A Preview of MPI 3.0: The Shape of Things to Come

20 Years of Excellence in Computational Science

OLCF

Oak Ridge Leadership Computing Facility

1992–2012

Manjunath Gorentla Venkata
manjugv@ornl.gov

Joshua Hursey
hurseyjj@ornl.gov (or jhursey@uwlax.edu)

U.S. DEPARTMENT OF ENERGY

Oak Ridge National Laboratory
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Overview of Seminar Series

- **Monday, June 25 - 3-4 pm:**
  - MPI Process (brief)
  - Timeline to 3.0
  - MPI 3.0 Fortran Bindings
  - MPI 2.2
- **Tuesday, June 26 - 3-4 pm**
  - Collectives:
    - Neighborhood
    - Nonblocking
  - Communicator Creation:
    - Noncollective
    - Nonblocking duplication
- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - **MPI_Comm_split_type()**
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

MPI_Comm_split_type

```
MPI_COMM_SPLIT_TYPE(comm, split_type, key, info, newcomm)
MPI Defines: MPI_COMM_TYPE_SHARED
```

New!

- **The 'split_type' specifies how to partition a communicator**
  - MPI Defines: `MPI_COMM_TYPE_SHARED`
    Split the communicator into subcommunicators, each of which can create a shared memory region.
  - Implementations can define additional types and/or use info argument (e.g., L2 cache, NUMA domain, I/O controller…)

- **What split_types would be useful to your application?**

MPI_Comm_split_type: Availability

- **Proposal: #287**
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/287
- **Open MPI**
  - MPI_COMM_TYPE_SHARED: All processes on the same "node"
  - **Available today**: Open MPI trunk
  - **Scheduled release**: Open MPI 1.7 (next feature series)
  - https://svn.open-mpi.org/trac/ompi/wiki/MPIConformance
- **MPICH2**
  - MPI_COMM_TYPE_SHARED: All processes on the same "node"
  - **Available today**: MPICH2 trunk, 1.5beta1
  - **Scheduled release**: MPICH2 1.5 (next release)

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - **MPI Matched Probe/Recv**
  - RMA / One-sided enhancements
  - Tool Interfaces
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

MPI Matched Probe/Recv

- **MPI_Probe/Recv cannot be used in a thread safe manner**
  - Probing for a message <u>does not</u> imply that a subsequent receive will actually receive that message.
  - Limits the ability to build some programming models on top of MPI
- **Need to couple the MPI_Probe with the following MPI_Recv**
  - New type: `MPI_Message`

```
MPI_Probe(MPI_ANY_SOURCE, 0, comm, &status);

MPI_Get_count(&status, MPI_BYTE, &len);
buf = malloc(len);

/* Thread B can jump in here an steal the message */

MPI_Recv(buf,len,MPI_BYTE,status.MPI_SOURCE,0,comm, MPI_STATUS_IGNORE);
```

Douglas Gregor, Torsten Hoefler, Brian Barrett, and Andrew Lumsdaine, *"Fixing Probe for Multi-Threaded MPI Applications*." Tech. Report, 2009. http://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR674

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

MPI Matched Probe/Recv

- **New Probe calls with an MPI_Message**
  - If successful, "keep" the message and store it in the MPI_Message
  - No other Probe/Recv can match this message except MRecv(msg)

```
MPI_IPROBE( source, tag, comm, flag,          status)
MPI_PROBE(  source, tag, comm,                status)

MPI_IMPROBE(source, tag, comm, flag, message, status)
MPI_MPROBE( source, tag, comm,       message, status)
```

New!

- **New Recv calls the reference an MPI_Message**
  - Receive only the MPI_Message previously probed

```
MPI_RECV(  buf, count, datatype, source, tag, comm, status )
MPI_IRECV( buf, count, datatype, source, tag, comm, request)

MPI_MRECV( buf, count, datatype,         message,   status )
MPI_IMRECV(buf, count, datatype,         message,   request)
```

New!

