The Message Passing Interface: On the Road to MPI 4.0 & Beyond

ISC High Performance
The HPC Event.

Martin Schulz
LLNL / CASC
Chair of the MPI Forum

MPI Forum BOF @ ISC 2016

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

LLNL-PRES-696804

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Where We Are

Available through HLRS
-> MPI Forum Website

- **MPI 3.0 ratified in September 2012**
  - Available at http://www.mpi-forum.org/
  - Several major additions compared to MPI 2.2
- **MPI 3.1 ratified in June 2015**
  - Inclusion for errata (mainly RMA, Fortran, MPI_T)
  - Minor updates and additions (address arithmetic and non-block. I/O)
  - Adaption in most MPIs progressing fast

Notable Additions since MPI 2.2

- Non-blocking collectives
- Neighborhood collectives
- RMA enhancements
- Shared memory support
- MPI Tool Information Interface
- Non-collective communicator creation
- Fortran 2008 Bindings
- New Datatypes
- Large data counts
- Matched probe
- Nonblocking I/O

Status of MPI-3.1 Implementations

	MPICH	MVAPICH	Open MPI	Cray MPI	Tianhe MPI	Intel MPI	IBM BG/Q MPI [1]	IBM PE MPICH [2]	IBM Platform	SGI MPI	Fujitsu MPI	MS MPI	MPC	NEC MPI
NBC	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	(*)	✔	✔
Nbrhood collectives	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	✔	✔
RMA	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	Q2'17	✔
Shared memory	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	*	✔
Tools Interface	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	*	Q4'16	✔
Comm-creat group	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	*	✔
F08 Bindings	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✘	✘	Q2'16	✔
New Datatypes	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔
Large Counts	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	Q2'16	✔
Matched Probe	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	Q2'16	✔
NBC I/O	✔	Q3'16	✔	✔	✘	✔	✘	✘	✘	✔	✘	✘	Q4'16	✔

Release dates are estimates and are subject to change at any time.

"✘" indicates no publicly announced plan to implement/support that feature.

Platform-specific restrictions might apply to the supported features

[1] Open Source but unsupported

[2] No MPI_T variables exposed

* Under development

(*) Partly done

On the Road to MPI 4.0

- **Some of the major initiatives discussed in the MPI Forum**
  - One Sided Communication (William Gropp)
  - Point to Point Communication (Daniel Holmes)
  - MPI Sessions (Daniel Holmes)
  - Hybrid Programming (Pavan Balaji)
  - Large Counts (Jeff Hammond)

  - Short updates on activities on tools, persistence and fault tolerance

- **How to contribute to the MPI Forum?**

Let's keep this interactive – Please feel free to ask questions!

MPI RMA Update

William Gropp
www.cs.illinois.edu/~wgropp

Brief Recap: What's New in MPI-3 RMA

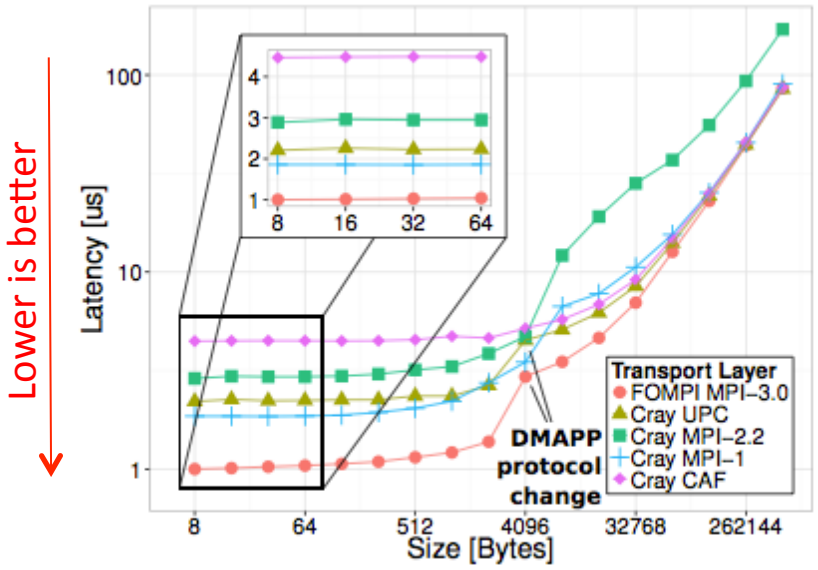
- Substantial extensions to the MPI-2 RMA interface
- New window creation routines:
  - MPI_Win_allocate: MPI allocates the memory associated with the window (instead of the user passing allocated memory)
  - MPI_Win_create_dynamic: Creates a window without memory attached. User can dynamically attach 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

What's new in MPI-3 RMA contd.

