Possible Malleability Support in MPI 5:

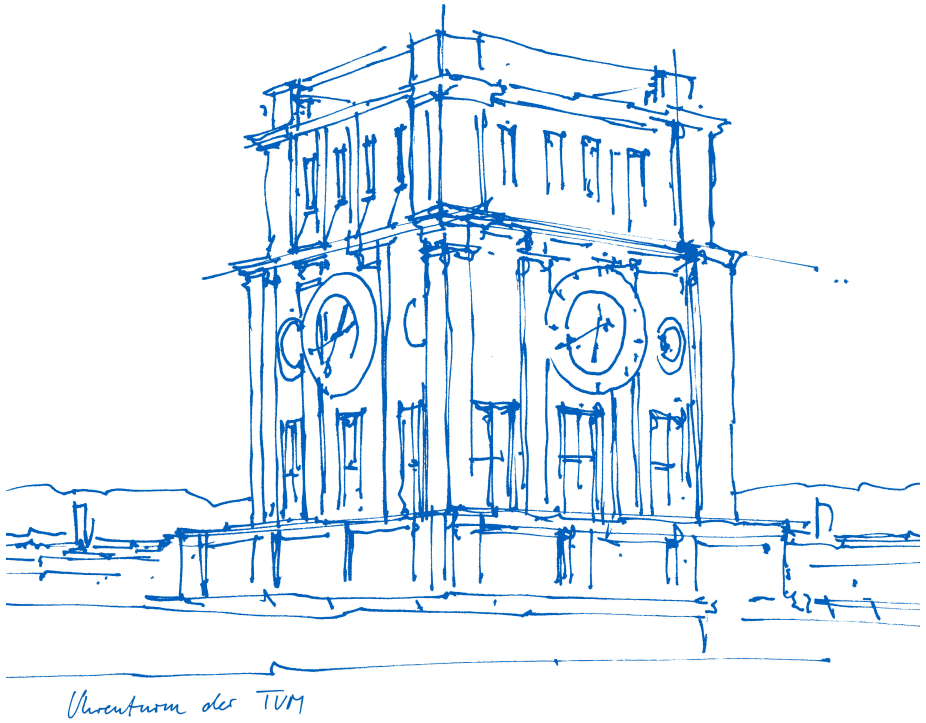
What Does it Mean for (MPI) Tools?

Martin Schulz

Chair for Computer Architecture and Parallel Systems
Technical University of Munich (TUM)

Scalable Tools Workshop 2022

Monday June 20th, 2022



Possible Malleability Support in MPI 5 (and all the other stuff the MPI Forum is cooking up):

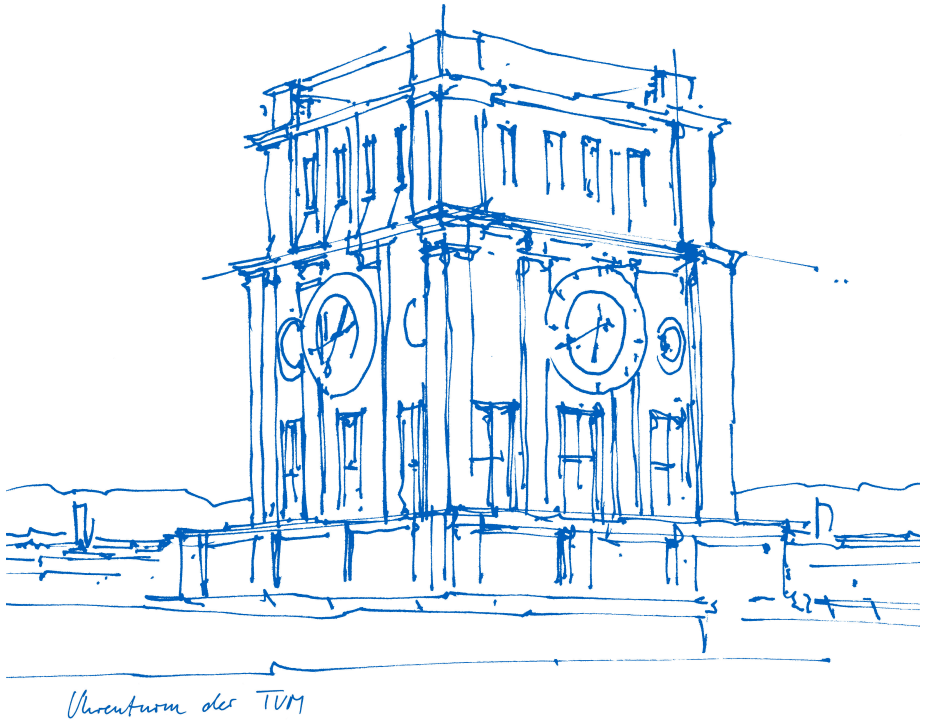
What Does it Mean for (MPI) Tools?

