Open MPI State of the Union XIV
Community Meeting SC21

We will start at 1:15pm US Eastern

Please ensure that you are muted

Open MPI State of the Union XIV
Community Meeting SC21

Jeff Squyres
cisco

George Bosilca
ICL INNOVATIVE COMPUTING LABORATORY

Joshua Hursey
Geoff Paulsen
Austen Lauria
IBM Spectrum MPI

Tomislav Janjusic
NVIDIA

Howard Pritchard
Los Alamos NATIONAL LABORATORY EST.1943

Takafumi Nose
FUJITSU

Open MPI versioning

Quick review

Open MPI versioning

- Open MPI uses “A.B.C” version number triple
- Each number has a specific meaning:

A This number changes when backwards compatibility breaks

B This number changes when new features are added

C This number changes for all other releases

Definition

- Open MPI vY is ***backwards compatible*** with Open MPI vX (where Y>X) if:
  - Users can compile a correct MPI / OSHMEM program with vX
  - Run it with the same CLI options and MCA parameters using vX or vY
  - The job executes correctly

What does that encompass?

- "Backwards compatibility" covers several areas:
  - Binary compatibility, specifically the MPI / OSHMEM API ABI
  - MPI / OSHMEM run time system
  - `mpirun` / `oshrun` CLI options
  - MCA parameter names / values / meanings

Version Roadmaps

v4.0.x (Previous stable)

- Release managers
  - Howard Pritchard,
    Los Alamos National Lab
  - Geoff Paulsen, IBM

- Current release: v4.0.7
- Released Nov 15, 2021

v4.1.x (Current stable)

- Release managers
  - Brian Barrett, AWS
  - Jeff Squyres, Cisco

- Current release: 4.1.1
  - 4.1.2rc3 is available
  - 4.1.2 final to be released "immanently"

Open MPI v5.0.0

Many improvements are coming!

v5.0.x (Future)

- Release managers
  - Austen Lauria, IBM
  - Tomislav Janjusic, NVIDIA
  - Geoff Paulsen, IBM

- Current release candidate:
  - v5.0.0rc2

Overview of Major Changes

- Replaced launcher ORTE with PRRTE
  - PMIx Reference Runtime Environment (v2.0.x)
  - Requires PMIx v4.1.x
- Shared memory component “vader” renamed to “sm”
- OpenIB BTL has been removed
  - IB / RoCE now supported via UCX
- Replaced Debugger (MPIR) interface with PMIx Tools Debugger interface
  - Tool implementers: see the transition guide
- Removed MPI C++ bindings

New and Improved Runtime Environment!

- PRRTE is our new launcher!
  - PMIx Reference RunTime Environment
  - PMIx standard based
- ORTE effectively forked into its own repo
- Run time no longer specifically tied to Open MPI
  - Also works with other MPI implementations
  - And also in other, non-MPI/HPC environments

What is PRRTE?

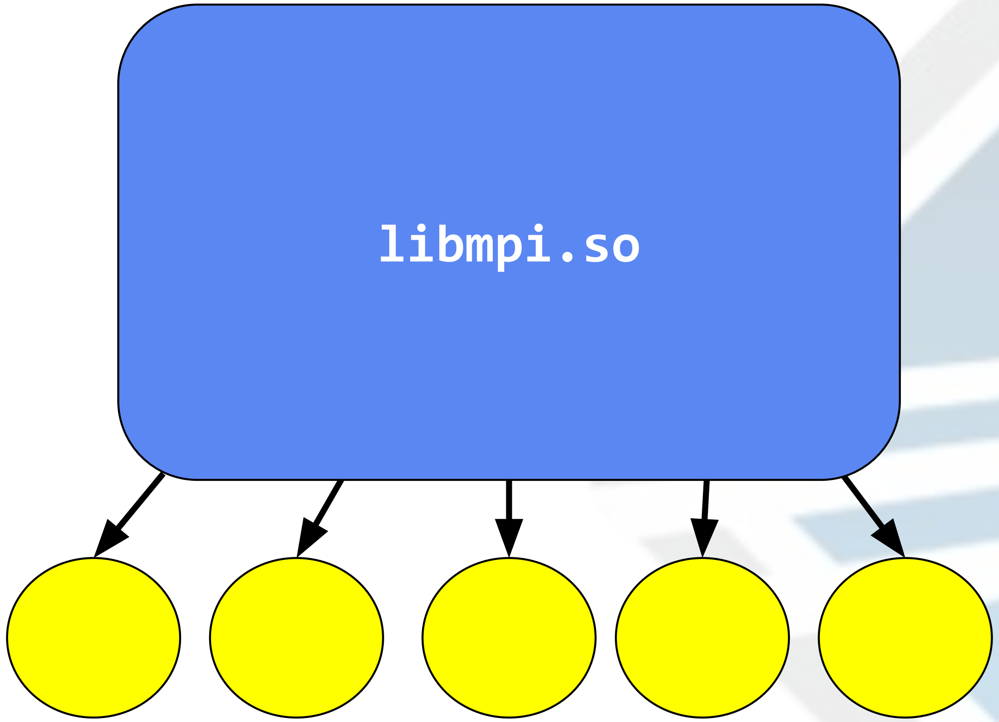
- **P**MIx **R**eference **R**un**T**ime **E**nvironment
  - Feature full, scalable, PMIx enabled runtime environment
- It is an open source, community supported, runtime implementation
  - Supports one-off jobs via `prterun` and multiple jobs via persistent daemons
  - Requires OpenPMIx >= 4.1.0
- Evolved from ORTE runtime in Open MPI to standalone project
  - Provides much of the same feature set as ORTE, plus more
  - Some `mpirun` command line options have changed
    - We are working on these differences via docs and aliases
  - `mpirun` and `mpiexec` are now small programs that exec `prterun`

What is OpenPMIx?

- Is a feature complete implementation of the PMIx standard
  - **P**rocess **M**anagement **I**nterface - E**x**ascale
- Provides an implementation to connect PMIx-enabled clients (e.g.,Open MPI) with tools (e.g., debuggers), and resource managers (e.g., PRRTE, SLURM, IBM JSM)
  - PMIx also provides event notification (fault tolerant libraries), and process wire-up, including “instant on” where supported
- OpenPMIx is an open source, community supported, scalable implementation
  - OpenPMIx releases tied to corresponding PMIx Standard releases
  - Proving ground for new PMIx Standard additions
  - Used on many large scale HPC systems, including all top 3 systems from Top 500 November 2021 list

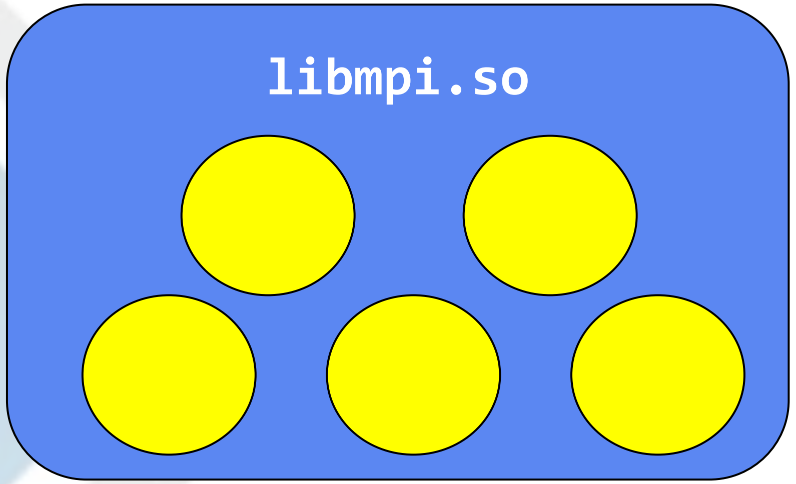
New Defaults

Open MPI ≤ v4.x



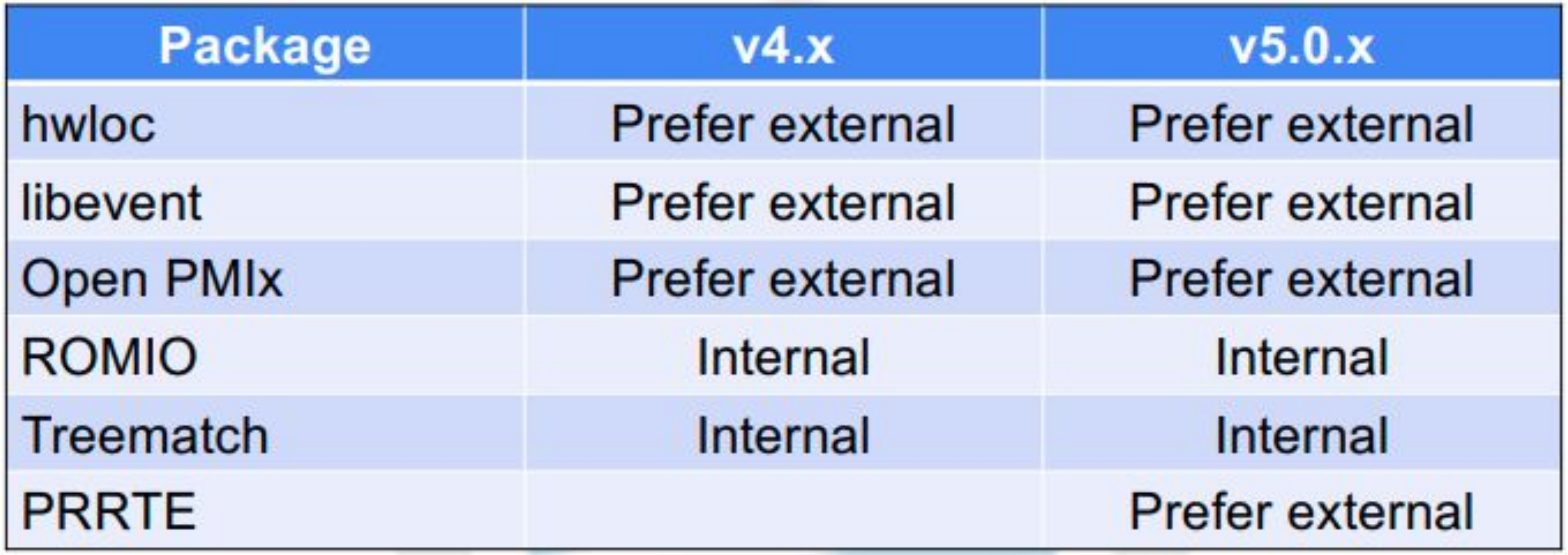
Plugins loaded dynamically at runtime

Open MPI 5.0.x



Plugins located in the library

Packaging



Package	v4.x	v5.0.x
hwloc	Prefer external	Prefer external
libevent	Prefer external	Prefer external
Open PMIx	Prefer external	Prefer external
ROMIO	Internal	Internal
Treematch	Internal	Internal
PRRTE		Prefer external

