The Message Passing Interface (MPI)
On the Path to MPI 5.0

Martin Schulz, Technische Universität München
Chair of the MPI Forum

Panelists:

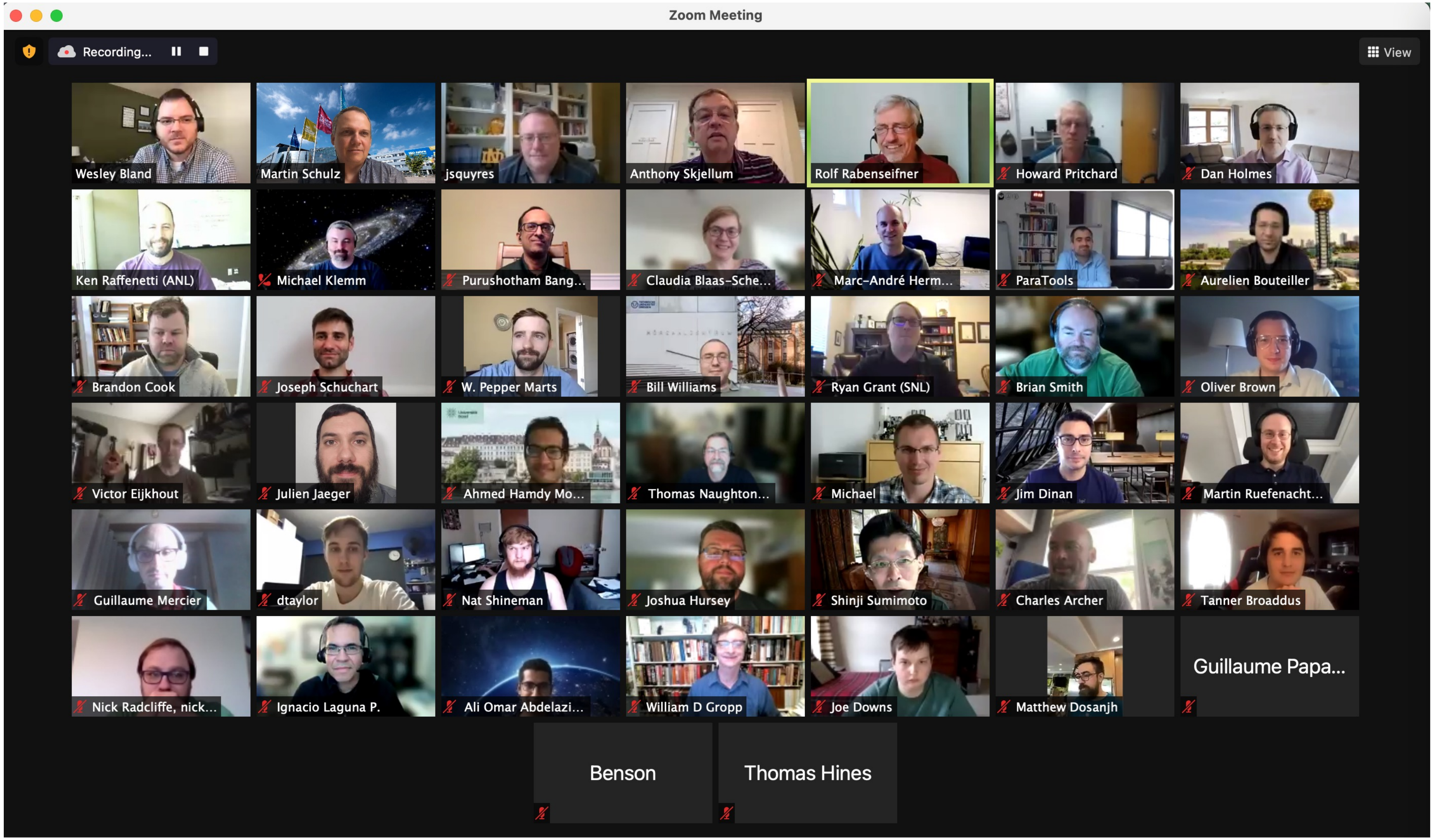
- Julien Jaeger, CEA
- Marc-André Hermanns, RWTH Aachen

+ the entire MPI Forum

ISC 2022 BoF, May 2022

MPI 4.0 got Ratified on June 9th 2021

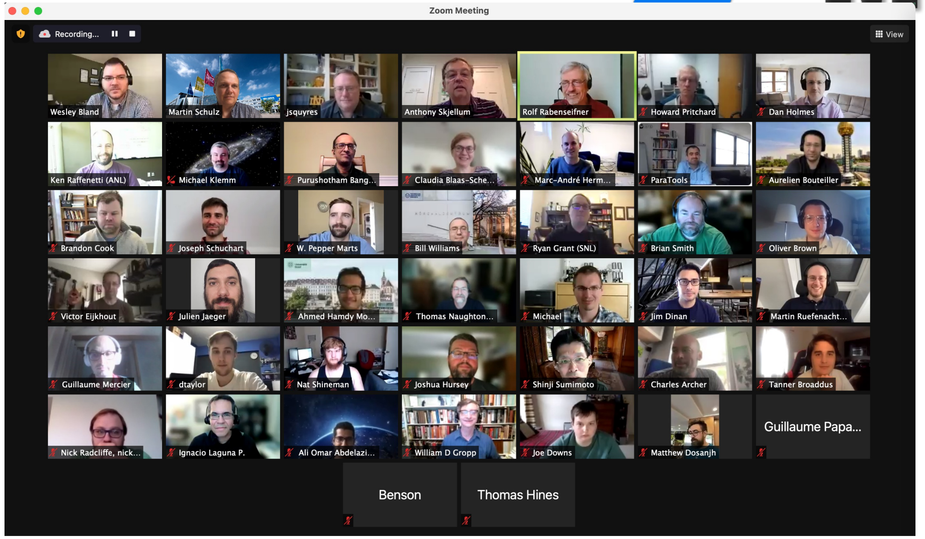
Available at http://www.mpi-forum.org/



MPI 4.0 (and what‘s Next)

Major additions for MPI 4.0

- Partitioned Communication
- New tool interface for events
- Solution for “Big Count” operations
- Persistent Collectives
- New init options via MPI Sessions
- Topology Solutions
- And much more …



MPI 4.0 Implementations in the Works

- The major implementations are already working towards MPI 4.0 some have complete support for full MPI 4.0 API
- In all major MPIS: several new core features already supported

The work of the MPI Forum Continues

- Next step: MPI 4.1 – minor changes/clarifications and cleanup/reorg
- Work on MPI 5.0 has begun as well
- http://www.mpi-forum.org/

Good Time to Join the MPI-Forum
The MPI-Forum is open to all interested in MPI.

