Evolving MPI to Address the Challenges of Exascale Systems

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

The vast majority of DOE's parallel scientific applications running on the largest HPC systems are written in a distributed-memory style using MPI as the standard interface for communication between processes. These application codes represent billions of dollars worth of investment. As we transition from today's petascale systems to exascale systems by the end of this decade, it is not clear what will be the right programming model for the future. However, until a viable alternative to MPI is available, and large DOE application codes have been ported to the new model, MPI must evolve to run as efficiently as possible on future systems. This situation requires that both the MPI standard and MPI implementations address the challenges posed by the architectural features, limitations, and constraints expected in future post-petascale and exascale systems.

The most critical issue is likely to be interoperability with intranode programming models with a high thread count. This requirement has implications both for the definition of the MPI standard itself (being considered now in the MPI Forum, in which we are major participants) and for MPI implementations. Other important issues, also impacting both the standard and its implementation, include scalability, performance, enhanced functionality based on application experience, and topics that become more significant as we move to the next generation of HPC architectures: memory utilization, power consumption, and resilience.

Our group at Argonne has been a leader in MPI from the beginning, including the MPI standardization effort; research into implementing MPI efficiently that has resulted in a large number of publications; and development of a high-performance, production-quality MPI implementation (MPICH) that has been adopted by leading vendors (IBM, Cray, Intel, Microsoft, Myricom) and runs on most of the largest machines in the world. This project continues the ongoing MPI-related research and development work at Argonne, with the overall goal of enabling MPI to run effectively at exascale. Specific goals of this 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.

We have close ties with various DOE applications that are targeted to scale to exascale, including the exascale codesign centers. We will work with these applications, particularly the mini-apps and skeleton codes from the codesign centers, to study the effectiveness of our MPI implementation and of the new features in the MPI standard. We will also continue our collaboration with vendors, particularly IBM, Cray, and Intel, to codesign MPICH such that it remains the leading implementation running on the fastest machines in the world.

Recent Accomplishments

MPI-3 Standard and Implementation The MPI-3 Standard was released in September 2012, and we have been actively involved in its definition for the past three years. 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.

The MPICH implementation has closely tracked the evolution of the MPI standard for a long time, and our goal was to be the first implementation to support MPI-3. We successfully developed and released at SC12 a major new version of MPICH (3.0) that supports all of MPI-3. (Although the implementation is functionally complete, performance tuning of many parts remains, which we continually work on.)

Collaboration with Vendors 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. We similarly work closely with Cray on MPI for the Cray systems and with Intel on MPI for 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.

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.

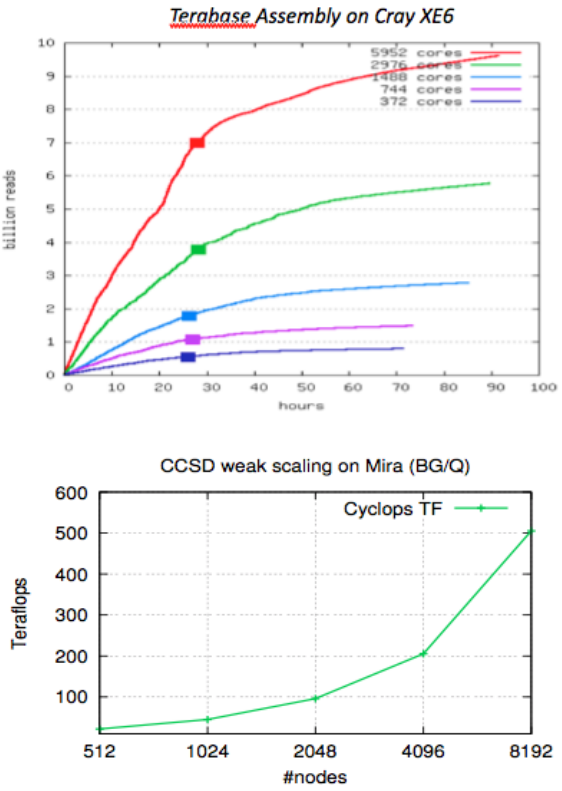

Figure 1: Application success stories: Genome assembly and Cyclops Tensor Framework (chemistry)

One-Sided Communication MPI-3's 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).

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 massive-scale computations in various domains including chemistry, biology, and nuclear physics (see Figure 1 for example).