Evolving MPI to Address the Challenges of Exascale Systems

Rajeev Thakur

Deputy Director

Mathematics and Computer Science Division

Argonne National Laboratory

Joint work with Pavan Balaji, Darius Buntinas, Jim Dinan, Dave Goodell, Bill Gropp, Rusty Lusk, Marc Snir

Current Situation with Production Applications (1)

- The vast majority of DOE's production parallel scientific applications today use MPI
  - Increasing number use (MPI + OpenMP) hybrid
  - Some exploring (MPI + accelerator) hybrid

- Today's largest systems in terms of number of regular cores (excluding GPU cores)

Sequoia (LLNL)	1,572,864 cores
Mira (ANL)	786,432 cores
K computer	705,024 cores
Jülich BG/Q	393,216 cores
Blue Waters	386,816 cores
Titan (ORNL)	299,008 cores

- **MPI already runs in production on systems with up to 1.6 million cores**

Current Situation with Production Applications (2)

- IBM has successfully scaled the LAMMPS application to over 3 million MPI ranks
- Applications are running at scale on LLNL's Sequoia and achieving 12 to 14 petaflops *sustained* performance
- HACC cosmology code from Argonne (Salman Habib) achieved **14 petaflops** on Sequoia
  - Ran on full Sequoia system using MPI + OpenMP hybrid
  - Used 16 MPI ranks * 4 OpenMP threads on each node, which matches the hardware architecture: 16 cores per node with 4 hardware threads each
  - **~ 6.3 million way concurrency: 1,572,864 MPI ranks * 4 threads/rank**
  - http://www.hpcwire.com/hpcwire/2012-11-29/sequoia_supercomputer_runs_cosmology_code_at_14_petaflops.html
  - SC12 Gordon Bell prize finalist

Current Situation with Production Applications (3)

- Cardioid cardiac modeling code (IBM & LLNL) achieved **12 petaflops** on Sequoia
  - Models a beating human heart at near-cellular resolution
  - Ran at scale on full system (96 racks)
  - Used MPI + threads hybrid: 1 MPI rank per node and 64 threads
  - OpenMP was used for thread creation only; all other thread choreography and synchronization used custom code, not OpenMP pragmas
  - http://nnsa.energy.gov/mediaroom/pressreleases/sequoia112812
  - SC12 Gordon Bell Prize finalist

- And there are other applications running at similar scales…

High-Level Goals of this Project

- Existing MPI applications developed over several years represent billions of dollars worth of investment

- As we progress from today's petascale to future exascale systems, MPI must evolve to run as efficiently as possible on these systems so that applications can continue to gain the performance benefits

- 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
  - E.g., lower memory per core, higher thread concurrency, power constraints, performance scalability, resilience to failures, …

Specific Goals of this Project (1)

- MPI Standardization
  - Work with the MPI Forum to ensure that the MPI *specification* evolves to meet the needs of future systems and of applications, libraries, and higher-level languages

- MPI Implementation
  - Continued enhancement of the MPICH implementation of MPI to support the new features in the MPI standard (MPI-3 and beyond) and to address the implementation challenges posed by exascale systems

- Transfer Technology to Vendors
  - Continue to collaborate with our vendor partners (IBM, Cray, Intel, …) to enable them to make the latest MPI features available to users on production systems

Specific Goals of this Project (2)

- Going Beyond Current MPI
  - Investigate new programming approaches for potential inclusion in future versions of the MPI standard
    - E.g., generalized user-defined callbacks, lightweight tasking, extensions for heterogeneous computing systems and accelerators, …
- Efficient Runtime for Implementing Higher-level Programming Models
  - Dynamic execution environments (e.g., Charm++, ADLB)
  - Global communication models (e.g., PGAS models, Global Arrays, GVR)
- Interoperability with other Programming Models
  - ***We are interested in working with you to ensure that the new models you are developing can interoperate with MPI***

Project Accomplishments

MPI Standardization

- The MPI Forum released the MPI-3 standard in Sept 2012 after about three years of effort

- Several of us played leading roles in the definition of MPI-3
  - ***Rajeev Thakur***: Co-chair of RMA working group
  - ***Pavan Balaji***: Chair of hybrid programming working group
  - ***Marc Snir***: Author of the main proposal in hybrid programming
  - ***Darius Buntinas***: Active role in fault tolerance working group
  - ***Dave Goodell***: Active role in defining the new tools interface in the tools working group
  - ***Jim Dinan***: Active in hybrid and RMA 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 since the beginning of MPI
- For MPI-3, we set an aggressive goal of implementing all of MPI-3 by SC12, 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 (3.0) that supports all of MPI-3
- 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 Interactions

- Continue to collaborate with our vendor partners to give them a running start toward supporting the latest MPI features on their platforms

- IBM has merged its 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 send us code patches that we incorporate.

