balaji chaired the hybrid programming working group, Marc Snir was the author of the main proposal in the hybrid programming working group, Darius Buntinas was active in the fault tolerance effort; Dave Goodell played a prominent role in the definition of the new tools interface in the tools working group, and Jim Dinan was active in the RMA and hybrid working groups.

The MPI Forum continues to meet every three months to define future versions of MPI (MPI 3.x, MPI 4.0), and we continue to be actively involved in those efforts.

MPI Implementation The MPI implementation that we develop, MPICH, has always closely tracked the evolving MPI standard, so that users are able to experiment with the new features, and vendors get an early start on incorporating the new features in their own implementations and make them available on production systems. For MPI-3, we set an aggressive goal of implementing all of MPI-3 before the SC12 conference, i.e., barely a month and a half after the standard was released. Thanks to the heroic efforts of our project members, we were successful in releasing at the MPICH BoF at SC12 an all-new version of MPICH, MPICH 3.0, that supports all of MPI-3. At the BoF, we also unveiled a new, redesigned web site for MPICH, www.mpich.org. Although this version of MPICH supports all of MPI-3, performance tuning is needed in many parts, which we continually work on.

Vendor Interaction We continue to collaborate with our vendor partners to ensure that they have a running start toward supporting a full MPI-3 implementation on their platforms. IBM has decided to merge its two separate MPI implementation efforts for the Blue Gene and POWER platforms into a single implementation based on MPICH. We have been working with them closely and share a common code base. They regularly send us code patches that we incorporate. We similarly work closely with Cray on MPI for the Cray systems and with Intel for MPI on Intel platforms. As a result, the majority of the largest machines in the Top500 list run MPICH. For example, seven of the top ten machines in the November 2012 Top500 list use MPICH. We are also working with Fujitsu and the University of Tokyo to port MPICH to the K computer.

One-Sided Communication MPI-3 has added significant new features for one-sided communication, which make it useful for implementing high-level programming models, libraries, and applications. In addition to supporting all these new features in MPICH 3.0, we have published papers on how to implement them efficiently and on using MPI for shared memory programming within a node (MPI+MPI) [2, 4, 5].

Active Messages in MPI MPI does not directly support active messages; nonetheless, they are useful for implementing higher-level programming models, such as PGAS and Charm++. Together with Xin Zhao and Bill Gropp at UIUC, we investigated approaches for supporting active messages within the context of MPI. This work was accepted for publication at CCGrid 2013 [3].

Tools Interface The new MPI-3 tools interface, commonly known as the "MPI_T" interface, has been completely implemented in MPICH. The new MPI_T interface exposes two different kinds of information to tools and applications: control variables and performance variables. All internal configuration in MPICH that was previously controlled exclusively by UNIX environment variables is now also accessible programmatically through MPI_T control variables. Examples include thresholds for selecting different algorithms for collective communication as well as options for runtime debugging. Through this interface, we have also exposed several new performance variables, primarily vending statistics from MPI message-matching queues and low-level communication data structures.

We have already used these new performance variables to great effect for studying the performance of NAMD/Charm++ implemented over MPI. A publication based on this exercise is under preparation.

Applications We interact closely with applications to enable them to use advanced functionality in MPI (either directly or through domain-specific models). We recently demonstrated performance improvements with large-scale computations in various domains including chemistry, biology, and nuclear physics.

Publications

1. James Dinan, David Goodell, William Gropp, Rajeev Thakur, and Pavan Balaji, "Efficient Multithreaded Context ID Allocation in MPI," in *Proceedings of the 19th European MPI Users' Group Meeting (EuroMPI 2012)*, September 2012.

2. Torsten Hoefler, James Dinan, Darius Buntinas, Pavan Balaji, Brian Barrett, Ron Brightwell, William Gropp, Vivek Kale, and Rajeev Thakur, "Leveraging MPI's One-Sided Communication Interface for Shared-Memory Programming," in *Proceedings of the 19th European MPI Users' Group Meeting (EuroMPI 2012)*, September 2012.

3. Xin Zhao, Darius Buntinas, Judicael Zounmevo, James Dinan, David Goodell, Pavan Balaji, Rajeev Thakur, Ahmad Afsahi, and William Gropp, "Towards Asynchronous, MPI-Interoperable Active Messages," in *Proceedings of the 13th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2013)*, May 2013.

4. Torsten Hoefler, James Dinan, Darius Buntinas, Pavan Balaji, Brian Barrett, Ron Brightwell, William Gropp, Vivek Kale, and Rajeev Thakur, "MPI+MPI: A New, Hybrid Approach to Parallel Programming with MPI Plus Shared Memory Computing," *Computing*, 2013. (accepted)

5. James Dinan, Pavan Balaji, Darius Buntinas, David Goodell, William Gropp, and Rajeev Thakur, "An Implementation and Evaluation of the MPI 3.0 One-Sided Communication Interface," submitted to *Concurrency and Computation: Practice and Experience*, December 2012.

Tutorials

1. William Gropp, Ewing Lusk, and Rajeev Thakur, "Advanced MPI," 19th European MPI Users' Group Meeting (EuroMPI 2012), Vienna, Austria, September 2012.

2. William Gropp, Ewing Lusk, and Robert Ross, and Rajeev Thakur, "Advanced MPI," SC12: International Conference on High Performance Computing, Networking, Storage, and Analysis, Salt Lake City, Utah, November 2012.

3. Pavan Balaji and Torsten Hoefler, "MPI for Dummies," 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), Shenzhen, China, February 2013.

4. Pavan Balaji and Torsten Hoefler, "Advanced Parallel Programming with MPI-1, MPI-2, and MPI-3," 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), Shenzhen, China, February 2013.

Talks

1. Rajeev Thakur, "Challenges for Communication Libraries and Runtime Systems at Exascale," Workshop on Clusters, Clouds, and Data for Scientific Computing, Dareizé, France, September 2012.