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

Martin Schulz
TU München
Chair of the MPI Forum

MPI Forum BOF @ SC 2017

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

SC17
Denver, CO | hpc connects.

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

Status of MPI-3.1 Implementations

	MPICH	MVAPICH	Open MPI	Cray	Tianhe	Intel		IBM			HPE	Fujitsu	MS	MPC	NEC	Sunway	RIKEN	AMPI
						IMPI	MPICH-OFI	BG/Q (legacy) [1]	PE (legacy) [2]	Spectrum								
NBC	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Nbr. Coll.	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔
RMA	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	(*)	✔	✔	✔	✔	Q2 '18
Shr. mem	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	Q1 '18
MPI_T	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	*	✔	✔	✔	✔	Q2 '18
Comm-create group	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	*	✔	✔	✔	✔	✔
F08 Bindings	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	✘	✔	✔	✔	✔	Q2 '18
New Dtypes	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Large Counts	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
MProbe	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	Q1 '18
NBC I/O	✔	✔	✔	✔	✘	✔	✔	✘	✘	✔	✔	✘	✘	*	✔	✘	✔	Q3 '18

Release dates are estimates; 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**
  - Support for Hybrid Programming (Pavan Balaji)
  - Update on FP16 and MPI Sessions (Daniel Holmes)
  - Error Management / Fault Tolerance (Aurelien Bouteiller)
  - Tools (Kathryn Mohror)
- **Additionally added after the BOF**
  - Persistence (Anthony Skjellum)
- **How to contribute to the MPI Forum?**

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

Argonne NATIONAL LABORATORY

MPI Forum: Hybrid Programming WG

Pavan Balaji

Hybrid Programming Working Group Chair

balaji@anl.gov

U.S. DEPARTMENT OF ENERGY

Active proposals

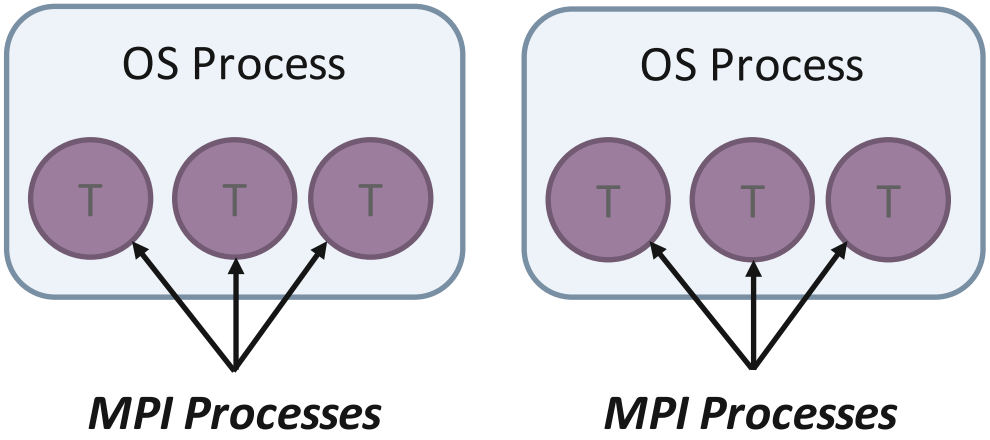
- Clarification of “process” in MPI Standard
- Detecting address space sharing
- Support for per-object thread serialization
- Permit creation of multiple user-visible endpoints
- Modified/new thread support levels

Proposal 1: clarify “process” (1/2)

- The current MPI standard has inconsistent usage of “process”
- Can each MPI process be implemented as a thread inside a single OS process?
- Application users assume that MPI process == OS process
  - If I have a global variable in my program, each MPI process will have a separate copy of that variable (updates do not need to be coordinated)
- The intention of the standard was that this need not be true
  - Already seen in MPI libraries such as MPC and FG-MPI
  - Have to jump through hoops to create multiple copies of global variables

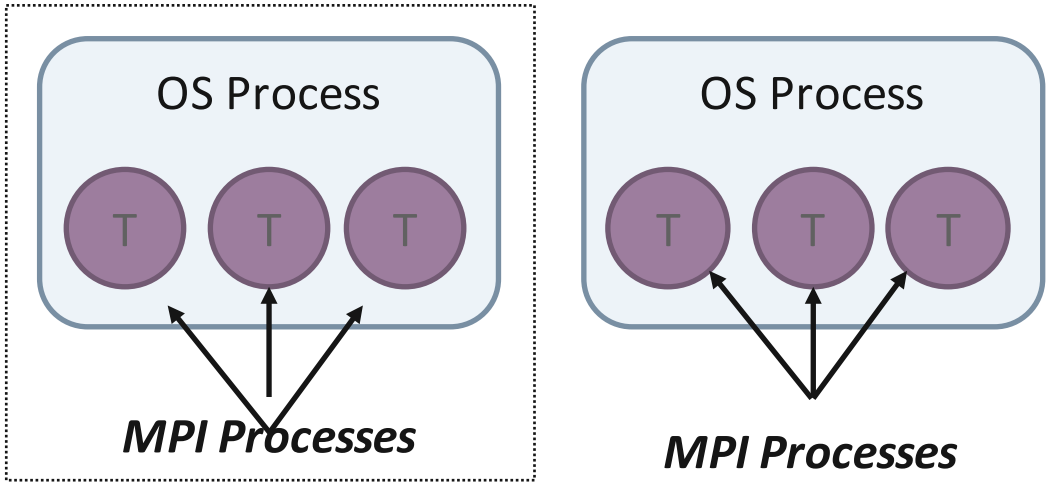
Proposal 1: clarify “process” (2/2)

- This proposal is attempting to clean up the entire MPI standard to make this consistent
- Multiple MPI processes can reside in a single address space
- Using global variables and assuming that they have private storage is incorrect (or at least not portable)
  - Potentially high impact to users, but then, this was always disallowed – most users simply ignored that requirement



