Open MPI Is…

- Evolution of several prior MPI’s
- Open source project and community
  - Production quality
  - Vendor-friendly
  - Research- and academic-friendly
- MPI-2.1 compliant
Members, Contributors, Partners
Open MPI has 2 concurrent release series
- “Feature series” → v1.<odd>
- “Super stable series” → v1.<even>

Both are tested and QA’ed
- Main difference between the two is time
Development trunk

Time

Branch to create Feature series

v1.5
v1.5.1
v1.5.2

New features, enhancements

Transition to super stable

v1.6
v1.6.1

Bug fixes only

v1.7 / v1.8 branch
v1.6 Roadmap

• v1.6.3 is the current stable release
  ▪ Bug fixes only
  ▪ v1.6.4 will likely happen… someday

• We encourage all users to move away from the v1.4 series

• v1.6 updates are boring (!)
  ▪ … as they are intended to be
v1.7 Series

Brian Barrett
1.7 Goals

- MPI-3.0 compliance
- Better resource exhaustion resilience
- Thread safety
- Cray XE/XK/XC support
- Memory usage at scale
v1.7.0 Features

- Better Fortran bindings
- Java bindings
- Improved locality control infrastructure
- Improved run-time infrastructure
- New collectives
- MPI-3 features…
v1.7.0 MPI-3.0 Compliance

- Matched probe
- Nonblocking collective operations
  - Intercommunicators may slip to 1.7.1
- Version query
- MPI-3 Fortran support (f08 bindings)
- `MPI_TYPE_CREATE_HINDEXED_BLOCK`
- `MPI_COMM_SPLIT_TYPE`
- `MPI_INFO_ENV` support
MPI-3.0 Plans

• Features for 1.7.1
  § New one-sided interface (including shared memory windows)

• Work in progress
  § Non-blocking collectives
  § Non-blocking/non-collective communicator duplication
  § MPIT Tools interface
Hierarchical Collectives Software Layers - Cheetah
Barrier – Comparison with Native MPI

The figure compares the latency (microsec.) of different MPI systems across varying numbers of processor cores. The systems compared are:

- Native MPI
- Cray MPI
- Cheetah radix 3
- Cheetah radix 6

The graphs show the latency in microseconds as the number of processor cores increases. The trend lines indicate that as the number of cores increases, the latency also increases for all systems, with Native MPI generally showing the highest latency across the range of cores tested.
Large-Scale Broadcast Performance: OMPI vs Native MPI large message 16 MBytes
Aggregate Broadcast Bandwidth

- **HierKNEM**
- **OMPI-Hierarch**
- **OMPI-Tuned**

30x speedup!

(a) Ethernet Cluster (32 nodes)

(b) InfiniBand Cluster (32 nodes)

768 processes, 32 nodes, 24 cores/node
Aggregate Reduce Bandwidth

HierKNEM  OMPI-Hierarch  OMPI-Tuned

MPICH2 on Ethernet or MVAPICH2 on IB

3-10 X

(a) Ethernet Cluster (32nodes)  (b) InfiniBand Cluster (32nodes)

768 processes, 32 nodes, 24 cores/node
Insensitive to process mapping

Impact of process mapping: aggregate Broadcast and Allgather bandwidth of the collective modules for two different process-core bindings: by core and by node (Parapluie cluster, IB20G, 768 processes, 24 cores/node).
A helper for starting parallel applications
- Launch
- Connect
- Control
- I/O

Critical for the scalability and the resilience of any programming paradigm
Communication Infrastructure

Undirected graph \( G := (V, E), |V| = n \) (any size)

Node \( i = \{0, 1, 2, \ldots, n-1\} \) has links to a set of nodes \( U \)

\( U = \{i \pm 1, i \pm 2, \ldots, i \pm 2^k \mid 2^k < n\} \) in a circular space

Merging all links creates binomial graph from each node of the graph

Broadcast from any node in \( \log_2(n) \) steps

Stay connected in spite of failures
Runtime deployments

Building a BMG from the initial startup tree
From a tree to a ring
Scalability

• **Startup**
  - Gracefully handle many processes per node
  - Minimize resource consumption while maximizing parallelism: build specialized network overlays

• **Business card (Modex) exchange**
  - Use the network overlays to exchange the business cards of the participating processes
  - Keep one single copy per node shared between all local processes
  - Update the data asynchronously
Self-adapting algorithms to evolve from any type of spanning tree toward BMG

Good candidate for resilient runtime
Supported C/R strategies

**Coordinated C/R**
- A complete checkpoint is taken at specified time intervals
- In case of a failure all processes rollback to the last valid checkpoint
- The time to checkpoint **strongly** depends on the checkpoint support (I/O bandwidth)