- Similar interactions 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 run MPICH
  - ***Seven of the top ten machines in the Nov. 2012 Top500 list use MPICH exclusively***
  - One machine uses MPICH together with other MPI implementations
  - We are 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
  - (Details later in this talk)

- 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)
  - "Leveraging MPI's One-Sided Communication Interface for Shared-Memory Programming", EuroMPI 2012
  - "MPI+MPI: A New, Hybrid Approach to Parallel Programming with MPI Plus Shared Memory Computing", *Computing* (journal), to appear
  - "An Implementation and Evaluation of the MPI 3.0 One-Sided Communication Interface," submitted to *Concurrency and Computation: Practice and Experience*

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
  - "Towards Asynchronous, MPI-Interoperable Active Messages", CCGrid 2013

Tools Interface

- The new MPI-3 tools interface, commonly known as the "MPI_T" interface, has been completely implemented in MPICH

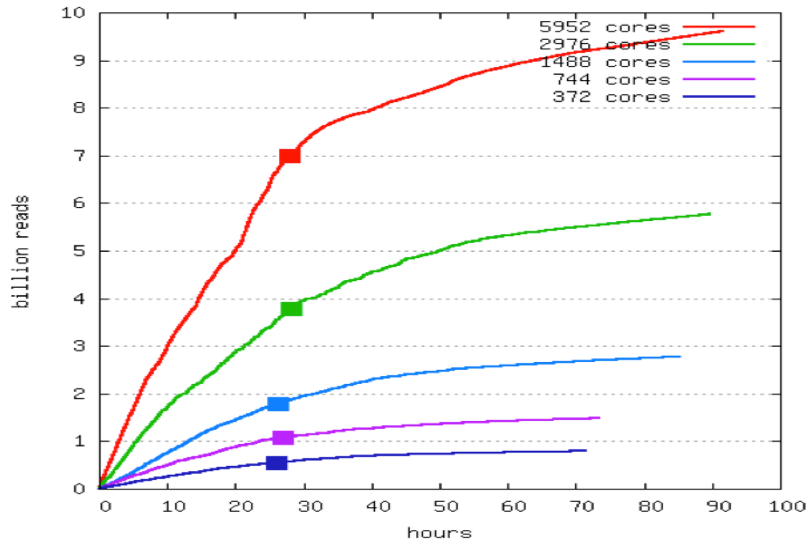
- All internal configuration in MPICH that was previously controlled exclusively by UNIX environment variables is now also accessible programmatically through MPI_T control variables
  - E.g., thresholds for selecting different algorithms for collective communication as well as options for runtime debugging

- Through MPI_T, we have also exposed several new performance variables, primarily vending statistics from MPI message-matching queues and low-level communication data structures

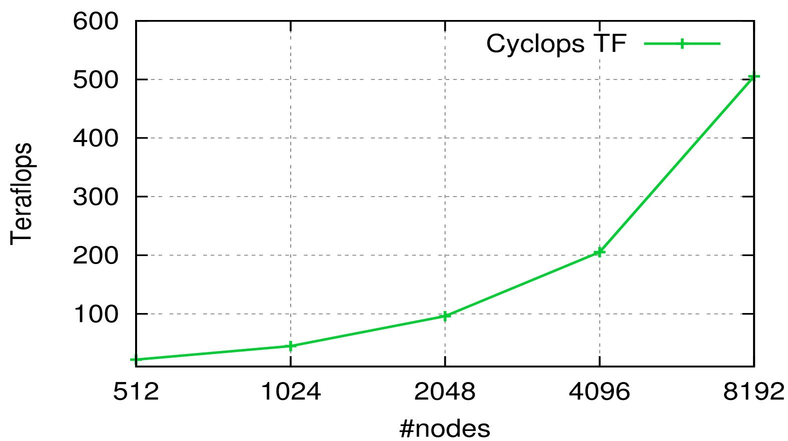
- We have used these new performance variables to great effect for studying the performance of NAMD/Charm++ implemented over MPI
  - a publication on this work is under preparation

Some Success Stories with Applications

Terabase Assembly on Cray XE6



CCSD weak scaling on Mira (BG/Q)



- **Terascale Genome Assembly**
  - Graph assembly problem that deals with finding an Hamiltonian path in a fuzzy graph with erroneous edges (or non-edges)
  - Highly communication intensive, with (seemingly) random global communication
  - Genome assembly at this scale (2.3TB on Cray XE6) has, for the first time, allowed scientists to study multiorganism genome colonies of completely or partially unknown species
- **Cyclops Tensor Framework (Chemistry)**
  - Fundamental component of quantum chemistry for coupled-cluster methods
  - Supersedes existing algorithms and software for parallel tensor contractions
  - Enabled quantum simulations of 250 electrons in 1000 orbitals (no point-group symmetry) on Argonne Mira
    - Order of magnitude larger scale problem than anything that has been previously done

What's New in MPI-3

MPI Standard Timeline

- MPI-1 (1994), presented at SC'93
  - Basic point-to-point communication, collectives, datatypes, etc
- MPI-2 (1997)
  - Added parallel I/O, RMA, dynamic processes, thread support, C++ bindings, …
- ---- Stable for 10 years ----
- MPI-2.1 (2008)
  - Minor clarifications and bug fixes to MPI-2
- MPI-2.2 (2009)
  - Small updates and additions to MPI 2.1
- MPI-3 (2012)
  - Major new features and additions to MPI