Proposal 2: detect address space sharing

- If multiple MPI processes can reside inside a single address space, it will be useful to provide a way to detect which processes are in the same address space
- Current proposal adds a new MPI_COMM_SPLIT_TYPE type that splits processes based on shared address space
  - Already implemented in MPC



Proposal 3: Per-object Thread Serialization

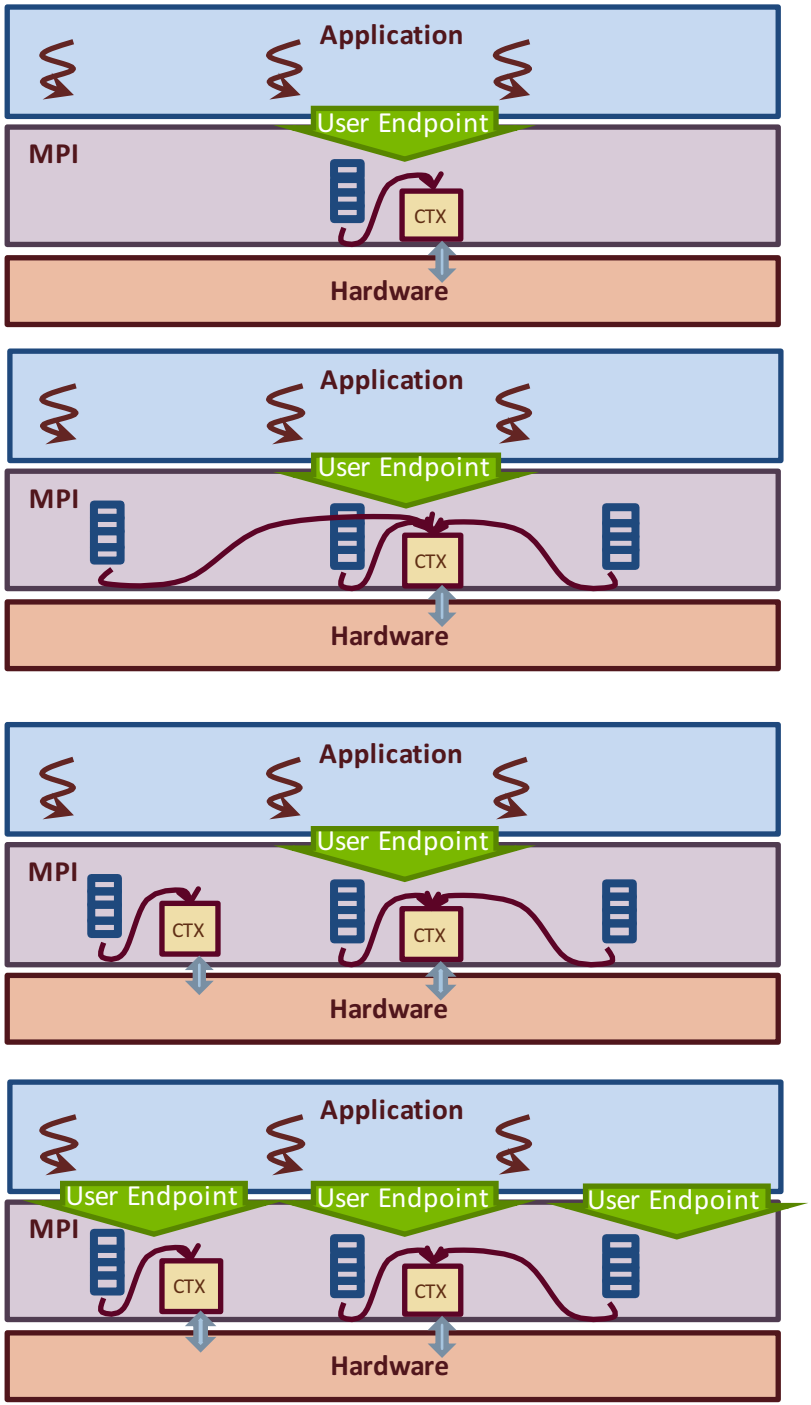
- MPI_THREAD_SERIALIZED says that only one thread can enter the MPI library at a time, while MPI_THREAD_MULTIPLE says that multiple threads can enter the MPI library at once
  - MPI_THREAD_MULTIPLE does not tell us which objects are really shared and which are not
- New per-object thread-safety capabilities
  - Info argument on a communicator, window or file specifying the thread-concurrency on that object (e.g., this communicator will only have one thread operating on it)
  - Might extend to other objects, such as datatypes in the future

Proposal 4: MPI endpoints

- Idea is to have multiple addressable communication entities within a single MPI process
  - Instantiated in the form of multiple ranks per MPI process
- Each rank can be associated with one or more threads
- Reduced 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)

Implementation phases/options

- Most common current approach
  - Single endpoint per MPI process
  - Worst case contention
- Possible optimization in MPI-3.1: multiple invisible endpoints
  - Multiple internal endpoints (BG/Q style)
  - Transparent to the user
  - E.g. one endpoint per comm, per neighbor process (regular apps)
- Endpoints proposal for MPI-4: multiple user-visible endpoints
  - Multiple endpoints managed by the user



State of MPI-3.1 implementations

- MPC, MPICH, and Open MPI are working on the third model from the previous slide
  - Multiple user-visible endpoints are available through different communicators
  - Each communicator maps to a single ordered software queue
  - One or more software queues maps to a single hardware context
- Current implementations do a MPSC queue (enqueue can be done by multiple threads; dequeue requires a lock, e.g., same lock as network context lock)
  - User can give hints to make it a SPSC queue or even an atomic-free queue (if we have enough hardware contexts)

Matching bits

- Using multiple user-visible endpoints in MPI-3.1 requires multiple communicators
- Using multiple user-visible endpoints in MPI-4 will require multiple destination ranks
- Both models require equivalent match bit reservation