- v4.x: OMPI prefers an external hwloc/libevent, but only allow the use of external libraries that are newer than what is shipped
- v5.0.x: OMPI prefers an external hwloc/libevent if they are at least minimum tested versions (RHEL7 or later)

ABI / Command Line Changes

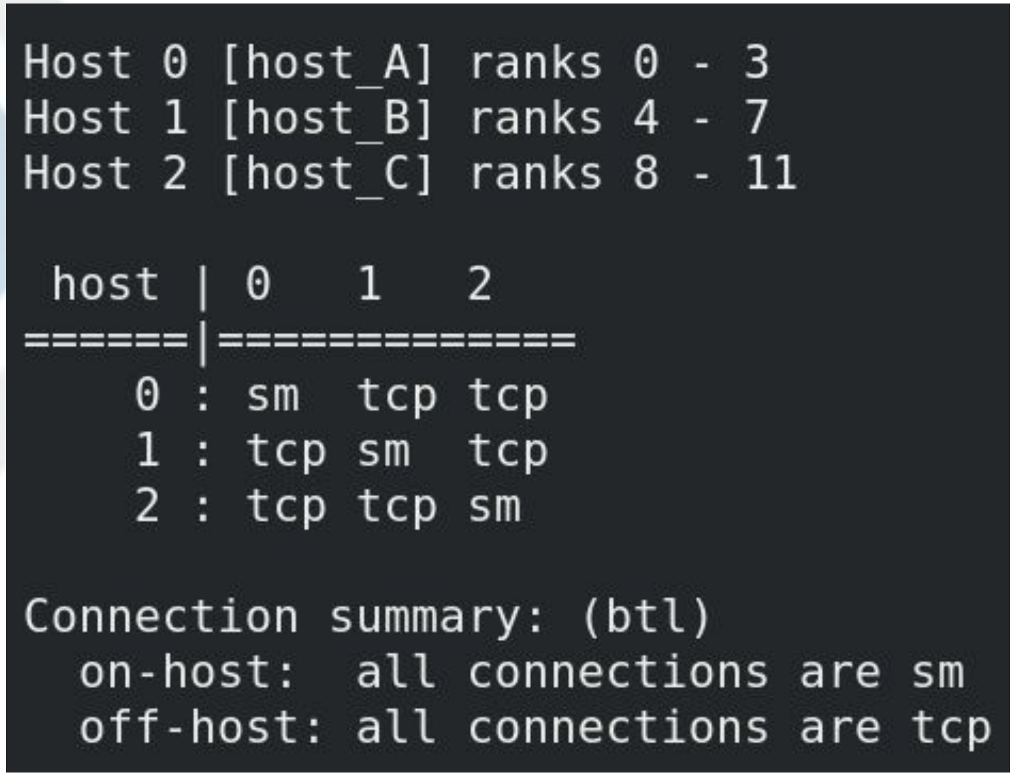
- Now following GNU CLI conventions
  - Multi-letter tokens are double-dash only (e.g., `--mca` not `-mca`)
  - Will automatically replace and print a warning if a single-dash is used
- Some MCA params have moved from `ompi_` to `prte_` or `pmix_`
  - Automatically convert MCA params to their proper project
    - Can be specified in default param files, environment, and command line
    - Work in progress: hope to have all complete before release
  - To see the full list of MCA parameters for each project, run `ompi_info --all`, or `prte_info --all`, or `pmix_info --all`
  - Actively working on masking and / or documenting these changes

New Features

- User Level Fault Mitigation (ULFM)
  - Conforms to the ULFM MPI Standard draft proposal
  - Enabled by default
  - New ULFM test suite in github.com/open-mpi/ompi-tests-public repo
- Threading MCA framework
  - pthreads (default), Argobots, and Qthreads
  - Specify by configuring with `--with-threads=X`
- Added CUDA support to mtl/ofi
  - Requires Libfabric >= v1.9
- New UCC (Unified Collective Component) from UCX community
  - coll/ucc and scoll/ucc
  - Configure with `--with-ucc=DIR` and increase priority

New Features (continued)

- Enable load-linked, store-conditional atomics for AArch64
  - Up to 40x performance improvement with multi-threaded lifo/fifo test benchmarks
- Added OMPIO GPFS filesystem support
- Added OMPIO atomicity support
- --mca ompi_display_comm 1
  - Displays a proc-by-proc communication matrix to stdout
- FP16 support via MPIX Extension
- memory/patcher:
  - Add ability to detect patched memory



```
Host 0 [host_A] ranks 0 - 3
Host 1 [host_B] ranks 4 - 7
Host 2 [host_C] ranks 8 - 11

 host | 0   1   2
======|============
    0 : sm  tcp tcp
    1 : tcp sm  tcp
    2 : tcp tcp sm

Connection summary: (btl)
  on-host:  all connections are sm
  off-host: all connections are tcp
```

Removed Features

- osc/pt2pt
  - Not maintained, very buggy
  - Replaced with osc/rdma + btl/tcp
- btl/openib
  - Note: IB / RoCE now supported via UCX
- btl/sm (legacy)
  - Replaced with btl/vader, which is renamed to btl/sm
  - Alias “vader” exists for backwards compatibility
- MXM support
  - pml/yalla, opal/atomic/mxm, oshmem/spml/ikrit all removed

Other Removed Features

- 32 bit builds are more limited
  - Now only supported with C11-compliant compilers
- Removed support for GNU gcc compilers < v4.8.1
- Removed PMI support
  - PMI-1 and PMI-2 are no longer supported
  - PMI shim is available here
- MPI C++ bindings.
  - Removed from MPI-3.0 standard (9 years ago)

More Removed Features

- fcoll/two_phase replaced with ROMIO/OMPIO
- The `--am` and `--amca` options are deprecated
- patcher/linux
  - Attempted to hook calls by patching dynamic symbol table, however it did not work in all cases
- Checkpoint restart (CR) functionality is now completely removed

MPIR Has Been Removed

- What is MPIR?
  - MPI Process Acquisition interface
  - Is not an API, but instead "... requires that a tool reads symbol table information and traces the starter process..."
  - Used by debuggers for MPI processes
- Replaced by the **PMIx tool interface**
  - gdb scripts that rely on `MPIR_*` routines will need to be updated
  - Contact your debugger provider for their timeline
  - To use non-PMIx enabled debuggers, see the MPIR-to-PMIx guide for a shim layer
- Initially announced at SC’17 BOF:
  - Deprecation notice in NEWS in early 2018
  - User runtime warnings in v4.0.0 (mid/late 2019)
  - Removed in v5.0.0

MPI-4.0 Support

- Open MPI v5.0.0 is not MPI-4.0 compliant
  - Officially, it is still MPI-3.1 compliant
  - However, it does have a number of MPI-4.0 features!
- MPI_ERRORS_ABORT infrastructure
  - New error classes to distinguish between aborting a communicator and aborting an entire application
- Initial error handler implementation
  - The error handler that is set before MPI is initialized and after it is finalized
  - Can be selected from `mpiexec` / `MPI_Comm_spawn()` info key parameters
- Error handling for “unbound” errors to MPI_COMM_SELF
  - Requires that unbound errors trigger the error handler on MPI_COMM_SELF instead of MPI_COMM_WORLD

MPI-4.0 Support (continued)

- Persistent collectives moved from the `MPIX_` namespace to `MPI_`
- `MPI_Comm_get_info()`, `MPI_File_get_info()`, and `MPI_Win_get_info()` updated to be MPI 4.0 compliant
  - Changes to modified keys are now reflected
- Partitioned communication implemented
- Support for `mpi_minimum_alignment` info key
- `MPI_info_get_string()` support
  - Replaces the deprecated `MPI_Info_get()` and `MPI_Info_get_valuelen()`
- Release Managers may take MPI-4.0 features on case-by-case basis
  - See Github project to track MPI-4.0 progress. Help is always appreciated!

Open MPI v5.0.0 Schedule

- Branched from master
  - March 11, 2021
- While still not release ready… https://www.open-mpi.org/software/ompi/v5.0/
  - Sept 30, 2021 - v5.0.0rc1
  - Oct 18, 2021 - v5.0.0rc2
- Best guess for release date: **Q1 2022**
- See Open MPI release timeline

Open Issues Before Release

- Various MPI one-sided issues
- Documentation
  - Mostly related to changes from `orte` → `prrte`
  - New/deprecated/changed `--mca` parameters
  - Revamped / ReadTheDocs documentation (hopefully before release!)
- Remaining v5.0.0 critical items: https://github.com/open-mpi/ompi/projects/3

