Open MPI State of the Union XVII Community Meeting

Howard Pritchard, LANL
George Bosilca, NVIDIA
Aurelien Bouteiller, AMD
Thomas Naughton, ORNL

Version Roadmaps

v4.1.x (Still supported)

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

- Current release: 4.1.8
  - February 2025
- Likely to be the last release

Open MPI v5.x (Current stable)

Many improvements !

v5.0.x

- Release managers
  Tomislav Janjusic, NVIDIA

- Current release
  v5.0.9 - November 2025
- Likely to have a v5.0.10 release - 2026

Open MPI v5.0.x Major Changes

- Replaced launcher ORTE with PRRTE
  - PMIx Reference Runtime Environment (v2.0.x)
  - Requires PMIx v4.1.x or newer
- Added a new *accelerator* framework
  - Allows for modularized implementation of various devices
- Support for MPI Sessions added
- OpenIB BTL has been removed
- Replaced Debugger (MPIR) interface with PMIx Tools Debugger interface

New and Improved Runtime Environment

- PRRTE is our new launcher!
  - PMIx Reference Runtime Environment (v2.0.x)
  - PMIx standard based
- Run time no longer specifically tied to Open MPI
  - Also works with other MPI implementation
  - And in other non-MPI/HPC environments

Open MPI v5.0.6 to v5.0.9

- **Collective Operations (coll)**
  - **coll/ucc**: Fixed initialization in non-blocking/persistent operations, set node local id, and improved MPI_IN_PLACE handling
  - **coll/tuned**: Added version identifier and extended dynamic rules file for alltoall_algorithm_max_requests
  - **coll/basic/adapt/han/hcoll/cuda**: Fixed MCA variable scopes and type mismatches
- **Point-to-Point Messaging (PML/MTL/BTL)**
  - **pml**: Fixed memory overrun bug in ob1, added node local id support for UCX
  - **mtl/ofi**: Added rcache for hmem, improved CXI provider support, set accelerator rcache flags
  - **btl/ofi**: Exported memory monitor, removed EFA cache hack, fixed multi init/fini scenarios
- **One-Sided Communication (OSC)**
  - **osc/ucx**: Added "no_locks" info key support to disable lock table
  - **osc/rdma**: Fixed component compatibility when not using ob1

Open MPI v5.0.6 to v5.0.9

- **Fortran Support**
  - Fixed common symbol sizes, alignments, and alignment detection
  - Fixed string copy errors in c2f conversion (off-by-one and length mismatches)
  - Added missing profile symbols in use-mpi-f08, improved complex(real16) testing
  - Added support for new compilers (flang-new, nvfortran)
- **Accelerator Support**
  - **opal/accelerator/rocm**: Added missing memcpy headers
  - Fixed ROCm 6.0 and ROCm 7.0 compatibility issues
  - Fixed accelerator rcache flag handling across mtl/ofi and btl/ofi
  - Improved CUDA selection and updated documentation
- **Memory Management**
  - **mpool/hugepage**: Fixed sizing for better memory management
  - **rcache**: Cleaned up base struct, fixed reference counting in UCX dynamic windows
  - Fixed opal_atomic_int128_t alignment for s390x architecture

Open MPI v5.0.6 to v5.0.9

- **MPI Sessions**
  - Added UCX support and improved multi init/fini handling
  - Enhanced argument checking for comm constructors and PMIx_Group_construct
  - Added hello_sessions example and fixed singleton session directory creation
- **I/O (io)**
  - **io/ompio**: Fixed file_seek calculation when using SEEK_END
  - **romio314**: Removed cudart linking dependency
- **ARM/AARCH64 Support**
  - Added SVE detection alongside NEON in aarch64 op component
- **Documentation**
  - Added "Last updated on" footer, fixed numerous typos and formatting issues
  - Updated MPI_Init/*Finalize*/Session* man pages and MCA variable documentation
  - Enhanced mpirun documentation for --pmixmca and --prtemca options

Open MPI v6.0.0

Even more improvements are coming!