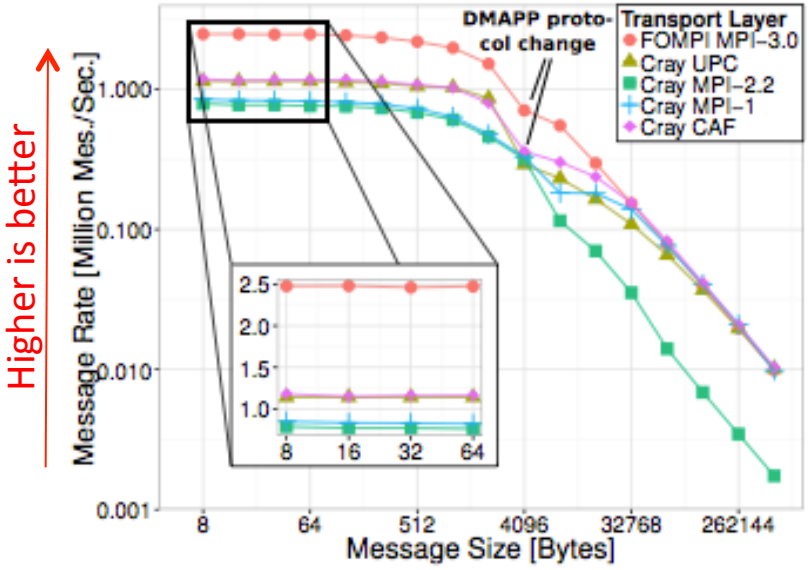
- A new "unified memory model" in addition to the existing memory model, which is now called "separate memory model"
- The user can query (via MPI_Win_get_attr) whether the implementation supports a unified memory model (e.g., on a cache-coherent system), and if so, the memory consistency semantics that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

MPI-3 RMA can be implemented efficiently

- "Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided" by Robert Gerstenberger, Maciej Besta, Torsten Hoefler (SC13 Best Paper Award)
- They implemented complete MPI-3 RMA for Cray Gemini (XK5, XE6, XK7) and Aries (XC30) systems on top of lowest-level Cray APIs
- Achieved better latency, bandwidth, message rate, and application performance than Cray's UPC and Cray's Fortran Coarrays



(a) Latency inter-node Put



(b) Message Rate inter-node

MPI RMA is Carefully and Precisely Specified

- To work on both cache-coherent and non-cache-coherent systems
  - Even though there aren't many non-cache-coherent systems, it is designed with the future in mind

- There even exists a formal model for MPI-3 RMA that can be used by tools and compilers for optimization, verification, etc.
  - See "Remote Memory Access Programming in MPI-3" by Hoefler, Dinan, Thakur, Barrett, Balaji, Gropp, Underwood. ACM TOPC, Volume 2 Issue 2, July 2015.
  - http://dl.acm.org/citation.cfm?doid=2798443.2780584

Some Current Issues Being Considered

- Clarifications to shared memory semantics
- Additional ways to discover shared memory in existing windows
- New assertions for passive target epochs
- Nonblocking RMA epochs

MPI ASSERTIONS
(PART OF THE POINT-TO-POINT WG)

Dan Holmes

Assertions as communicator INFO keys

- Three separate issues
  - #52 – remove info key propagation for communicator duplication
  - #53 – add function MPI_Comm_idup_with_info
  - #11 – allow INFO keys to specify assertions not just than hints plus define 4 actual INFO key assertions