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

MPI Matched Probe/Recv

- Without Matched Probe/Recv : Not thread safe

```
MPI_Probe(MPI_ANY_SOURCE, 0, comm, &status);

MPI_Get_count(&status, MPI_BYTE, &len);
buf = malloc(len);

/* Thread B can jump in here an steal the message */
MPI_Recv(buf,len,MPI_BYTE,status.MPI_SOURCE,0,comm, MPI_STATUS_IGNORE);
```

- With Matched Probe/Recv : Thread safe

New!

```
MPI_Message msg;
MPI_Status status;

MPI_MProbe(MPI_ANY_SOURCE, 0, comm, &msg, &status);

MPI_Get_count(&status, MPI_BYTE, &len);
buf = malloc(len);

MPI_Recv(buf, len, MPI_BYTE, &msg, MPI_STATUS_IGNORE);
```


Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

MPI Matched Probe/Recv: Availability

- **Proposal: #38**
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/38
- **Open MPI**
  - **Available today**: Open MPI trunk
  - **Scheduled release**: Open MPI 1.7 (next feature series)
  - https://svn.open-mpi.org/trac/ompi/wiki/MPIConformance
- **MPICH2**
  - **Available today**: MPICH2 trunk, 1.5beta1
  - **Scheduled release**: MPICH2 1.5 (next release)

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - **RMA / One-sided enhancements**
  - Tool Interfaces
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

RMA/One-Sided Enhancements

- **Disclaimer:**
  - I am not an RMA-guy!
  - RMA semantics are oftentimes subtle for good performance reasons
    - A full seminar on just this topic is needed to really understand how to use the model
  - Here are some references for those that want more details:
    - **Ticket 270: Updated RMA Proposal**
      https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/270
    - **Overview with discussion of Bonachea's and Duell's critique**
      http://meetings.mpi-forum.org/secretary/2011/05/slides/rma-overview-5-11-4up.pdf
    - **Torsten Hoefler: CSCS 2012 Tutorial Slides**
      http://www.unixer.de/teaching/mpi_tutorials/cscs12/
- **I'll present a general overview & present some highlights**

RMA/One-Sided Enhancements: Terminology

- **Origin Process**: Process with the source buffer, initiates the operation
- **Target Process**: Process with the destination buffer, does not explicitly call communication functions
- **Epoch**: Virtual time where operations are in flight. Data is consistent after new epoch is started.
  - **Access Epoch**: Rank acts as origin for RMA calls
  - **Exposure Epoch**: Rank acts as target for RMA calls
- **Ordering**: Only for accumulate operations: order of messages between two processes (default: in order, but can be relaxed)
- **Assert**: Assertions about how the one-sided functions are used. Think of them as "fast" optimizations hints

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Creating a Window

- **Expose Consecutive Memory (memory allocated by user):**

```
MPI_WIN_CREATE(base, size, disp_unit, info, comm, win)
MPI_WIN_FREE(win) – This will –not- deallocate memory.
```

- **Expose Consecutive Memory (memory allocated by MPI):**

```
MPI_WIN_ALLOCATE(size, disp_unit, info, comm, baseptr, win)
MPI_WIN_FREE(win) – This will deallocate memory!
```

New!

  - This can improve the performance on systems with RDMA

- **Window of Dynamically Attached Memory (Dynamic Win.):**

```
MPI_WIN_CREATE_DYNAMIC(info, comm, win)
MPI_WIN_ATTACH(win, base, size)
MPI_WIN_DETACH(win, base)
```

New!

  - Irregular applications that need to expand the window size after creation
  - Allows registration of non-overlapping regions of memory locally

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Communication

- **All communication calls are <u>nonblocking</u>:**
  - Call initiates the transfer, but transfer may continue after the call returns
  - Transfer is completed when a synchronization call is issued
- **Put memory to a target and Get memory from a target**
  - Nonblocking, bulk completion at the end of the epoch

```
MPI_PUT(       origin_addr, origin_count, origin_datatype,
  target_rank, target_disp, target_count, target_datatype, win)
MPI_GET(       origin_addr, origin_count, origin_datatype,
  target_rank, target_disp, target_count, target_datatype, win)
```

- **Request Based Put/Get :**
  - Request used to query local completion (local buffer consistency)

```
MPI_RPUT(origin_addr, ..., target_rank, target_disp, ..., win, req)
MPI_RGET(origin_addr, ..., target_rank, target_disp, ..., win, req)
```

New!