Open MPI v6.0.0

- Release managers
  - Edgar Gabriel, AMD
  - Howard Pritchard, LANL
- Pre-releases expected in December


Open MPI v6.0.0 Major Enhancements

- MPI 4.1 compliant
  - Support for MPI_Count (aka big count)
- Enhanced GPU support
  - Extended *accelerator* framework to support IPC methods provided by CUDA and ROCm
  - Support for memkind (➡see AMD presentation)
  - Intel Level-Zero support (without IPC)
- Significant enhancements to collective operations
- ROMIO removed
  - Very old version, no support big count
- Support for Eviden BXI networks

Open MPI v6.0.0 Runtime Changes

- Replaced PRRTE with our own fork (https://github.com/open-mpi/prrte)
  - Renamed PRRTE executables to avoid name clashes with native PRRTE install (should be transparent to users using mpirun/mpiexec)
  - Emphasis on stability rather than features
  - Open MPI can still be built against external OpenPMIx and PRRTe installations
  - Requires PMIx 6.0.0 or later if building Open MPI with the internal OMPI PRRTE

Collective operations

- ***xhc***: new shared memory collective component using xpmem
- ***acoll***: new ➡ see AMD presentation
- ***ucc***: added support for persistent collective (or ➡ see NVIDIA presentation, to be clarified with George)
- ***han***: alltoall performance enhancements
- ***accelerator***: ➡ see AMD presentation
- ***tuned***: new json based configuration file (do we want to show an example?)

- ***hcoll***: component removed
- ***sm***: component removed

Collective operations

Enable user to determine which component is used for a collective operation:

```
$ mpirun --mca coll_base_verbose 10 -np 2 ./osu_bcast
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 allgather -> accelerator
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 allgatherv -> tuned
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 allreduce -> accelerator
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 alltoall -> accelerator
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 alltoallv -> tuned
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 alltoallw -> basic
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 barrier -> tuned
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 bcast -> accelerator
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 iallgather -> libnbc
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 iallgatherv -> libnbc
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 iallreduce -> libnbc
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 ialltoall -> libnbc
 coll:base:comm_select: communicator MPI_COMM_WORLD rank 0 ialltoallv -> libnbc
```

Note: some components forward execution to another component under certain conditions

MPI 5.0 ABI Support

- May be too late for 6.0.x release stream but still under consideration
- Plan is to first support ‘c’ ABI, including helper functions for external Fortran bindings implementations
- Lots of thanks to Lisandro Dalcin for testing/providing fixes
- Leverages Python framework used for big count support
  - Thanks to Jake Tronge (GATech) and Joe Downs (Univ. Louisville)
- ABI mpi.h generated from MPI standard using https://github.com/mpi-forum/pympistandard + patches
- Plans for Fortran ABI less certain, may rely on external Fortran bindings (e.g. vapaa) to provide this support

Backup

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

What does that encompass?

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

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

Open MPI v5.0.0 to v5.0.9

- Documentation
  - Documentation updates
  - Improved installation and configuration guides
  - Updated man pages and error messages
  - Added better guidance for packagers
- Third-party Component Updates
  - Updated OpenPMIx and PRRTE to latest versions
  - Improved integration with third-party libraries
  - Better handling of dependencies
- Testing improvements
  - Added CUDA and ROCm compile checks
  - Enhanced CI/CD workflows
  - Added macOS testing
  - Improved mpi4py + HDF5 testing support