Benefits of Endpoints

- More flexible interface
  - Individually addressable threads
  - Multiple threads concurrently in MPI without needing full MPI_THREAD_MULTIPLE support
- Multiple threads active in collective operations
  - Multi-thread packing/unpacking
  - Multi-thread processing of reduction operations
  - Avoids OpenMP reduction followed by MPI reduction
- Other things
  - Allocating receive and freeing send (ownership passing, true zero-copy point-to-point)

Proposal 5: modified/new thread levels

- Modified (or new) thread support levels for MPI_INIT_THREAD
  - MPI_THREAD_ENDPOINT_FUNNELED and MPI_THREAD_ENDPOINT_SERIALIZED
  - MPI_THREAD_COMM_FUNNELED and MPI_THREAD_COMM_SERIALIZED (what about win/file)
- Benefits (beyond INFO key assertion alternative):
  - INIT-time declaration allows MPI implementations to setup global resources "appropriately”

FP16 Support in MPI

Point-to-Point WG update
MPI BoF at SC17

FP16

- FP16 = 2-byte floating-point storage
- IEEE format specification exists
- Languages are starting to include support
- Useful for DL and other mixed-precision workloads
- Proposal: add built-in datatype(s) to MPI
- Status: draft standard text written, some issues
- Aiming for formal reading in March 2018 meeting

https://github.com/mpi-forum/mpi-issues/issues/65

“Sessions” Endpoints / Teams

Sessions WG update
MPI BoF at SC17

Sessions Problem Statement

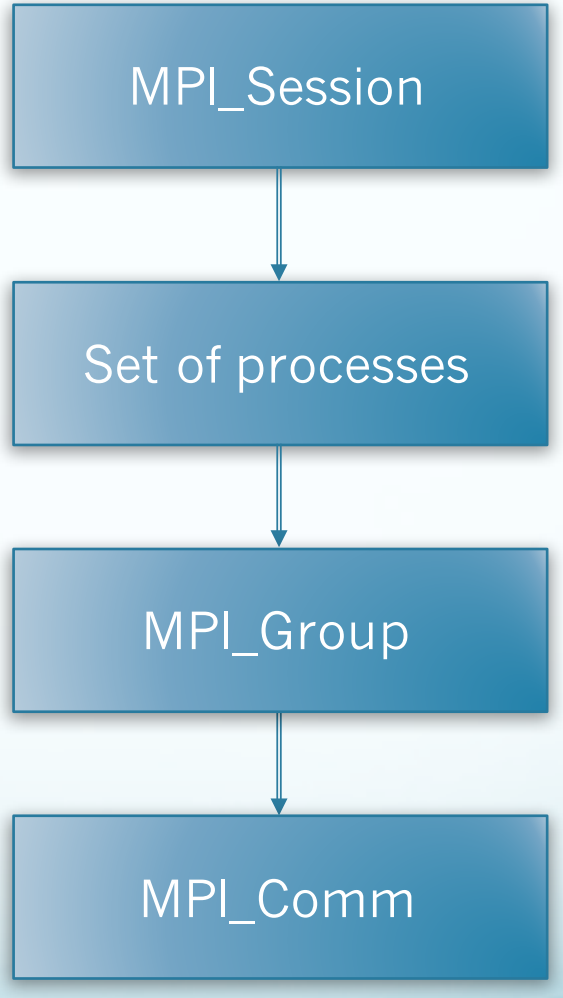
- Original problem: improve initialisation of MPI
  - Avoid multi-library, multi-threaded race-conditions
- Expanded problem: meta-programming for MPI
  - Interaction with resource manager / job scheduler
  - Move beyond process-centric world view (e.g. threads)
  - Obtain useful HW topology information from system
  - Support coupled applications: visualisation, steering
  - Unify with tool functionality (debug, performance)
  - Handle dynamic resources – grow/shrink/adapt/fault

(Some) Design Questions

- Supporting full/arbitrary/flexible addressability
  - What is an MPI_Group member/MPI process/“rank”?
  - SW/HW Thread, OS process, CPU core, GPU, node?
- Permitting/limiting access to “external” resources
  - Growing a job, coupling applications, connecting tools
  - Discovery (publish/subscribe?), security, logistics
- Isolation of sessions – software and hardware
  - Load a different MPI library for each session?
  - Fault isolation domain, thread-support domain, etc.

Sessions Overview

- General scheme:
  - Query the underlying run-time system
    - Get a “set” of processes
  - Determine the processes you want
    - Create an MPI_Group
  - Create a communicator with just those processes
    - Create an MPI_Comm



Previously Proposed API

- Get a local handle that distinguishes each caller
  - MPI_SESSION_INIT
- Discover named sets of (abstract/HW) resources
  - MPI_SESSION_GET_NAMES
- Convert resources into normal MPI objects
  - MPI_GROUP_CREATE_FROM_SESSION
- Create objects – MPI_Comm, MPI_Win, & MPI_File
  - MPI_CREATE_<XXX>_FROM_GROUP

“Set of processes”

- This is not a proper name
- **Suggestion: use “Team” instead**
- Same concept as before, just a new name
- Teams have properties: name, size, <others>
- Built-in teams: “mpi://world” and “mpi://self”
- Command-line: “mpiexec –np 100 –team ocean;
  –np 200 –team atmos”
- Runtime: “arch://x86”, “location://rack6”, etc

Sessions + Tools

- Debug existing application
  - Start 2nd job
  - Use MPI_SESSION to find app job
  - <permissions check>
  - “Attach” to app job
  - Query teams => debug teams + app teams
  - Stuff (e.g. debug via remote RMA operations!)
- Workflow – coupled applications
  - Start independent jobs
  - Attach from one to other and from other to one
  - Query teams => job 1 teams + job 2 teams

Sessions + Tools

- MPIR2
  - Interaction with runtime, e.g. for co-location
  - mpirun -np 500 –team app://ocean ocean.exe;
    -np 500 –team debug://tv tv.exe
  - Ocean.c int main {MPI_INIT; …}
  - Tv.c int main {MPI_SESSION_INIT;
    MPI_SESSION_GET_NAMES; …}