The Bulk of Work is in the Working Groups

Collective Communication, Topology, Communicators, Groups
- Torsten Hoefler, Andrew Lumsdaine and Anthony Skjellum

Fault Tolerance
- Wesley Bland, Aurélien Bouteiller

HW Topologies
- Guillaume Mercier

Hybrid and Accelerator Programming
- Jim Dinan

Language Bindungs
- Martin Ruefenacht

Persistence
- Anthony Skjellum

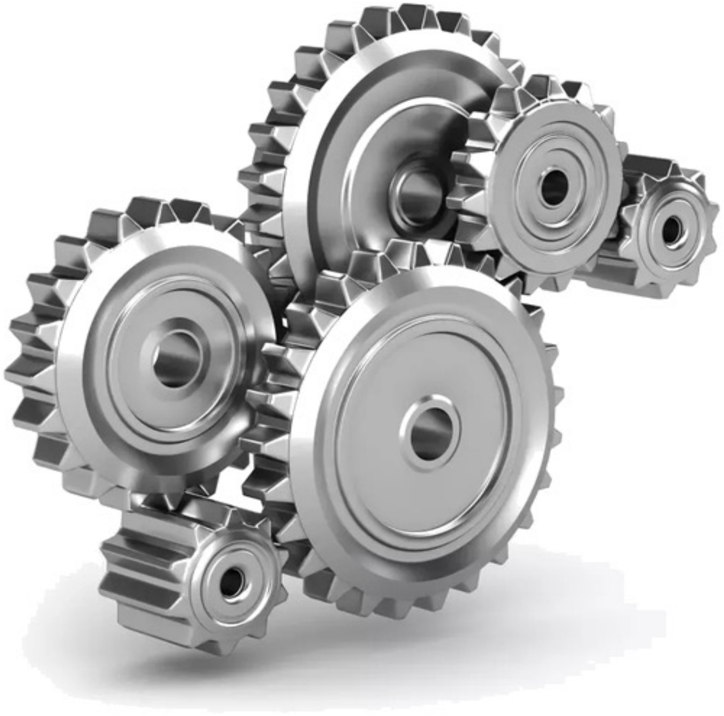
Point to Point Communication
- Rich Graham and Dan Holmes

Remote Memory Access
- Bill Gropp, Rajeev Thakur and Joseph Schuchart

Semantic Terms
- Rolf Rabenseifner and Purushotham Bangalore

Sessions
- Dan Holmes, Howard Pritchard

Tools
- Marc-Andre Hermanns



The Bulk of Work is in the Working Groups

Collective Communication, Topology, Communicators, Groups
- Torsten Hoefler, Andrew Lumsdaine and Anthony Skjellum

Fault Tolerance
- Wesley Bland, Aurélien Bouteiller

HW Topologies
- Guillaume Mercier

Hybrid and Accelerator Programming
- Jim Dinan

Language Bindungs
- Martin Ruefenacht

Persistence
- Anthony Skjellum

Point to Point Communication
- Rich Graham and Dan Holmes

Remote Memory Access
- Bill Gropp, Rajeev Thakur and Joseph Schuchart

Semantic Terms
- Rolf Rabenseifner and Purushotham Bangalore

Sessions
- Dan Holmes, Howard Pritchard

Tools
- Marc-Andre Hermanns

Partitioned/Collective Persistent Joint WG

ISC BOF of the MPI Forum

Slides by Tony Skjellum, 30-May-22

Examples of WG Activities

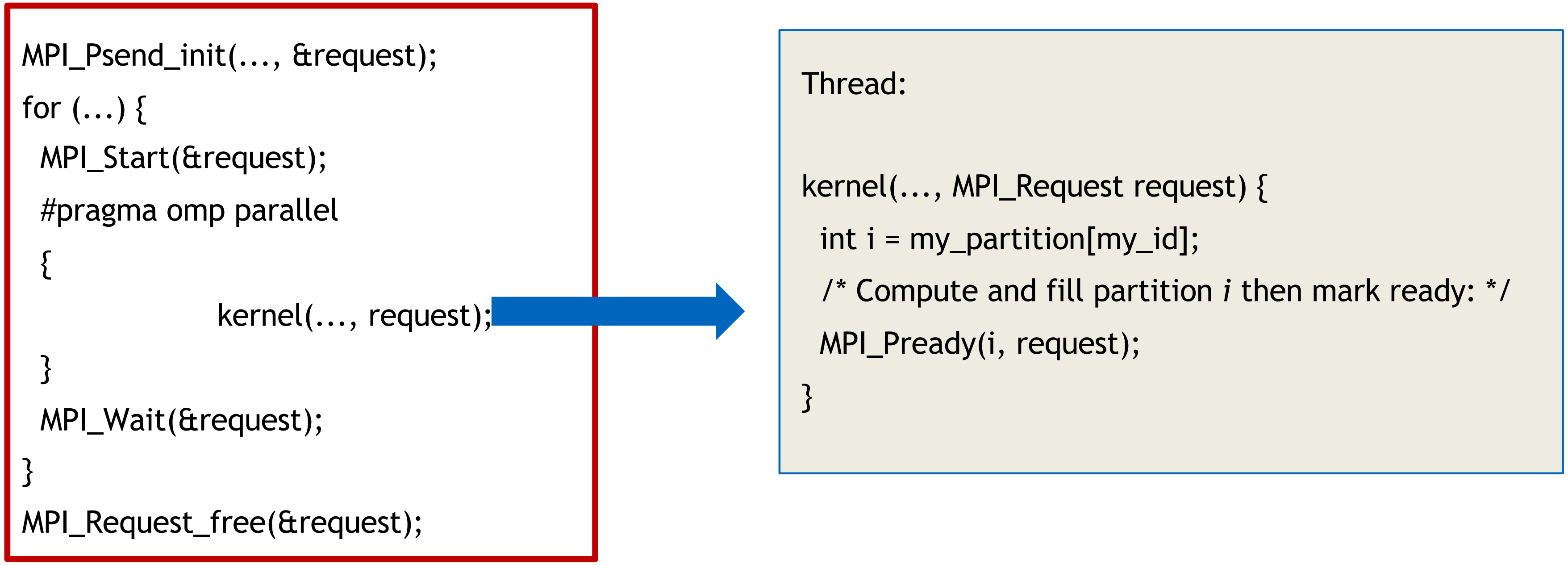
Towards MPI 4.1

- Assertion total ordering of persistent collectives across group at initialization – allows optimizations for some networks
  - By default, persistent collectives are not tied to same group-wide ordering rules as blocking(completing) and nonblocking collectives

Towards MPI 5.0

- Expanding the concept of persistent communication
  - Make it orthogonal and pervasive across the entire standard
- Channel or Buffer pools
  - Combined with lower-level interfaces for accelerator concerns
  - Includes better support for partitioned communication

