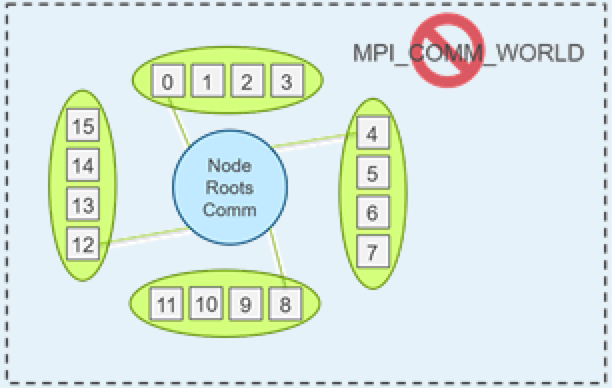
IMPLEMENTING TRUE MPI SESSIONS



EUROMPI 25

Oct. 2, 2025, Charlotte, NC, USA

Hui Zhou, Ken Raffenetti, Yanfei Guo, Michael Wilkins, and Rajeev Thakur
Argonne National Laboratory

BACKGROUND

MPI SESSIONS TUTORIAL

The World Model:

```
MPI_Init();

[MPI_COMM_WORLD]

MPI_Finalize();
```

The Sessions Model:

```
MPI_Session_init();

MPI_Group_from_session_pset();
MPI_Comm_create_from_group();

[On-demand communicators]

MPI_Session_finalize();
```

MPI SESSION TIMELINE

2016
Sep. 17

Holmes, D., et al.: *MPI Sessions: leveraging runtime infrastructure to increase scalability of applications at exascale*

2017
Sep. 28

Castain, R.H. et al.: *PMIx: Process Management for Exascale Environment*

2019
Sep. 25

Hjelm N. et al.: *MPI Sessions: Evaluation of an Implementation in Open MPI*

2021
Feb. 26

MPICH v4.0a1 is released.

2021
Jun. 9

MPI 4.0 standard is released, adding MPI Sessions.

2021
Jun. 21

MPICH v4.0a2 is released.

2021-2025

MPI Sessions adoptions, including resource management, fault tolerance, job malleability.

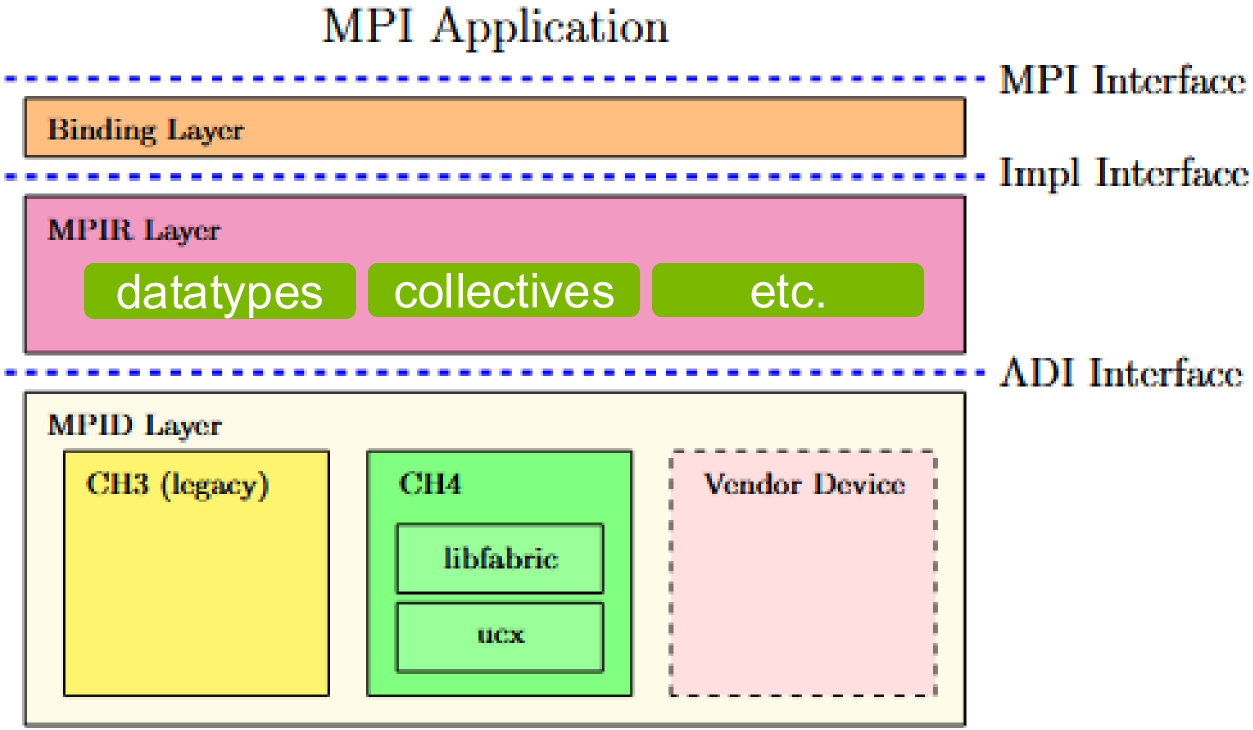
2025
Oct. 2

Zhou H. et al.: *Implementing True MPI Sessions and Evaluating MPI Initialization Scalability*

