The Next Generation of the Message Passing Interface: MPI 4.0

Martin Schulz, Technische Universität München
Chair of the MPI Forum

Ryan Grant, Sandia National Laboratories
Chapter Chair for Partitioned Communication

Marc-André Hermanns, RWTH Aachen
Working Group Chair for Tools

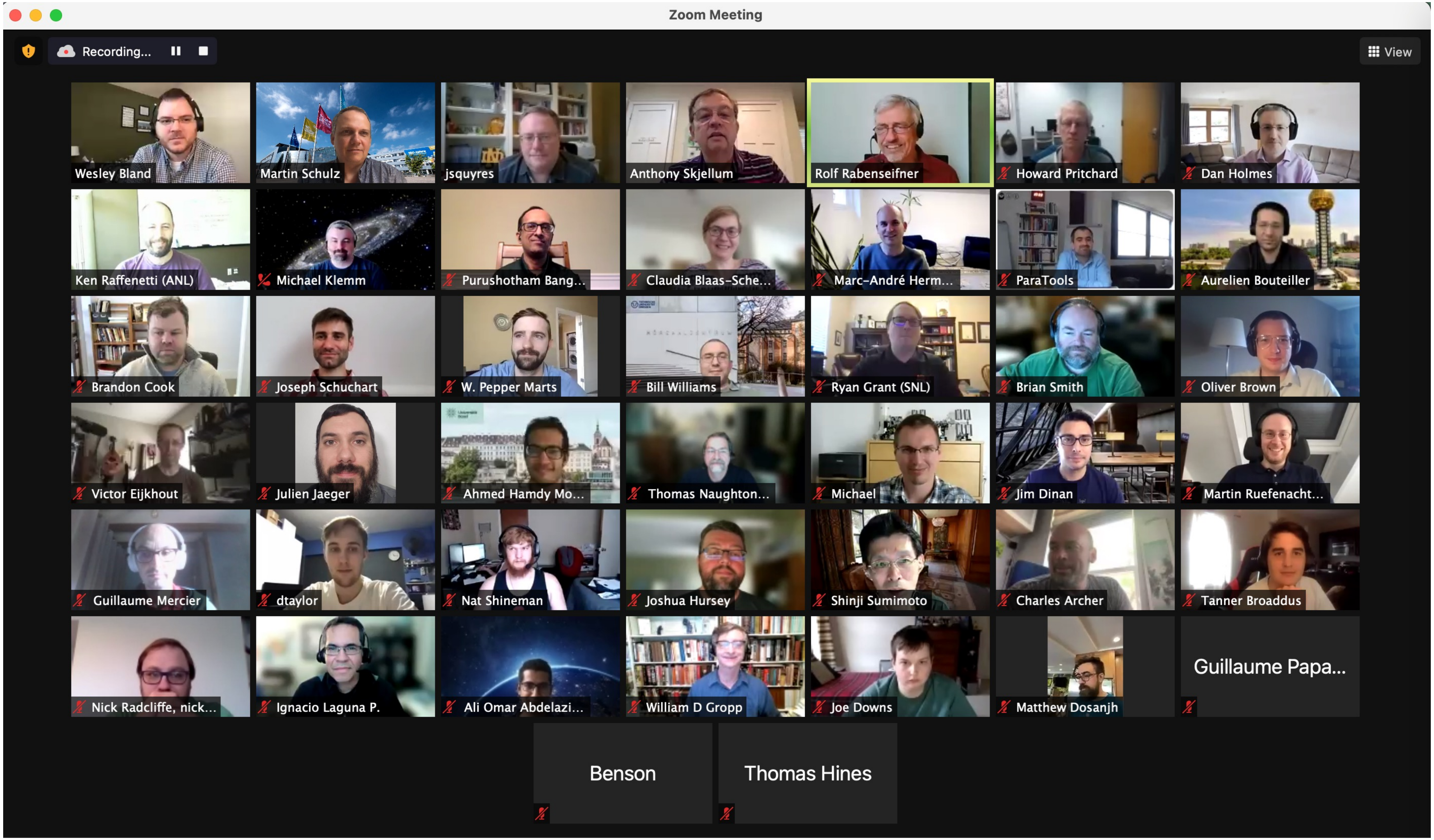
Anthony Skjellum, University of Tennessee at Chattanooga
Working Group Chair for Persistent Communication