Martin Schulz

Chair for Computer Architecture and Parallel Systems
Technical University of Munich (TUM)

Scalable Tools Workshop 2022

Monday June 20th, 2022



Overview of (some) items in MPI that will affect tools

MPI 4.0 introduced MPI Sessions

- Dynamic initialization and finalization
- Resource isolation

Plans for MPI 5.0 include support for malleability

- Based on the MPI Sessions model
- Changing resource availability and usage on the fly

Other MPI features

- Fault tolerance
- Partitioned communication and accelerator bindings
- Continuations

Finally, perhaps some good news: QMPI

MPI Sessions

Instead of MPI_Init / MPI_COMM_WORLD:

1. Get local access to the MPI library

   Get a Session Handle

2. Query the underlying run-time system

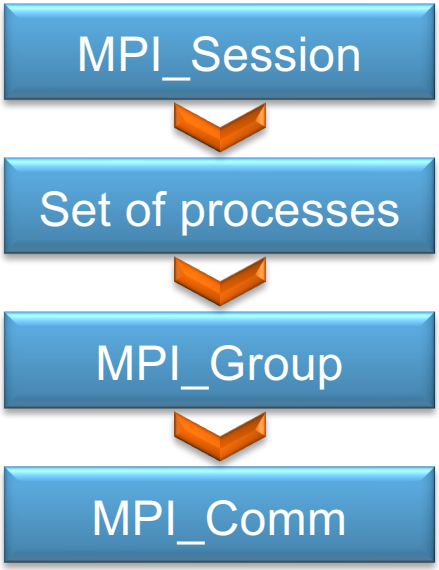
   Get a "set" of processes

3. Determine the processes you want

   Create an MPI_Group

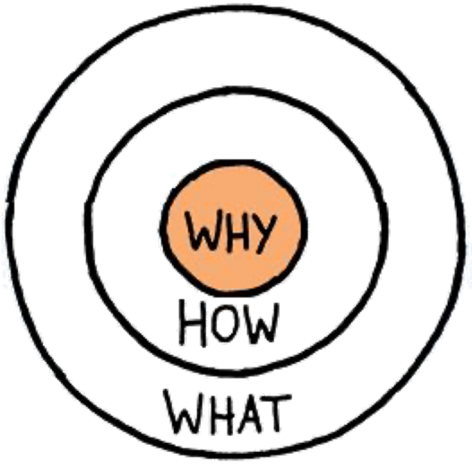
4. Create a communicator with just those processes

   Create an MPI_Comm



Never mix handles derived from two sessions in any call!

MPI Sessions



What does this do?

- Deliver runtime information to the MPI library
- Enable resource isolation between sessions
- Eliminate the static resource MPI_COMM_WORLD

Where do Process Sets Come From?