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);
}
```

- Current extension proposals focus on accelerators
  - Optimizations to ensure buffers are “ready”
  - Bindings for CUDA and SYCL
- Further additions
  - Collective versions for partitioned communication

Hybrid and Accelerator WG

ISC BOF of the MPI Forum

Slides by Jim Dinan, 30-May-22

Meetings: Wed. 10-11am US Eastern Time

https://github.com/mpiwg-hybrid/hybrid-issues/wiki

Hybrid & Accelerator Working Group

Mission: Improve interoperability of MPI with other programming models

Active topics:

1. Supporting partitioned communication from accelerators
   - Partitioned communication buffer preparation [Ryan Grant, Queen's U.]
   - Accelerator bindings [Jim Dinan, NVIDIA + Maria Garzaran, Intel]

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

```
__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)
```

Add device bindings

Hybrid & Accelerator Working Group

Mission: Improve interoperability of MPI with other programming models

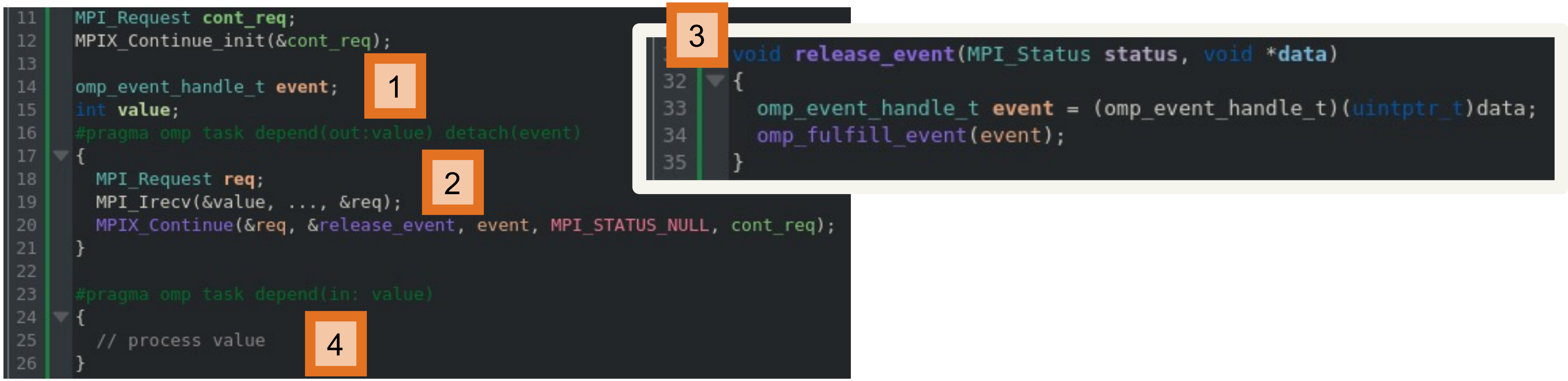
Active topics:

1. Supporting partitioned communication from accelerators
    - Partitioned communication buffer preparation [Ryan Grant, Queen’s U.]
    - Accelerator bindings [Jim Dinan, NVIDIA + Maria Garzaran, Intel]
2. Integration with accelerator programming models:
    - Accelerator info keys [Jim Dinan, NVIDIA]
    - Stream/Graph Based MPI Operations [Jim Dinan, NVIDIA]
3. Continuations proposal [Joseph Schuchart, UTK]

Proposal for Thread Continuations

Idea: Treat the completion of an MPI operation as continuation of some activity

Ability to couple with OpenMP events and dependencies



```c
MPI_Request cont_req;
MPIX_Continue_init(&cont_req);

omp_event_handle_t event;
int value;
#pragma omp task depend(out:value) detach(event)
{
  MPI_Request req;
  MPI_Irecv(&value, ..., &req);
  MPIX_Continue(&req, &release_event, event, MPI_STATUS_NULL, cont_req);
}