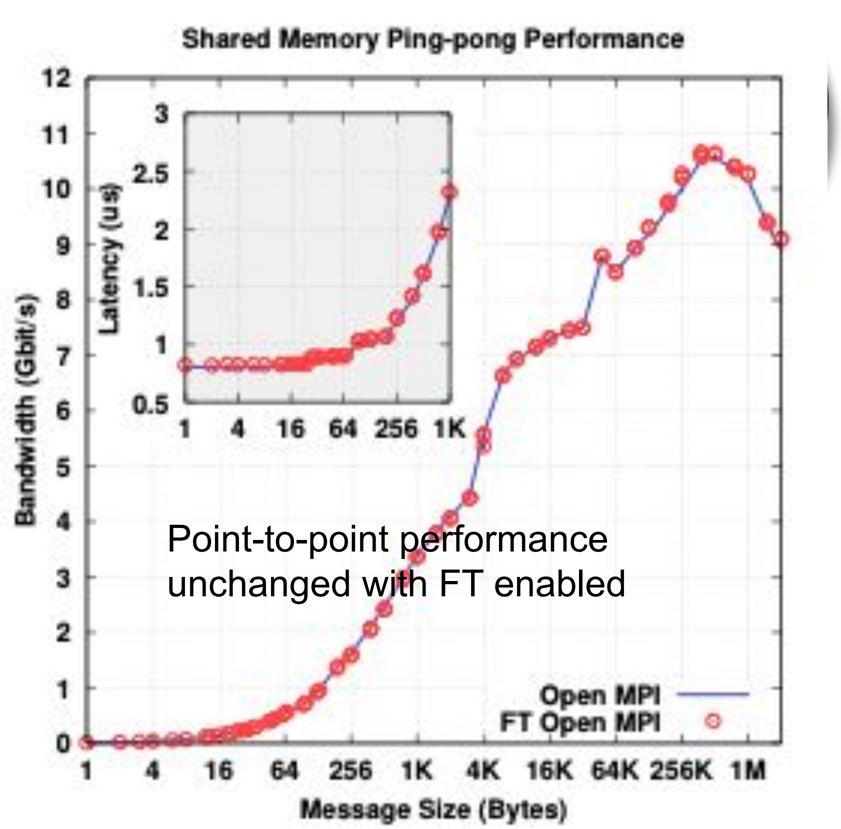
UTK Open MPI Activities

George Bosilca
University of Tennessee

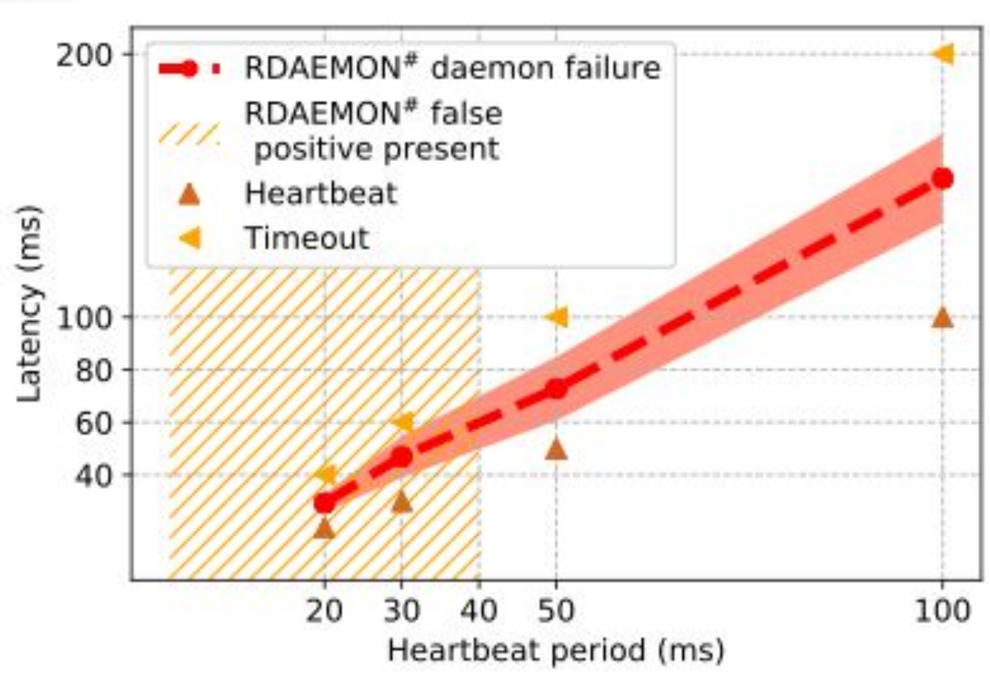
ICL
INNOVATIVE
COMPUTING LABORATORY

User Level Failure Mitigation

- ULFM now integrated in master and in 5.0
  - `--with-ft=mpi`
- Move the underlying resilient mechanisms outside ULFM/OMPI
  - Failure detector and reliable broadcast in PPRTE
  - Used in OMPI ULFM and SUNY OpenSHMEM
- Scalable fault tolerant algorithms demonstrated in practice for revoke, agreement, and failure detection (SC’14, EuroMPI’15, SC’15, SC’16)
- Extensions are in development for additional asynchrony (async spawn, shrink) and info keys for more flexible behaviors (automatic revocation, consistent collectives, consistent communicator creation)

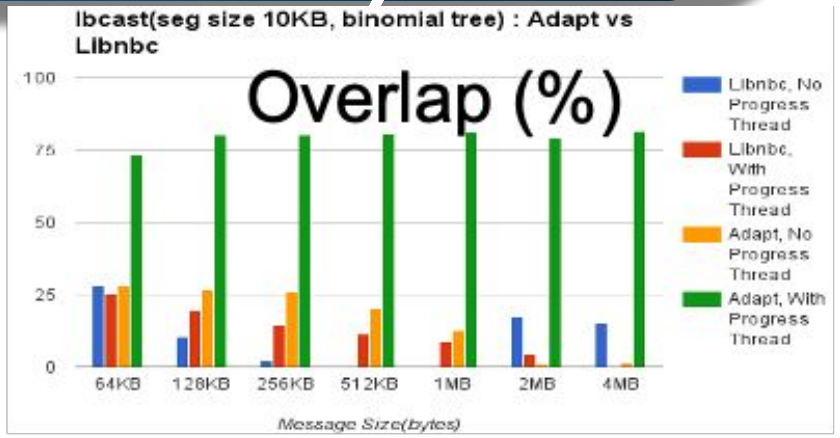


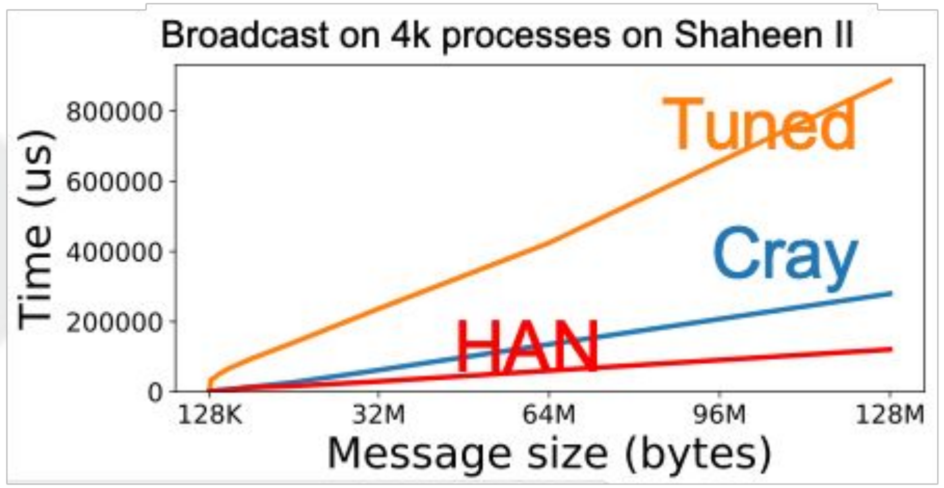
Point-to-point performance unchanged with FT enabled

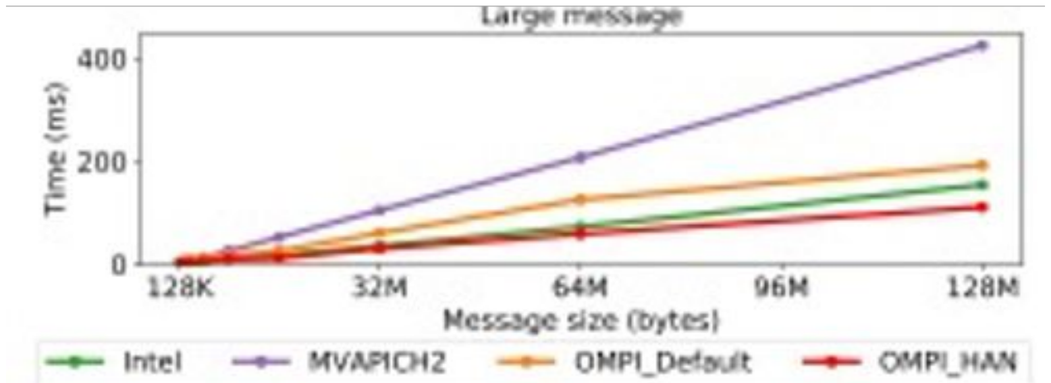


Adaptive Collective Communications Framework (HAN and ADAPT)

- More complex architectures demand more complex collective frameworks
  - Tuned was not built to support hierarchical architectures nor cope with system noise nor provide overlap
- Event-driven collective based on a dataflow state machine
  - Architecture aware
  - Reactive to network noise
  - Provide support for overlap for non-blocking collectives
- Summit tuning in underway

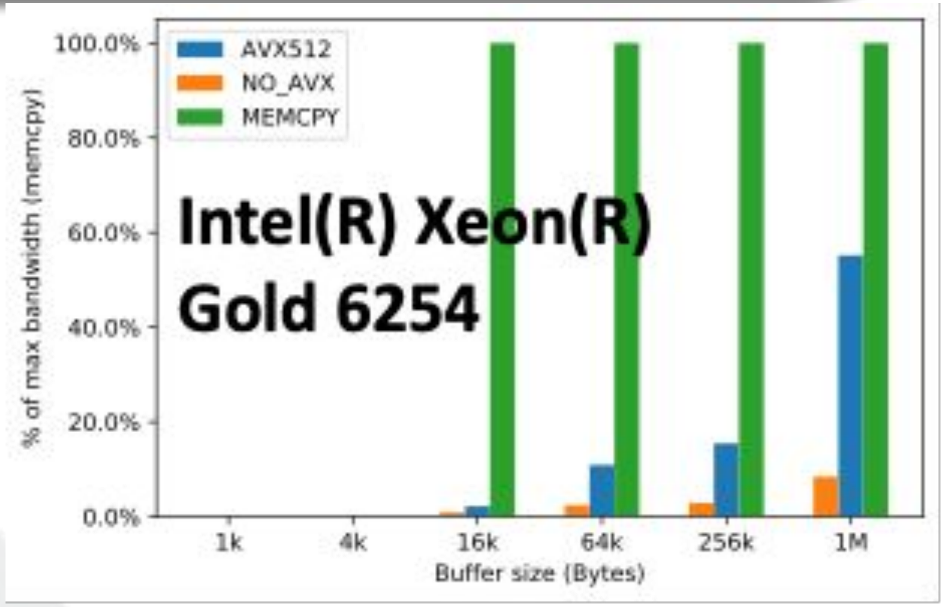


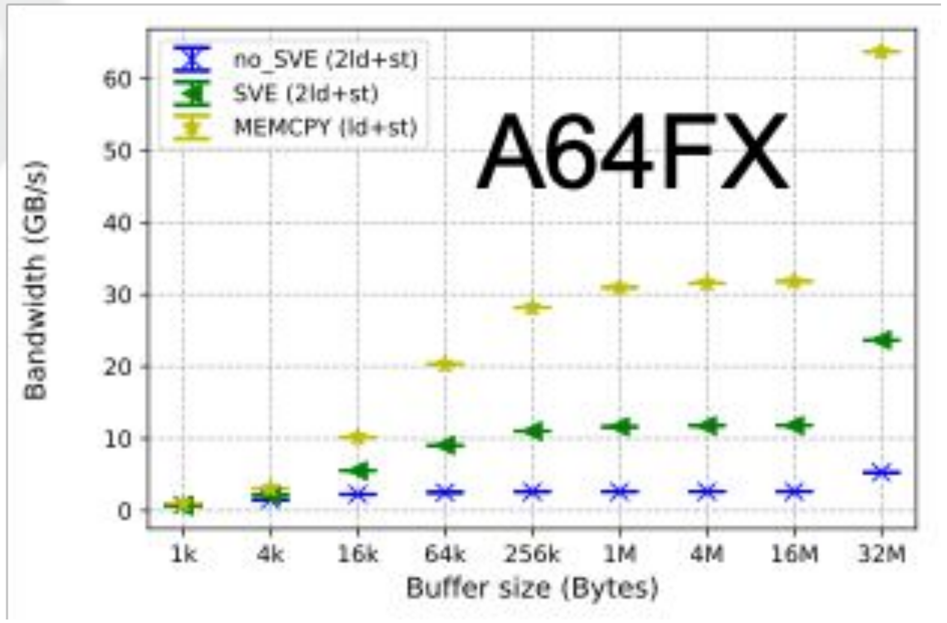




Explicit Support for Vectorial ISA

- First step of adding support for vectorial instructions to MPI_Op
  - Intel AVX* and ARM SVE
  - Distinction between build capabilities and execution capabilities
- Lessons learned are now impacting other parts of the code, especially the datatype engine
  - Prefetching, gather/scatter for non-contiguous datatypes