- Two predefined sets: **`mpi://WORLD`** and **`mpi://SELF`**
- Runtimes can provide system configurations, like **`location://rack/17`**
- Users can specify process sets, like **`app://ocean`**

Intended Usage Patterns

- Scalable initialization on subsets of processes
- Separate library initialization
- Separation of application components (ice, ocean, atmosphere, …)

Consequences

No MPI_COMM_WORLD (potentially, ever)

- Applications do not have to call MPI_Init
  - No longer the first call that can be intercepted
  - More calls allowed before MPI_Init
- Applications can call MPI_Session_init multiple times
  - Every session can have different process sets
  - Even mpi://WORLD can be different

Tools need to track all Session initializations and finalizations

- May have to deal with re-initializations
- Can only rely on communicators derived from particular process sets
- Have to obey the rule of not mixing handles from multiple sessions
- Will be hard to impossible to create a global communicator

Advantages: Tools can utilize separate/isolated sessions themselves

PMPI and MPI Sessions

MPI Session calls can be tracked with PMPI

- Standard interception possible
- Can influence thread levels, if needed

Currently: single PMPI tool stack across all MPI Sessions

- Tools need to keep track of session handles and associations
  - Forum is considering a query Session ID procedure – helpful?
- Tool invocations may not make sense in the calling context

Is this the right model?

- Or should there be a tool(stack) per session
  - If so, how to specify the tool?
  - How to name/associate the sessions?
- Or should there for be a different model?

- Do we need other routines?

Malleability on top of MPI Sessions

Enables path from the runtime to the application

- Runtime can add new process sets in a session (possibly with versioning)
- New sessions can have new process set lists (arguments at session start)

MPI Forum working on APIs to provide handshake

- Detection of new resources
- Negotiations for and acceptance of new resources

Connection to fault tolerance proposals

- Set of sessions from multiple processes can form a transitive “bubble”
- Bubbles can be seen as inherent fault domains (connection to FT)

Should maintain MPI look & feel

EuroHPC Time-X: Weather and climate [1]

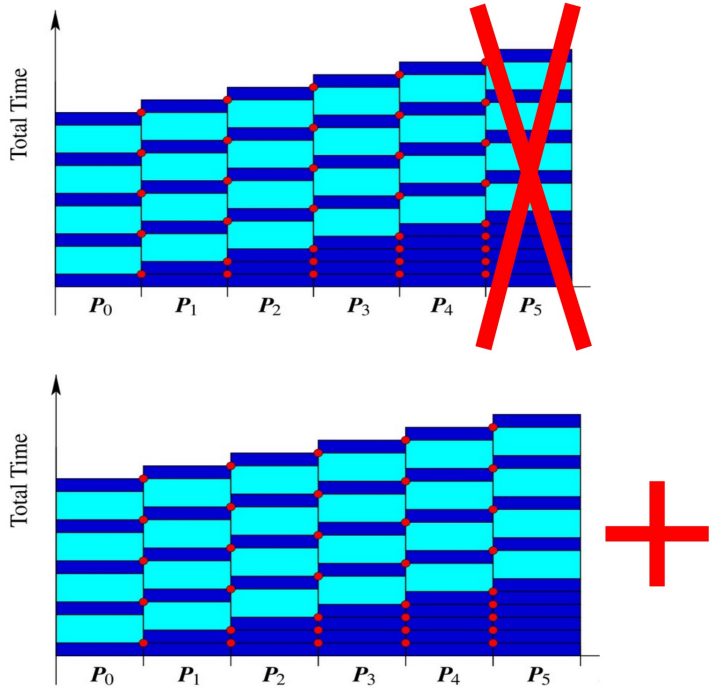
Exploit malleable MPI to support varying resources ‘for parallel-in-time applications

- Vary number of parallel steps
- Based on efficiency predictions
- Based on resource availability

Key design points

- Who triggers malleability?
- Cost model

Each column represents a parallel-in-time simulation instance for a particular time interval



Support removing time-parallel simulation instances (top) or adding them (bottom)

EU Grant #955701
BMBF #16HPC050

Impact of Malleability

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Add {1,13}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Add {6}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Remove {1,4}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Add {15}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Remove {5,12,13}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Resource Change: Add {1,213}

- Resources can come and go
- Final set can be completely different than starting
- But: negotiation routines can be intercepted