Remove propagation of INFO

- Currently MPI_Comm_dup creates an exact copy of the parent communicator including INFO keys and values
- The MPI Standard is not clear on which version of an INFO key/value to propagate
  - The one passed in by the user or the one used by the MPI library?
- If INFO keys can specify assertions then propagating them is a bad idea
  - Libraries are encouraged to duplicate their input communicator
  - Libraries expect full functionality, i.e. no assertions
  - Libraries won't obey assertions they didn't set and don't understand
- Removal is backwards incompatible but
  - Propagation was only introduced in MPI-3.0

Add MPI_Comm_idup_with_info

- Non-blocking duplication of a communicator
  - Rather than blocking like MPI_Comm_idup
- Uses the INFO object supplied as an argument
  - Rather than propagating the INFO from the parent communicator like MPI_Comm_dup_with_info
- Needed for purely non-blocking codes, especially libraries

Allow assertions as INFO keys

- Language added to mandate that user must comply with INFO keys that restrict user behaviour
- Allows MPI to rely on INFO keys and change its behaviour
- New optimisations are possible by limiting user demands
- MPI can remove support for unneeded features and thereby accelerate the functionality that is needed

New assertions

- Four new assertions defined:

- mpi_assert_no_any_tag
  - If set true, MPI_ANY_TAG will not be used
- mpi_assert_no_any_source
  - If set true, MPI_ANY_SOURCE won't be used
- mpi_assert_exact_length
  - If set true, all sent messages will exactly fit their receive buffers
- mpi_assert_allow_overtaking
  - If set true, messages can overtake even if not logically concurrent

Point-to-point WG

- Fortnightly meetings, Monday 11am Central US webex
  - All welcome!

- Future business:
  - Allocating-receive, freeing-send operations
  - Further investigation of INFO key ambiguity

(Streams/channels has been moved to the Persistence WG)

MPI SESSIONS

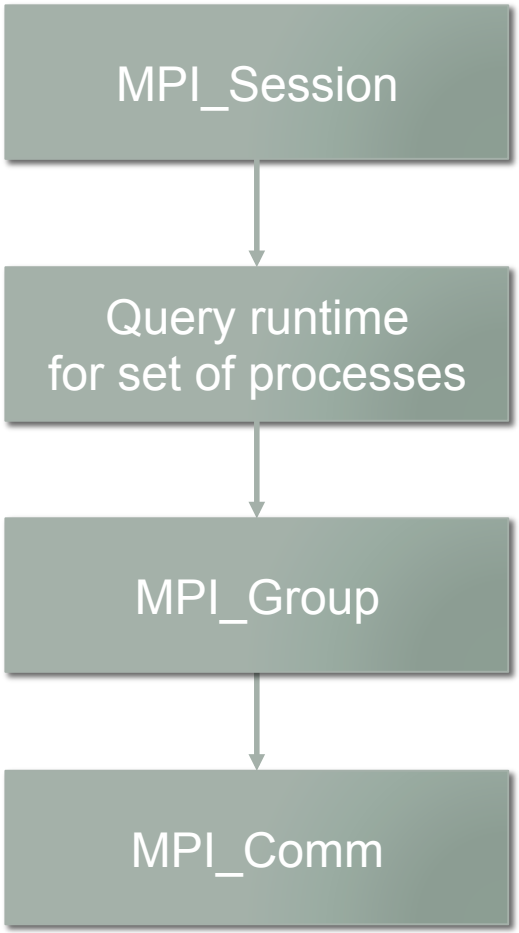
Dan Holmes

What are sessions?

- A simple handle to the MPI library
- An isolation mechanism for interactions with MPI
- An extra layer of abstraction/indirection
- A way for MPI/users to interact with underlying runtimes
  - Schedulers
  - Resource managers
- An attempt to solve some threading problems in MPI
  - Thread-safe initialisation by multiple entities (e.g. libraries)
  - Re-initialisation after finalisation
- An attempt to solve some scalability headaches in MPI
  - Implementing MPI_COMM_WORLD efficiently is hard
- An attempt to control the error behaviour of initialisation

How can sessions be used?

- Initialise a session
- Query available process “sets”
- Obtain info about a “set” (optional)
- Create an MPI_Group directly from a “set”
- Modify the MPI_Group (optional)
- Create an MPI_Communicator directly from the MPI_Group (without a parent communicator)
  - Any type, e.g. cartesian or dist_graph



Why are sessions a good idea?