Open MPI v5.0.0 to v5.0.9

- Performance & Reliability
  - Fixed race conditions in various components
  - Improved memory handling and leak fixes
  - Enhanced error handling in several components
  - Added AARCH64 OP component with NEON and SVE ISA support
  - Added MCS-based implementation for SHMEM locks for better scaling in OSHMEM
- CUDA/ROCm Updates
  - Fixed ROCm 6.0 and ROCm 7.0 compatibility issues
  - Improved handling of CUDA memory management
  - Fixed build system and linkage for non-standard libcuda.so locations
- Build System & Configuration
  - Added support for new compilers (flang-new, nvfortran)
  - Improved DSO (Dynamic Shared Objects) handling
  - Updated build configurations for better packaging

Open MPI Updates from AMD

Edgar Gabriel
Aurelien Bouteiller
Manu Shantharam
Nithya VS

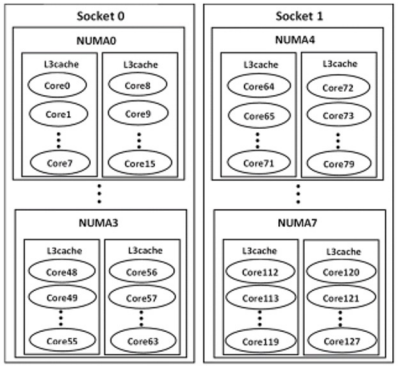
edgar.gabriel@amd.com
aurelien.bouteiller@amd.com
manu.shantharam@amd.com
nithya.vs@amd.com

AMD
together we advance_

Upcoming in Open MPI 6.0: acoll collective component

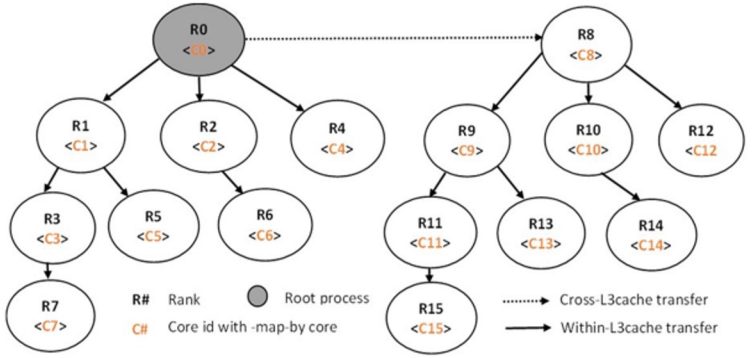
Affinity-Aware collectives for intra-node communication (acoll)

- "acoll" is optimized for communications within a single node of AMD "Zen"-based processors
- provides support for many commonly used collective operations, including MPI_Bcast, MPI_Allreduce, MPI_Reduce, MPI_Gather, MPI_Allgather, and MPI_Barrier



Two stages: Instantiation and Communication

- **Instantiation** – once during initialization of communicator
  - **Identify affinity** related information through hardware locality software package like 'hwloc', form groups of processes/ranks with similar affinities
- **Communication** stage: 2 steps
  - perform communication among **local ranks** within a group
  - perform communication **across affinity groups**



acoll: How-to and what's next?