From: Jan Fecht, TUM – published at HPCMALL 2022

Example API

Resource Changes

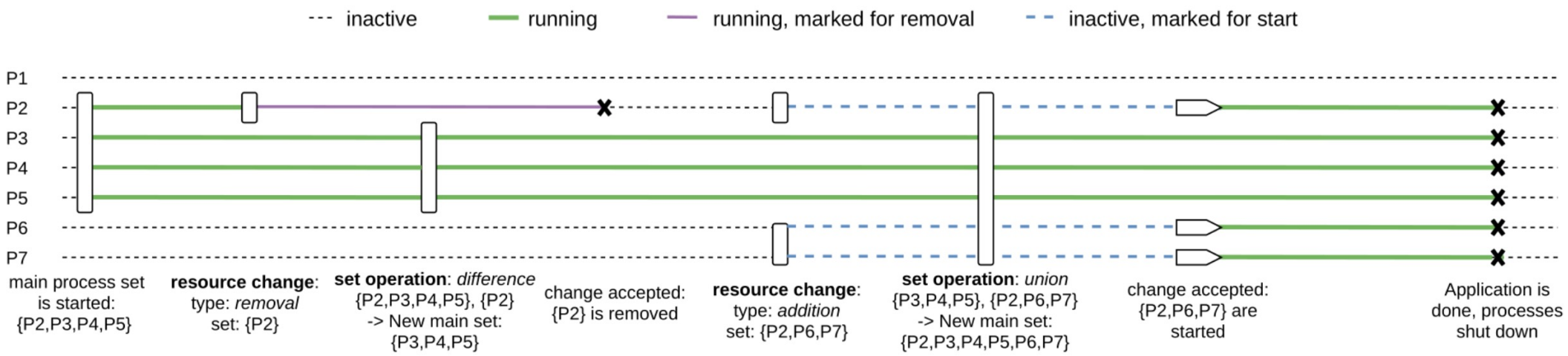
- Resource Change = **new process set** + **resource change type**
- The application:
  1. **polls** for a resource change
  2. does **load balancing**
  3. **accepts** the resource change
- Optional **info object** for passing information to new processes

```
int MPIDYNRES_RC_get(
    MPI_Session session,
    MPIDYNRES_RC_type *o_rc_type,
    char o_diff_pset_name[],
    MPIDYNRES_RC_tag *o_tag,
    MPI_Info *o_info
);
```

```
int MPIDYNRES_RC_accept(
    MPI_Session session,
    MPIDYNRES_RC_tag i_tag,
    MPI_Info i_info
);
```

From: Jan Fecht, TUM – published at HPCMALL 2022

Example for Malleability



Creation of new Psets

- New routines are an option for this

Processes come and go

- Need to intercept “adjustment phase”
- Need to see if application accepts the changes