Sessions + Tools

- How to initialise the MPI_T interface?
  - **MPI_SESSION_INIT(*sess1)**
  - MPI_T_SESSION_INIT(*t_sess1)
  - MPI_T_SESSION_CREATE(sess1, *t_sess2)
  - Scope **all** MPI_T functions with a t_sess parameter

- NB: do we need to scope all normal MPI functions with a session parameter?
  - MPI_TYPE_COMMIT, MPI_ERRHANDLER_CREATE, MPI_INFO_CREATE, MPI_ALLOC_MEM, lots of others

“Session”

- The name “Session” has already been used in MPI
  - Performance variables in MPI_T require a session
- Currently looking at various alternative names

- **Suggestion: a session is actually an endpoint!**

Sessions –> Endpoints

- Similarities
  - Isolated resources
  - Independent “identities”
- Differences
  - Lifetime: endpoints created via a new communicator, but can survive its destruction
  - Scope: endpoints returned in array to a single thread, not useful for isolation of libraries

Sessions –> Endpoints

```
MPI_Endpoint ep; MPI_Team team;
MPI_Group grp; MPI_Comm comm;

MPI_ENDPOINT_INIT(MPI_FLAGS_NULL,
                  MPI_INFO_NULL, &ep);

MPI_GROUP_FROM_TEAM(MPI_TEAM_WORLD, ep, 1,
                    “MyFirstTag”, &group);

MPI_COMM_FROM_GROUP(group, &comm);
```

Multiple MPI Libraries

- Suggestion: different MPI library for each session
  - Should that be “endpoint” or “team” in future?
- Use-case: bridging between clusters/systems
- Sessions could be a shim layer
  - Redirect MPI functions to the correct real MPI library
- Main issue: no ABI between MPI libraries
  - Could session shim translate? Yes
  - In critical path? With zero overhead? Probably no

PMPI/QMPI

- If sessions is a shim layer for multiple MPI libraries
- And PMPI/QMPI is an interception layer
- Are they basically the same thing?
- What does that look like/mean?

Summary

- Lots of new ideas – check ✔
- Exciting/interesting – check ✔
- Contentious – check ✔

- Please get involved!

Error Management Working Group Update

Aurélien Bouteiller

MPI Forum Bof @SC'17

Summary of activities

- Default error handlers and error/abort behavior
- Non-catastrophic errors
- Integration between global C/R and scoped recovery models
- User Level Failure Mitigation

Default and Fatal Errors

MPI_ERRORS_ARE_FATAL The handler, when called, causes the program to abort on all executing processes. This has the same effect as if MPI_ABORT was called by the process that invoked the handler.

- In Section 8.3, the above statement is self contradictory
  - It aborts "all" executing processes, but MPI_ABORT has a communicator argument
  - The later is more useful to contain errors in domains
- Proposed changes:
  - **MPI_ERRORS_ARE_FATAL will by default be attached to MPI_COMM_WORLD, MPI_COMM_SELF** and the communicator obtained from **MPI_COMM_GET_PARENT**;
  - It is fatal at all *connected processes*
  - New handler **MPI_ERRORS_ABORT** aborts (only) the communicator (window/file)
  - MPI errors during operations that are **not attached to a communicator/window/file** will be raised on **MPI_COMM_SELF** (instead of MPI_COMM_WORLD)
  - Clarification of the inheritance rules: after MPI_COMM_DUP(comm1, &comm2), comm2 has the same error handler as comm1
- More info on the MPI Forum ticket #1:
  - https://github.com/mpi-forum/mpi-issues/issues/1

(Non-)Catastrophic Errors

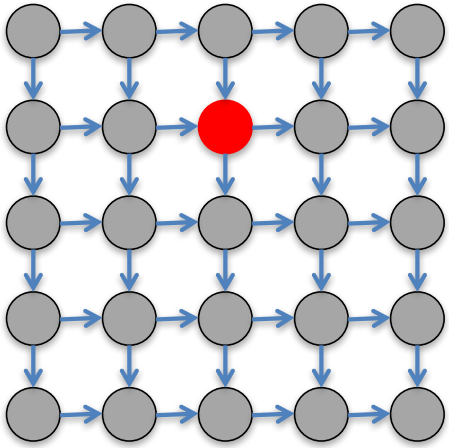
- After an error is detected, the state of MPI is undefined *if the error is catastrophic*, that is …
- MPI is in a correct, defined state after a "non-catastrophic" error
- **`MPI_Get_state(OUT state)`**
  - When state is **MPI_IS_OK,** the application may continue to use MPI (that is, communicating with MPI will yield correct results).
  - When state is **MPI_IS_CATASTROPHIC**, continued use of MPI interfaces may result in undefined behavior.
- Motivating examples
  - When an error is returned during MPI_WIN_ALLOCATE_SHARED, the user can try to use non-shared memory window, or resort to 2-sided MPI instead.
  - Posting multiple iRecv, creating multiple communicators, etc, running out of MPI resources
- More information on the MPI Forum ticket #28: https://github.com/mpi-forum/mpi-issues/issues/28

Interactions between multiple recovery models

- Global C/R recovery proposed by I. Laguna & friends
  - Simpler to program and deploy
  - Limited to global C/R, no support for localized or scoped recovery
  - *Full text not produced yet* (devil is in the details ☺)
- ULFM
  - Expressive support for localized and communicator scoped recovery
  - Support for user CR and non-CR models
  - Implementing global recovery over ULFM is possible but requires more work from the user level
- WG tasked with evaluating if these models may coexist in the standard
  - WG confident that these may coexist and may be selected at runtime
  - WG still working to understand if/how an application may switch over time from one mode to the other and forth
  - WG investigating if an application may use simplified C/R on a subgroup of the processes, ULFM on another