+ the entire MPI Forum

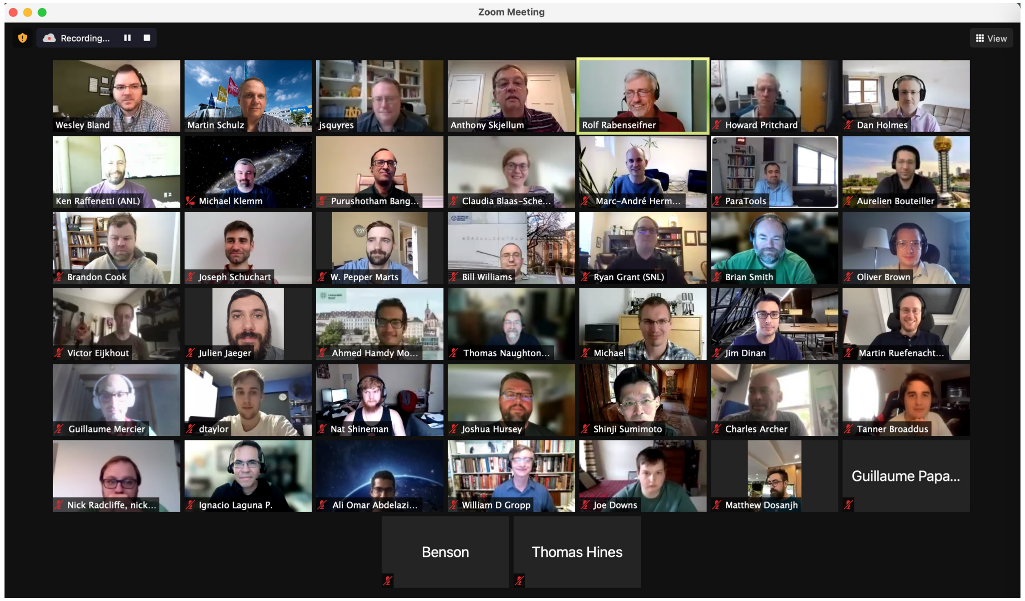
ISC 2021 BoF, June 2021

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
- Several features already supported
- Full support expected by late fall

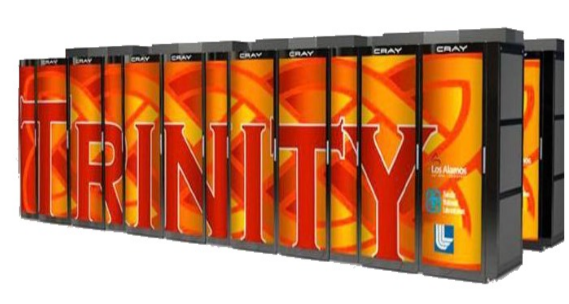
The work of the MPI Forum Continues

- 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.

MPI Partitioned Communication

MPI BoF – ISC 2021



PRESENTED BY

Ryan E. Grant

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Images courtesy of: Sandia National Laboratories, Cray Inc.,

SAND2021-7618 PE

MPI Partitioned Communication Concepts

- Many actors (threads) contributing to a larger operation in MPI
  - Same number of messages as today!
- Many threads/actors work together to assemble a message
  - MPI only has to manage knowing when completion happens
- Persistent-style communication
  - Init…(Start…test/wait)…free
- No heavy MPI thread concurrency handling required
- Triggering of data movement useful for GPU/accelerators
  - Coming improvements and support in MPI 4.1

How to use Partitioned MPI

- Like persistent communications, setup the operation

  int MPI_Partitioned_send_init(buf, etc….)

- Start the request

  int MPI_Start(MPI_Request request)

- Add items to the buffer

  int MPI_Pready(int partition, MPI_Request request)