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Accumulation

- **Remote Accumulations**
  - Accumulate origin into the target

```
MPI_ACCUMULATE( origin_addr, origin_count, origin_datatype,
   target_rank, target_disp, target_count, target_datatype, op, win)
MPI_RACCUMULATE(origin_addr, origin_count, origin_datatype,
   target_rank, target_disp, target_count, target_datatype, op, win,
   request)
```

New!

- **Remote Get and Accumulate**
  - Accumulate origin into the target, returns content before accumulation
  - Generalized fetch and add (use MPI_REPLACE for fetch & set)

New!

```
MPI_GET_ACCUMULATE( origin_addr, origin_count, origin_datatype,
                    result_addr, result_count, result_datatype,
       target_rank, target_disp, target_count, target_datatype, op,win)
MPI_RGET_ACCUMULATE(origin_addr, origin_count, origin_datatype,
                    result_addr, result_count, result_datatype,
       target_rank, target_disp, target_count, target_datatype, op,win,
       request)
```


Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Accumulation

- **Fetch and Op:**
  - Common use case: A single element fetch & op
  - Similar to MPI_Get_accumulate, but a more limited interface

```
MPI_FETCH_AND_OP(origin_addr, result_addr, datatype,
                 target_rank, target_disp, op, win)
```

New!

- **Compare and Swap:**
  - Compares the compare buffer with the target buffer
  - If compare and target are identical then replaces the value at target with origin.
  - Original target value is returned in result.

```
MPI_COMPARE_AND_SWAP(origin_addr, compare_addr, result_addr, datatype,
                     target_rank, target_disp, win)
```

New!

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Synchronization Modes

- **Active Target:**
  - Data moved from one process to another, and both are explicitly involved in the transfer.
- **Passive Target:**
  - Data moved from one process to another, and only the origin process is explicitly involved in the transfer.

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Synchronization: Active Target

- **MPI_WIN_FENCE**
  - All RMA calls started before fence will complete
  - Ends/starts access, and/or exposure epochs
- **Specify access/exposure epochs separately**
  - **Post**: Begin exposure epoch to group
  - **Start**: Begin access epoch to group
  - **Complete**: Finish access epoch (origin completion, not target)
  - **Wait**: Finish exposure epoch (completes at target)

```
MPI_WIN_FENCE(assert, win)
MPI_WIN_POST( group, assert, win) – exposure epoch
MPI_WIN_START(group, assert, win) – access epoch
MPI_WIN_COMPLETE(win)         – access epoch
MPI_WIN_WAIT(    win)         – exposure epoch
MPI_WIN_TEST(    win, flag) – exposure epoch
```

 Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Synchronization: Passive Target

- Lock/Unlock
  - Starts an access epoch of the type specified to a specific rank

```
MPI_WIN_LOCK(lock_type, rank, assert, win)
MPI_WIN_UNLOCK(         rank,         win)
```

- Lock_all/Unlock_all
  - Starts a shared access epoch from origin to all ranks (not collective)

New!

```
MPI_WIN_LOCK_ALL(assert, win)
MPI_WIN_UNLOCK_ALL(      win)
```

- Passive synchronization primitives
  - Can only be called within lock/unlock or lockall/unlockall epochs

New!

```
MPI_WIN_FLUSH(      rank, win)
MPI_WIN_FLUSH_LOCAL(rank, win)
MPI_WIN_FLUSH_ALL(        win)
MPI_WIN_FLUSH_LOCAL_ALL(win)
MPI_WIN_SYNC(win)
```


Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Synchronization: Passive Target

New!