#pragma omp task depend(in: value)
{
  // process value
}
```

```c
void release_event(MPI_Status status, void *data)
{
  omp_event_handle_t event = (omp_event_handle_t)(uintptr_t)data;
  omp_fulfill_event(event);
}
```

"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.

Hybrid & Accelerator Working Group

Mission: Improve interoperability of MPI with other programming models

Active topics:

1. Supporting partitioned communication from accelerators
    - Partitioned communication buffer preparation [Ryan Grant, Queen's U.]
    - Accelerator bindings [Jim Dinan, NVIDIA + Maria Garzaran, Intel]
2. Integration with accelerator programming models:
    - Accelerator info keys [Jim Dinan, NVIDIA]
    - Stream/Graph Based MPI Operations [Jim Dinan, NVIDIA]
3. Continuations proposal [Joseph Schuchart, UTK]
4. Clarification of thread ordering rules [Daniel Holmes, Intel]

More information: https://github.com/mpiwg-hybrid/hybrid-issues/wiki

Fault Tolerance WG

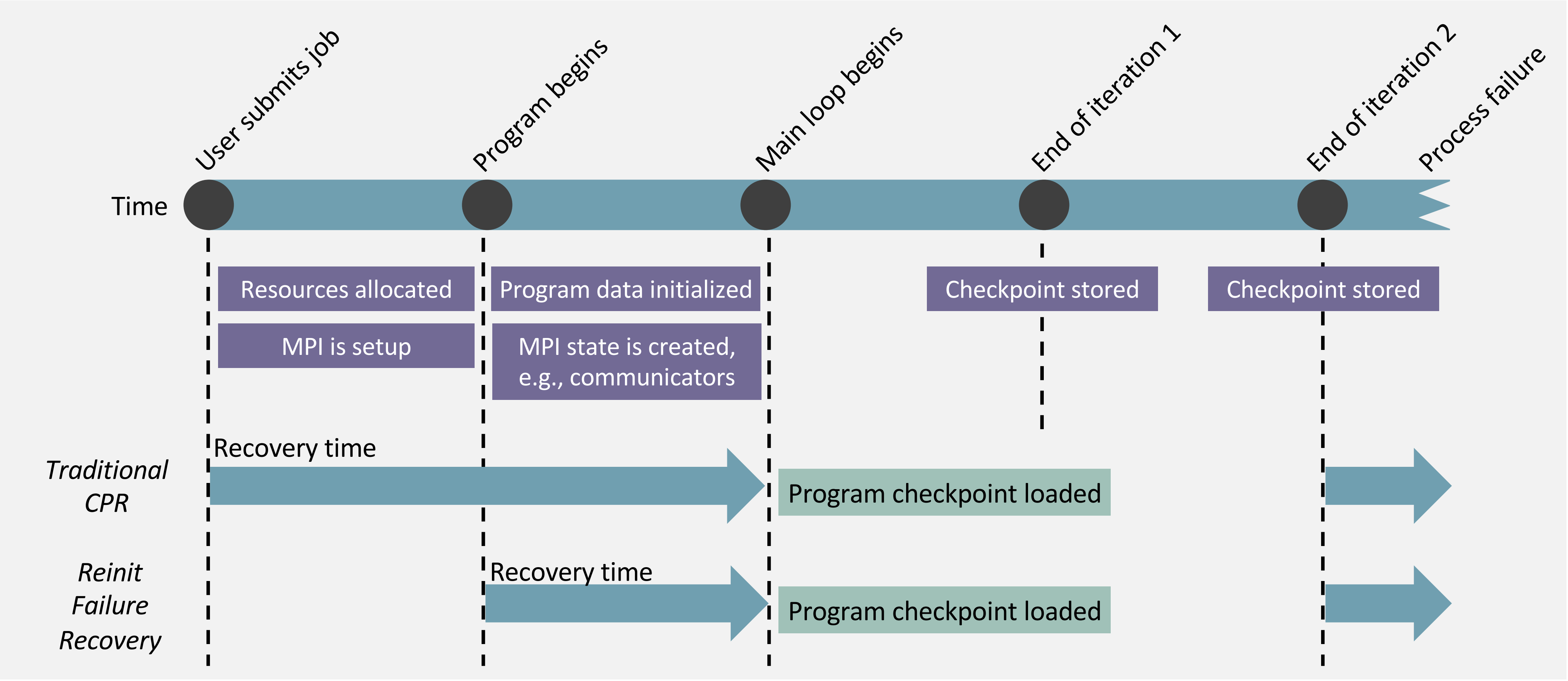
ISC BOF of the MPI Forum

Based on Slides by Aurelian Bouteiller, 30-May-22

FT WG Mission Statement

- Commissioned to work on fault tolerance.
- Work has expanded to include all error handling.
- The focus includes more than just the well-known ULFM proposal:
  - Finer control on what gets aborted after an error
  - Let programs fallback to TCP/other if MPI has an error; **increase the appeal to non-HPC folks**
  - Clarification of what the state of the MPI library should be after an error (i.e., **POSIX-like error handling**)