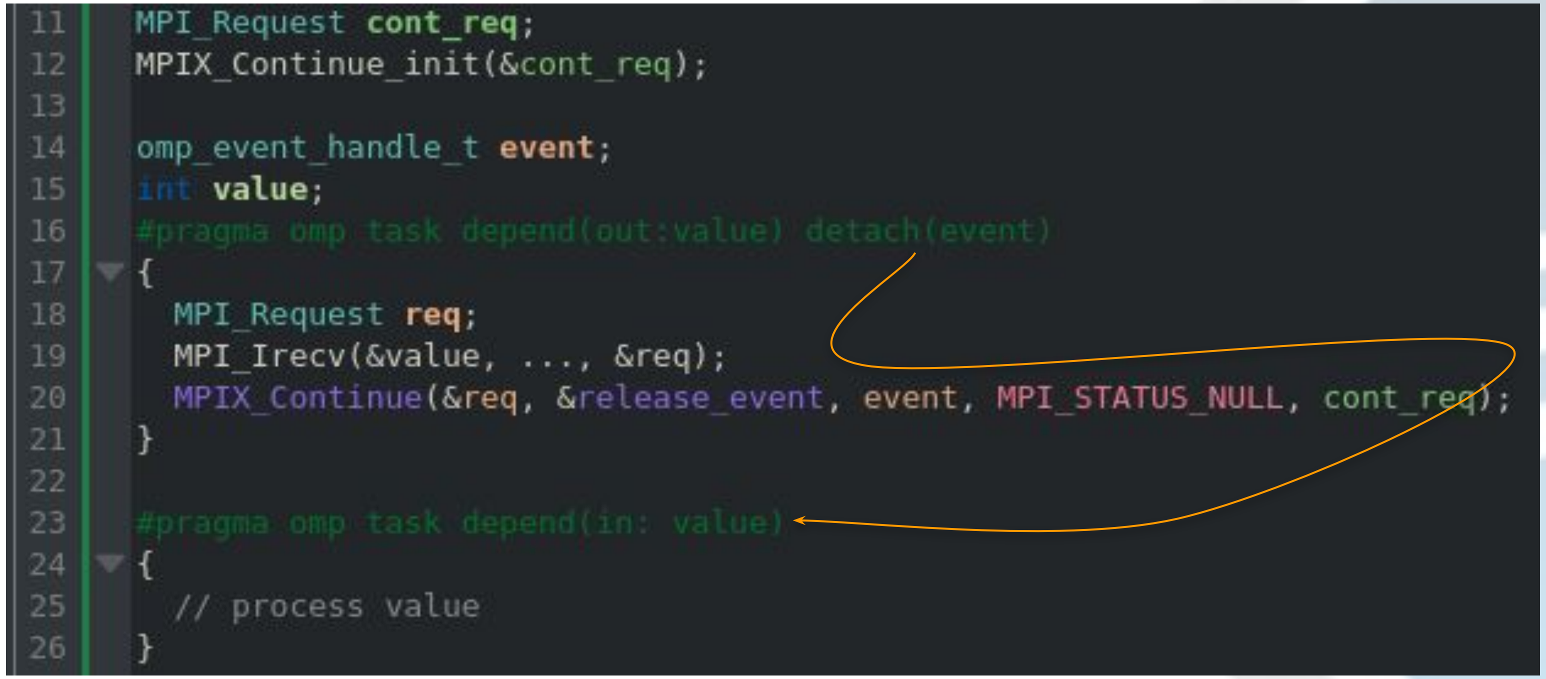




Completion Continuations

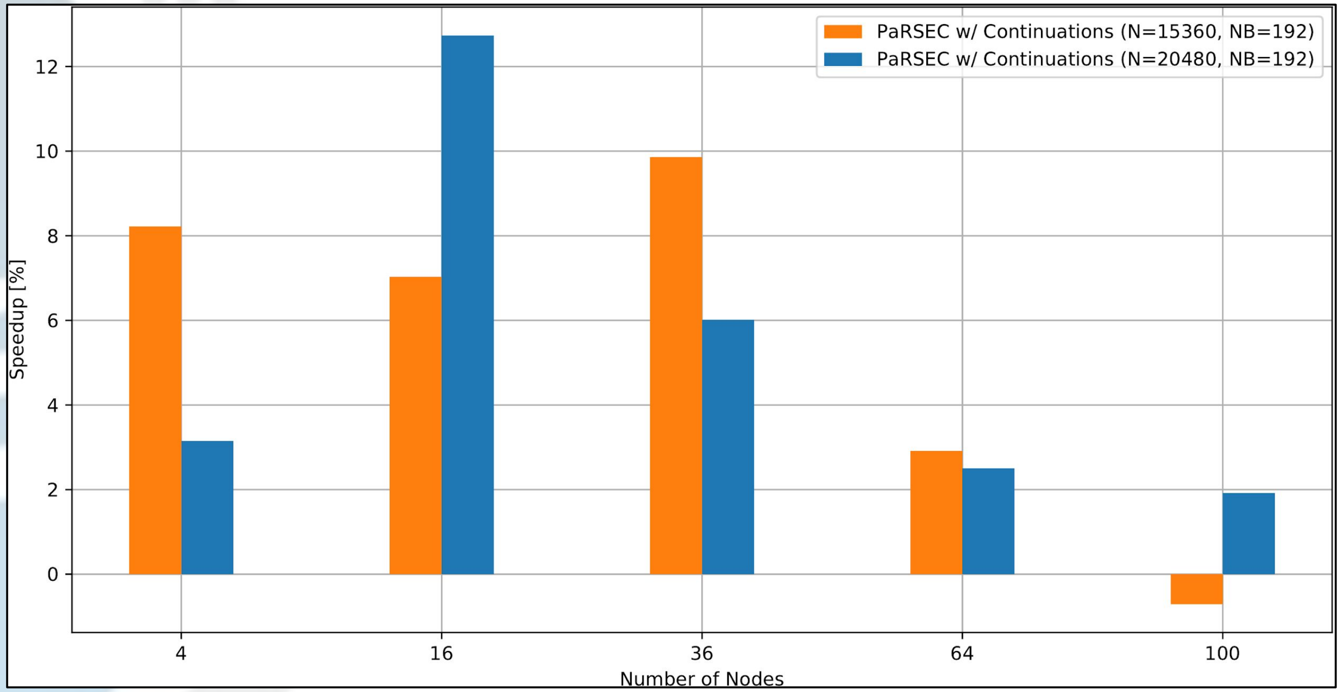
Continue activity after an MPI operation completed

- Interoperability with asynchronous and multithreaded programming models
- Preparing integration with main development branch
- Proposal under discussion in HACC-WG



```
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
}
```

QR Factorization in DPLASMA

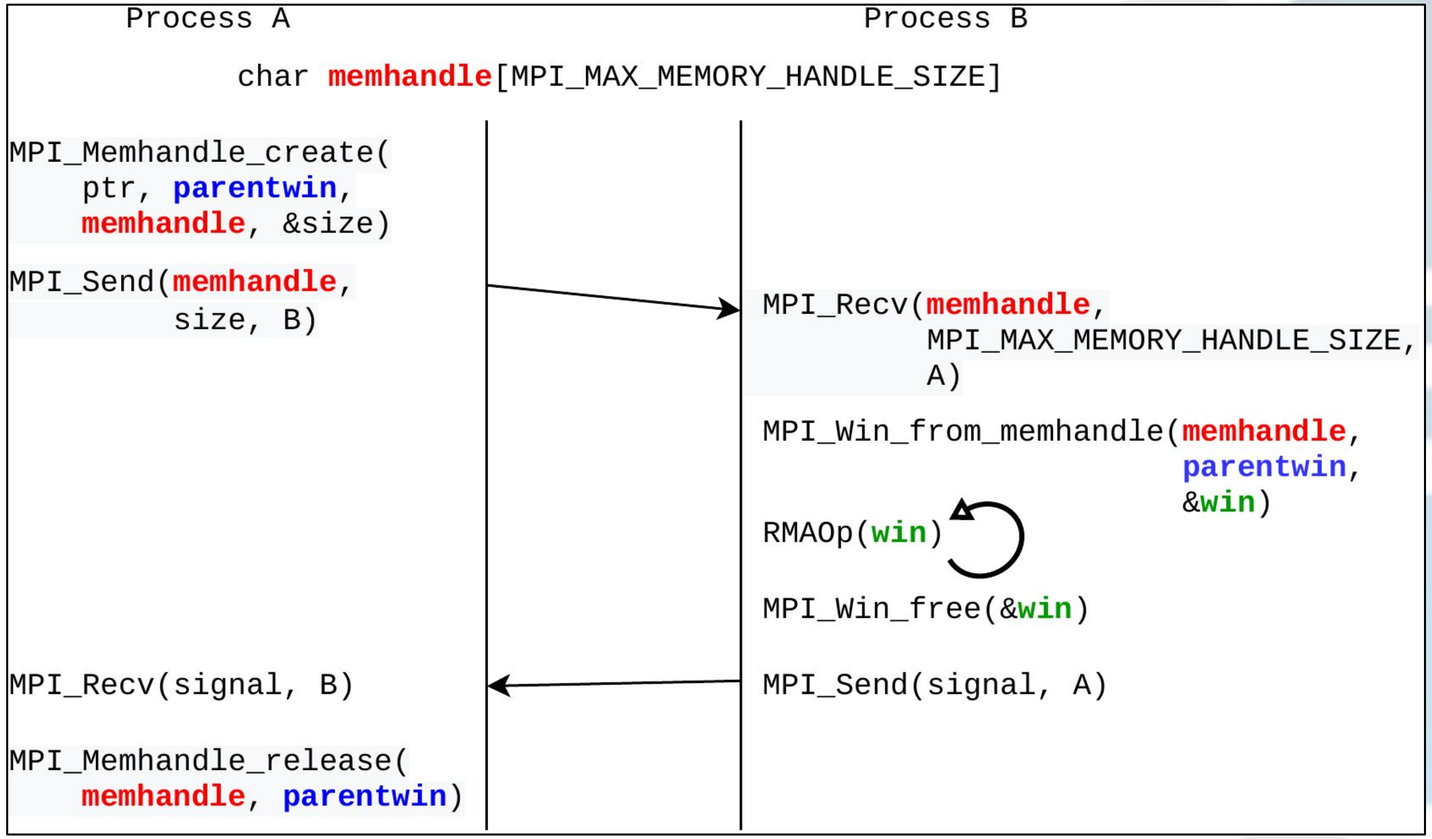


“Callback-based completion notification using MPI Continuations.” Joseph Schuchart, Christoph Niethammer, José Gracia, George Bosilca, Parallel Computing, 2021.