**Uncoordinated C/R**
- A single checkpoint is taken at specified time intervals
- In case of a failure one process rollback to the last valid checkpoint
- The time to checkpoint **barely** depends on the checkpoint support (I/O bandwidth)
Correlated Set Coordinated Message Logging

- **Hybrid** between *coordinated* and *uncoordinated*
- **Codependent failures** are defined as sets of processes prone to fail simultaneously (cores of a same node)
- Codependent processes use coordinated checkpoint: relieves the need for expensive sender-based logging
- Non codependent processes are still uncoordinated and benefit from faster recovery
Correlated Set in Message Logging

Shared Memory NetPipe

Non deterministic events are still logged, but payload in a correlated set is not
 MPI Forum Fault Tolerance Working Group

Define a minimal set of semantics and interfaces to enable fault tolerant applications and libraries to be constructed portably

- User Level Failure Mitigation

- Prototype in Open MPI is guiding proposal development
  - http://fault-tolerance.org/
Cisco Ultra Low Latency Ethernet

- Cisco Ethernet Virtual Interface Card (VIC)
- “Userspace NIC” (USNIC) mode
  - OS bypass
  - Hardware offload
- Exports UD verbs interface
Cisco Ultra Low Latency Ethernet

- Back-to-back verbs latency
  - 1.7us HRT ping-pong

Diagram: Cisco VIC hardware connected to Cisco VIC hardware.
Cisco Ultra Low Latency Ethernet

- Back-to-back verbs latency
  - 1.7us HRT ping-pong
- Cisco’s lowest latency switch
  - 190ns port-to-port
Cisco Ultra Low Latency Ethernet

• Back-to-back verbs latency
  ▪ 1.7us HRT ping-pong
• Cisco’s lowest latency switch
  ▪ 190ns port-to-port
• Prototype Open MPI BTL plugin
  ▪ 300-400ns
• Total: ~2.2-2.3us
Mo’ Betta Fortran Bindings

- Revamped “F90” bindings support
  - use mpi
- Prototypes for all MPI subroutines
  - …but not for gfortran 😞
Mo’ Betta Fortran Bindings

- “F08” bindings (“use mpi_f08”)
  - New for MPI-3
- Many new features, including:
  - MPI handle type safety!
  - Type(MPI_Comm) :: my_comm
- Tested with:
  - Intel, Absoft, Portland compilers
  - …no gfortran support 😞 (YET)
Better Processor / Memory Affinity

• Uses Hardware Locality (hwloc)
  ▪ Sub-project of Open MPI

• Shameless plug:
Indexes: physical
Date: Tue Oct 30 09:29:46 2012

Hwloc of Sandy Bridge Server
Better Processor / Memory Affinity

• Probe nodes for topology at run-time
• Smallest unit of affinity is hyperthread
  ▪ “mpirun –bind-to-core” binds to all hyperthreads in a core
• “mpirun –report-bindings” much more readable
Better Processor / Memory Affinity

- Affinity is complicated!
- Evolving hardware architectures
  - Evolving application affinity needs
- Location Aware Mapping Algorithm
  - New / additional affinity options
  - v1.7.x (probably: x=1)
VampirTrace at Scale

- Last scalability limit for tracing $x \cdot 10^5$ procs: No HPC FS handles one file per process
- I/O Forwarding Scalability Layer (IOFSL)
  - Forwarding, buffering, aggregation, of I/O ops.
  - Map many logical files to few physical files on few IOFSL servers in “atomic append” mode
  - Open Source project, see http://www.iofsl.org/
- Full-system run on ORNL’s JaguarPF (XT5)
- In cooperation with ORNL and ANL
Trace of S3D combustion code with 200,448 procs on ORNL’s JaguarPF, recorded using 672 IOFSL servers (\(9.4 \cdot 10^{11}\) events, 4.2 TB compressed)
VampirTrace for MPI + CUDA

- VampirTrace supports MPI + CUDA
  - One/multiple CUDA devices per MPI process
  - API calls (host) and kernel executions (device)
  - GPU hardware performance counters
  - Host interactions (allocation, transfers, sync.)
  - NVIDIA‘s CUPTI tool interface
- Now also supports NVIDIA CARMA devices

Visit ZIH booth 4036, hall 2, win a NVIDIA Tesla K20 card!
Many more projects are occurring in the Open MPI community.
- clang compiler extensions
- MOSIX support
- ...

Come get involved!
Come Join Us!

http://www.open-mpi.org/