  MPI_Pready is thread-safe and meant to be called from separate threads

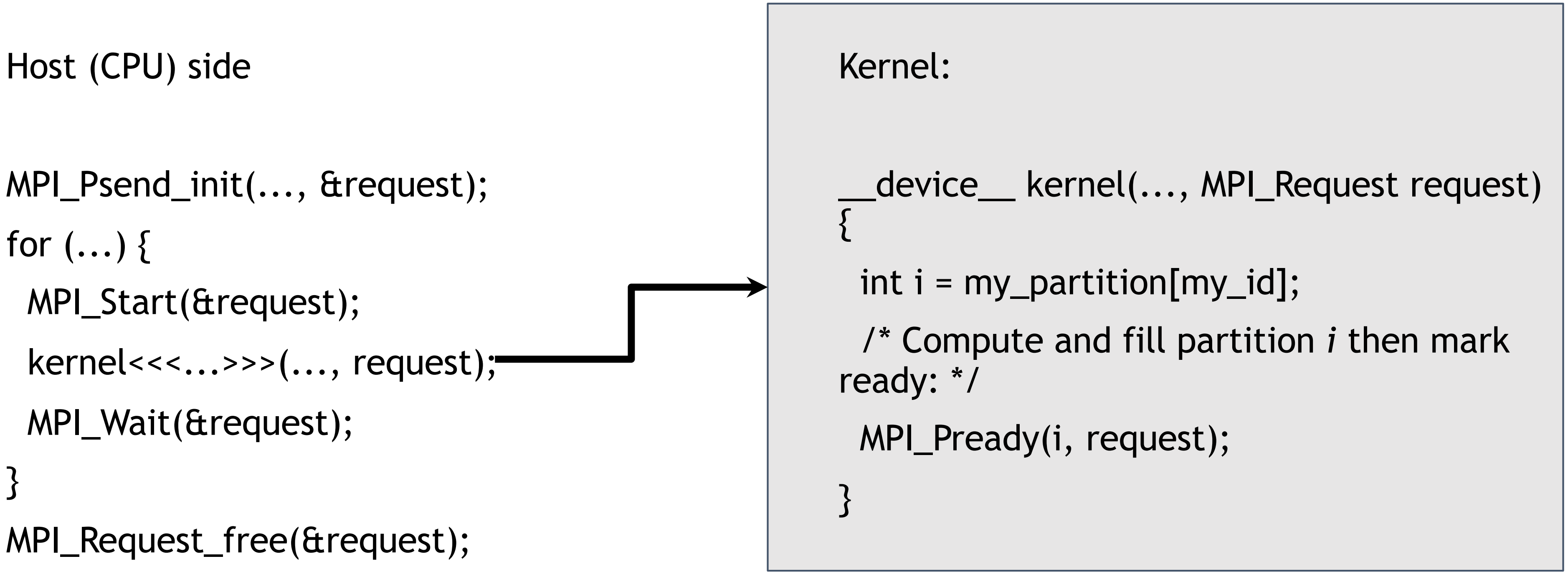
- Wait on completion

  int MPI_Wait(MPI_Request request)

- Optional: Use the same partitioned send over again

  int MPI_ Start(MPI_Request request)

Usage model - Kernel communication triggering



Host (CPU) side

```
MPI_Psend_init(..., &request);
for (...) {
  MPI_Start(&request);
  kernel<<<...>>>(..., request);
  MPI_Wait(&request);
}
MPI_Request_free(&request);
```

Kernel:

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

Note: CPU does communication setup and completion steps for MPI. Setup commands on NIC and poll for completion of entire operation. Kernel just indicates when NIC/MPI can send data. Ideally want to trigger communication from GPU to fire off when data is ready without communication setup/completion in kernel

Pbuf_prepare/Psync Example

MPI_PSEND_INIT
MPI_START
MPI_PBUF_Prep (blocking)
MPI_PREADY…(nonblocking)
MPI_WAIT (completing)

MPI_START, MPI_PSYNC
MPI_PREADY...MPI_PREADY
MPI_WAIT

MPI_PRECV_INIT
MPI_START
MPI_PBUF_Prep (blocking)
Optional - parrived (nonblocking)
MPI_WAIT (completing)

MPI_START, MPI_PSYNC
MPI_PARRIVED...MPI_PARRIVED
MPI_WAIT

In discussion for MPI 4.1

for Tools

Dr. Marc-André Hermanns
RWTH Aachen University

MPI 4.0 BoF @ ISC 2021

MPI_T Events: Callback-driven event information

Motivation

- PMPI does not provide access to MPI internal state information
- MPI_T performance variables only show aggregated information

New interface to query available runtime event types

- Follows the MPI_T variable approach
- No specific event types mandated
- Event structure can be inferred at runtime

Register callback functions to be called by the MPI runtime

- Runtime may defer callback invocation (tool can query event time)
- Runtime may reduce restrictions on callback functions per invocation
- Callback can query event information individually or copy data en bloc

Count Solution”

Prof. Anthony Skjellum
University of Tennessee at Chattanooga

MPI 4.0 BoF @ ISC 2021

Persistent Collectives

Following the basic ideas of persistent point to point

- One-time initialization to pass all arguments, which returns a request
- Use of this request to start communication
- Completion using Test/Wait
- Reuse request to restart the operation as often as one wants

Available for all MPI collective communication operations (and barriers)

Benefits

- Specify repeated operations
- Ability to lock down resources and to cache execution plan
- Performance optimization after (small) 1x cost
- Allows for continuous plan optimization

Big Count aka. Embiggenment

Problem: in previous interface “count” arguments are “int”

