MPI 3.0

SciNet
www.scinet.utoronto.ca
University of Toronto
Toronto, Canada

October 22, 2013
Outline

1. MPI History
2. MPI Implementations
3. MPI 3.0 New Features
### Message Passing Interface (MPI)

**What is it?**

- An open standard library interface for message passing, ratified by the MPI Forum
- Version: 3.0 (2012)
# Message Passing Interface (MPI)

## What is it?

- An open standard library interface for message passing, ratified by the MPI Forum
- Version: 3.0 (2012)
Features added in MPI-2

- **Dynamic Processes** - extensions that remove the static process model of MPI. Provides routines to create new processes after job startup.

- **One-Sided Communications** - provides routines for one directional communications. Include shared memory operations (put/get) and remote accumulate operations.

- **Extended Collective Operations** - allows for the application of collective operations to inter-communicators.

- **External Interfaces** - defines routines that allow developers to layer on top of MPI, such as for debuggers and profilers.

- **Additional Language Bindings** - describes C++ bindings and discusses Fortran-90 issues.

- **Parallel I/O** - describes MPI support for parallel I/O.
New for MPI-3.0

- **Non-blocking Collectives** - Permits tasks in a collective to perform operations without blocking, possibly offering performance improvements.
- **Neighborhood Collectives** - Extends the distributed graph and Cartesian process topologies with additional communication power.
- **New One-Sided Functions and Semantics** - Better handle different memory models.
- **New Communicator Creation Functions** - New group-collective communicator creation.
New for MPI-3 - continued

- **Fault Tolerance/Resiliency** - Attempt at user-level failure notification.
- **MPI Tool interface** - Exposes certain internal variables, counters (primarily for performance tools).
- **Matched Probe** - Fixes a bug in MPI-2 where one could not probe for messages when using MPI and threads.
- **Large counts** - Added `MPI_COUNT`.
MPI Implementations

MPI Version Support

- MPI-3
  - MPICH 3.x (3.1b1) (no longer MPICH1 & MPICH2)
  - MVAPICH 2.0a (MPICH 3.0.4)
- MPI-2.2 + some MPI-3
  - OpenMPI 1.7.2, 1.7.3 and 1.9.x (svn)
  - MPICH2 1.5 (BGQ)
- MPI-2.2
  - IntelMPI 4.1

SciNet GPC

module load intel/13.1.1 use.experimental mvapich2
Many applications benefit from overlapping communication and computation using non-blocking MPI point-to-point operations.

i.e. `MPI_ISEND/MPI_IRECV` with `MPI_WAIT/MPI_TEST`
Non-blocking Collectives

Non-blocking Communication

- Many applications benefit from overlapping communication and computation using non-blocking MPI point-to-point operations.
- i.e. MPI_ISEND/MPI_IREDUCE with MPI_WAIT/MPI_TEST

Non-blocking Collectives

- Non-blocking versions of all collective operations
  - MPI_IBCAST, MPI_IBARRIER, MPI_IGATHER, MPI_IALLTOALL, etc.
- Can have multiple outstanding collectives on the same communicator.
Non-blocking Collectives

**MPI_IBARRIER**

- Sounds counter-intuitive, but can be useful.
  - Overlap barrier latency, and do other work while waiting.
  - Use the split semantics to processes notify non-collectively but synchronize collectively.

- Semantics:
  - **MPI_IBARRIER** - calling process enters the barrier, no synchronization happens.
  - Synchronization may happen asynchronously.
  - **MPI_TEST/MPI_WAIT** - synchronization happens if necessary.
Non-blocking Collectives

**MPI_IBARRIER**

- Sounds counter-intuitive, but can be useful.
  - Overlap barrier latency, and do other work while waiting.
  - Use the split semantics to processes notify non-collectively but synchronize collectively.

- Semantics:
  - MPI_IBARRIER - calling process enters the barrier, no synchronization happens
  - Synchronization may happen asynchronously
  - MPI_TEST/MPI_WAIT - synchronization happens if necessary

```c
MPI_Ibarrier(comm, request);
...
/* computation, other MPI communications */
...
MPI_Wait(request, status);
```
Non-blocking Collectives

Examples

- **Dynamic Sparse Data Exchange (DSDE)**
  - Dynamic: communication pattern varies across iterations
  - Sparse: number of neighbors is limited
  - Data exchange: only senders know neighbors

- **Parallel 3D FFT**
  - Traditionally implemented with MPI_ALLTOALL's
  - Subdivide into blocks and use MPI_IALLTOALL
Topology & Neighborhood Collectives

**MPI Topologies**

Specify application/algorithm communication topology via virtual topology creation functions (since MPI-1.0).

- **MPI_CART_CREATE** - a k-dimensional Cartesian application topology
- **MPI_DIST_GRAPH_CREATE** - scalable distributed graph

Many applications are written (compute, communicate, compute, ...)

High temporal locality in communication patterns!

Specify the communication pattern statically along a virtual topology

- **MPI_NEIGHBOR_ALLGATHER** - same buffer to all
- **MPI_NEIGHBOR_ALLTOALL** - specialized send buffer

Blocking and non-blocking variants.
## Topology & Neighborhood Collectives

### MPI Topologies

Specify application/algorithm communication topology via virtual topology creation functions (since MPI-1.0).