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

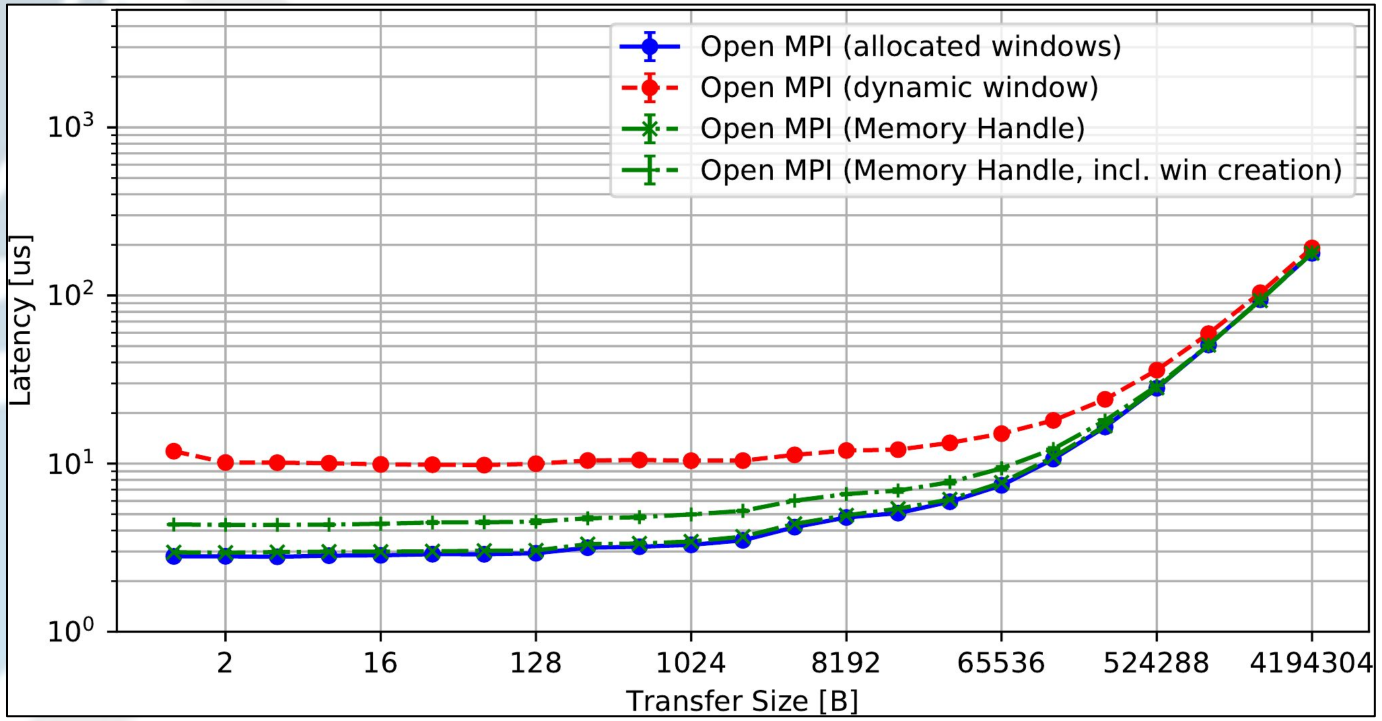
RMA Memory Handles

- Expose dynamic memory registration to user
- Exploit RDMA capabilities with P2P windows
- Avoid collective window memory allocation



```
      Process A                          Process B

            char memhandle[MPI_MAX_MEMORY_HANDLE_SIZE]

MPI_Memhandle_create(
    ptr, parentwin,
    memhandle, &size)

MPI_Send(memhandle,
         size, B)
                                   MPI_Recv(memhandle,
                                            MPI_MAX_MEMORY_HANDLE_SIZE,
                                            A)

                                   MPI_Win_from_memhandle(memhandle,
                                                          parentwin,
                                                          &win)
                                   RMAOp(win)

                                   MPI_Win_free(&win)

MPI_Recv(signal, B)                MPI_Send(signal, A)

MPI_Memhandle_release(
    memhandle, parentwin)
```

osu_put_latency: allocated vs dynamic vs memory handle windows



Quo Vadis MPI RMA? Towards a More Efficient Use of MPI One-Sided Communication. J. Schuchart, C. Niethammer, J. Gracia, G. Bosilca. Presented at EuroMPI'21.

Fujitsu

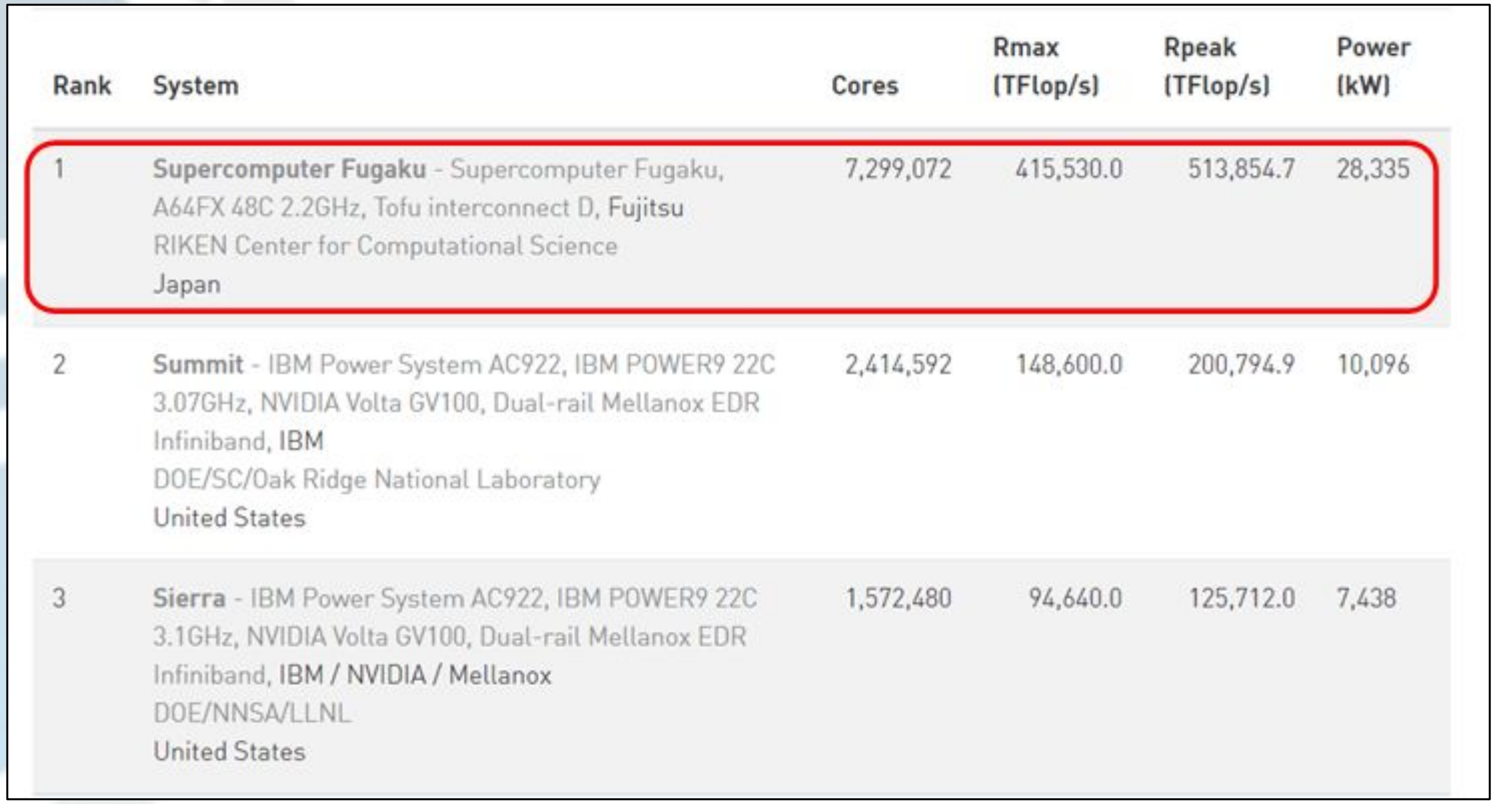
Takafumi Nose

Fujitsu MPI

- Based on Open MPI 4.0.x
- Optimized for A64FX + Tofu interconnect D
  - A64FX: Armv8.2-A + SVE (SIMD) + FP16 (half-precision floating point)
  - TofuD: 6D Torus/Mesh interconnect

Success of Fugaku

- Fugaku held the top spot on the TOP500 list from June 2020 to June 2021
- 635,904 MPI processes worked perfectly
- This work could not have been done without Open MPI
- Thank you!



Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438

Fujitsu’s Contributions

- Persistent collectives in MPI-4.0
  - E.g., `MPI_Bcast_init()`
- Datatype for half-precision floating point
- Bug fixes and man page improvements

IBM

Joshua Hursey
Geoff Paulsen
Austen Lauria

IBM Spectrum MPI

Delivering Robust & Sustained High Performance for Scalable MPI Applications

Accelerated & Enhanced MPI Point-to-Point

- Driving maximum performance from POWER9, InfiniBand, and GPU hardware.
- Supports direct transfer of GPU buffers between GPUs and across the InfiniBand network.

Dynamic & Optimized MPI Collectives

- Best algorithm selected per call at runtime.
- Includes Power optimized and hardware accelerated (e.g., SHARP) algorithms.