- **MPI_WIN_FLUSH(rank, win)**
  - Completes all RMA operations with target rank at both origin and target
- **MPI_WIN_FLUSH_LOCAL(rank, win)**
  - Completes all RMA operations with target rank at origin
- **MPI_WIN_FLUSH_ALL(win)**
  - Completes all RMA operations with all ranks at both origin and target
- **MPI_WIN_FLUSH_LOCAL_ALL(win)**
  - Completes all RMA operations with all ranks at origin
- **MPI_WIN_SYNC(win)**
  - Synchronize private and public window copies (~memory barrier)

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: Shared Memory Windows

- **Allocate a shared memory segment in the window**
  - All processes in comm must be in shared memory MPI_Comm_split_type()!
  - Returns a pointer to the start of local rank's part of memory
  - Memory can be accessed with direct load/store instructions
  - Two allocation modes:
    - **Contiguous** (default): Process i's memory starts where process (i-1)'s memory ends
    - **Non-Contiguous** (info=alloc_shared_noncontig): Possible memory optimizations
  - Query operation to determine remote rank's memory location
    - Important for non-contiguous cases

```
MPI_WIN_ALLOCATE_SHARED(size, info, comm, baseptr, win)
MPI_WIN_SHARED_QUERY(win, rank, size, baseptr)
```

New!

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements: …

- **Two Memory Models**
  - **Unified:** public and private window are identical
  - **Separate:** public and private window are separate
- **See document and slides for more details**
  - **Ticket 270: Updated RMA Proposal**
    https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/270
  - **Torsten Hoefler: CSCS 2012 Tutorial Slides**
    http://www.unixer.de/teaching/mpi_tutorials/cscs12/

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL), Brian Barrett (SNL)

RMA/One-Sided Enhancements

- **Proposal: #270, #284**
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/270
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/284
- **Open MPI**
  - **Available today**: Open MPI branch (under development)
  - **Scheduled release**: Open MPI 1.7 (next feature series)
  - https://svn.open-mpi.org/trac/ompi/wiki/MPIConformance
- **MPICH2**
  - **Available today**: In development
  - **Scheduled release**: MPICH2 1.5 (next release)

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - **Tool Interfaces**
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

Tools Interface (MPI_T)
*New "Tools" Chapter in MPI 3.0

- **Implementation independent API to access (and potentially modify) internal MPI layer information**
  - All API routines prefixed with MPI_T_
- **Goals:**
  - **Provide access to MPI internal information**
    - Configuration and control information
    - Performance information
    - Debugging information
  - **Standardized access to this information** (build on success of PMPI)
  - **MPI_T is an MPI implementation agnostic specification**
    - No particular implementation model assumed
    - Ability to provide no/varying amount of information

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T) General Approach

- **Basic concept: Implementation exposes a set of named variables**
  - Set of variables and naming left to MPI implementation
  - MPI_T provides query functions to detect variables
  - Semantics of the variable are provided as clear text
  - Routines provided to read and write values of these variables
- **Split into performance and control variables**
  - **Performance**: Internal performance data (software counters in MPI)
    - Number of packets sent, time spent blocking, memory allocated, …
  - **Control**: Configuration information/environment variables
    - Eager limit, startup control, buffer sizes, buffer management, …

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T) General Approach

- **Mainly intended for tool development!**
  - Document environment (list all configuration variables)
  - Set configurations on particular platforms
- **Number of variables can change at runtime**
  - Implementations my load variables on-demand (lazy loading)
- **Mechanisms to write control variables**
  - Opportunities for (auto) tuning
  - Might be limited:
    - Some configurations cannot be changed
    - Some configurations are fixed after a certain point (e.g., MPI_Init)
    - Some configurations must be applied globally