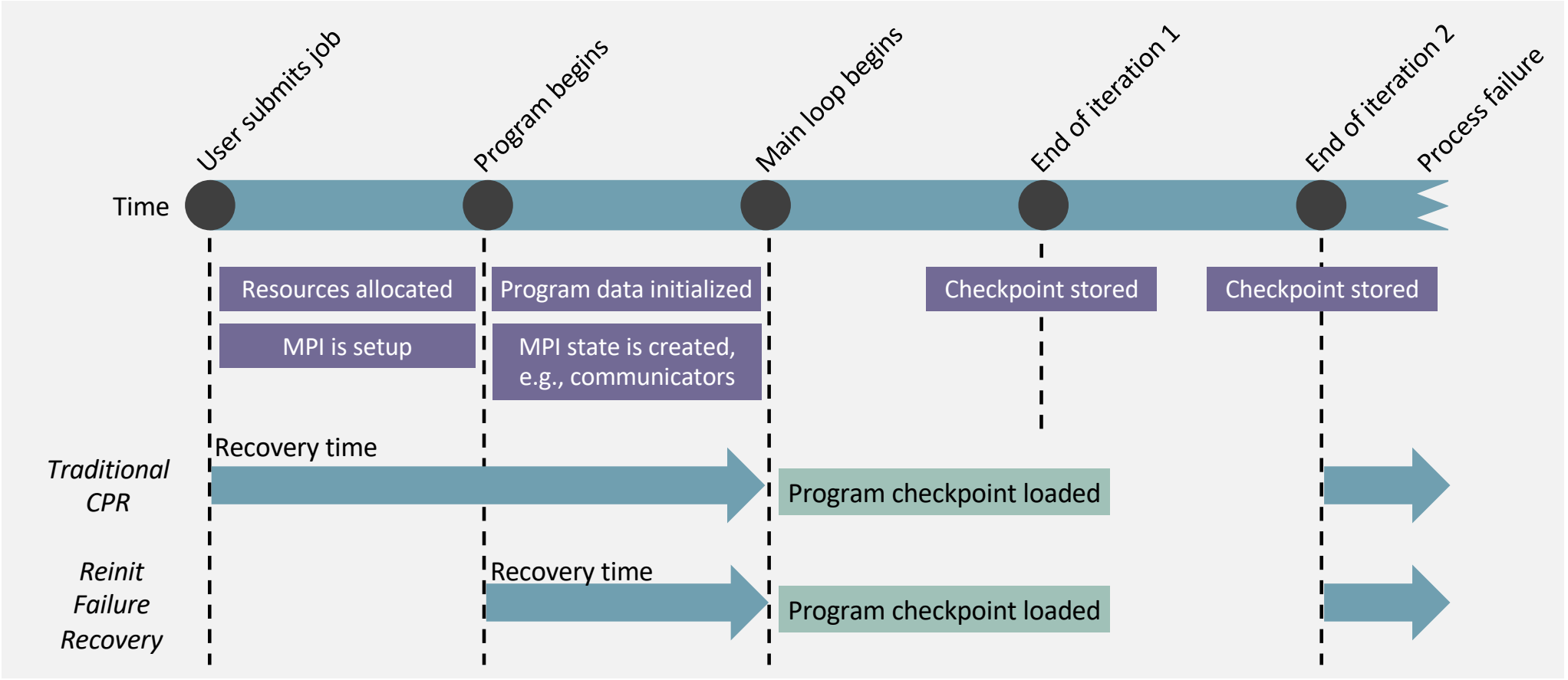
From: Jan Fecht, TUM – published at HPCMALL 2022

A different view on Malleability: Fault Tolerance

MPI Forum is making a push (again) for fault tolerance

- Multiple options/models on the table, tools will have to track that
- Implicitly changes to set of processes of applications
- Already changes to error handling to allow for continued execution

Coarse-grained Recovery (Reinit)



From: Ignacio Laguna, LLNL

A different view on Malleability: Fault Tolerance

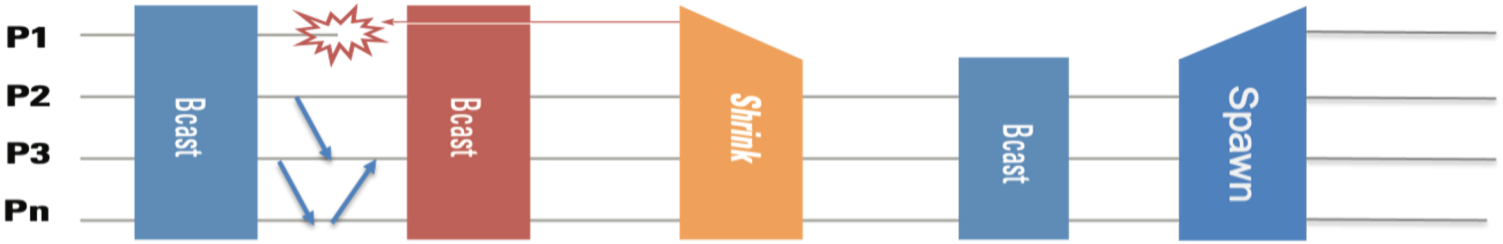
MPI Forum is making a push (again) for fault tolerance

- Multiple options/models on the table, tools will have to track that
- Implicitly changes to set of processes of applications
- Already changes to error handling to allow for continued execution

Option 1: Coarse-Grained Recovery (“Reinit” proposal)

- On failure in the system, automatic jump to start of main / No finalize
- Processes will be re-used so that data can be maintained
- But some processes may be restarted from scratch
- All MPI state is lost and has to be rebuilt (including tool state!)