2025
Nov. tbd

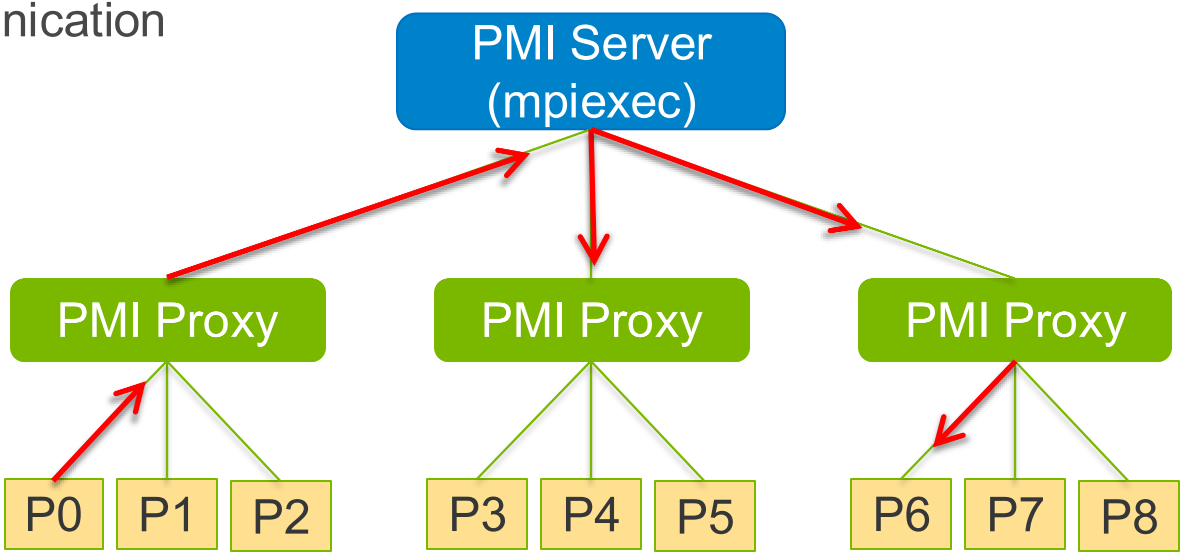
MPICH v5.0.0b1 planned release

MPICH ARCHITECTURE



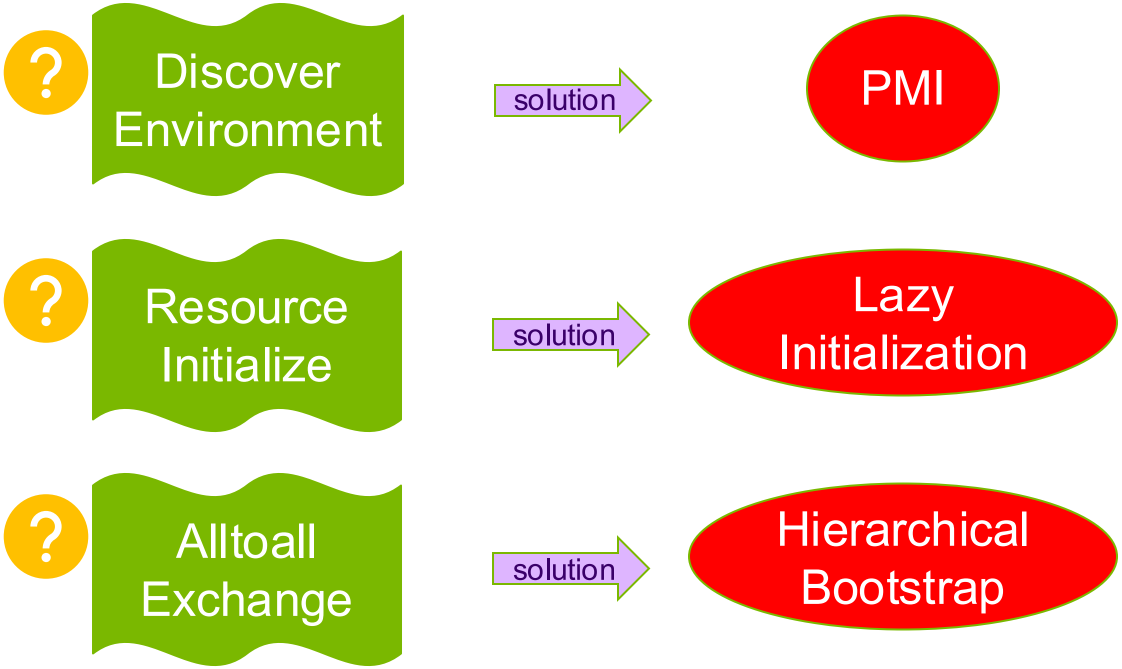
PROCESS MANAGEMENT INTERFACE (PMI)

1. Launching MPI processes
2. Initial process identification
3. Bootstrap communication
    1. PMI_Put
    2. PMI_Barrier
    3. PMI_Get