  - Consult on error management in new additions (MPI Sessions, MPI_INFO before MPI_INIT, etc.)

Coarse-grained Recovery (Reinit)


User submits job

Program begins

Main loop begins

End of iteration 1

End of iteration 2

Process failure

Time

Resources allocated

MPI is setup

Program data initialized

MPI state is created, e.g., communicators

Checkpoint stored

Checkpoint stored

Traditional CPR

Recovery time

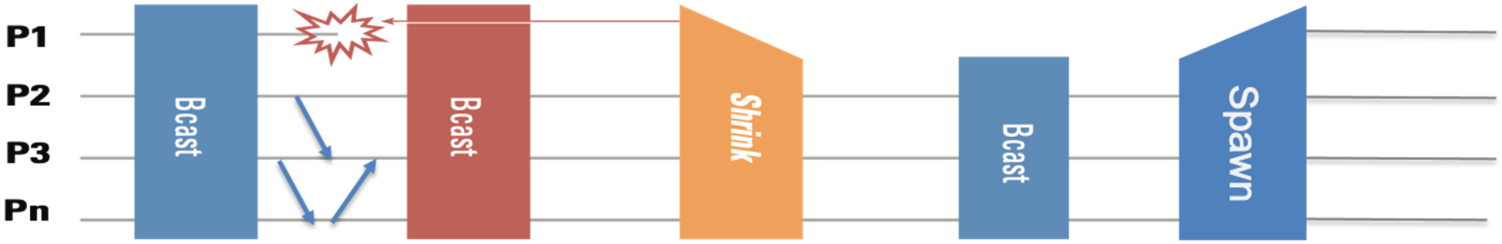
Program checkpoint loaded

Reinit Failure Recovery

Recovery time

Program checkpoint loaded

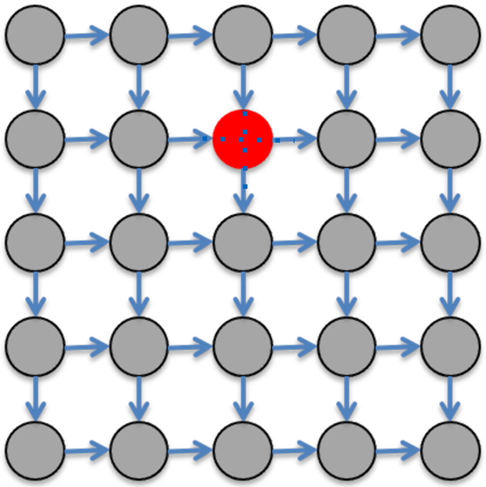
ULFM MPI Crash Recovery (Background)



- 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

Sessions WG

ISC BOF of the MPI Forum

WG leads: Howard Pritchard, Dan Holmes

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

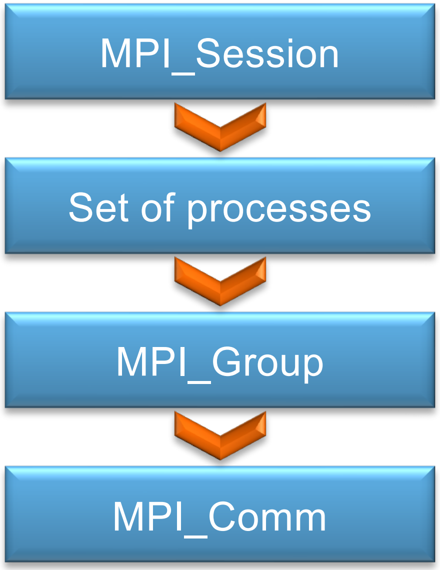
   Get a "set" of processes

3. Determine the processes you want

   Create an MPI_Group

4. Create a communicator with just those processes

   Create an MPI_Comm



MPI_Session

Set of processes

MPI_Group

MPI_Comm

What does this do?

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

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)

Active discussion in the MPI Forum Sessions WG – please join us

- Join us at the HPCMALL workshop on Thursday

HARDWARE TOPOLOGIES ISC BOF

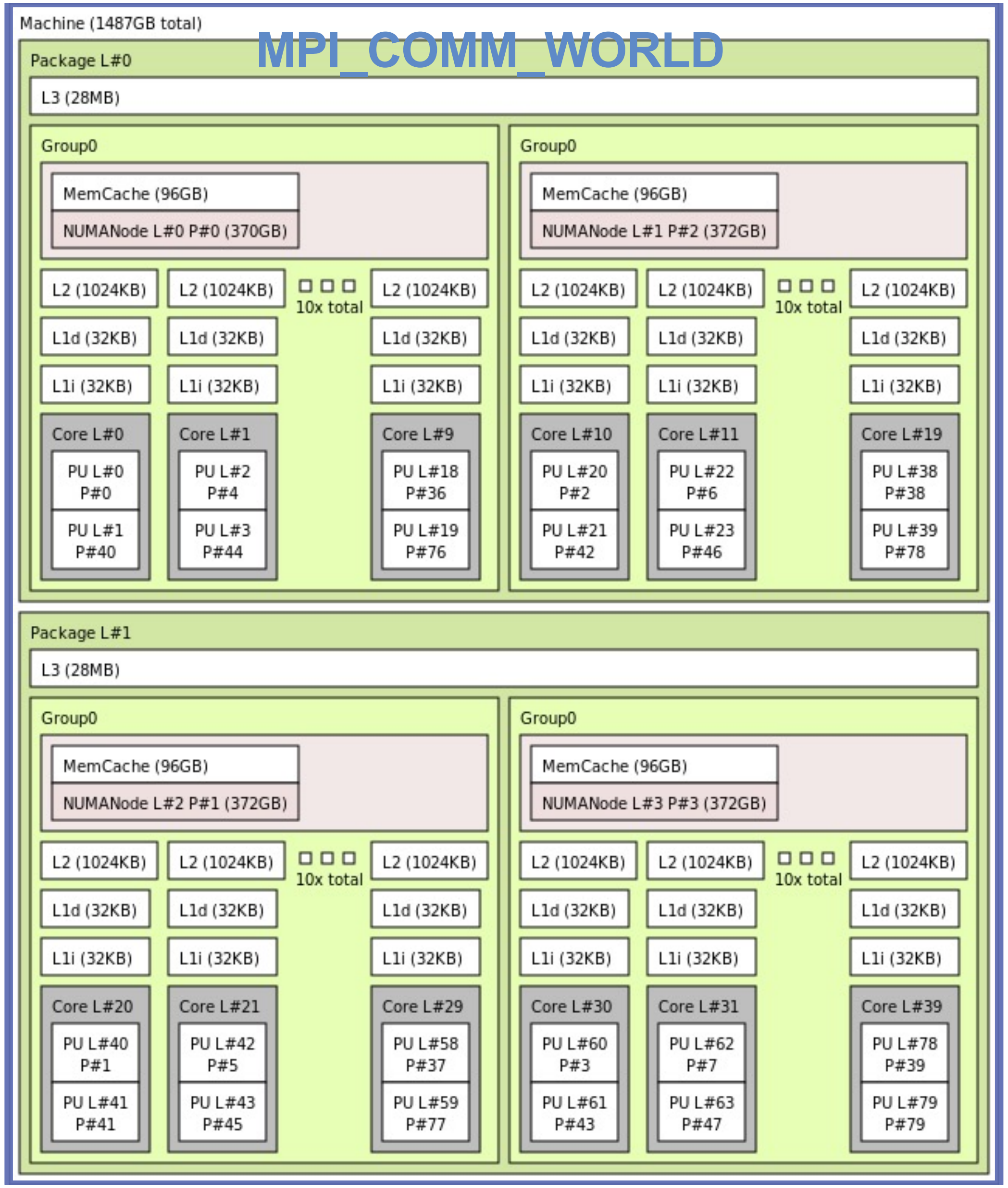
2022-05-31

Current MPI 4.0 status

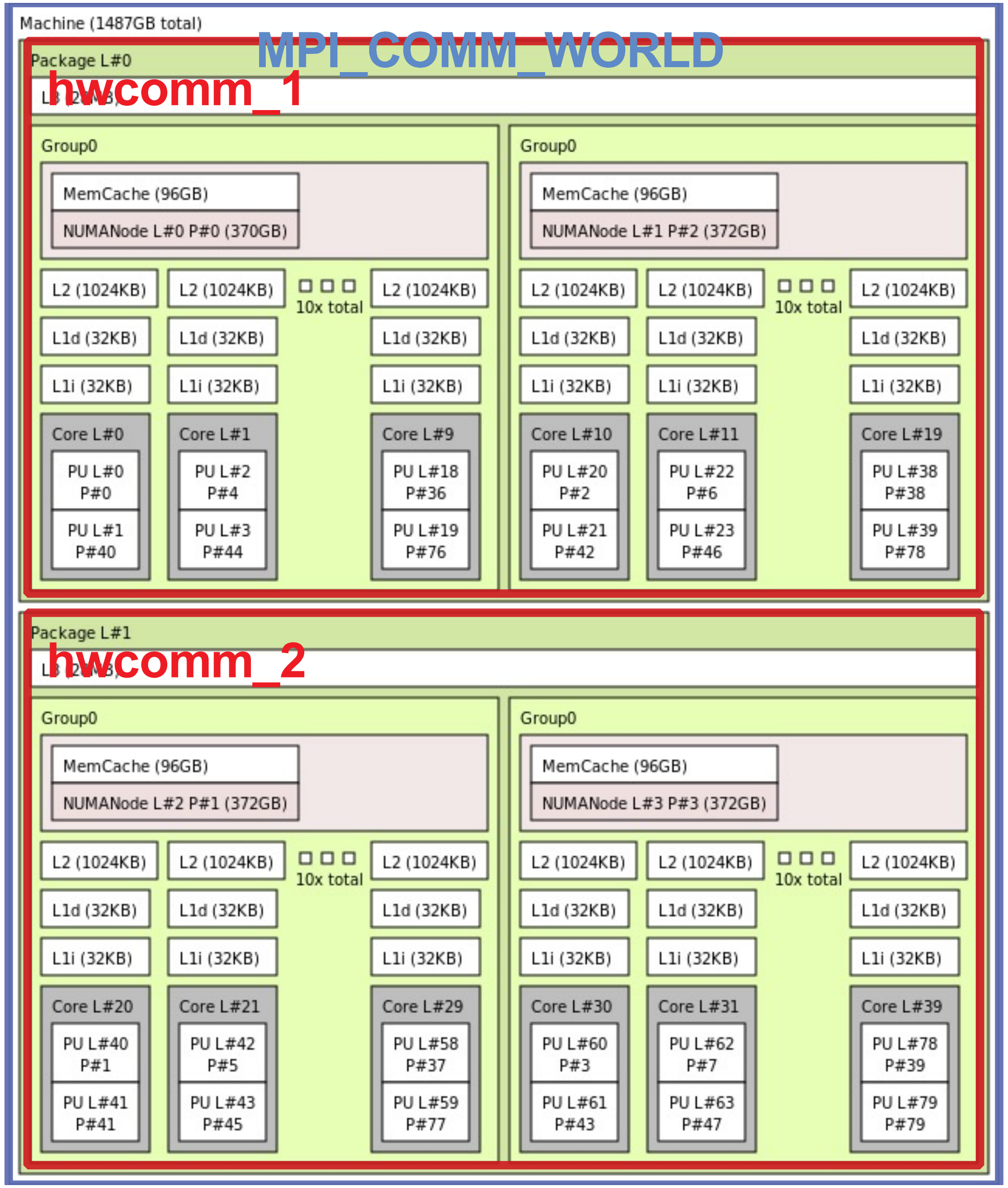
- Introduced two new split_types for MPI_Comm_split_type:
  - MPI_COMM_TYPE_HW_UNGUIDED
    - Splits to the “next” level in the hw hierarchy