ULFM MPI Crash Recovery

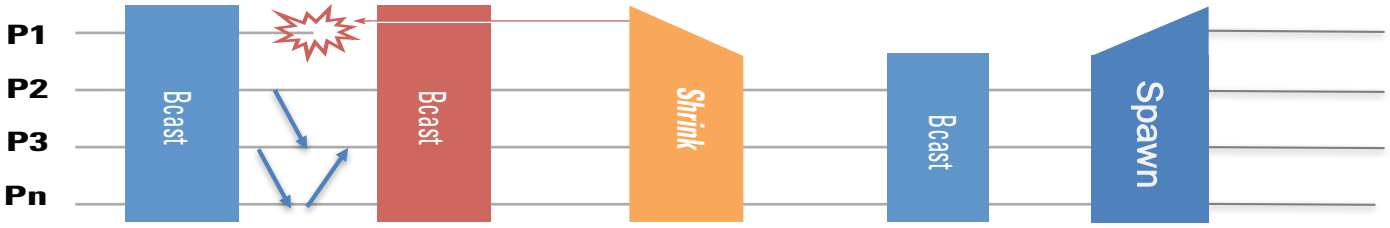
What is the scope of a failure?
Who should be notified about?
What actions should be taken?



- Failure Notification
- Error Propagation
- Error Recovery
- Respawn of nodes
- Dataset restoration

Not all recovery strategies require all of these features, that's why the interface should split notification, propagation and recovery.

- Some applications can continue w/o recovery
- Some applications are maleable
  - Shrink creates a new, smaller communicator on which collectives work
- Some applications are *not* maleable
  - Spawn can recreate a "same size" communicator
  - It is easy to reorder the ranks according to the original ordering
  - Pre-made code snippets available



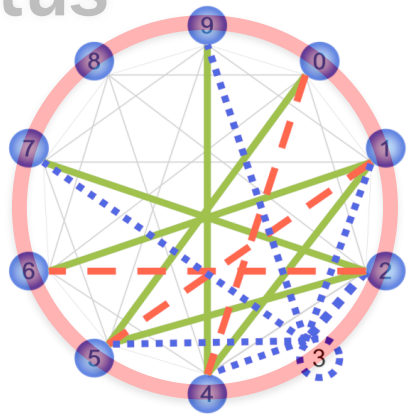
- Adds **3 error codes** and **5 functions to manage process crash**
  - **Error codes:** interrupt operations that may block due to process crash
  - **MPI_COMM_FAILURE_ACK / GET_ACKED:** continued operation with ANY-SOURCE RECV and observation known failures
  - **MPI_COMM_REVOKE** lets applications interrupt operations on a communicator
  - **MPI_COMM_AGREE**: synchronize failure knowledge in the application
  - **MPI_COMM_SHRINK**: create a communicator excluding failed processes
  - More info on the MPI Forum ticket #20: https://github.com/mpi-forum/mpi-issues/issues/20

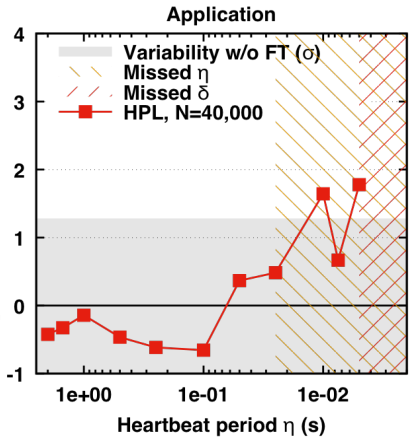
WG Researching ULFM Expansions

- Simplification of “global” recovery patterns
  - ULFM designed to provide “scoped” recovery
  - Addition of function “REVOKE_ALL” to revoke all communicators at once
- Automations
  - In many cases, one wants to discard failed communicators and requests
  - Addition of error handler “MPI_ERRORS_REVOKE, MPI_ERRORS_FREE” to automate these common usage patterns
- Run-through failures RMA
  - ULFM current design limited to ”stopping” RMA operations on a window impacted by a failure (the window may be rebuild from a communicator later)
  - Investigating more ambitious recovery models with continued operation on windows

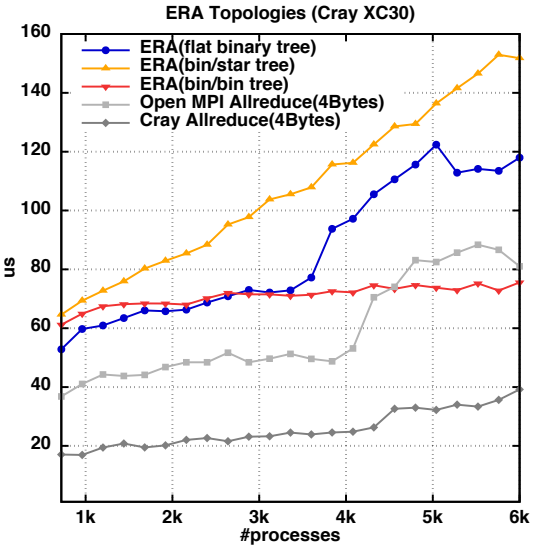
User Level Failure Mitigation: Implementation status

- ULFM available in Open MPI and MPICH
  - ULFM in MPICH release
  - Open MPI ULFM implementation updated in-sync with Open MPI master
- Scalable fault tolerant algorithms
  - Research on algorithms dedicated to HPC resilience bearing fruits
  - New algorithms demonstrated in practice (SC'14, EuroMPI'15, SC'15, SC'16)