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

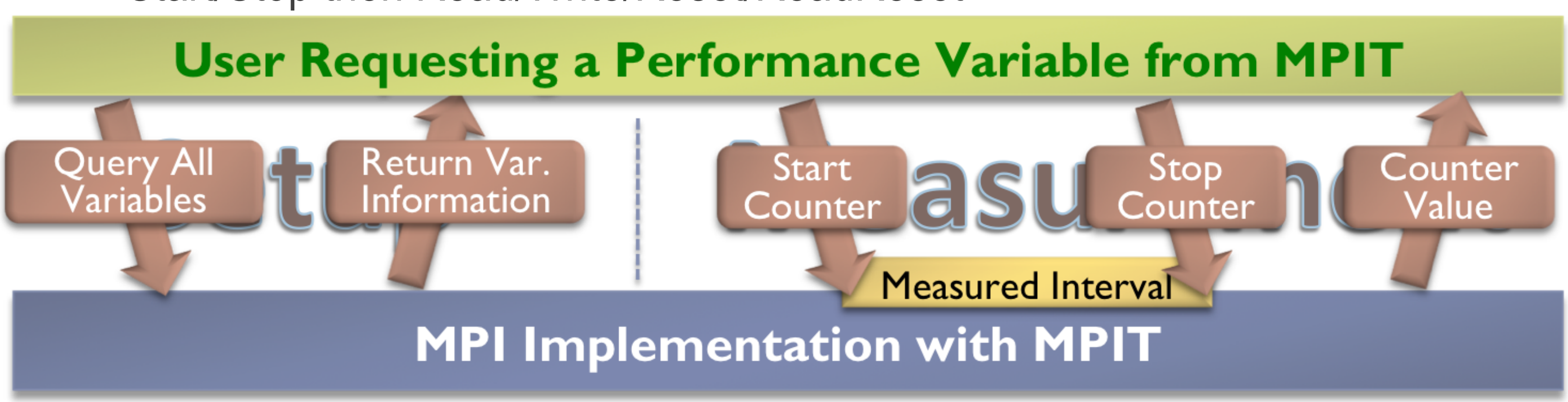
Tools Interface (MPI_T) General Approach

- **Binding variables to MPI Objects**
  - Message traffic on a given communicator
  - Remote accesses to a specific RMA window
  - I/O buffer setting for a particular MPI file

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T) General Approach

- **Access to internal performance variables**
  - Example: # of messages sent, # cycles waited, total memory allocated
  - Usage scenarios:
    - Calipers within a PMPI tool
    - Used within a signal handler for a sampling tool
  - Variables can be started and stopped, and accessed within "sessions"
    - Sessions: An object to provide isolation between multiple users of MPI_T
    - Start/Stop then Read/Write/Reset/ReadReset



Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T) Summary

- **Query interface to ask for provided variables**
  - Variables numbered from 0 to N-1
  - Routine to ask for N
  - Routine to ask for metadata for each variable
- **Handle allocation and free**
  - Enable access to a particular variable
  - Binds an MPI_T variable to an MPI object
- **Binding of variables**
  - Enables the restriction of a variable to a particular object
  - Instantiates the concrete variable in the context of the object
  - One variable can be bound to multiple objects

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T) Summary

- **Performance Variables:**
  - Allocate session
  - Allocate handle
  - Reset/Write variables
  - Start/Stop variables
  - Read/Readreset variables
  - Free handle & Free Session
- **Control Variables**
  - Allocate handle
  - Read/Write variable
    - Scoping to define to which ranks a configuration change must be applied to
  - Free handle

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Tools Interface (MPI_T)

- **Proposal: #266**
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/266
- **Open MPI**
  - **Available today**: In development
  - **Scheduled release**: Unknown
  - https://svn.open-mpi.org/trac/ompi/wiki/MPIConformance
- **MPICH2**
  - **Available today**: MPICH2 trunk, 1.5beta1
  - **Scheduled release**: MPICH2 1.5 (next release)
  - Some limitations – see release notes

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - **<u>Tool Interfaces (MPIR)</u>**
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

Process Acquisition Interface (MPIR)