```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split_type(MPI_COMM_WORLD,
           MPI_COMM_TYPE_HW_UNGUIDED,
           rank,info,&hwcomm);
```

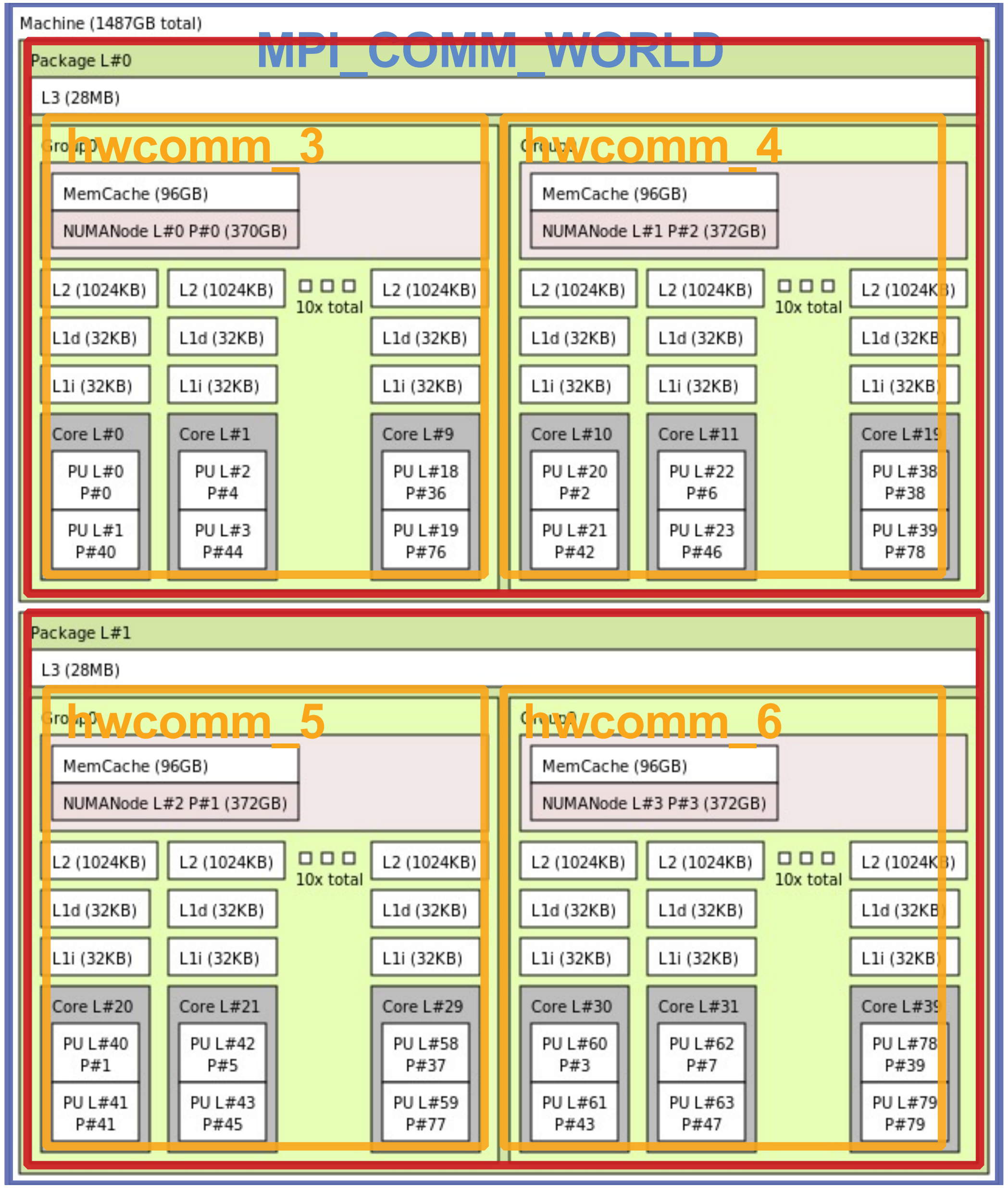
```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split_type(MPI_COMM_WORLD,
        MPI_COMM_TYPE_HW_UNGUIDED,
        rank,info,&hwcomm);
```