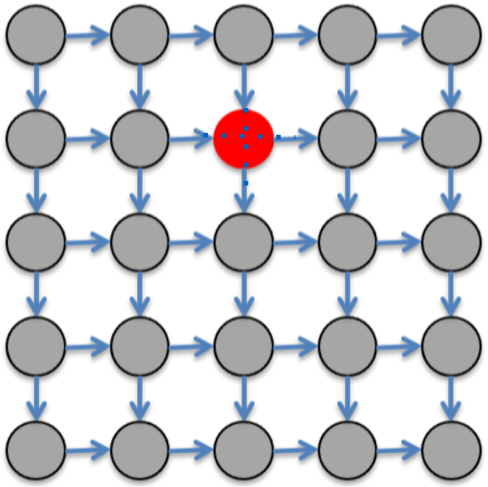
ULFM MPI Crash Recovery



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

- 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.



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

- 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

From: Aurelien Bouteiller, UTK

A different view on Malleability: Fault Tolerance

MPI Forum is making a push (again) for fault tolerance

- Multiple options/models on the table, tools will have to track that
- Implicitly changes to set of processes of applications
- Already changes to error handling to allow for continued execution

Option 1: Coarse-Grained Recovery (“Reinit” proposal)

- On failure in the system, automatic jump to start of main / No finalize
- Processes will be re-used so that data can be maintained
- But some processes may be restarted from scratch
- All MPI state is lost has to be rebuilt (including tool state!)

Option 2: Fine-Grained Recovery (“ULFM” proposal)

- Contiued, but limited operation after failure
- Tools would have to follow same limitations (and not change the state!)
- Communicator can be revoked, shrunk and deleted/replaced
- Mechanisms built on top of ULFM may need PMPI

And Now to Something Completely Different!

Partitioned Communication (MPI 4.0)

```
MPI_Psend_init(..., &request);
for (...) {
  MPI_Start(&request);
  #pragma omp parallel
  {
                kernel(..., request);
  }
  MPI_Wait(&request);
}
MPI_Request_free(&request);
```

```
Thread:

kernel(..., MPI_Request request) {
  int i = my_partition[my_id];
  /* Compute and fill partition i then mark ready: */
  MPI_Pready(i, request);
}
```

Next steps:

- Collective versions for partitioned communication

Current extension proposals focus on accelerators

- Optimizations to ensure buffers are “ready”
- Bindings should allow exection from the accelerator

Accelerator Bindings for MPI Partitioned APIs

CUDA and SYCL Language Bindings Under Exploration

```
int MPI_Psend_init(const void *buf, int partitions, MPI_Count count,
                   MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Info info,
                   MPI_Request *request)

int MPI_Precv_init(void *buf, int partitions, MPI_Count count,
                   MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Info info,
                   MPI_Request *request)

int MPI_[start,wait][_all](...)
```

Keep host only

Add device bindings

```
__device__ int MPI_Pready(int partition, MPI_Request request)

__device__ int MPI_Pready_range(int partition_low, int partition_high, MPI_Request request)

__device__ int MPI_Pready_list(int length, const int array_of_partitions[], MPI_Request request)

__device__ int MPI_Parrived(MPI_Request request, int partition, int *flag)
```

Interception options unclear

- Does PMPI work on this and, if so, how? Do we need it?
- How would this impact tools for other functions (e.g., RMA Put/Get)?