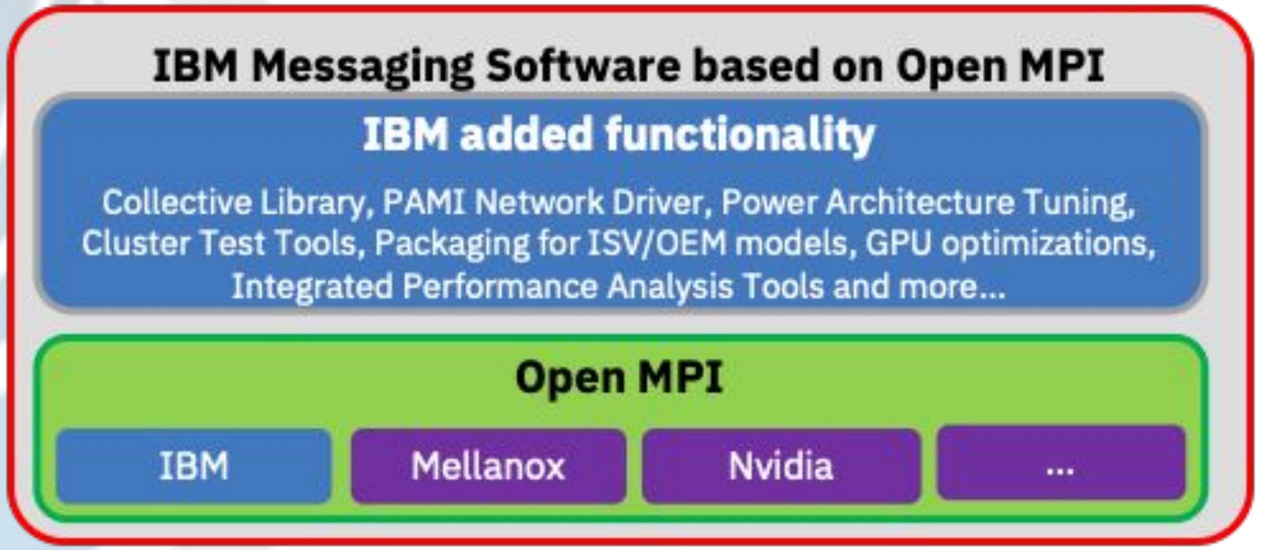
Usability Features Targeting Installation, Startup, Debugging, and Profiling

- Scalable to thousands of nodes and nearly a million processes!

Integration with IBM solutions such as LSF, JSM, ESSL, and Spectrum Scale

Supports 5 of the top 21 systems in the Top500 as of June 2021

Built on the open source Open MPI project with IBM value add and IBM service and support



IBM Messaging Software based on Open MPI

IBM added functionality

Collective Library, PAMI Network Driver, Power Architecture Tuning, Cluster Test Tools, Packaging for ISV/OEM models, GPU optimizations, Integrated Performance Analysis Tools and more…

Open MPI

IBM | Mellanox | Nvidia | …

Spectrum MPI 10.4 is based on Open MPI 4.0.x with PMIx 3.2.x

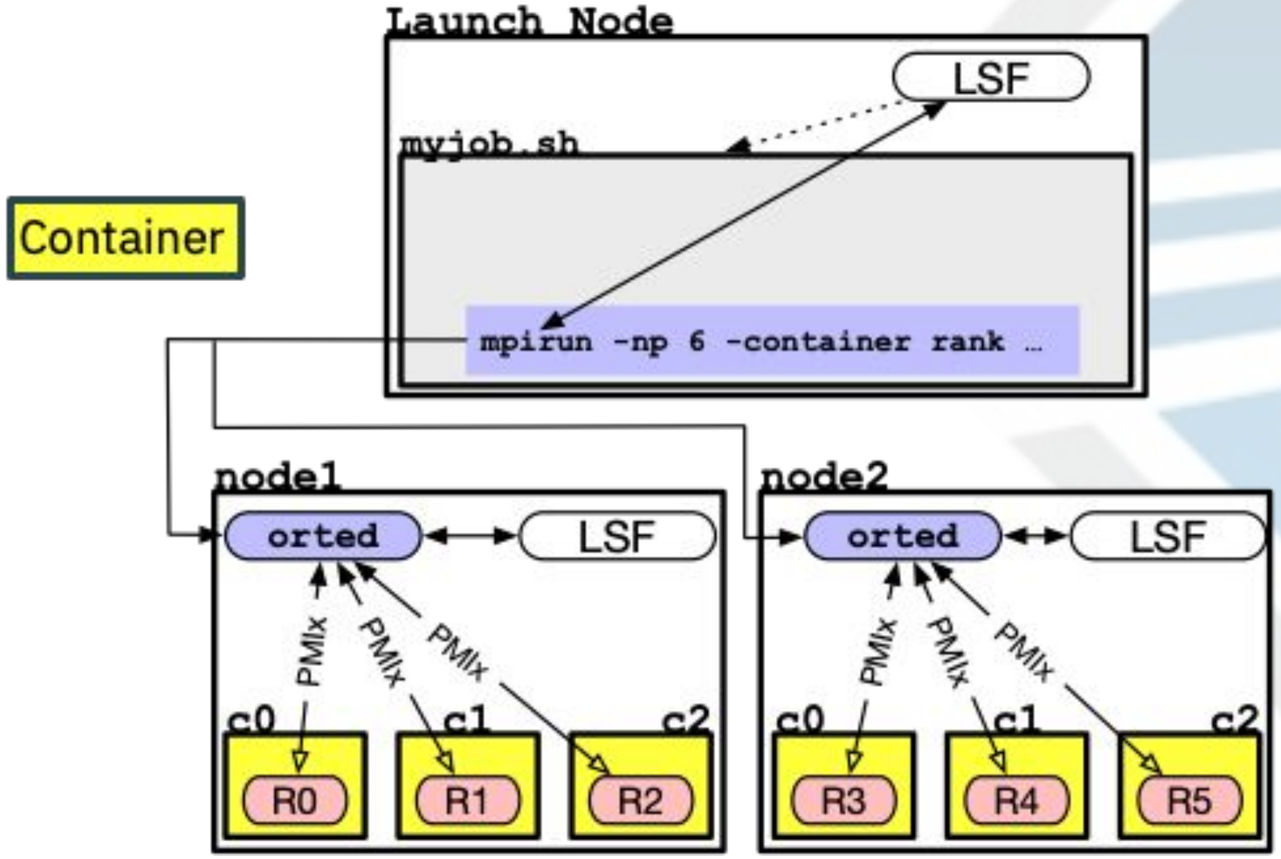
Spectrum MPI Community Edition Available!

Container Ready Supporting Applications on Bare Metal & Private/Public Cloud

Spectrum MPI options to support launching applications in the two different container modes commonly used in HPC environments

Rank Contained mode

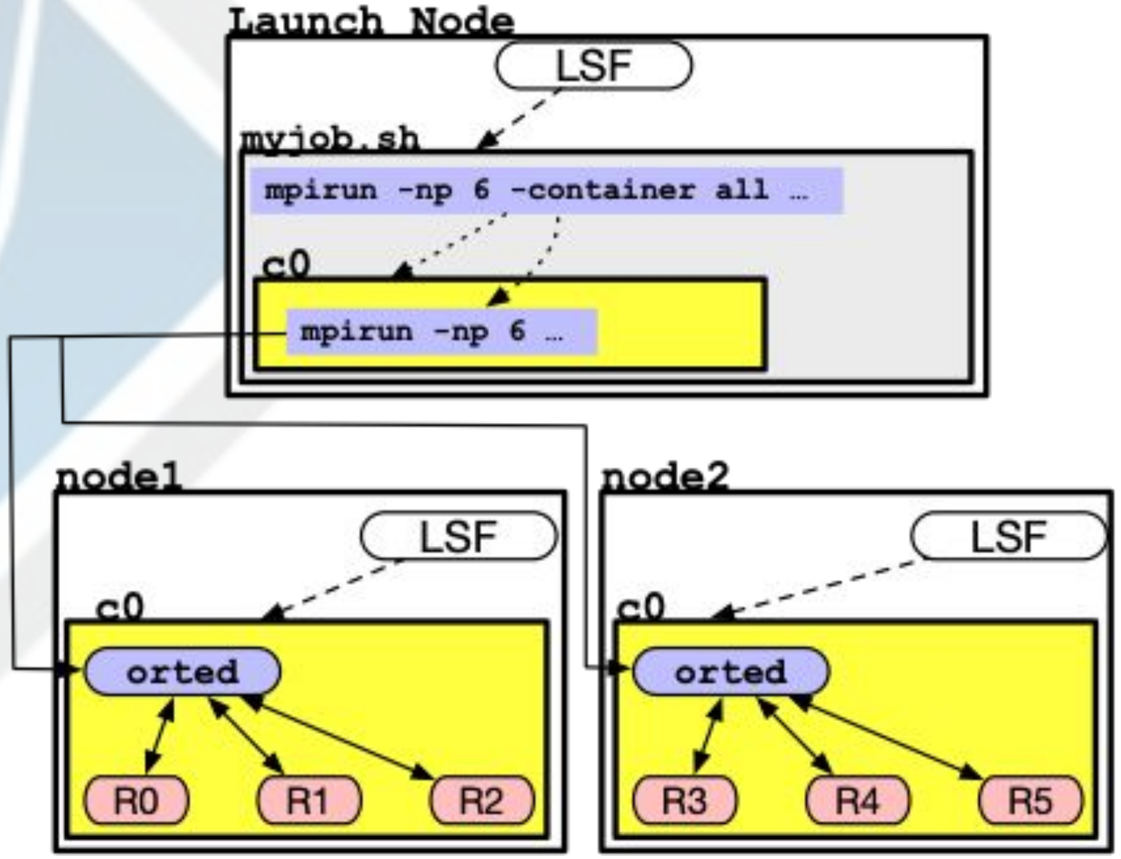
One container per application process



```
mpirun -np 6 -container rank …
```

Fully Contained mode

One container per node



```
mpirun -np 6 -container all …
mpirun -np 6 …
```

ECP OMPI-X

Howard Pritchard

ECP OMPI-X

A major focus of the project over the past year has been on the release of the MPI-4 Standard itself as well as implementation of some of the features new to this version of the MPI Standard.

ECP OMPI-X MPI-4

MPI Partitioned Communication

- New MPI multi-threading interface
- Better efficiency with minimal app changes
- Allows new type of overlap in communication
- Leverages hardware capabilities
- Will be a primary tool on GPU-based systems
- Formerly known as Finepoints
- Demonstrated improvements with ECP mini-app
  - ~5% improvement in runtime
  - ~25% improvement in communication
- Part of MPI 4.0
- Part of Open MPI (master)
- MPI_Psync refinement will further improve GPUs (MPI 4.1)

MPI Sessions

- More scalable and flexible “construction” of MPI applications
  - MPI startup is local rather than collective
  - MPI groups can be constructed locally prior to minting communicators
  - Supports multiphysics and other use cases
- Part of MPI 4.0
- Prototype implementation available for Open MPI
  - Plan to officially merge after OMPI 5.0.0 release

ECP OMPI-X and User Space Threads

User-Level Lightweight Threads in MPI

- Supporting “MPI+X” hybrid execution scenarios
- Supporting CPU threading and high interconnect concurrency
- User-level thread (ULT) have lower work scheduling overheads compared to Pthreads
- User-level thread (ULT) support in Open MPI is functional
  - Supports both Qthreads and Argobots
  - Similar support in MPICH
- Exploring and addressing lower-level interactions between MPI and threading with high concurrency, process and thread bindings, etc.