- **MPI_CART_CREATE** - a k-dimensional Cartesian application topology
- **MPI_DIST_GRAPH_CREATE** - scalable distributed graph

### Neighborhood Collectives (new for MPI-3.0)

- Many applications are written (compute, communicate, compute, ...)
  - High temporal locality in communication patterns!
- Specify the communication pattern statically along a virtual topology
  - **MPI_NEIGHBOR_ALLGATHER** - same buffer to all
  - **MPI_NEIGHBOR_ALLTOALL** - specialized send buffer
- Blocking and non-blocking variants.
### One-sided Communication - Remote Memory Access

- Allow one process to specify all communication parameters, both for the sending side and for the receiving side.
- Can be advantageous as avoids message matching overhead and reduce memory overhead.
- Separate communication and synchronization.
  - Allocate/Deallocate memory: `MPI_WIN_ALLOCATE, MPI_WIN_FREE`
  - Send/Receive: `MPI_PUT, MPI_GET`
- See Chapter 11.0 in MPI standard.
One-sided Communication in MPI

One-sided Communication - Remote Memory Access

- Allow one process to specify all communication parameters, both for the sending side and for the receiving side.
- Can be advantageous as avoids message matching overhead and reduce memory overhead.
- Separate communication and synchronization.
  - Allocate/Deallocate memory: MPI_WIN_ALLOCATE, MPI_WIN_FREE
  - Send/Receive: MPI_PUT, MPI_GET
- See Chapter 11.0 in MPI standard.

Status

- Initially implemented in MPI-2.0, good for non-coherent systems.
- Hard to use and slow on coherent systems.
One-sided Communication in MPI

New Features in MPI-3

- Improved one-sided semantics and extended operations.
- Dynamic window creation.
- Lightweight local and remote synchronization.
- Flush operations.
- Request-based operations.
Scalable Communicator Creation

Communicator Creation

- Creating a communicator in MPI-2 is an all-collective operation.
- Can lead to performance/scaling issues with many small groups of communicators.

Non-Collective Communicator Creation

Create communicators without involving all processes in the parent communicator.

Very useful for some applications, dynamic load balancing, fault tolerance.

Collective only in the members of the new communicator.

No unnecessary global synchronization.

Reduced overhead when creating small communicators.
Scalable Communicator Creation

**Communicator Creation**
- Creating a communicator in MPI-2 is an all-collective operation.
- Can lead to performance/scaling issues with many small groups of communicators.

**Non-Collective Communicator Creation**
- Create communicators without involving all processes in the parent communicator.
- Very useful for some applications, dynamic load balancing, fault tolerance.
- Collective only in the members of the new communicator.
- No unnecessary global synchronization.
- Reduced overhead when creating small communicators.
Fault Tolerance

- Application involved fault tolerance (not transparent, no magic)
  - Focus on user-level failure notification for Algorithm Based Fault Tolerance (ABFT)
  - Management through communicators
  - Requires a robust implementation
  - Still a work in progress
- FT modes
  - **Run-through stabilization** (MPI-3.0) - non-failed processes can continue to use MPI and can determine which ranks have failed
  - **Process recovery** (targeted for MPI-3.1) - replace the failed process in all existing communicators, windows and file handles
MPI Tool Interface

**MPI_T**

- Provide hooks for tools on MPI internal information
- Query and set internal MPI variables and counters
- Query internal state of the MPI library at runtime
- Design similar to PAPI counters
- Implementation agnostic
- Complements the existing PMPI interface
- Primarily for MPI performance tools (Scalasca, Vampir, Tau, etc.)
Matched Probe

**MPI-2.2**
- point-to-point communication is not thread safe!
  - Message probed in multiple threads but received in only one.
  - Leads to race conditions.
Matched Probe

MPI-2.2
- point-to-point communication is not thread safe!
  - Message probed in multiple threads but received in only one.
  - Leads to race conditions.

MPI-3.0
- Matched Probes and Receives
  - Fix returns a message handle from probe.
  - Receive this message only through the handle.
Matched Probe

MPI-2.2

- point-to-point communication is not thread safe!
  - Message probed in multiple threads but received in only one.
  - Leads to race conditions.

MPI-3.0

- Matched Probes and Receives
  - Fix returns a message handle from probe.
  - Receive this message only through the handle.

```c
MPI_Message msg;
MPI_Mprobe(...,msg, status)
size=get_count(status)*size_of(datatype)
buffer=malloc(size)
MPI_recv(buffer,...,msg,...)
```
Language Bindings

- New Fortran 2008
- Deprecated C++ (use C)
## Language Bindings

- New Fortran 2008
- Deprecated C++ (use C)

## Counts

- **MPI-2.2** All counts are `int / INTEGER`
- **MPI-3.0**
  - New “long” count type
  - Fortran: `INTEGER(KIND=MPI_COUNT_KIND)`
  - C: `typedef < some long type > MPI_Count`
  - No new communication routines
http://www.mpi-forum.org/
MPI: A Message-Passing Interface Standard V3.0
“New and old Features in MPI-3.0: The Past, the Standard, and the Future” - Torsten Hoefler
“MPI 3.0 An overview of the proposed features” - Hristo Iliev
“Non-Blocking Collective Operations for MPI-3” - Torsten Hoefler
“ADVANCED MPI 2.2 AND 3.0 TUTORIAL “ - Torsten Hoefler
“MPI 3 and Beyond: Why MPI is Successful and What Challenges it Faces” - William Gropp