EFFICIENT MPI STARTUP

Bottlenecks In A Large-scale MPI Startup



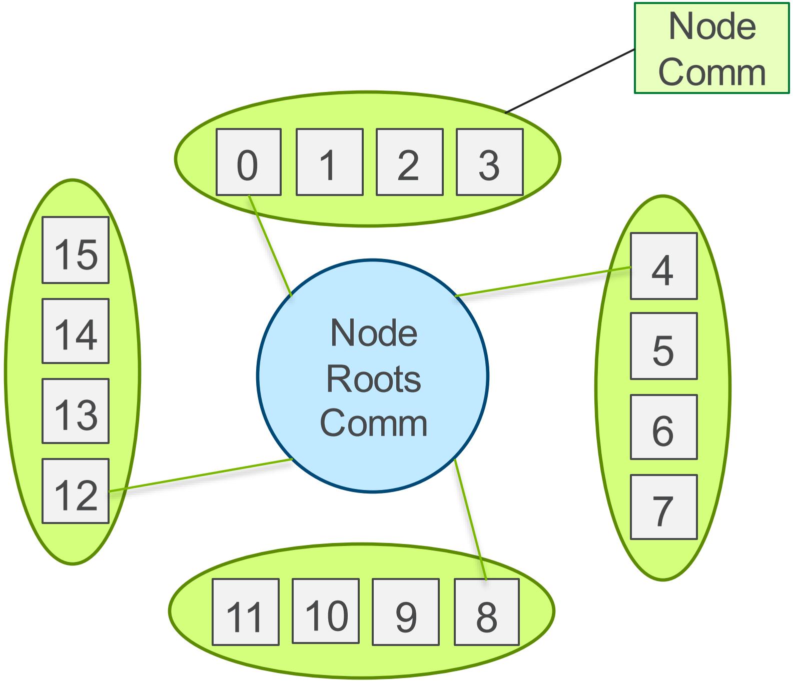
EFFICIENT ALL-TO-ALL ADDRESS EXCHANGE

Hierarchical bootstrapping

Step 1:
Use PMI to establish node-roots communicator

Step 2:
Use shared memory to establish intra-node communicator

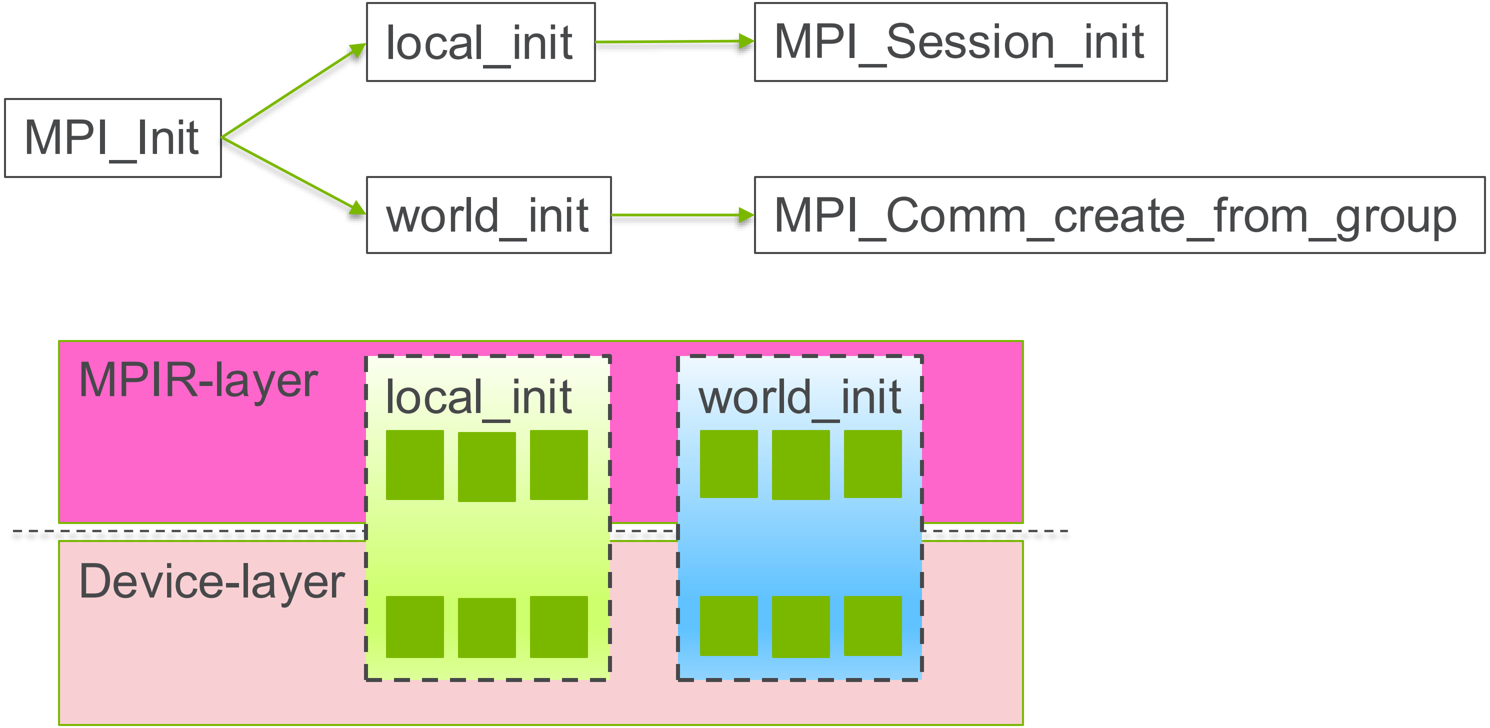
Step 3:
Perform hierarchical allgather using fast MPI communication