```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split_type(MPI_COMM_WORLD,
        MPI_COMM_TYPE_HW_UNGUIDED,
        rank,info,&hwcomm);
```



```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_split_type(MPI_COMM_WORLD,
        MPI_COMM_TYPE_HW_UNGUIDED,
        rank,info,&hwcomm);
```

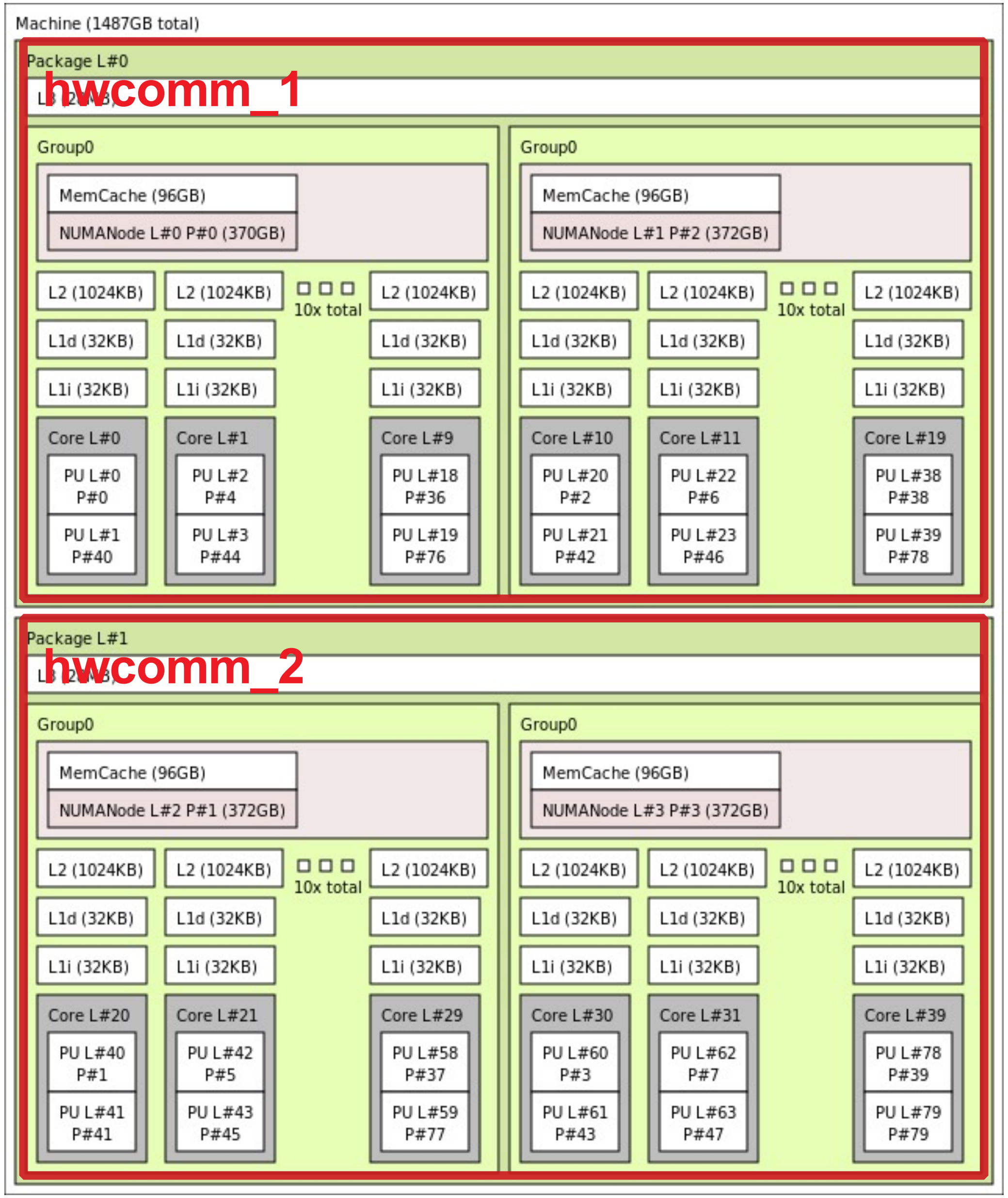


Current MPI 4.0 status

- Introduced two new split_types for MPI_Comm_split_type:
  - MPI_COMM_TYPE_HW_UNGUIDED
    - Splits to the “next” level in the hw hierarchy
  - MPI_COMM_TYPE_HW_GUIDED
    - Uses a new info key: mpi_hw_resource_type

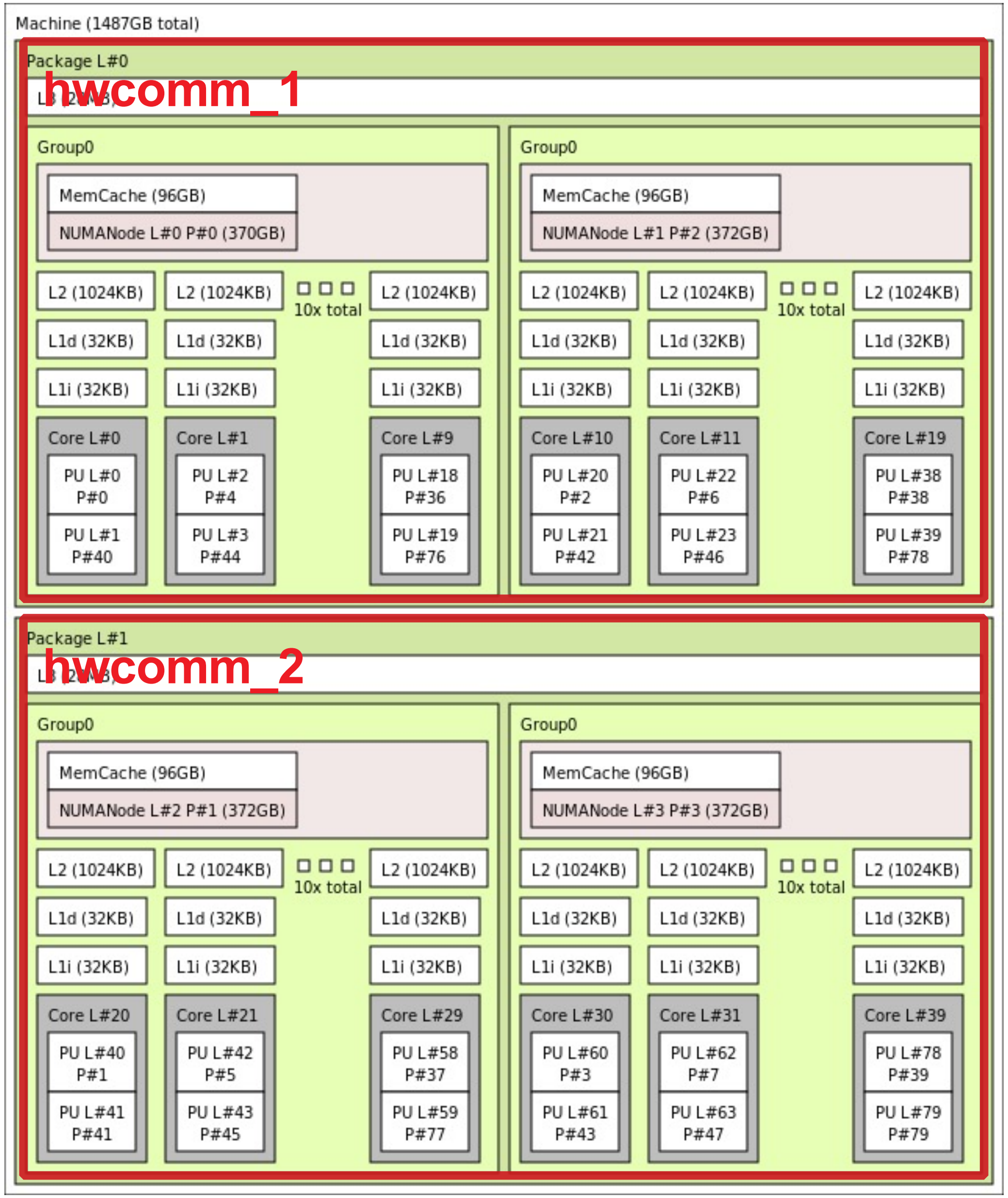
```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Info_create(&info);
MPI_Info_set(info,"mpi_hw_resource_type","NUMANode");
MPI_Comm_split_type(MPI_COMM_WORLD,
                    MPI_COMM_TYPE_HW_GUIDED,
                    rank,info,&hwcomm);
```

```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Info_create(&info);
MPI_Info_set(info,"mpi_hw_resource_type",
    "NUMANode");
MPI_Comm_split_type(MPI_COMM_WORLD,
    MPI_COMM_TYPE_HW_GUIDED,
    rank,info,&hwcomm);
```