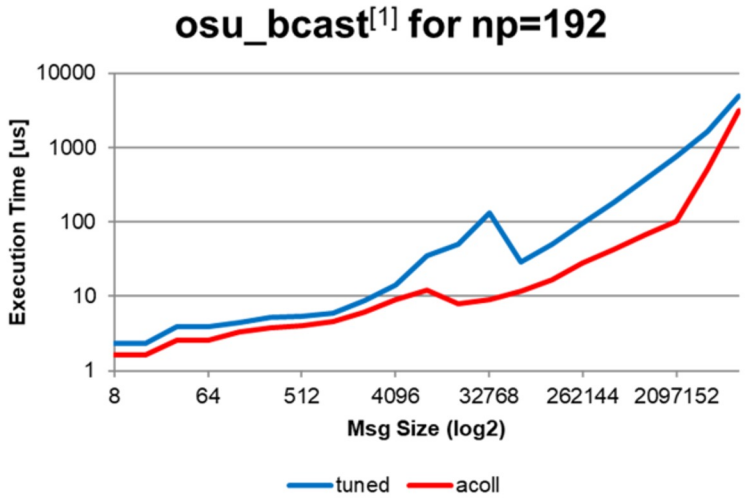
How to build and run with acoll

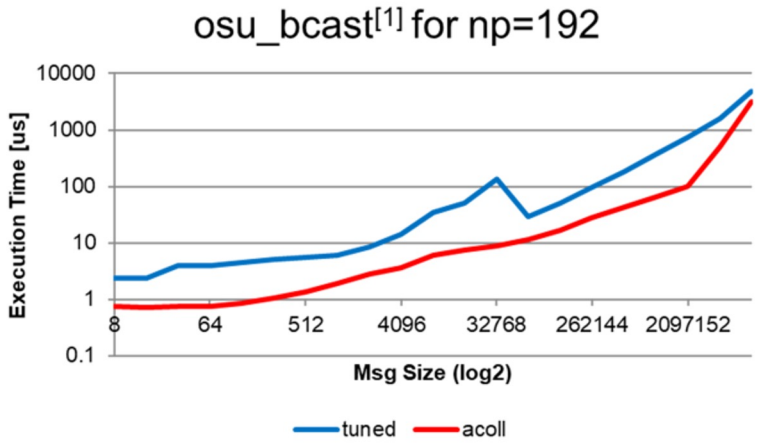
- Part of the main branch and will be part of 6.0.x
- Clone the Open MPI repository, build and install according to documentations
- Running with acoll component

```
mpirun <common mpi runtime parameters> --mca coll
acoll,tuned,libnbc,basic -mca coll_acoll_priority 40
./<executable>
```

What's next

- Extend acoll to interact with accelerator component
- Inter-node support

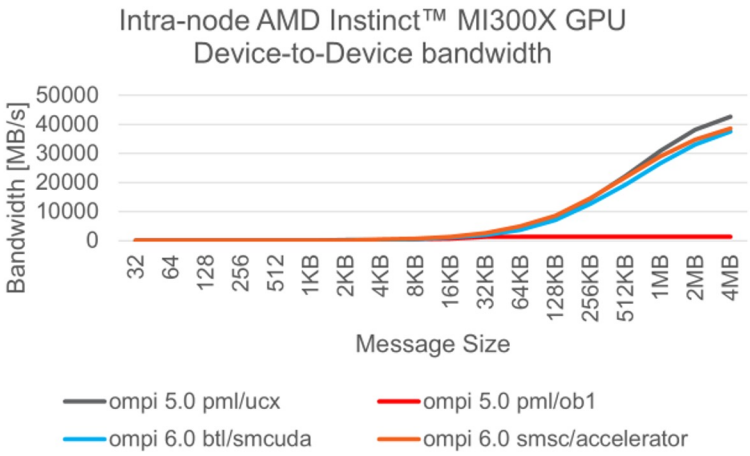




[1] OSU Benchmark Suite https://mvapich.cse.ohio-state.edu/benchmarks/ (BSD License)

Open MPI 6.0 upcoming enhancements for GPU devices

- **Added support for Open MPI native intra-node device-to-device transfers**
  - **Single copy** accelerator component (***smsc/accelerator***)
  - Generalized the existing ***btl/smcuda*** component to work **with any accelerator** supported by Open MPI
  - Both components achieve similar performance to UCX for device-to-device transfers
  - Brings **performance parity** for system configuration that are not supported by UCX
  - Components have to be explicitly requested by user