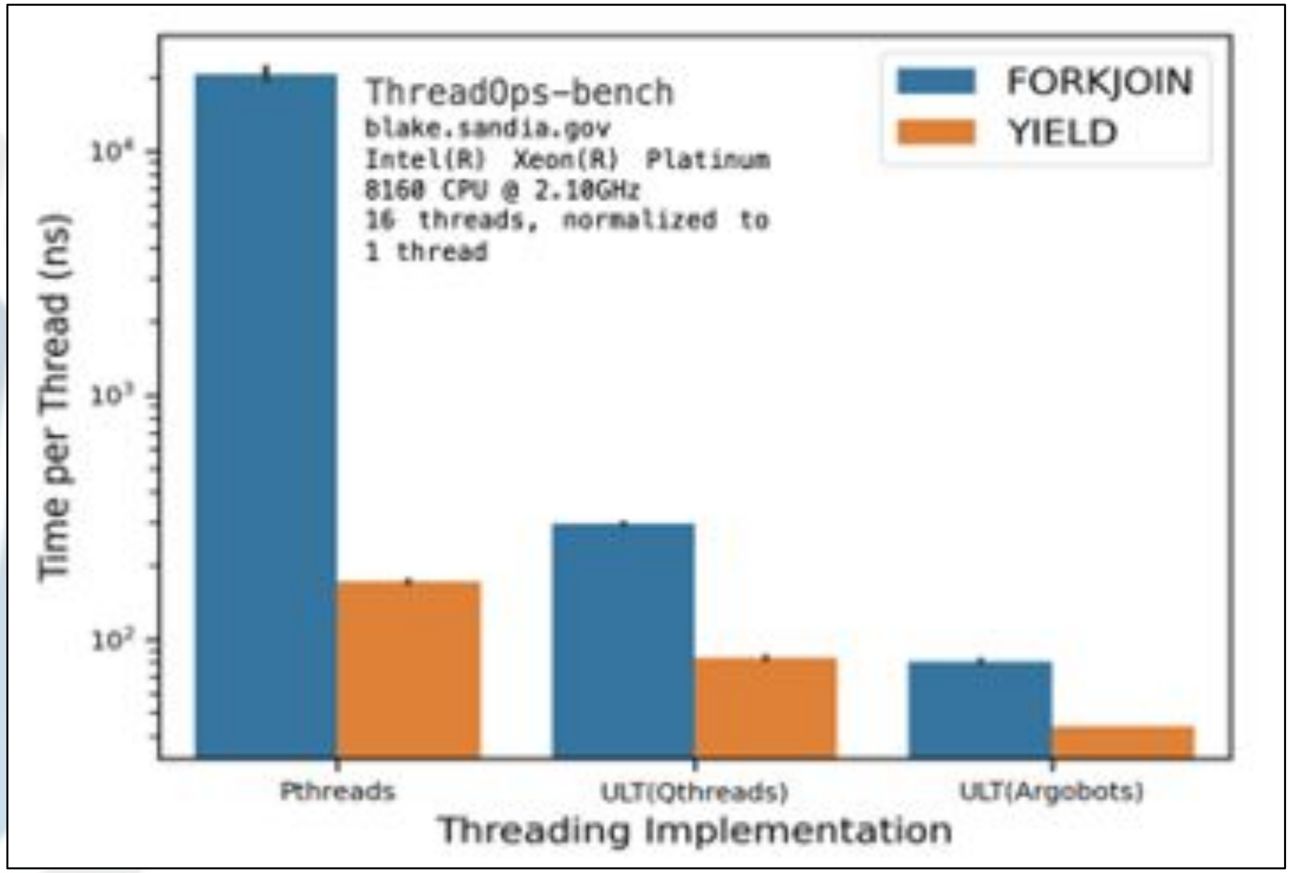
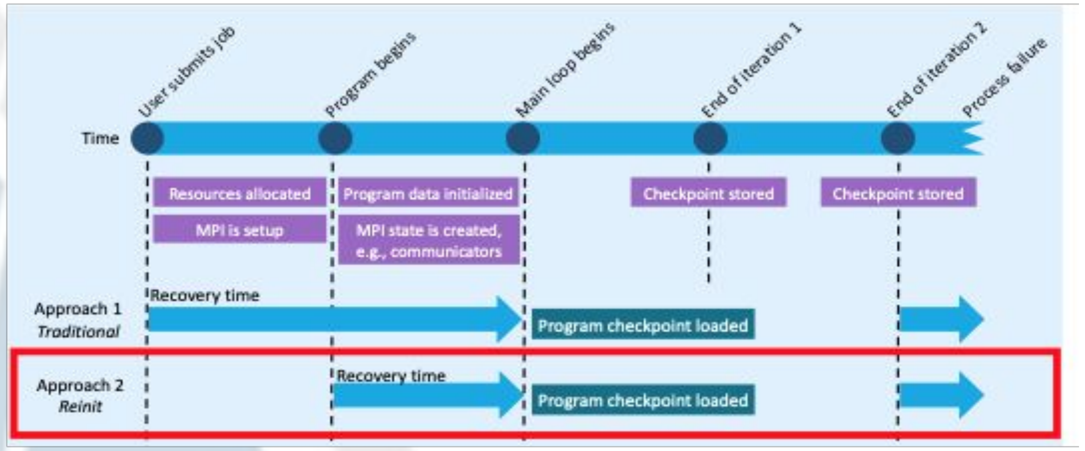


Illustration of relative costs of PThreads vs ULT threads (Qthreads, Argobots) using ThreadOps benchmark for fork-join and yield operations

ECP OMPI-X Beyond MPI 4.1

Failure Recovery using Reinit

- Reinit recovers from process and node failures
  - Implements backwards recovery for checkpointed applications
- Easy-to-use interface of a single MPI function
- Protocol has been formally verified
- Performs and scales better than traditional checkpoint/restart across a wide range of failure scenarios
- Prototype implementation available for Open MPI
- Under discussion for future MPI standard



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

Approach 1 *Traditional*

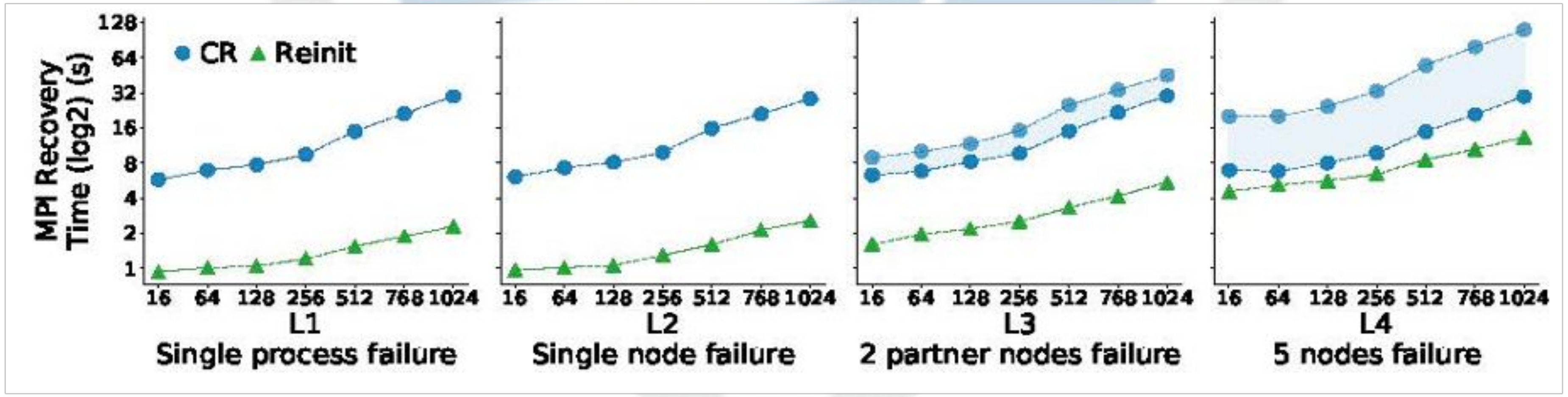
Recovery time

Program checkpoint loaded

Approach 2 *Reinit*

Recovery time

Program checkpoint loaded



Comparison of recovery times for traditional checkpoint/restart (blue) and Reinit (green) for the HPCG benchmark. Each graph represents different types/severities of failures.

ECP OMPI-X Next Steps

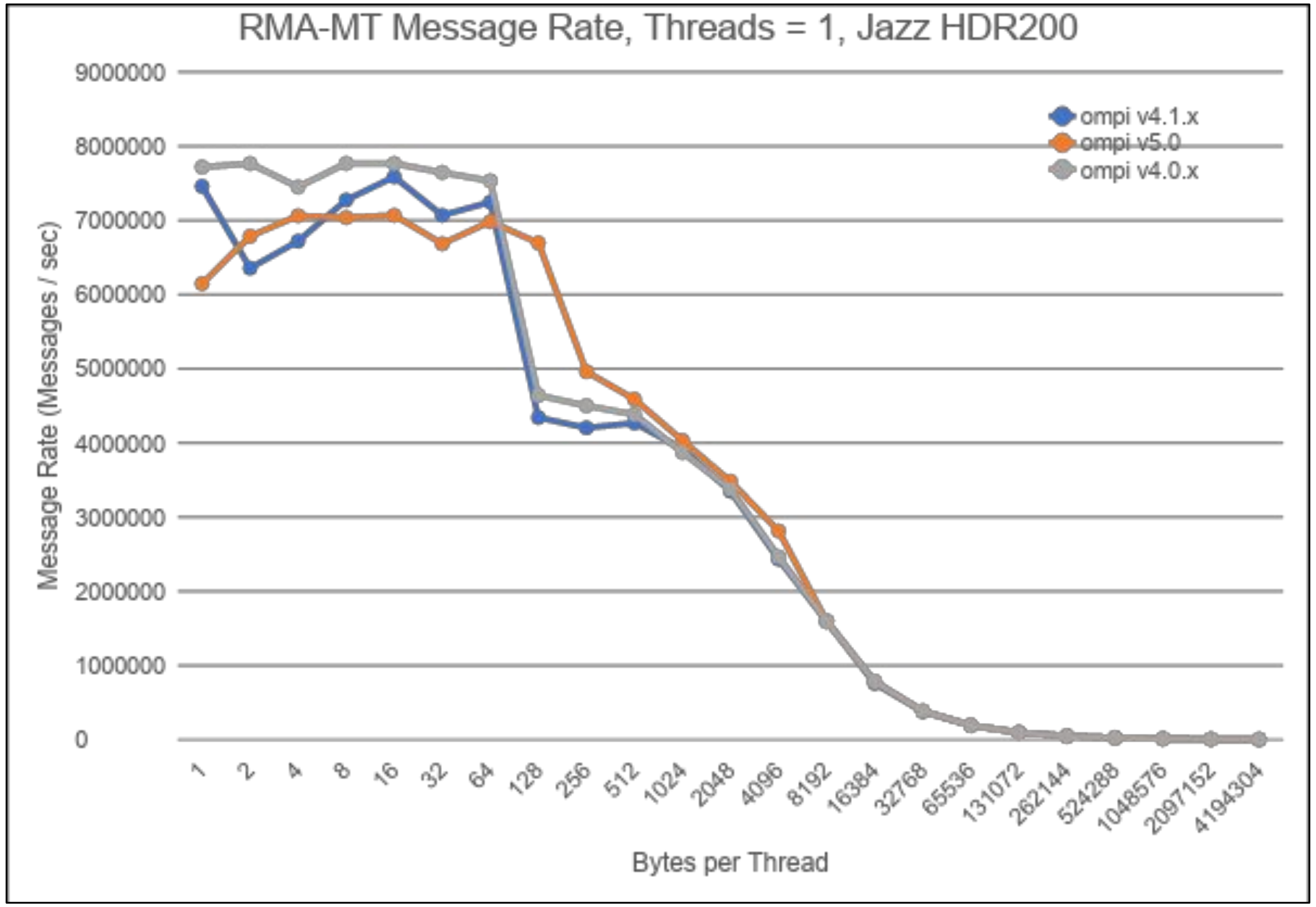
- Functionality and performance on networks of primary interest to ECP - HPE Slingshot11
- Explore methods for reducing GPU thread/host thread synchronization for MPI data motion operations