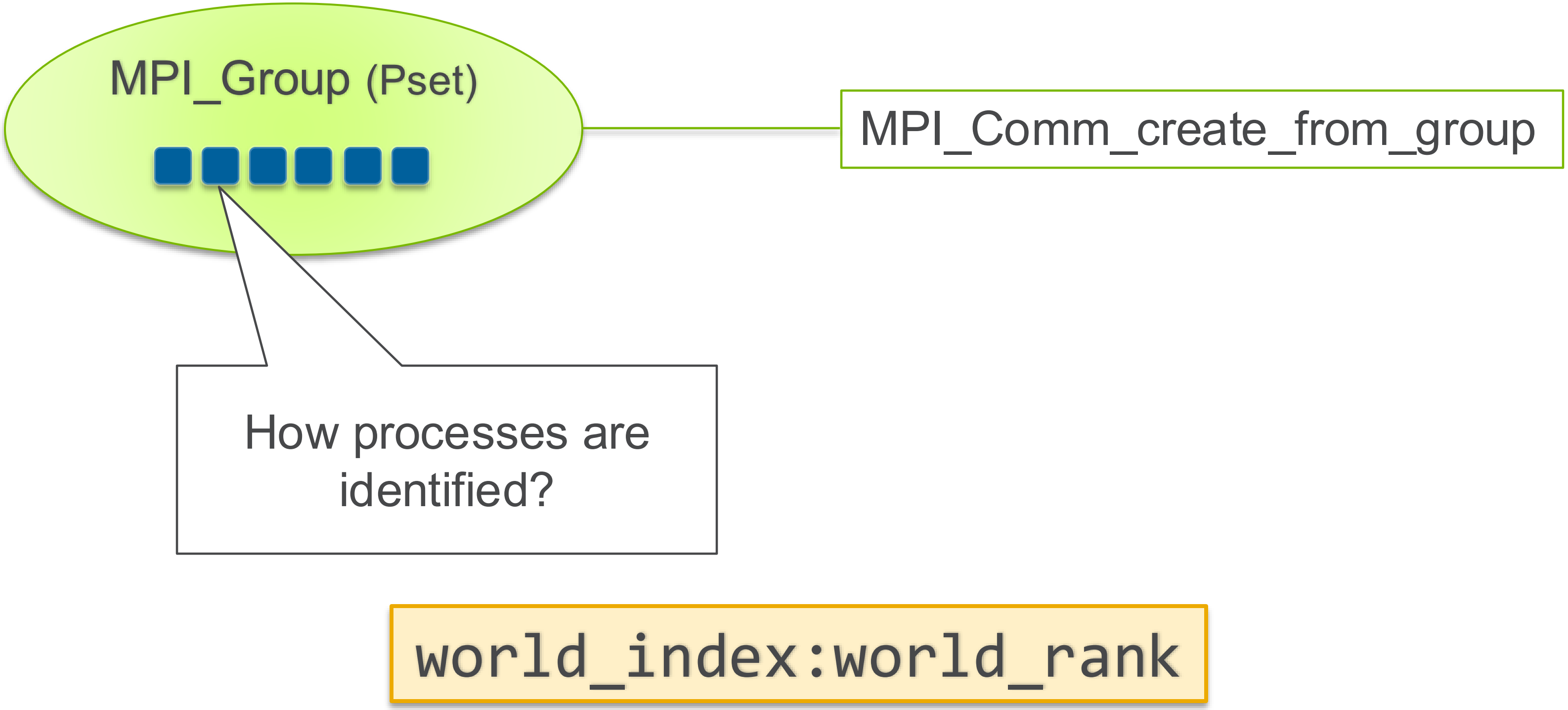
IMPLEMENTATION

IMPLEMENTING MPI SESSIONS

Separating Local and Collective Initialization



IMPLEMENTING TRUE MPI SESSIONS
Communicator-Independent Process IDs



IMPLEMENTING TRUE MPI SESSIONS

Group-level address exchange via PMI

- PMI-1: PMI_Barrier extension (PMI v1.2)

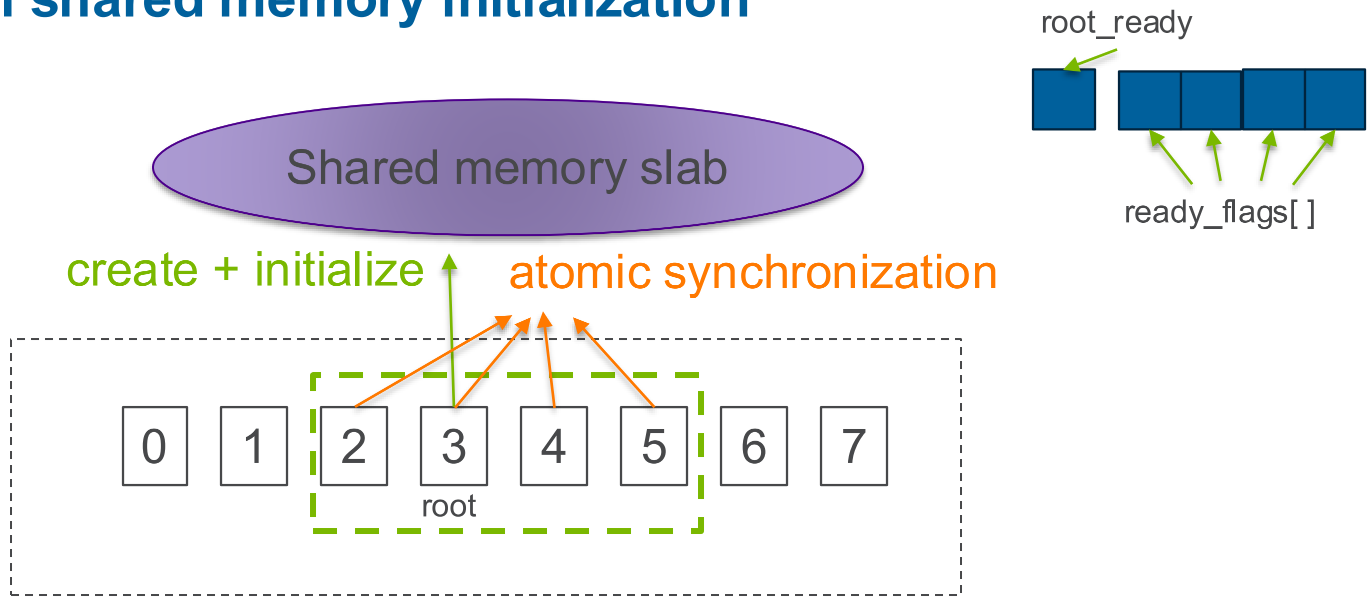
```
int PMI_Barrier_group(const int *group, int count
                      const char *tag);
```

- PMI-2: deprecate
- PMI-x: PMIx_Fence

```
pmix_status_t
PMIx_Fence(const pmix_proc_t procs[], size_t nprocs,
           const pmix_info_t info[], size_t ninfo);
```