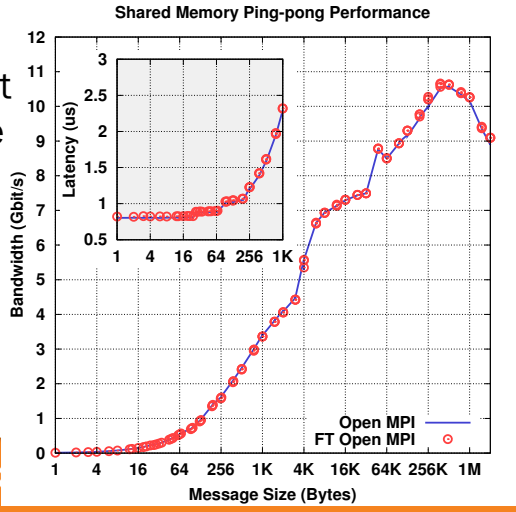




Fault Tolerant Agreement costs approximately 2x Allreduce



Point to point performance unchanged With FT enabled



User Level Failure Mitigation: User Adoption

Judicael A. Zounmevo, Dries Kimpe, Robert Ross, and Ahmad Afsahi. 2013. **Using MPI in high-performance computing services.**

Fenix Framework: user-level C/R With scoped recovery

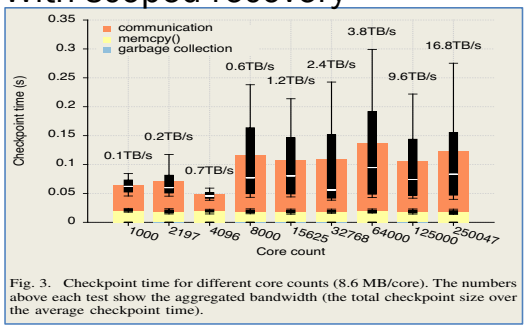


Fig. 3. Checkpoint time for different core counts (8.6 MB/core). The numbers above each test show the aggregated bandwidth (the total checkpoint size over the average checkpoint time).

SAP: Resilient Databases over MPI

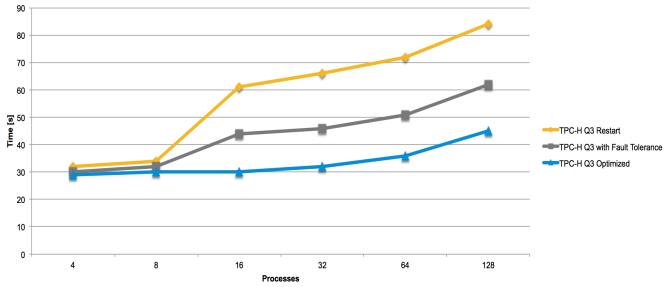


Figure 5.24: Optimization: Runtime of TPC-H Benchmark Query 3 with Failure in Phase 4 (1GB Data per Process)

Master-Thesis von Jan Stengler aus Mainz April 2017

MapReduce

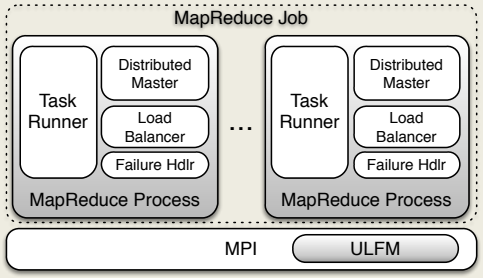


Figure 2: The architecture of FT-MRMPI.

Domain Decomposition PDE

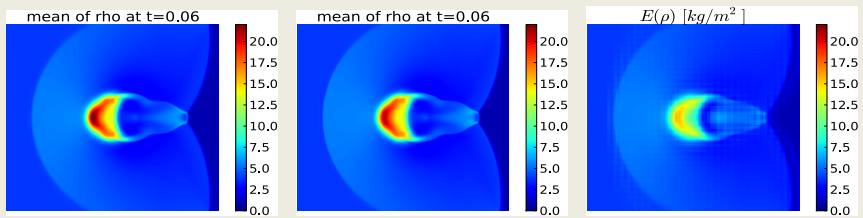
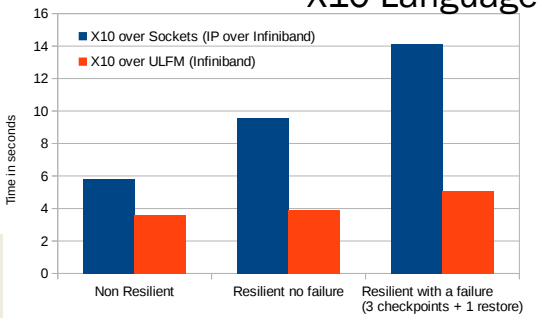


Figure 5. Results of the FT-MLMC implementation for three different failure scenarios.

Source: Sara Hamouda, Benjamin Herta, Josh Milthorpe, David Grove, Olivier Tardieu. ***Resilient X10 over Fault Tolerant MPI.***

X10 Language



The performance improvement due to using ULFM v1.0 for running the LULESH proxy application [3] (a shock hydrodynamics stencil based simulation) running on 64 processes on 16 nodes with

And many more…

- Fortran CoArrays “failed images” uses ULFM-RMA to support Fortran TS 18508 in gcc-7.2

Stefan Pauli, Manuel Kohler, Peter Arbenz: ***A fault tolerant implementation of Multi-Level Monte Carlo methods.*** PARCO 2013: 471-480

Thanks

Participate!

- WG mailing list
  - https://lists.mpi-forum.org/mailman/listinfo/mpiwg-ft
- WG issue tracker
  - https://github.com/mpiwg-ft/ft-issues
- WG meeting notes, documents, and telecon info
  - https://github.com/mpiwg-ft/ft-issues/wiki

MPI Tools Working Group Update

Kathryn Mohror
Lawrence Livermore National Lab
Marc-Andre Hermanns
Forschungszentrum Jülich

What is the Tools WG about?