- **Standard support for 3rd party tools (e.g., debuggers)**
  - Tools loaded independently from the application
- **Requirement:**
  - Where to find all MPI Processes?
  - How to attach or inspect them?
- **Typical work flow:**
  - Point debugger to this mechanism
  - Gather all host/PID information
  - Launch daemons on all hosts
  - Daemons use PID information to attach to all MPI processes
  - Central debugger controls MPI processes through daemons

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Process Acquisition Interface (MPIR)

- **MPIR: Process Acquisition Interface for MPI**
  - Not actually part of the MPI standard
    "The MPIR Process Acquisition Interface, Version 1.0" – side document
  - Established as the de-facto standard
  - Implemented by all major MPI version
- **Components:**
  - Handshake protocol to gain control over MPI processes
    - Support for both launch and attach cases
  - Access to a process table listing all MPI processes in a job
- **Limitations (plans for a MPIR-2 in the future)**
  - MPI process table is static, monolithic data structure
  - Support for fault tolerance unclear

Thanks to Torsten Hoefler (ETH Zürich), Martin Schulz (LLNL)

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - **MPI <next>**
    - Fault Tolerance
    - Hybrid, collectives, …

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - **MPI <next>**
    - **Fault Tolerance**
    - Hybrid, collectives, …

Fault Tolerance (#323)
User-Level Failure Mitigation (ULFM RTS)

- **Fault tolerance is important to HPC applications**
  - Large scale and long runtimes lead to increased opportunity for failure to disrupt the application (MTTI, MTBF, …)
  - Projected that process failure will become a normal event in the future
  - C/R techniques alone will not be enough to handle the rate of failure
    - Natural & Algorithm Based Fault Tolerance (ABFT)
      e.g., checksums stored in peers, rewinding computation, redundant computation
- **Entire HPC software stack lacks support for portable, fault tolerant applications.**

International Exascale Software Project (IESP):

Fault Tolerant MPI (2012) → Applications (2016)

Dongarra, J., Beckman, P., et al., "*The International Exascale Software Roadmap*," International Journal of High Performance Computer Applications, 2011 (to appear).

Algorithm-Based Fault Tolerance (ABFT) Techniques

- **Faulty Subgroups**
  - Ensemble-style applications
  - Extensive reliance on error handlers
- **Recovery Blocks**
  - Iterative applications
  - Execution block followed by an acceptance test
- **Linear Algebra Libraries**
  - Encapsulate fault tolerant versions of commonly used linear algebra operations.
  - FT-LA project to support ScaLAPACK

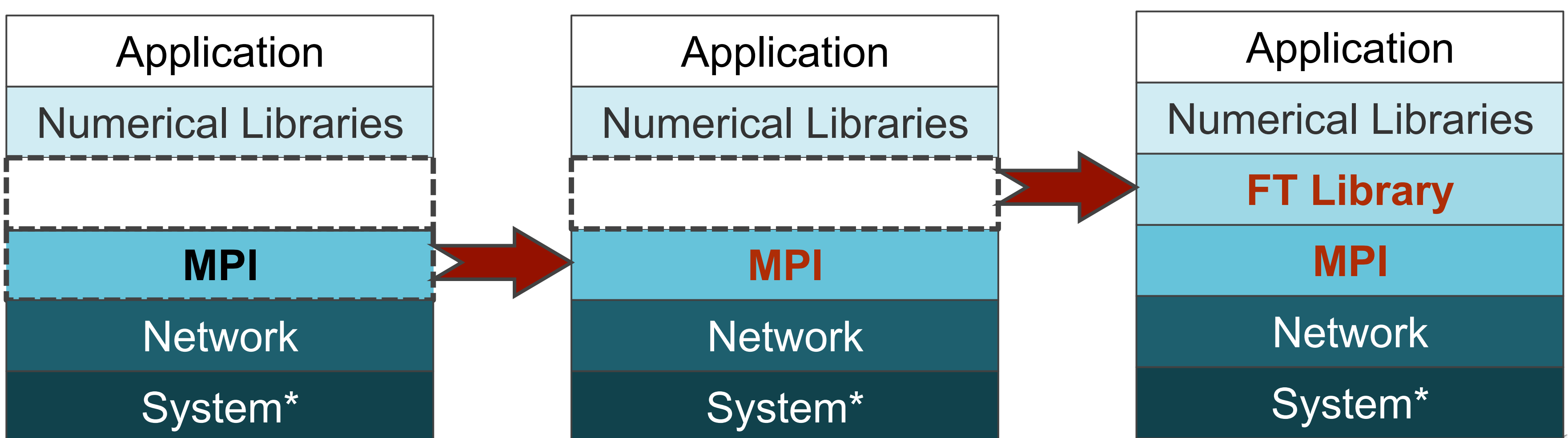
```
int rc, allsucceeded;

// Recovery Block
  rc = MPI_Allreduce( ..., comm );
  if( MPI_ERR_PROC_FAILED == rc ) {
    goto acceptance_test;
  }
  rc = MPI_Allreduce( ..., comm );
  if( MPI_ERR_PROC_FAILED == rc ) {
    goto acceptance_test;
  }

// Acceptance Test
acceptance_test:
  // Check result of computation
  // The return code in this example.
  allsucceeded = (MPI_SUCCESS == rc);
  // Agree upon acceptance test
  MPI_Comm_agree(comm, &allsucceeded);
  // If failed, then the allsucceeded will be 'false'
  if( !allsucceeded ) {
    // Start recovery action
  }
```