- Any thread/library/entity can use MPI whenever it wants
- Error handling for sessions is defined and controllable
- Initialisation and finalisation become implementation detail
- Scalability (inside MPI) should be easier to achieve
- Should complement & assist endpoints and fault tolerance

Who are sessions aimed at?

- Everyone!
- Library writers: no more reliance on main app for correct initialisation and provision of an input MPI_Communicator
- MPI developers: should be easier to implement scalability, resource management, (fault tolerance, endpoints, …)
- Application writers: MPI becomes ‘just like other libraries’

Sessions WG

- Fortnightly meetings, Monday 1pm Eastern US webex
  - All welcome!
- https://github.com/mpiwg-sessions/sessions-issues/wiki
- Future business:
  - Dynamic “sets”? Shrink/grow – user-controlled/faults?
  - Interaction with tools? Isolation causes issues?
  - Different thread support levels on different sessions?

Argonne NATIONAL LABORATORY

Towards Enhanced Support for Hybrid Programming

(Hybrid WG)

Pavan Balaji

Hybrid Programming Working Group Chair

balaji@anl.gov

U.S. DEPARTMENT OF ENERGY

MPI Forum Hybrid WG Goals

- Ensure interoperability of MPI with other programming models
  - MPI+threads (pthreads, OpenMP, user-level threads)
  - MPI+CUDA, MPI+OpenCL
  - MPI+PGAS models
- Active Proposals
  - Generalizing MPI processes
  - MPI Endpoints

Generalizing MPI Processes

- The MPI-3.1 standard is, in some places, ambiguous with respect to what an "MPI Process" is
  - Is it an OS process (are global variables shared between MPI processes)?
  - Is it a thread (are global variables shared between a subset of MPI processes)?
- Problematic for some MPI implementations such as MPC (MPI processes are implemented as threads)
- Proposal is to clarify throughout the standard that the MPI process can be implemented as threads
  - Separate proposal to query for such information

MPI Endpoints

- Resource sharing between MPI processes
  - System resources do not scale at the same rate as processing cores
    - Memory, network endpoints, TLB entries, …
    - Sharing is necessary
  - MPI+threads gives a method for such sharing of resources
- Performance Concerns
  - MPI-3.1 provides a single view of the MPI stack to all threads
    - Requires all MPI objects (requests, communicators) to be shared between all threads
    - Not scalable to large number of threads
    - Inefficient when sharing of objects is not required by the user
  - MPI-3.1 does not allow a high-level language to interchangeably use OS processes or threads
    - No notion of addressing a single or a collection of threads
    - Needs to be emulated with tags or communicators

Single view of MPI objects

- MPI-3.1 specification requirements
  - It is valid in MPI to have one thread generate a request (e.g., through MPI_IRECV) and another thread wait/test on it
  - One thread might need to make progress on another's requests
  - Requires all objects to be maintained in a shared space
  - When a thread accesses an object, it needs to be protected through locks/atomics
    - Critical sections become expensive with hundreds of threads accessing it
- Application behavior
  - Many (but not all) applications do not require such sharing
  - A thread that generates a request is responsible for completing it
    - MPI guarantees are safe, but unnecessary for such applications

```
P0 (Thread 1)
MPI_Irecv(…, comm1, &req1);
pthread_barrier();

pthread_barrier();
MPI_Wait(&req1, …);
```

```
P0 (Thread 2)
MPI_Irecv(…, comm2, &req2);
pthread_barrier();
MPI_Wait(&req2, …);
pthread_barrier();
```

```
P1
MPI_Ssend(…, comm1);
MPI_Ssend(…, comm2);
```