```
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Info_create(&info);
MPI_Info_set(info,"mpi_hw_resource_type",
    "NUMANode");
MPI_Comm_split_type(MPI_COMM_WORLD,
    MPI_COMM_TYPE_HW_GUIDED,
    rank,info,&hwcomm);
```

- How can you know which “mpi_hw_resource_type” are available in your MPI implementation ?
- Can I split a communicator on something else than hw resource ?



Currently explored ideas

- **Query functions** to retrieve implementation-dependent info key values
  - Names are still not standardized
  - Standard way to access this piece of information
- **Introduce a new** MPI_COMM_TYPE_RESOURCE_GUIDED split_type value
  - Allows splitting according to **other kinds of resources**
  - New Info key: mpi_pset_name
  - Cohesive way to manage hw features in MPI
    - **Process sets** (MPI Sessions) can be used to achieve the same goal
      - **Bottom-up approach** vs. top-down (World Process Model)
    - A process set name can be that of a hw resource (e.g., hwloc://L3cache)
  - Allow to easily isolate “modules” in MPI applications

Long term items (MPI 5.0)

- **Explicit access** to underlying HW topology
  - Hardware communicators are implicit, topology is not described explicitly
  - Distance Functions
    - Determine criteria to take into account
- Support **memory types**
  - Same kind of support as in OpenMP?
- Dealing with **mapping and binding policies** for processes
  - mpirun/mpiexec standardized set of arguments?
  - Express and enforce policies at the MPI application level

Overhaul for Topologies and Collective Communication Interfaces

- **Three topological structures**
  - No virtual topological attached (ie fully connected + unweighted) → intracomms
    - Could be used to perform Processes to CPU mapping
  - Unweighted, bipartite → intercomms
  - Virtual topology attached (weighted, directed graph) → neighborhood intracomm
- **Remove Cartesian and (non-scalable) random graph interfaces**
- **The topological structure dictates the behaviour of collectives**
  - No communication allowed between non-neighbor processes
- **What about physical topologies?**
  - Fully-connected communicators used for process mapping
  - Introduce two classes of topologies (virtual vs. hw)

Tools WG

ISC BOF of the MPI Forum

Marc-André Hermanns, 30-May-22

Tools Working Group Topics

QMPI: Callback-driven interception of MPI calls

- Overcome single-tools restriction of PMPI interface
  - Retain full capabilities
- Allow dynamic registration of tools
  - Evolution of interface anticipated (future proofing)
- Allow hierarchy of tools
- Allow multiple registrations of the same tool
- Target: MPI 5.x
  - Text chapter in discussion
  - Corresponding reference implementation in MPICH

Tools Working Group Topics

Handle Introspection

- Provide Debugger with information about opaque MPI handles
  - Targeted support for all MPI handle types
  - Infer parent/child relationships
  - Infer state of handles
  - Provide handle specific information

- Ensure compatibility of debuggers for any MPI implementation
  - MPI implementation provides information library via standard interface

- Target: MPI 5.x
  - API currently still in design process

Tools Working Group Topics

Unique IDs for MPI_T entities

- Extend current MPI_T semantics
  - Mitiate zoo of MPI_T entities
- Provide reliable names for portable tools
  - Currently names are free to change
  - Semantics only communicated via text description
- IDs given for a specific semantic behavior
  - Entities of same ID must have same semantic behavior
    - Entities with same behavior may exist via multiple Ids
  - Implementors may reuse IDs for entities with same semantics
- Target: MPI 5.x
  - API currently still in design process

Tools Working Group Topics

Unique IDs for MPI_T entities

- Extend current MPI_T semantics
  - Mitiate zoo of MPI_T entities
- Provide reliable names for portable tools
  - Currently names are free to change
  - Semantics only communicated via text description
- IDs given for a specific semantic behavior
  - Entities of same ID must have same semantic behavior
    - Entities with same behavior may exist via multiple Ids
  - Implementors may reuse IDs for entities with same semantics
- Target: MPI 5.x
  - API currently still in design process

Tools Working Group Topics

Other Topics

- Additional calls for querying request statuses with a single call
  - ANY, SOME, ALL siblings to MPI_Request_get_status
  - Needs clarification of progress behavior first

- Further compound types for MINLOC and MAXLOC
  - Currently not all integer value types supported
  - Ongoing discussion on implementation path:
    - Type creation function vs. Static type definition

- Standardized MPI_T performance variable to query operation state
  - Needs discussion of differen operation states with semantics term group

  - API currently still in design process

The Message Passing Interface (MPI)
On the Path to MPI 5.0

Martin Schulz, Technische Universität München
Chair of the MPI Forum

Panelists:

- Julien Jaeger, CEA
- Marc-André Hermanns, RWTH Aachen

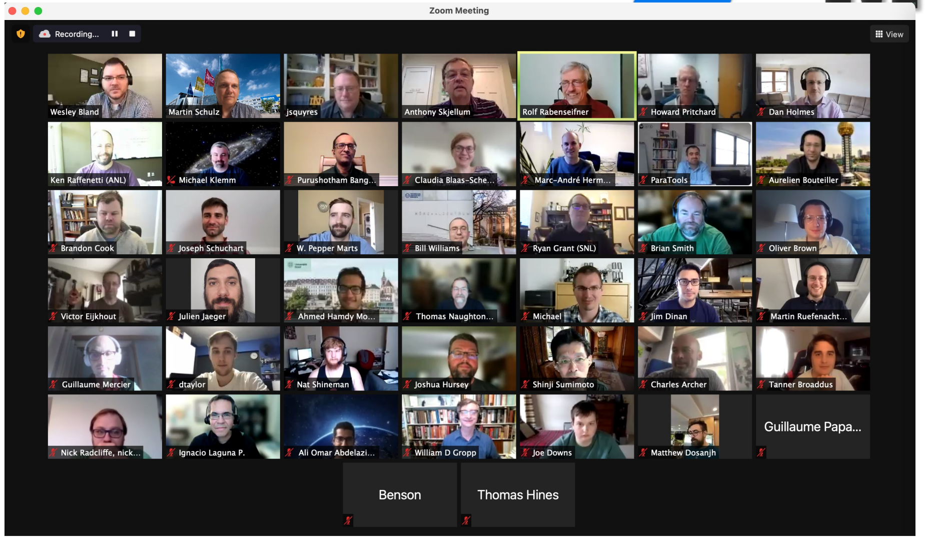
+ the entire MPI Forum

ISC 2022 BoF, May 2022

MPI: Version 4.0 and Beyond – Q&A

Major additions for MPI 4.0

- Partitioned Communication
- New tool interface for events
- Solution for “Big Count” operations
- Persistent Collectives
- New init options via MPI Sessions
- Topology Solutions
- And much more …



MPI 4.0 Implementations in the Works

- The major implementations are already working towards MPI 4.0
- Several features already supported
- Full support across most implementations soon

The work of the MPI Forum Continues

- Next step: MPI 4.1 – minor changes/clarifications and cleanup/reorg
- Work on MPI 5.0 has begun as well
- http://www.mpi-forum.org/

Good Time to Join the MPI-Forum
The MPI-Forum is open to all interested in MPI.

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 (Martin Schulz, TUM/LRZ)
- Secretary (Wesley Bland, Intel)
- Treasurer (Brian Smith, ORNL)
- Editor (Bill Gropp, UIUC/NCSA)

Open membership

- Any organization is welcome to participate
- Consists of working groups and the actual MPI forum (plenary)
- Voting (plenary) meetings 4 times each year (3 in the US, one with EuroMPI/Asia/USA)
- Voting rights depend on attendance