Algorithm-Based Fault Tolerance (ABFT) Techniques

- **Portable, Transparent, Scalable Fault Tolerance Libraries**
  - Combinations of:
    - Application level checkpoint/restart,
    - Message logging,
    - Replication,
    - Containment domains,
    - Transactions, …
  - All sitting above a fault tolerant message passing environment



Fault Tolerance (#323)
User-Level Failure Mitigation (ULFM RTS)

Define a set of semantics and interfaces to enable fault tolerant applications and libraries to be portably constructed on top of MPI.

- **Application involved fault tolerance (not transparent FT)**
- **Starting with fail-stop process failure**
  - A process failure in which the MPI process permanently stops communicating with other MPI processes, and its internal state is lost.
- **Two Complementary Proposals:**
  - Run-Through Stabilization: ~~(*Target: MPI-3.0*)~~ (*Target: MPI-3.1*)
    - Continue running and using MPI even if one or more MPI processes fail
  - Process Recovery: ~~(*Target: MPI-3.1*)~~ (*Target: MPI-3.2?*)
    - Replace MPI processes in existing communicators, windows, file handles

MPI Forum Fault Tolerance Working Group:
https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage

User-Level Failure Mitigation (ULFM) Run-Through Stabilization (RTS) Proposal

- **Failures are managed on a per-communicator basis**
  - `MPI_ERR_PROC_FAILED`: operation failed due to process failure
- **Point-to-Point Communication**
  - Communication between active processes is *unaffected* by the failure of a non-participating process.
- **Collective Communication**
  - Fault-aware: Will not hang in the presence of process failure, but may not return the same return code at all processes.
- **Communicator Creation**
  - Behave as other collectives. Therefore, it is possible that some processes see a valid communicator while others do not.
  - `MPI_COMM_SHRINK(comm, &newcomm)`

MPI Forum Fault Tolerance Working Group:
https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage

User-Level Failure Mitigation (ULFM) Run-Through Stabilization (RTS) Proposal

- `MPI_COMM_SHRINK(comm, &newcomm)`
  - A special fault tolerant creation operation that creates a new communicator with just the alive processes of an input communicator.
- `MPI_COMM_REVOKE(comm)`
  - Any one process can revoke the communication context of a communicator at all processes
  - All subsequent, non-local operations on that communicator will return an error `MPI_ERR_REVOKED`
  - Eventually all other processes will see the error, even if they did not call `MPI_COMM_REVOKE()`.
- `MPI_COMM_AGREE (comm, &flag)`
  `MPI_COMM_IAGREE(comm, &flag, &req)`
  - Collective fault tolerant agreement operation that will return uniformly at all processes with the same return code and value for `flag`.
  - `flag` is boolean argument & agreement on logical AND of input values.