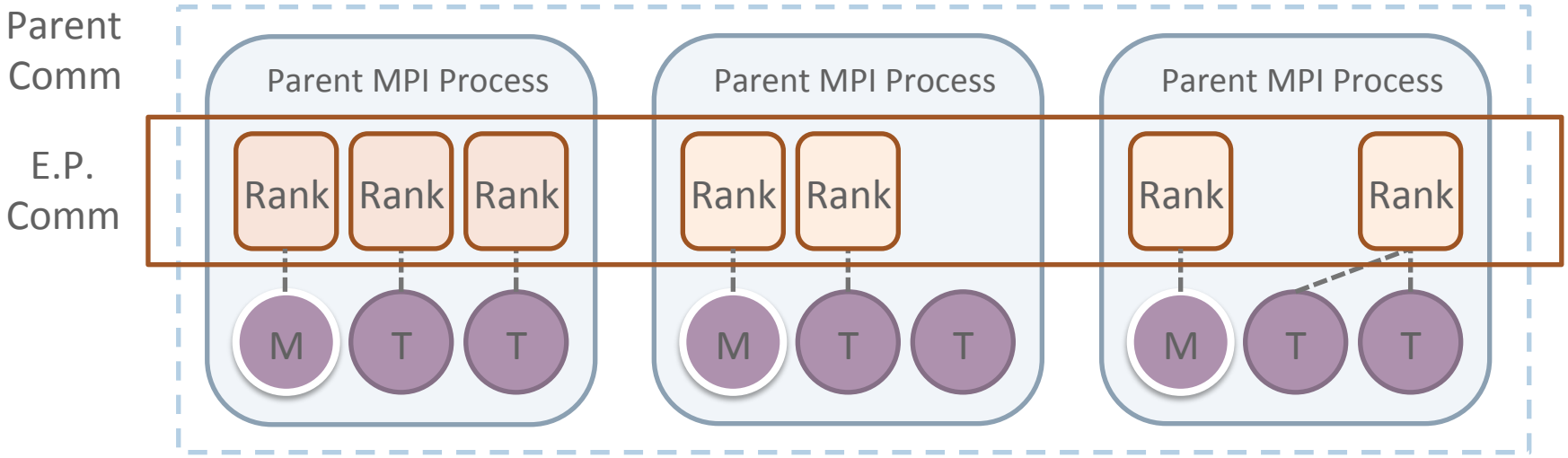
Interoperability with High-level Languages

- In MPI-3.1, there is no notion of sending a message to a thread
  - Communication is with MPI processes – threads share all resources in the MPI process
  - You can emulate such matching with tags or communicators, but some pieces (like collectives) become harder and/or inefficient
- Some high-level languages do not expose whether their processing entities are processes or threads
  - E.g., PGAS languages
- When these languages are implemented on top of MPI, the language runtime might not be able to use MPI efficiently

MPI Endpoints: Proposal for MPI-4

- Idea is to have multiple addressable communication entities within a single process
  - Instantiated in the form of multiple ranks per MPI process
- Each rank can be associated with one or more threads
- Lesser contention for communication on each "rank"
- In the extreme case, we could have one rank per thread (or some ranks might be used by a single thread)

MPI Endpoints Semantics




```
MPI_Comm_create_endpoints(MPI_Comm parent_comm, int my_num_ep,
        MPI_Info info, MPI_Comm out_comm_handles[])
```

- Creates new MPI ranks from existing ranks in parent communicator
  - Each process in parent comm. requests a number of endpoints
  - Array of output handles, one per local rank (i.e. endpoint) in endpoints communicator
  - Endpoints have MPI process semantics (e.g. progress, matching, collectives, …)
- Threads using endpoints behave like MPI processes
  - Provide per-thread communication state/resources
  - Allows implementation to provide process-like performance for threads

Hybrid MPI+OpenMP Example

With Endpoints

```
int main(int argc, char **argv) {
  int world_rank, tl;
  int max_threads = omp_get_max_threads();
  MPI_Comm ep_comm[max_threads];

  MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &tl);
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

#pragma omp parallel
  {
    int nt = omp_get_num_threads();
    int tn = omp_get_thread_num();
    int ep_rank;
#pragma omp master
    {
      MPI_Comm_create_endpoints(MPI_COMM_WORLD, nt, MPI_INFO_NULL, ep_comm);
    }
#pragma omp barrier
    MPI_Comm_rank(ep_comm[tn], &ep_rank);
    ... // Do work based on 'ep_rank'
    MPI_Allreduce(..., ep_comm[tn]);

    MPI_Comm_free(&ep_comm[tn]);
  }
  MPI_Finalize();
}
```