- Tools support
  - Debuggers
  - Correctness tools
  - Performance analysis tools
  - Could really be anything

MPI Tool Support Chapter

- The MPI Tools Information Interface (MPI_T for short)
  - Developed over many years, by the folks in the MPI Tools Working Group
  - Led by Martin Schulz
- Included in MPI 3.0 in 2012
  - Now there is a new chapter “Tool Support”
  - Replaces the existing MPI profiling interface chapter
  - PMPI included as a new subchapter (unchanged)

Chapter 14

Tool Support

14.1 Introduction

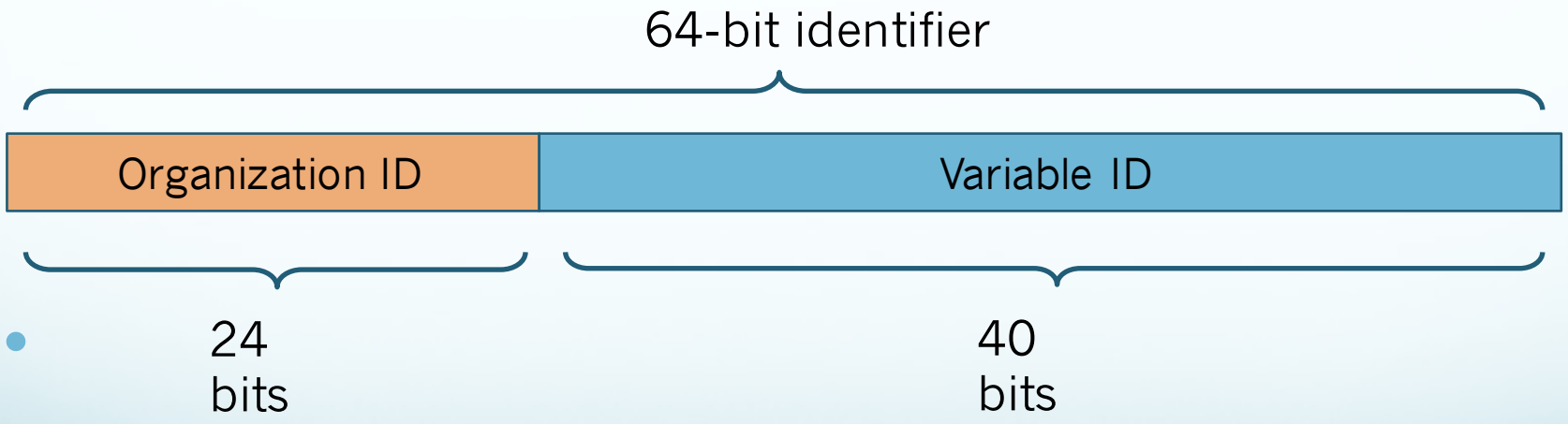
This chapter discusses interfaces that allow debuggers, performance analyzers, and other tools to extract information about the operation of MPI processes. Specifically, this chapter defines both the MPI profiling interface (Section 14.2), which supports the transparent interception and inspection of MPI calls, and the MPI tool information interface (Section 14.3)

What's happening now?

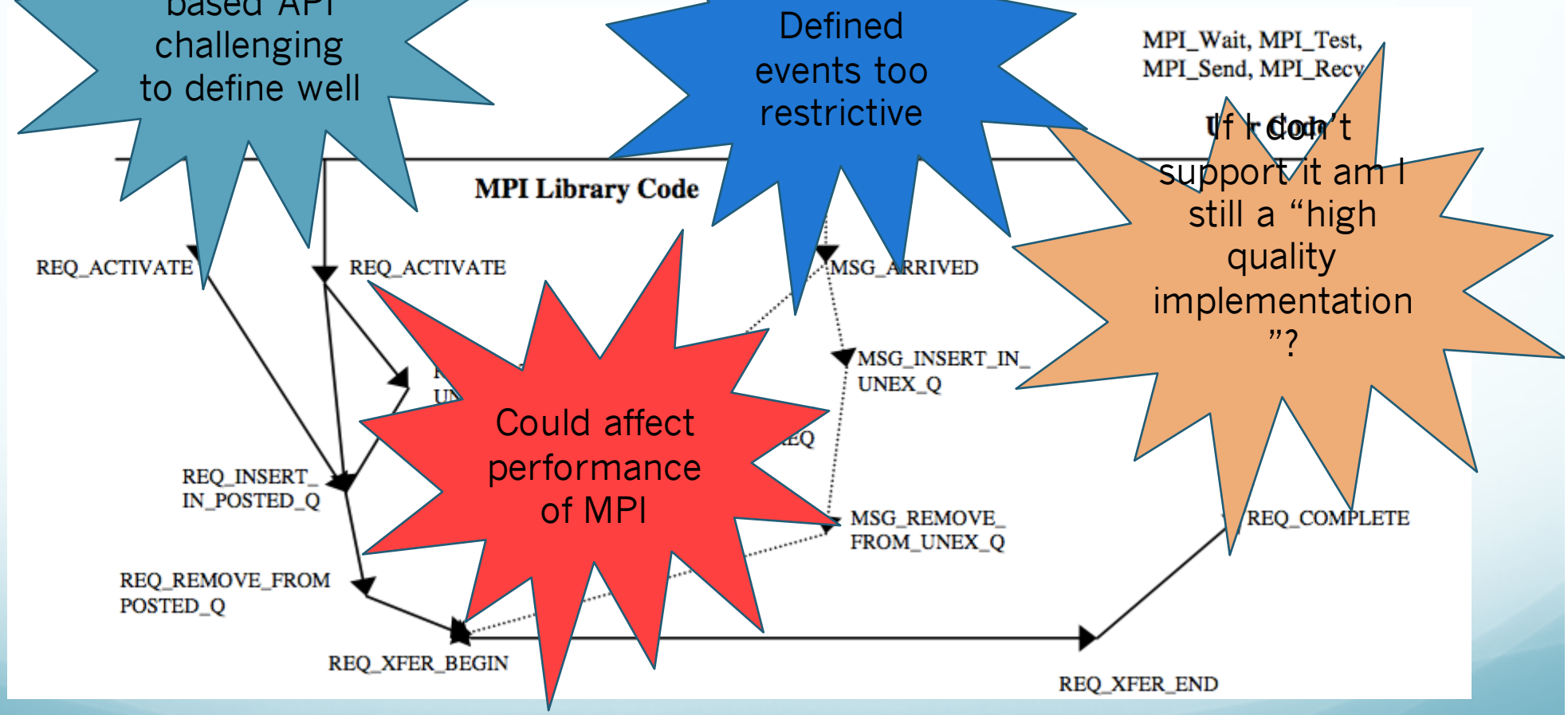
- **Variables**
  - MPI_T Variable Registration
- **QMPI**
  - A new interception interface for MPI
- **Events**
  - Get event notification from MPI