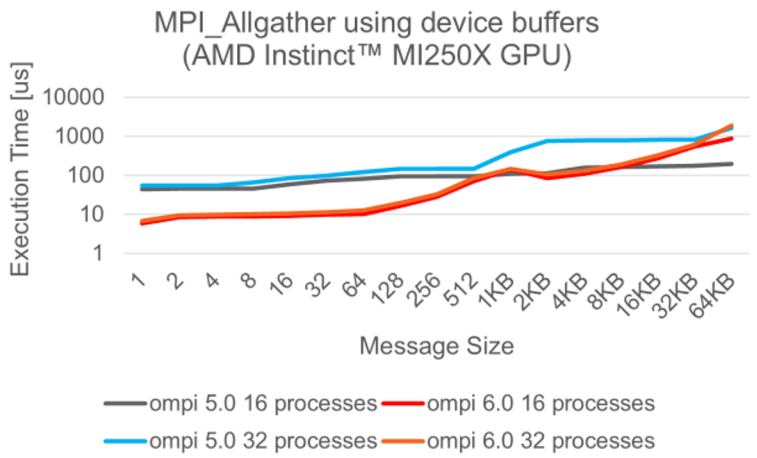


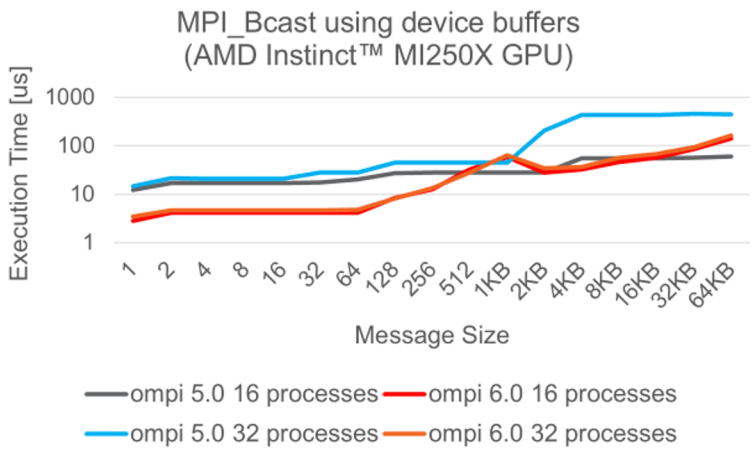

```
mpirun --mca pml ob1 --mca smsc_accelerator_priority 80 -np 4 ./<executable>
```

- **Support for MPI 4.1 memory-allocation kinds**
  - User can restrict memory types supported for an `MPI_Comm`, `MPI_Win`, `MPI_File`, or `MPI_Session`
  - A few optimizations taking advantage of this feature already available in Open MPI 6.0
    - Some components can disqualify themselves during object creation based on the info object value
    - Buffer-pointer type-check bypass depending on the info object value in the parallel file I/O component
    - More to follow

Enhancements to collective operations on device buffers

- Added support to the ***coll/accelerator*** component for MPI_Bcast, MPI_Allgather, and MPI_Alltoall
- Significant performance improvements for short messages





DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18


AMD

Open MPI Activities at DOE – ORNL/LANL

Howard Pritchard (LANL)
Thomas Naughton (ORNL)
Amir Shehata (ORNL)