MPI Forum Fault Tolerance Working Group:
https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage

Early Experimentation Results: ULFM RTS MPI Prototype

- **NetPIPE Latency/Bandwidth**
  - <ins><1% overhead</ins> in shared memory latency
  - <ins>Negligible</ins> impact on shared memory bandwidth
  - <ins>Negligible</ins> impact on performance over the Gemini interconnect
- **Collectives:**
  - Existing collectives over point-to-point did not need to be modified
  - The collectives only needed to error out when a failure is encountered
  - <ins>No additional overhead</ins> for collective operations
- **Agreement:**
  - Log scaling performance results presented at EuroMPI 2011
  - Performance <ins>similar to an MPI_Allreduce</ins> over the alive processes.

Wesley Bland, Aurelien Bouteiller, Thomas Herault, Joshua Hursey, George Bosilca, and Jack J. Dongarra. "*An Evaluation of User-Level Failure Mitigation Support in MPI*." EuroMPI, 2012
Hursey, J., Naughton, T., Vallee, G., Graham, R., "*A Log-Scaling Fault Tolerant Agreement Algorithm for a Fault Tolerant MPI*," EuroMPI, 2011.

Fault Tolerance (#323)
User-Level Failure Mitigation (ULFM RTS)

- **Proposal: #323**
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/323
- **Open MPI**
  - **Available today**: Beta release
    - http://www.open-mpi.org/~jjhursey/projects/ft-open-mpi/
  - **Scheduled release**: Unknown
- **MPICH2**
  - Partial support in the MPICH2 trunk, but not to the current proposal.
  - **Available today**: Unknown
  - **Scheduled release**: Unknown
- **Other implementations working on support at the moment.**

Overview of Seminar Series

- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - **MPI <next>**
    - Fault Tolerance
    - **Hybrid, collectives, …**

MPI <next>: 3.1/4.0
A small sampling of what is in the works

- **Hybrid Programming Models**
  https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid
  - How to support co-existence with other models?
  - **Endpoints** (#310, #311)
  - **Helper Threads** (#217)
- **Collectives**
  http://lists.mpi-forum.org/mailman/listinfo.cgi/mpi3-coll
  - **Scalable variants of vector collectives** (#264) (e.g., MPI_GATHERDV)
- **File I/O**
  http://lists.mpi-forum.org/mailman/listinfo.cgi/mpi3-io
  - **Immediate versions of nonblocking I/O collectives** (#273)
  - **MPI_File_stat** (#295)

Overview of Seminar Series

- **Monday, June 25 - 3-4 pm:**
  - MPI Process (brief)
  - Timeline to 3.0
  - MPI 3.0 Fortran Bindings
  - MPI 2.2
- **Tuesday, June 26 - 3-4 pm**
  - Collectives:
    - Neighborhood
    - Nonblocking
  - Communicator Creation:
    - Noncollective
    - Nonblocking duplication
- **Thursday, June 28 - 3-4 pm**
  - MPI_Comm_split_type()
  - MPI Matched Probe/Recv
  - RMA / One-sided enhancements
  - Tool Interfaces
  - MPI <next>
    - Fault Tolerance
    - Hybrid, collectives, …

A Preview of MPI 3.0: The Shape of Things to Come

- **Thanks to:**
  - Brian Barrett (SNL)
  - Torsten Hoefler (ETH Zürich)
  - Rolf Rabenseifner (HLRS)
  - Craig Rasmussen (University of Oregon)
  - Martin Schulz (LLNL)
  - Jeff Squyres (Cisco Systems)
- **MPI Forum:**
  - **Meetings**: http://meetings.mpi-forum.org
  - **Documents**: http://www.mpi-forum.org
- **ORNL Representatives:**
  - **Manjunath Gorentla Venkata**: manjugv@ornl.gov
  - **Brian Smith**: smithbe@ornl.gov