IMPLEMENTING TRUE MPI SESSIONS

Group-level shared memory initialization

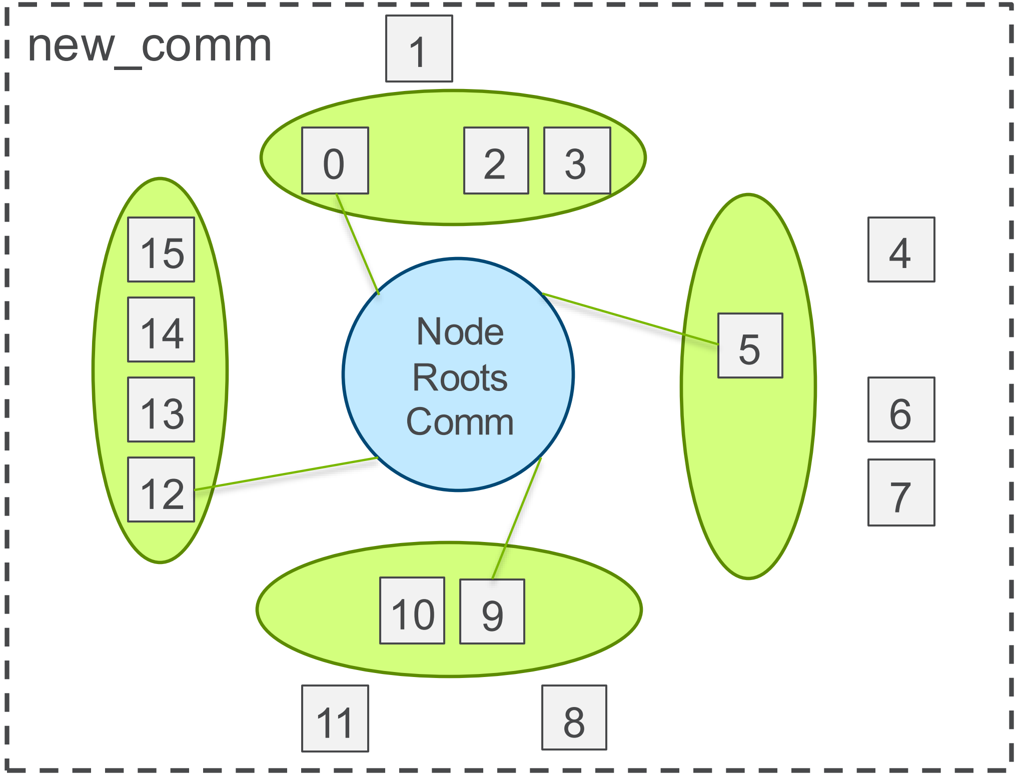


We redesigned shared memory init to an async atomic semantics

IMPLEMENTING TRUE MPI SESSIONS

Update Hierarchical Bootstraping in a group context

1. Node-roots-comm via PMI
2. Node-comm via atomic SHM
3. Bootstrap new_comm via hierarchical collectives



EXPERIMENTS

EXPERIMENTAL EVALUATION