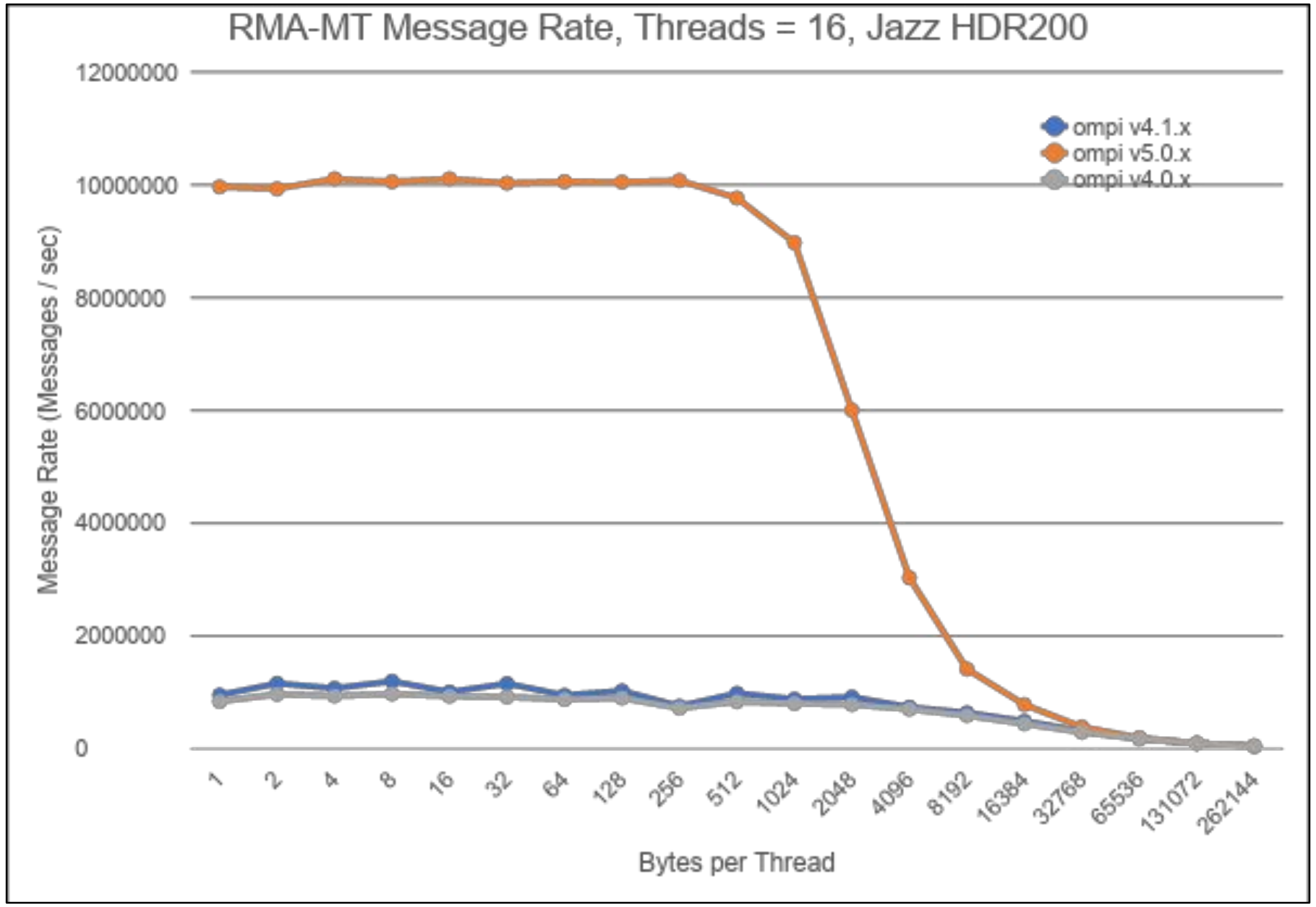
NVIDIA

Tomislav Janjusic
Joshua Ladd

Scaling Open MPI to Exascale

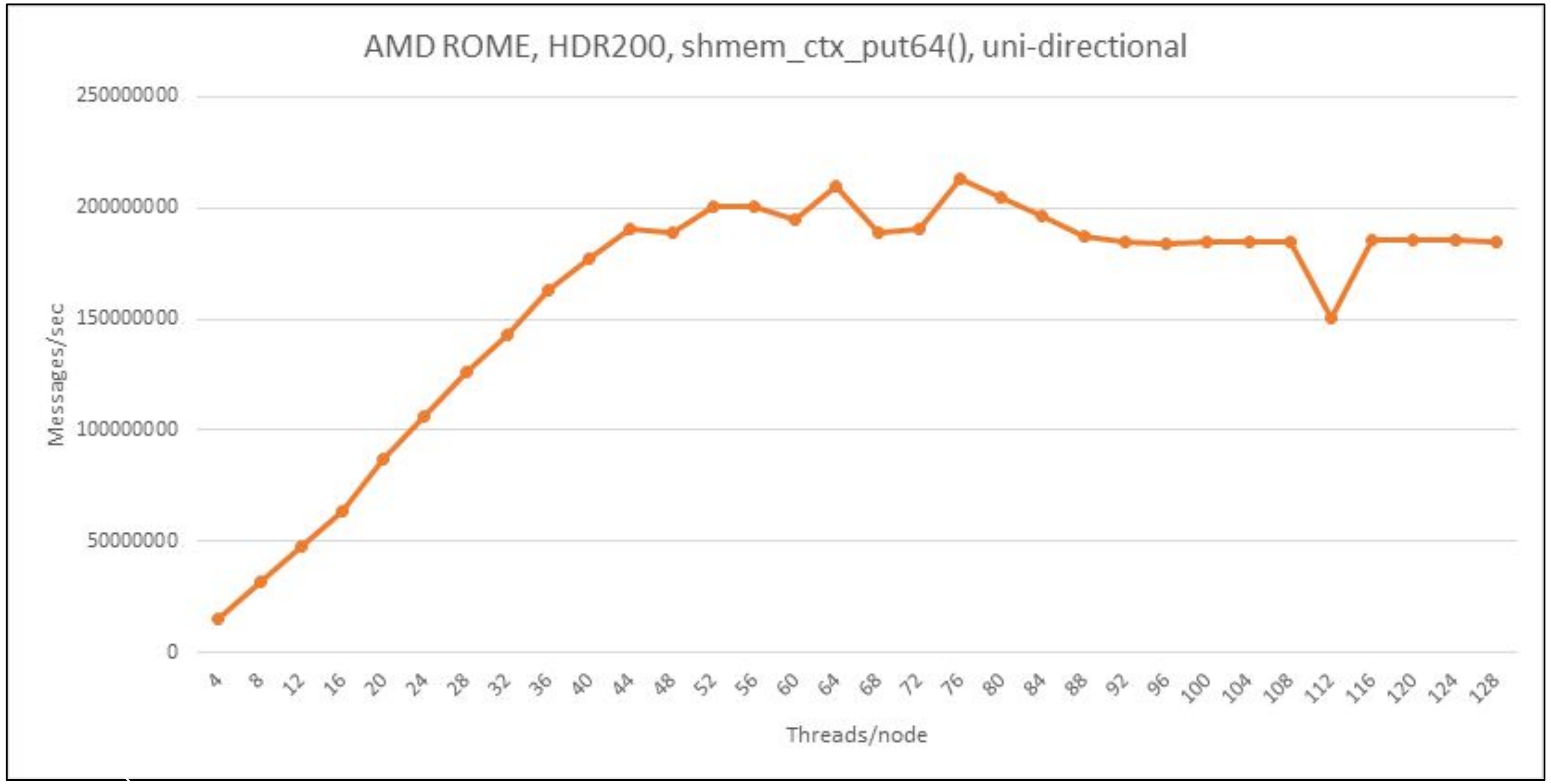
MPI one-sided thread multiple performance optimized out-of-the-box in v5.0.x





Scaling Open MPI to Exascale

- OSHMEM thread multiple performance also optimized out-of-the-box
  - `--mca spml_ucx_nb_progress_thresh_global 2048`
  - `--mca spml_ucx_nb_ucp_worker_progress 64`
  - `--mca spml_ucx_default_ctx_ucp_workers 2`



AWS

William Zhang

HPC on AWS

OFI Enhancements

- Added GPUDirect RDMA support for CUDA buffers in OFI
- Multi-NIC and OFI provider selection
  - Use hwloc to select providers with the closest NIC
- Unified memory monitors between Open MPI and Libfabric using new import monitor API

Our Next Year

- Improve GPUDirectRDMA support in Open MPI
  - Improve performance
  - Add integration with the OFI BTL to support one-sided operations
- OFI performance improvements

Questions?

Where Do We Need Help?

- Code
  - Any bug that bothers you
  - Any feature that you can add
- ***User documentation***
- Testing (CI, nightly)
- Usability

We ❤️ ⎇

Come join us!