Additional Notes

- Useful for more than just avoiding locks
  - Semantics that are "rank-specific" become more flexible
    - E.g., ordering for operations from a process
    - Ordering constraints for MPI RMA accumulate operations
- Supplementary proposal on thread-safety requirements for endpoint communicators
  - Is each rank only accessed by a single thread or multiple threads?
  - Might get integrated into the core proposal
- Implementation challenges being looked into
  - Simply having endpoint communicators might not be useful, if the MPI implementation has to make progress on other communicators too

More Info

- Endpoints:
  - https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/380
- Hybrid Working Group:
  - https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid

LARGE COUNT WG

Jeff Hammond, Intel

Large Count WG

- Add MPI_Foo_x for relevant Foo?

See http://dx.doi.org/10.1109/ExaMPI.2014.5 for details and discussion.

- Use Fortran interface for int and MPI_Count?
- Use C11 _Generic for int and MPI_Count?
- C++ provides equivalent of C11 _Generic?

Challenges / Potential

- Forum more receptive to lots of MPI_Foo_x compared to MPI 3.0.
  - MPI_Alltoallv_x solves large-disp issue.
  - Needed for most nonblocking collectives.
- Fortran interface viewed favorably.
- C11 _Generic is rather tricky w.r.t. pointers.
- C++ bindings were deleted. Odd to bring back now just to have C11. Non-standard wrapper?

UPDATE ON TOOL INTERFACES IN MPI

Slides by Kathryn Mohror, LLNL

MPI_T interface is in good shape

- Between MPI 3.0 and MPI 3.1
  - Lots of feedback from community
    - Tools people and MPI implementors
  - Errata
    - 19 Errata to MPI 3.0
    - Good thing! People are using the interface!
- Feature update in MPI 3.1
  - Handful of small changes
    - Quick look up of variables
    - Add a couple new return codes
    - Minor clarifications
    - Specify that some function parameters are optional

Tools Interfaces Designs

- New interface to replace PMPI
  - Known, longstanding problems with the current profiling interface
    - One tool at a time can use it
    - Forces tools to be monolithic (a single shared library)
    - The interception model is OS dependent
  - New interface
    - Callback design
    - Multiple tools can potentially attach
    - Maintain all old functionality
- New feature for event notification in MPI_T
  - Keep good ideas from PERUSE without matching MPI_T approach
  - Tool registers for interesting event and gets callback when it happens

Debugging Interfaces Designs

- Fixing some bugs in the original “blessed” MPIR document
  - Missing line numbers!
- Working on design for new API-based support for debuggers – MPIR2
  - Support non-traditional MPI implementations
    - Ranks are implemented as threads
  - Support for dynamic applications
    - Commercial applications/ Ensemble applications
    - Fault tolerance
  - Handle Introspection Interface
    - See inside MPI to get details about MPI Objects
      - Communicators, File Handles, etc.

I have ideas. Can I join in the fun?

- Yes! The MPI tools WG is open to everyone!
- Join the mailing list
  - http://lists.mpi-forum.org/
  - List name: mpiwg-tools
- Join our meetings
  - https://github.com/mpiwg-tools/tools-issues/wiki/Meetings
- Look at the wiki for current topics
  - https://github.com/mpiwg-tools/tools-issues/wiki

Collective Persistent Communication

MPI Forum BoF
ISC 2016

WG lead: Anthony Skjellum, Auburn University

Persistent Communication

- Exists for P2P from the beginning
  - Create persistent request with MPI_*_Init
  - Start communication with MPI_Start
  - Complete as any other asynchronous operation
- Advantages
  - Optimization potential for repeated operations
- Proposal to add same concept for collectives
  - Similar concept by adding “Init” routines
  - Closely related to non-blocking collectives
  - High optimization potential

Fault Tolerance and Error Handling

MPI Forum BoF
ISC 2016

Slides by Wesley Bland, Intel

User Level Failure Mitigation (ULFM)

- Support process level fault tolerance across MPI
- Provide mechanisms for detecting and recovering from failures
- Specify how MPI should react to failures
- Still being debated by the MPI Forum
  - Hope to bring up for a vote within a few months

Error Handling Clarifications

- Error Handlers
  - New error handlers could provide more information to users
  - Give more flexibility to handle specific types of errors in different ways
  - Combine the different types of error handlers into a single function (for more general use)
- Error Reporting
  - Allow MPI to decide if errors are catastrophic or not
  - Non-catastrophic errors would not cause MPI to become "undefined"
- Other Cleanup
  - Generally clean up definitions around errors throughout the standard

The MPI-Forum
How it Works / How to Participate

ISC High Performance
The HPC Event.