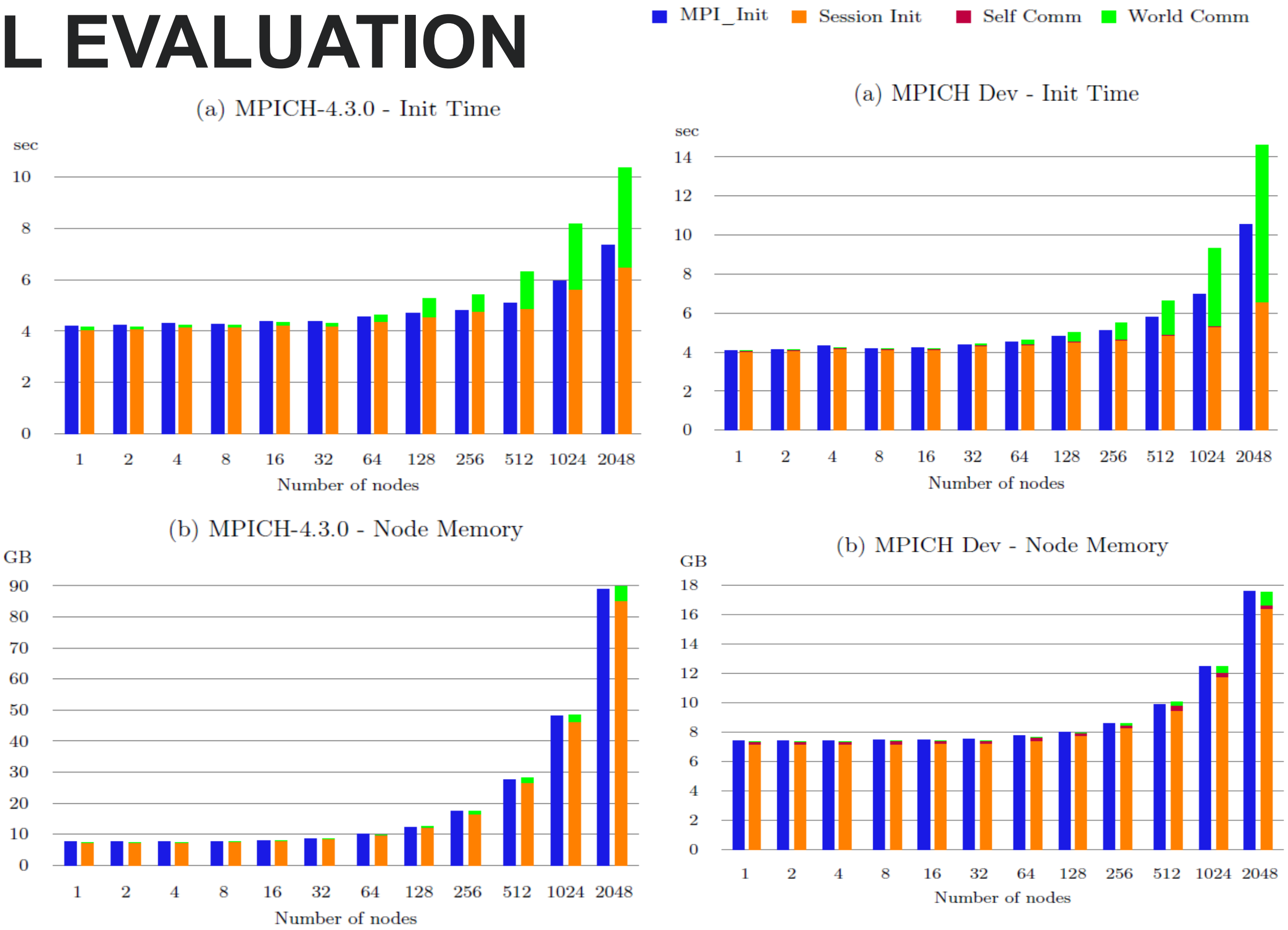
MPICH

Expectation:

- Equivalency between the world and sessions model.
- Flat local initialization.
- No performance degradation before and after supporting true sessions.

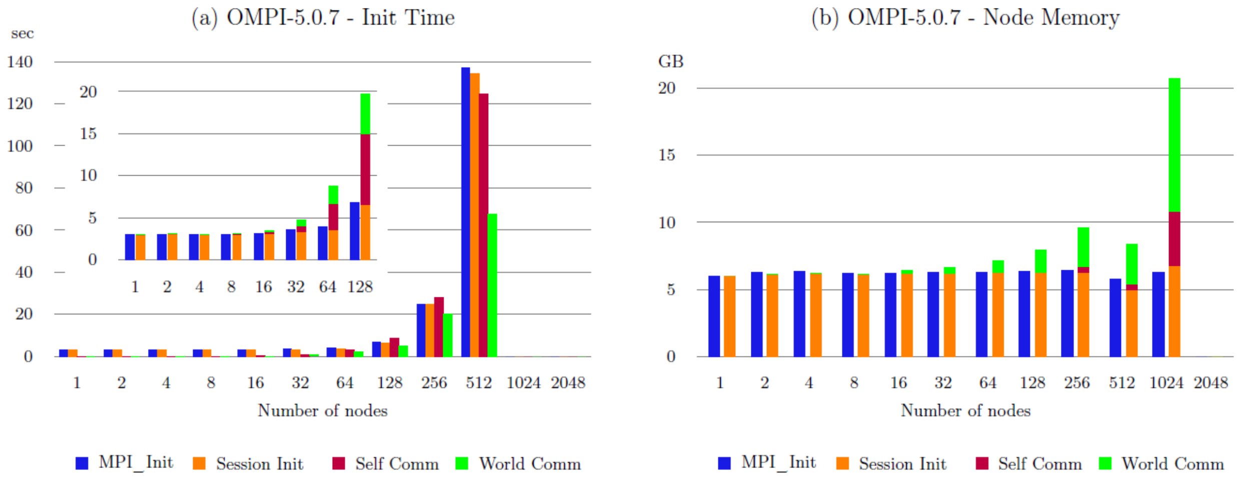
Results:

- Internal world comm slightly more efficient.
- Local initialization get slower and takes more memory as num of nodes increase.
- Slight performance degradation