- Limits communication volumes to 32bit x Datatype
- Significant number of applications need more
- Initial datatype “trick” no longer sufficient

Solutions discussed included:

- Just changing “int” arguments to “MPI_Count” arguments → ☹ ☹ ☹
- Polymorphic bindings → ☹ ☹
- Duplication of interfaces: with int and with MPI_Count (“_c” suffix) → ☹

Last option was selected

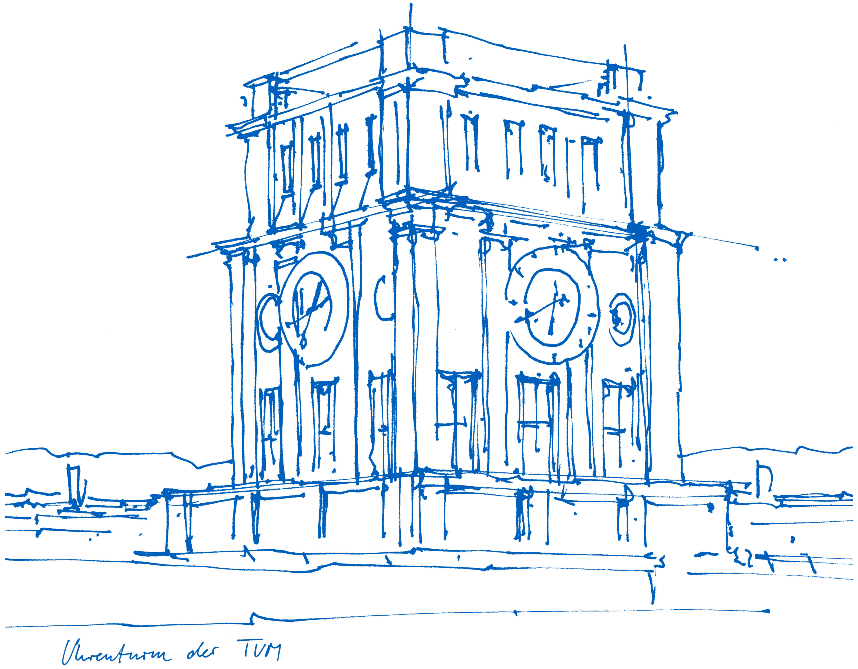
- Update of the general type rules for bindings
- Verification of all bindings, which led to errata tickets
- Addition of many new routines with “_c”

Additional features in MPI 4.0:

MPI Sessions and More

Prof. Martin Schulz
TU Munich

MPI 4.0 BoF @ ISC 2021



A New Way to Use MPI: MPI Sessions

Basic scheme

1. Get local access to the MPI library

   Get a Session Handle

2. Query the underlying run-time system

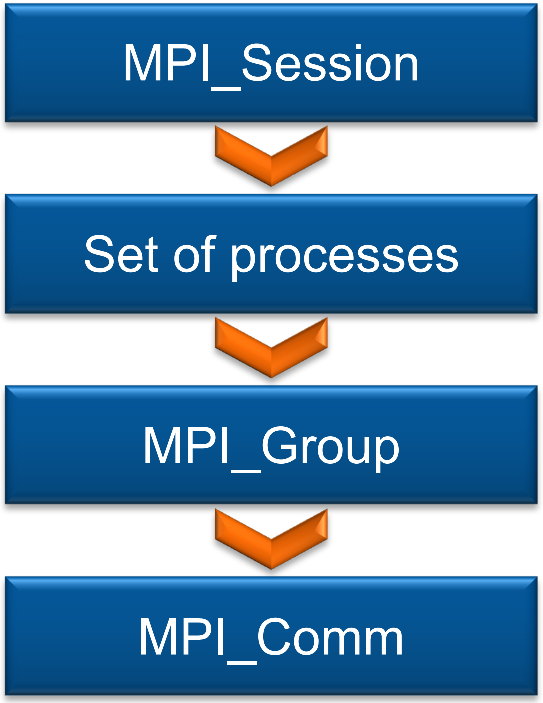
   Get a "set" of processes

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

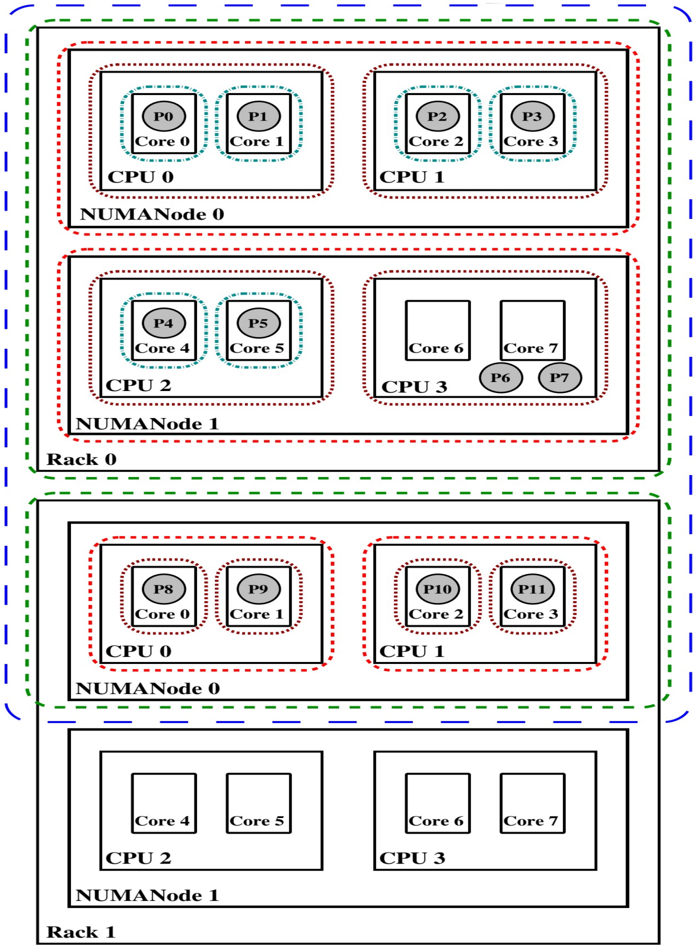
MPI_Group

MPI_Comm

MPI Session's intended goals

- No more implicit MPI_COMM_WORLD
- Enable runtime information to flow into MPI
- Creation of communicators without parent communicators
- Re-initialization of MPI
- Resource isolation
- Many future uses …

Other Additions



Assertions for message traffic to guide optimization

- Can state that an application doesn't use wildcards
- Enables traffic optimizations
- Great opportunities for implementations to optimize

HW Topology Functions

- Split communicators based on HW topologies
- Guided mode: along user given lines
- Unguided mode: detection of HW hierarchy

Better Error handling that allows:

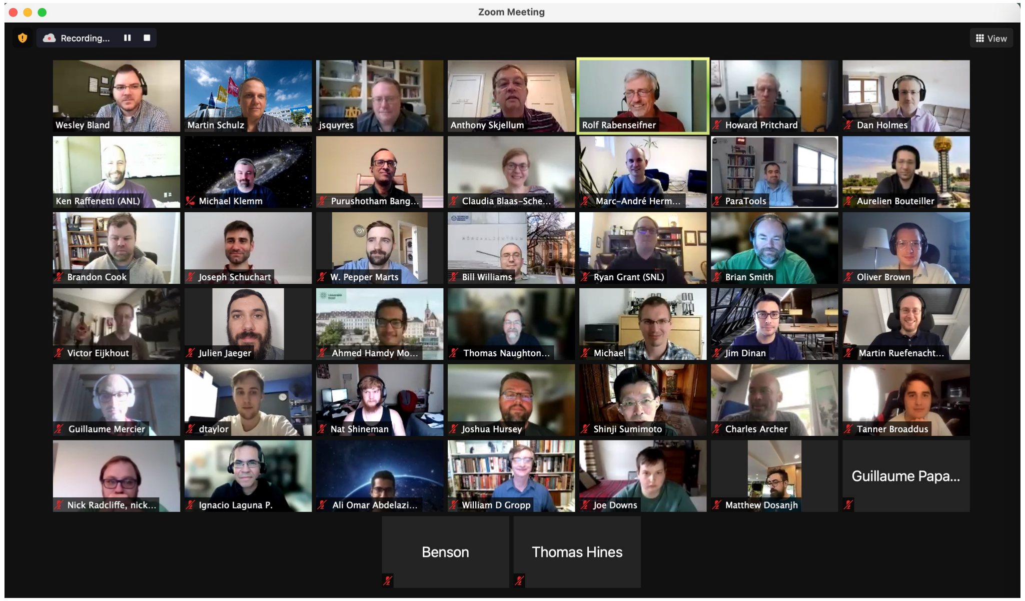
- Point to Point communication with sockets-like error handling
- Enables manager/worker and other non-traditional types of applications
- Enterprise applications that want to move from sockets to MPI can do so.

Access to MPI Info before MPI initialization (needed for Sessions, MPI_T, FT, …)

MPI 4.0 – Live 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 expected by late fall

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.