Martin Schulz
LLNL / CASC
Chair of the MPI Forum

MPI Forum BOF @ ISC2016

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

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

The MPI Forum Drives MPI

- **Standardization body for MPI**
  - Discusses additions and new directions
  - Oversees the correctness and quality of the standard
  - Represents MPI to the community
- **Organization consists of chair, secretary, convener, steering committee, and member organizations**
- **Open membership**
  - Any organization is welcome to participate
  - Consists of working groups and the actual MPI forum
  - Physical meetings 4 times each year (3 in the US, one with EuroMPI/Asia)
    - Working groups meet between forum meetings (via phone)
    - Plenary/full forum work is done mostly at the physical meetings
  - Voting rights depend on attendance
    - An organization has to be present two out of the last three meetings (incl. the current one) to be eligible to vote

Typical Way to Add Items to MPI

1. **New items should be brought to a matching working group for discussion**
   - Creation of preliminary proposal
   - Simple (grammar) changes are handled by chapter committees

Current Working Groups and Topics

- **Collectives & Topologies**
  - Torsten Hoefler, ETH
  - Andrew Lumsdaine, Indiana
- **Fault Tolerance**
  - Wesley Bland, ANL
  - Aurelien Bouteiller, UTK
  - Rich Graham, Mellanox
- **Fortran**
  - Craig Rasmussen, U. of Oregon
- **Generalized Requests**
  - Fab Tillier, Microsoft
- **Hybrid Models**
  - Pavan Balaji, ANL
- **I/O**
  - Quincey Koziol, HDF Group
  - Mohamad Chaarawi, HDF Group
- **Large count**
  - Jeff Hammond, Intel
- **Persistence**
  - Anthony Skjellum, U. of Alabama
- **Point to Point Comm.**
  - Dan Holmes, EPCC
  - Rich Graham, Mellanox
- **Remote Memory Access**
  - Bill Gropp, UIUC
  - Rajeev Thakur, ANL
- **Tools**
  - Kathryn Mohror, LLNL
  - Marc-Andre Hermans, RWTH Aachen
- **New working groups**
  - Added on demand
  - Support of 4 organizations at a physical MPI forum meeting

Typical Way to Add Items to MPI

1. **New items should be brought to a matching working group for discussion**
   - Creation of preliminary proposal
   - Simple (grammar) changes are handled by chapter committees
2. **Socializing of idea driven by the WG**
   - Could include plenary presentation to gather feedback
     - Focused on concepts not details like names or formal text
   - Make proposal easily available through WG wiki
   - Important to keep overall standard in mind
3. **Development of full proposal**
   - Latex version that fits into the standard
   - Creation of ticket to track voting
4. **MPI forum reading/voting process**

The MPI Voting Process

- **Quorum**
  - 2/3 of eligible organizations have to be present
  - 3/4 of present organization have to vote yes
  - Goal: standardize only if there is consensus
- **Steps**
  1. Reading: "Word by word" presentation to the forum
  2. First vote
  3. Second vote
- **Each step has to be at a separate physical meeting**
  - Ensure people have time to think about additions
  - Avoid hasty mistakes, which are hard to fix
  - Prototypes are encouraged and helpful to convince people

Summary

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

- **MPI Forum is an open forum**
  - Everyone / every organization can join
  - Voting rights depends on attendance of physical meetings
- **Major initiatives towards MPI 4.0**
  - Active discussion in the respective WGs
  - Need/want community feedback
- **Get involved**
  - Let us know what you or your applications need
    - mpi-comments@mpi-forum.org
  - Participate in WGs
    - Email list and Phone meetings
    - Each WG has its own Wiki
  - Join us at a MPI Forum F2F meeting
    - Next meetings: Edinburg, UK (Sep.), Dallas, TX (Dec.)