Aurora, PPN = 96

EXPERIMENTAL EVALUATION
Open MPI

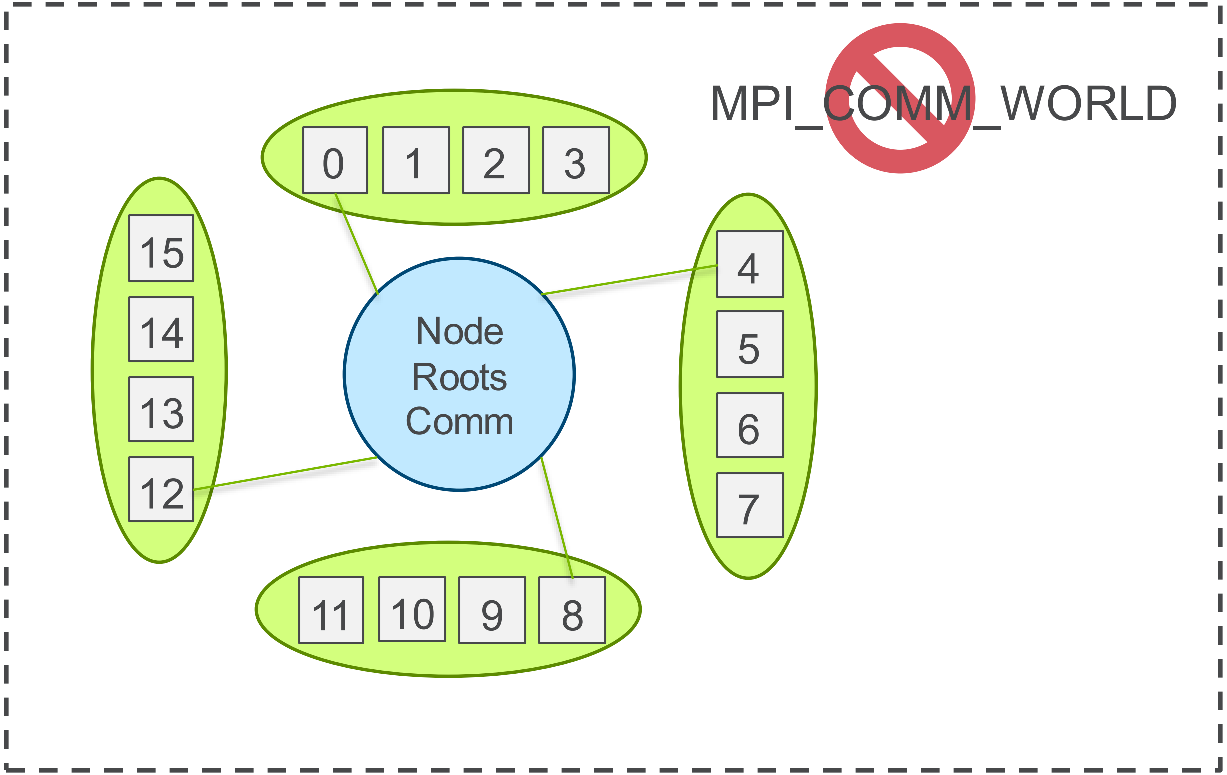


Aurora, PPN = 96

EXPERIMENTAL EVALUATION

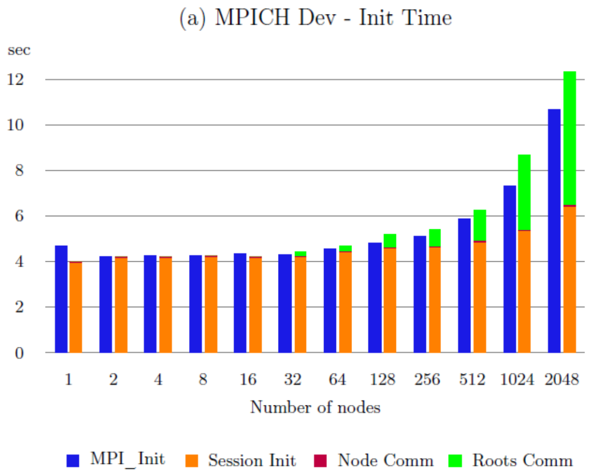
Sparse World

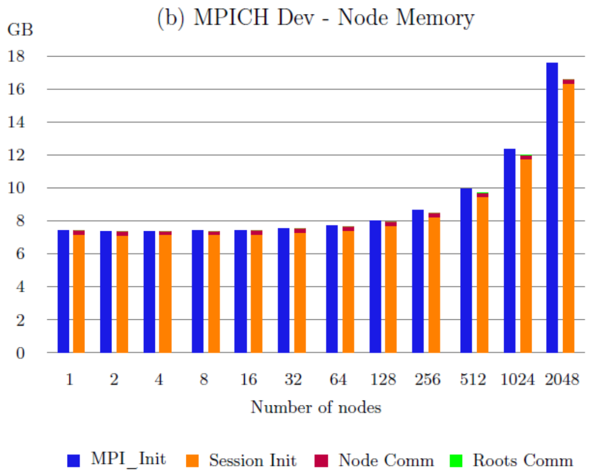
- Dense World
  - 192 internode connections
- Sparse World
  - 12 internode connections
  - Reduction by $\frac{1}{(PPN)^2}$



EXPERIMENTAL EVALUATION
Sparse World

1. Not much savings over hierarchical bootstrapping
2. More significant memory savings
3. Require user-layer hierarchical code