From: Work by Jim Dinan, NVIDIA and Maria Garzaran, Intel

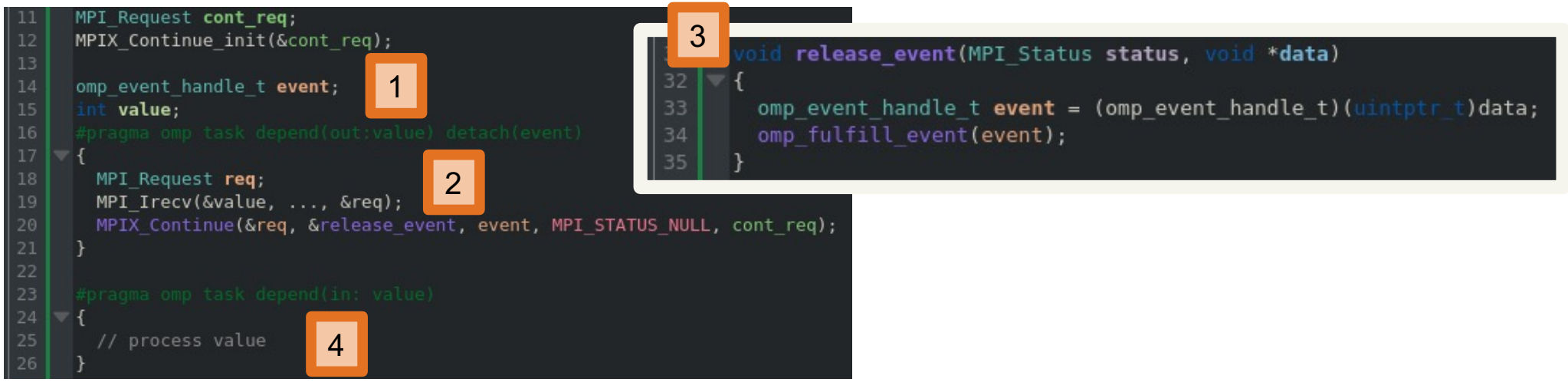
MPI Continuations

Opening Up MPI for Hybrid Runtimes

Several Proposals for Continuations

- Treat the completion of an MPI operation as continuation of some activity
- Ability to couple with OpenMP events and dependencies

Proposal for Thread Continuations



"Callback-based completion notification using MPI Continuations,"
Joseph Schuchart, Christoph Niethammer, José Gracia, George Bosilca, Parallel Computing, 2021.

"MPI Detach - Asynchronous Local Completion,"
Joachim Protze, Marc-André Hermanns, Ali Demiralp, Matthias S. Müller, Torsten Kuhlen. EuroMPI '20.

MPI Continuations

Opening Up MPI for Hybrid Runtimes

Several Proposals for Continuations

- Treat the completion of an MPI operation as continuation of some activity
- Ability to couple with OpenMP events and dependencies

Consequences for Tools

- Must track MPI and OpenMP
- Request tracking needs new approach
- May need wrapping of callback function (limitations tbd.)

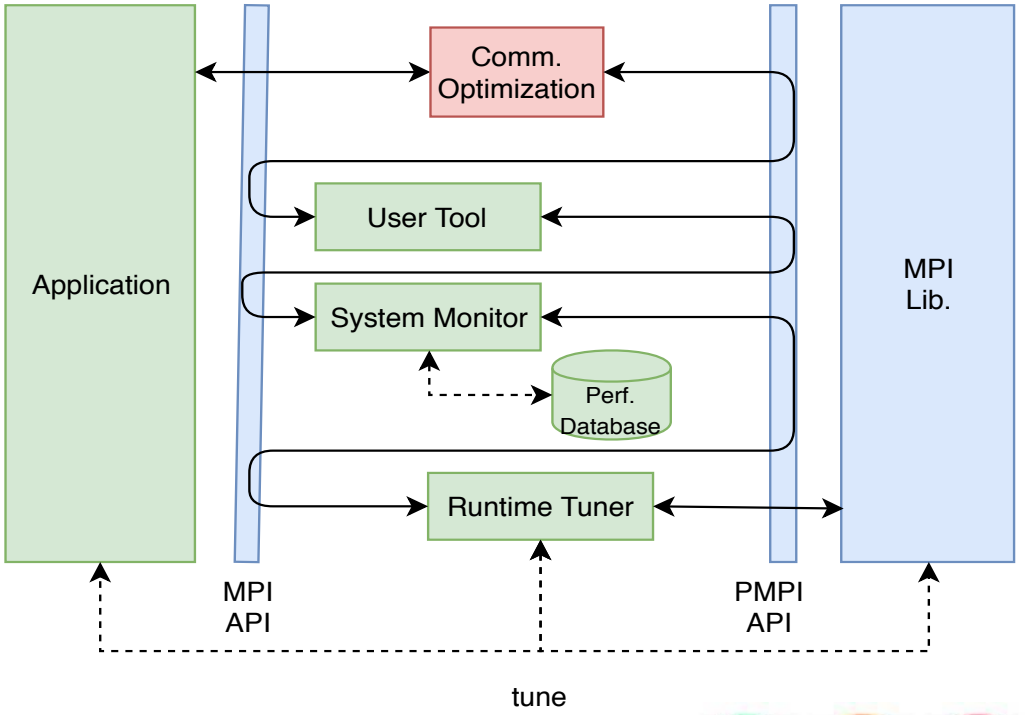
Good News (hopefully ☺): QMPI is Making Progress