MPI_T Variable registration

- MPI implementations are free to provide whatever variables make sense for their implementation
  - Variables are allowed to change between versions of the library and across hardware
  - Want to provide some stability for tools and keep the freedom for implementations
- Organization IDs and variable identifiers registered with MPI Forum
  - API for retrieving runtime variable ID to permanent registered ID
  - Now tool has a stable ID for understanding the meaning of MPI_T variables



The inspiration for MPI_T_events comes from PERUSE



We're going back to the drawing board to design MPI_T_events

- Provide access to MPI implementation-internal information about events
  - What happens and when it happens
- Do not mandate specific implementation of MPI functionality
  - No requirement to implement specific events
- Blend into existing MPI_T interface
  - Interface to query available events, register, read
- Notification of events can be immediate or deferred
  - Queuing of events can reduce overhead
  - It may be impossible to provide immediate notification of some events

MPI_T_events interface is nearly ready for consideration by the Forum

- Query for events that are available and get their information with MPI_T_event_get_info()
- Register for events of interest with MPI_T_event_register()
  - Provide: index, an object handle, a pointer to user data, a pointer to event handler callback function
- Callback handler can query event data via MPI_T_event_read() MPI_T_event_read_all()
- Event handlers have safety requirements to indicate the behavior allowed in the callback
  - Can you do more than just read the event queue and get a time stamp?
  - E.g. none, thread-safe, async signal safe

I have ideas. Can I join in the fun?

- Yes!
- Join the mailing list
  - http://lists.mpi-forum.org/
  - 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

Persistence Working Group Update:
Persistent Collectives

BoF – The Message Passing Interface: On the Road to MPI 4.0 and Beyond

SC17

Persistent collectives

- New initialization function for every collective operation
  - Traditional collectives
  - Nearest-neighbor collectives
- Each _init has an info argument to express how function will be used
- Syntax similar to persistent point-to-point
  - MPI_Allreduce_init(…, &request)
  - MPI_Start(&request)
  - MPI_Wait(&request)
  - MPI_Request_free(&request) --- only on inactive request
- Allows a ‘plan step’ like FFTW
- High quality implementations are meant to do as much optimization and resource allocation as possible within the _init

Persistent collectives

- Consider existing non-blocking collectives
- Expensive ‘plan step’ could be done there too
- Cache the plan for each collective operation
- Heuristics to keep most frequently re-used plans
- Pros
  - amortize expensive setup over multiple re-uses
  - choose best algorithm
  - lockdown resources
  - reduce time to solution and/or predictability
- Cons
  - much more complex MPI implementation
  - less user control/expertise helping MPI
- Strong evidence that MPI’s don’t pick the best collective with ‘late binding’

Semantic Rules, I

- Clear, consistent set of rules for initialization and start:
  - All processes in a communicator's group call a sequence of _inits in the same order
  - Processes in a communicator's group can call MPI_Start on the persistent request in any order (no need to call start in the same order as the corresponding init)
- _init functions have the following properties:
  - nonblocking collective operations that do not return a separate request object
  - they simply return the request object of the persistent collective
  - it is legal to start a request as soon as _init returns

Semantic Rules, II

- It is always legal to use a persistent request zero or more times before MPI_Request_free(); *but must be called only on inactive request*
- We discourage delaying "the work of _init" till first transfer is done
- Arguments posed at _init are fixed for the duration of the collective
- The values in data arrays are the only thing that can change (that's the key outcome of a collective)
- No changing displacements, counts, data types
- Info key values at _init must be copied and cached by MPI
- Data arrays cannot go out of scope before the request is freed
- No other related objects can go out of scope either before request is freed

Prototype Implementation Status

- Uses the OpenMPI development release for implementation
- Borrows heavily from OpenMPI's LibNBC Modular Component Architecture (MCA) component
- Breaks each collective function into `_init` and `_start` functions
- Provides a method for collective schedule caching and retrieval in the `MPI_Request`
- Leverages MCA automation and the existing capabilities of LibNBC by replicating its code into a new MCA component, LibPNBC, performing search and replace operations to re-identify it, and adding the core functionality required by persistence

Open issues

- Overload meaning of MPI_Request_free for persistent collectives
  OR
  Provide a new, collective, request-free type operation
- Issues:
  - We want resources back across the group in a bounded amount of time
  - MPI may need a synchronization to support resource recovery
- Unlike pt2pt requests, only can free inactive requests

Where to learn more?

- https://github.com/mpi-forum/mpi-issues/issues/25
- Contact us:
  - Tony Skjellum – tony-skjellum@utc.edu
  - Dan Holmes – dholmes@epcc.ed.ac.uk
  - Puri Bangalore – puri@uab.edu

The MPI-Forum
How to Get Involved!

Martin Schulz
TU-München
Chair of the MPI Forum
MPI Forum BOF @ SC2017

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

The MPI Forum Drives MPI

- **Standardization body for MPI**
  - Discusses additions and new directions
  - Voting procedures for actually changing the standard or realizing a new one
  - 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

Current Working Groups and Topics

- **Collectives & Topologies**
  - Torsten Hoefler, ETH
  - Andrew Lumsdaine, Indiana
- **Fault Tolerance**
  - Wesley Bland, ANL
  - Aurelien Bouteiller, UT Knoxville
  - 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, UT Chattanooga
- **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

We Want to Grow the MPI Community!
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: San Jose (Dec), Portland (Feb), Austin (Jun)