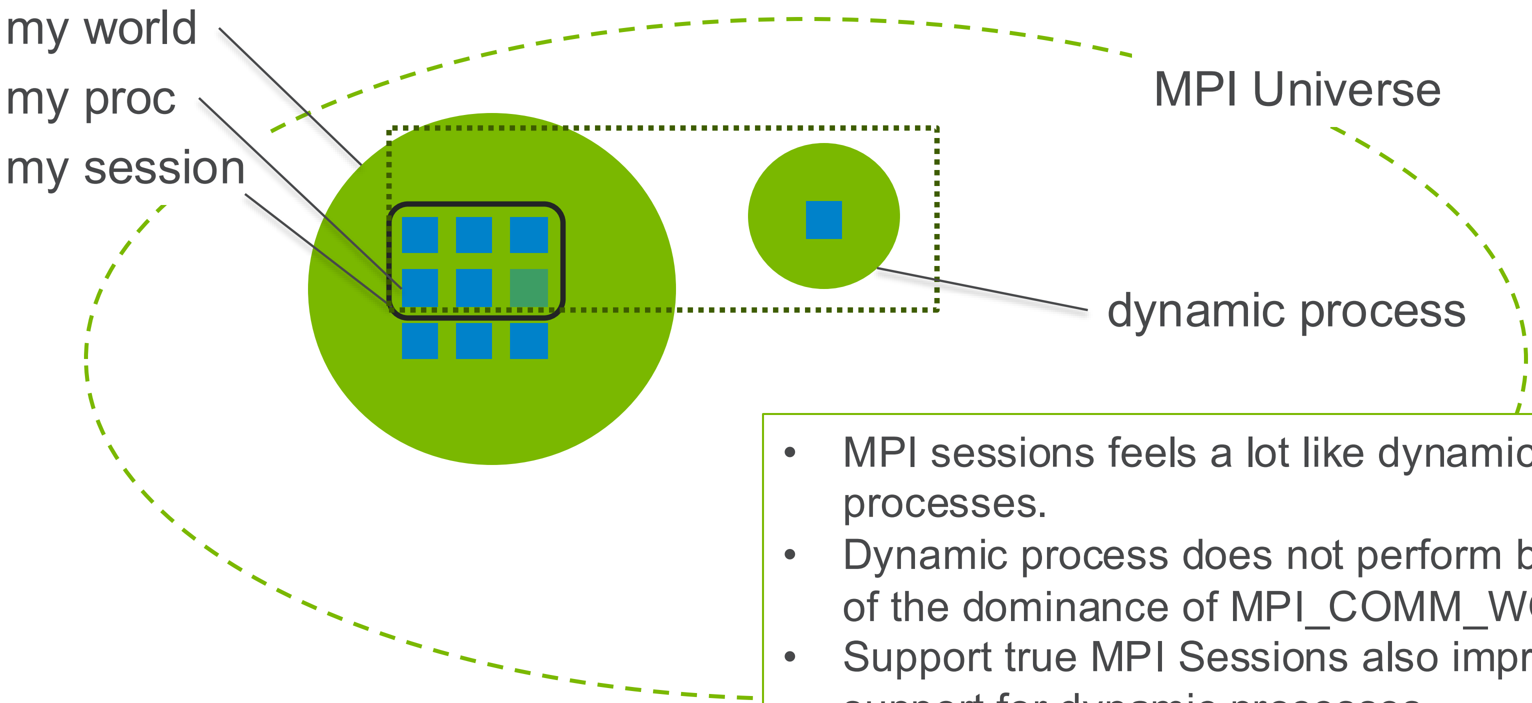




DISCUSSIONS

DISCUSSIONS

Lesson 1: MPI Sessions vs. MPI Dynamic Processes



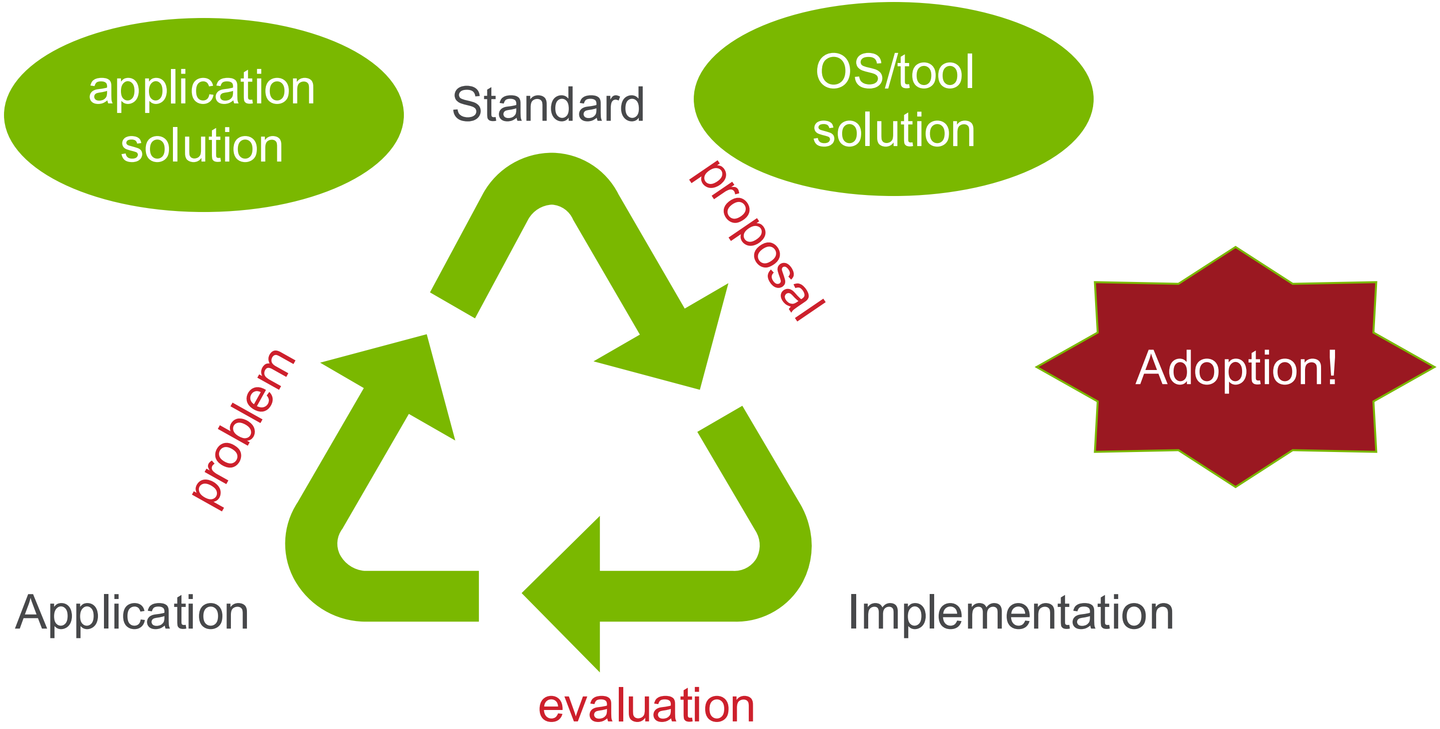
- MPI sessions feels a lot like dynamic processes.
- Dynamic process does not perform because of the dominance of MPI_COMM_WORLD
- Support true MPI Sessions also improves support for dynamic processes

DISCUSSIONS

Lesson 2: The Importance of Implementation

DISCUSSIONS

Lesson 3: The Importance of Complete Cycle



SUMMARY

- MPICH supports MPI Sessions since v4.0 in 2021. However, it relies on an internal comm_world to bootstrap communicators in the sessions model.
- We reimplemented in MPICH to support true sessions that does not depend on comm_world.
- Our evaluations show no significant scaling advantage between world model and sessions model if the world communicator is still constructed.
- We show improved initialization time and memory consumption when sparse communicators are used instead.
- Supporting true MPI sessions greatly improves MPICH’s dynamic process support.

Q & A