Replacement idea for PMPI

- Multiple intercepts
- Runtime loading
- Options for user vs. system tools

Status

- Initial PMPI prototype
  (works on any MPI)
- Text proposal almost complete
- Prototype in MPICH available
- Open questions
  - Tool specification at runtime
  - How to deal with PMPI in transition period
  - Long term: dynamic tool loading



From: Bengisu Elis, TUM – published at EuroMPI 2019, MPICH Implementation by Wesley Bland when at Intel (now Meta)

MPI is Changing and Tools will Need to Adapt

Sessions change process tracking

- No global process set is guaranteed
- Process sets can change

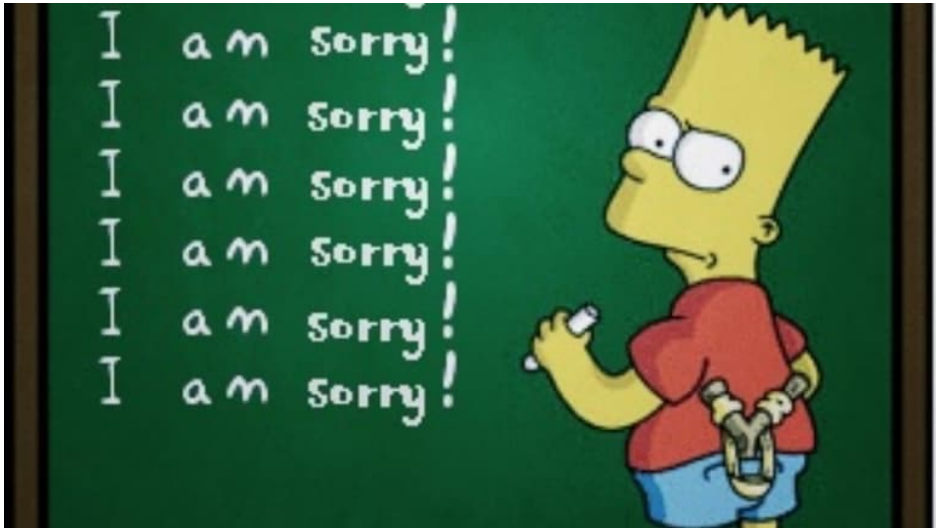
Malleability → Dynamic Tools

- Track process sets even within sessions
- Capture negotiations with runtime
- Match adjustment phases of apps
- Fault tolerance complicates things

Other new features likely on the way

- Accelerator bindings
  (especially for partitioned communication)
- Push towards hybrid runtimes

QMPI is making progress



MPI www.mpi-forum.org

DEEP-SEA
EU Grant #955606
BMBF #16HPC014

TIME-X
EU Grant #955701
BMBF #16HPC050

REGALE
EU Grant #956560
BMBF #16HPC039K

SPONSORED BY THE
Federal Ministry of Education and Research