Overview of New Features in MPI-3

- Major new features
  - Nonblocking collectives
  - Neighborhood collectives
  - Improved one-sided communication interface
  - Tools interface
  - Fortran 2008 bindings
- Other new features
  - Matching Probe and Recv for thread-safe probe and receive
  - Noncollective communicator creation function
  - "const" correct C bindings
  - Comm_split_type function
  - Nonblocking Comm_dup
  - Type_create_hindexed_block function
- C++ bindings removed
- Previously deprecated functions removed

Nonblocking Collectives

- Nonblocking versions of all collective communication functions have been added
  - MPI_Ibcast, MPI_Ireduce, MPI_Iallreduce, etc.
  - There is even a nonblocking barrier, MPI_Ibarrier
- They return an MPI_Request object, similar to nonblocking point-to-point operations
- The user must call MPI_Test/MPI_Wait or their variants to complete the operation
- Multiple nonblocking collectives may be outstanding, but they must be called in the same order on all processes

Neighborhood Collectives

- New functions MPI_Neighbor_allgather, MPI_Neighbor_alltoall, and their variants define collective operations among a process and its neighbors
- Neighbors are defined by an MPI Cartesian or graph virtual process topology that must be previously set
- These functions are useful, for example, in stencil computations that require nearest-neighbor exchanges
- They also represent sparse all-to-many communication concisely, which is essential when running on many thousands of processes.
  - Do not require passing long vector arguments as in MPI_Alltoallv

Improved RMA Interface

- Substantial extensions to the MPI-2 RMA interface

- New window creation routines:
  - MPI_Win_allocate: MPI allocates the memory associated with the window (instead of the user passing allocated memory)
  - MPI_Win_create_dynamic: Creates a window without memory attached. User can dynamically attach and detach memory to/from the window by calling MPI_Win_attach and MPI_Win_detach
  - MPI_Win_allocate_shared: Creates a window of shared memory (within a node) that can be can be accessed simultaneously by direct load/store accesses as well as RMA ops

- New atomic read-modify-write operations
  - MPI_Get_accumulate
  - MPI_Fetch_and_op (simplified version of Get_accumulate)
  - MPI_Compare_and_swap

Improved RMA Interface contd.

- A new "unified memory model" in addition to the existing memory model, which is now called "separate memory model"
- The user can query (via MPI_Win_get_attr) if the implementation supports a unified memory model (e.g., on a cache-coherent system), and if so, the memory consistency semantics that the user must follow are greatly simplified.

- New versions of put, get, and accumulate that return an MPI_Request object (MPI_Rput, MPI_Rget, …)
- User can use any of the MPI_Test/Wait functions to check for local completion, without having to wait until the next RMA sync call

Tools Interface

- An extensive interface to allow tools (debuggers, performance analyzers, etc.) to portably extract information about MPI processes
- Enables the setting of various control variables within an MPI implementation, such as algorithmic cutoff parameters
  - e.g, eager v/s rendezvous thresholds
  - Switching between different algorithms for a collective communication operation
- Provides portable access to performance variables that can provide insight into internal performance information of the MPI implementation
  - e.g., length of unexpected message queue

- Note that each implementation defines its own performance and control variables; MPI does not define them

Fortran 2008 Bindings

- An additional set of bindings for the latest Fortran specification
- Supports full and better quality argument checking with individual handles
- Support for choice arguments, similar to (void *) in C
- Enables passing array subsections to nonblocking functions
- Optional ierr argument
- Fixes many other issues with the old Fortran 90 bindings

What did not make it into MPI-3

- There were some evolving proposals that did not make it into MPI-3
  - e.g., fault tolerance and improved support for hybrid programming
- This was because the Forum felt the proposals were not ready for inclusion in MPI-3
- These topics may be included in a future version of MPI

- Current activities of the MPI Forum (for MPI 3.x and MPI 4) can be tracked at http://meetings.mpi-forum.org/

Summary

- Different programming models have picked different tradeoffs in the space of portability, performance, expressiveness, and ease of use

- MPI as a runtime system has chosen to be highly feature rich and portable, and has enabled high-level libraries to be built on top of it to provide domain-specific algorithms and simplistic use of a subset of the features (e.g., PETSc, Trilinos, FFTW, ADLB, ...)

- This model has been extremely successful and has resulted in a wide and rich ecosystem built around MPI that includes high-level domain-specific libraries, performance and debugging tools, and applications in almost every domain of science

- ***We are interested in working with you to enable interoperability of your programming models with MPI***

Thanks!

- MPICH Leads
  - Argonne National Laboratory
  - University of Illinois, Urbana-Champaign
- Core MPICH developers
  - IBM
  - INRIA
  - Microsoft
  - Intel
  - University of British Columbia
  - Queen's University
- Derivative implementations
  - Cray
  - Myricom
  - Ohio State University
- Other Collaborators
  - Absoft, Pacific Northwest National Laboratory, Qlogic, Sandia, Totalview Technologies, University of Utah