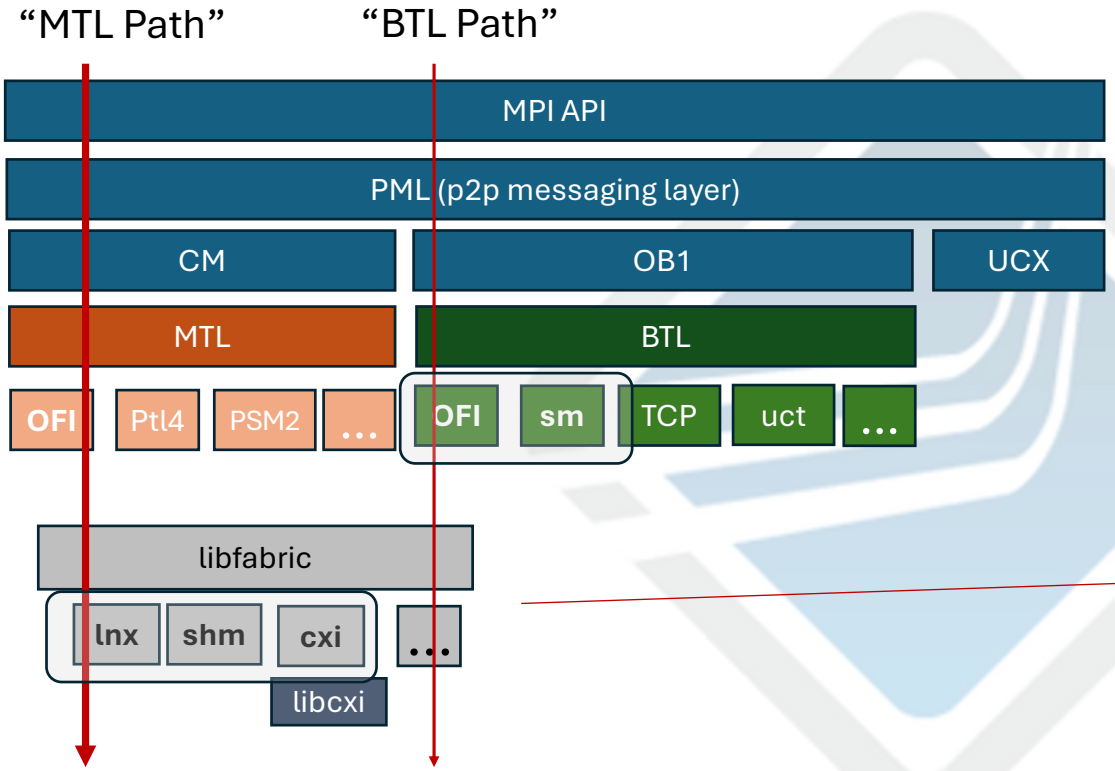
Open MPI on DOE Systems

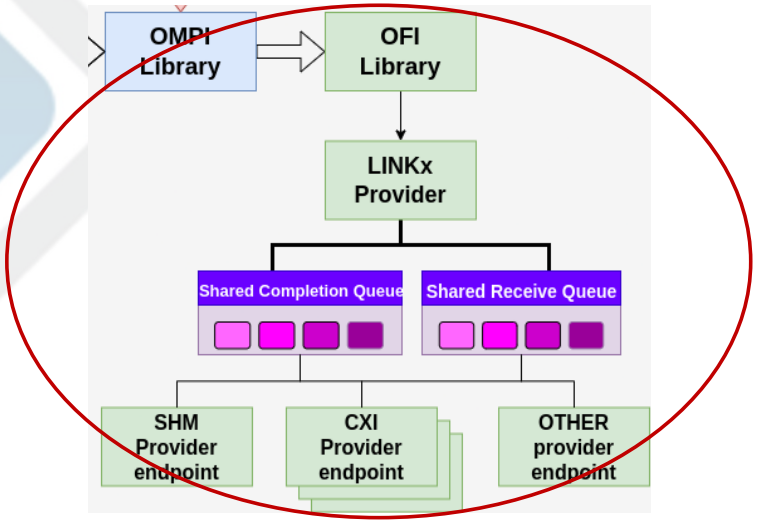
- DOE Exascale systems
  - Slingshot 11, different GPU devices & node topologies
- ORNL & LANL working on SS11 items for
  - Different resource managers via OpenPMIx/PRRTE
  - Both MTL & BTL paths usable with libfabric provider

https://s4pst.org

https://cass.community

Where Libfabric fits in Open MPI Architecture Diagram





Libfabric “lnx” provider

- Links two or more providers for seamless use
  - Renamed “LINKx” → “LNX
  - Provider must support FI_PEER capability
  - Useful for linking Slingshot 11 (CXI) and Shared Memory (SHM) providers
  - Can be used for multi-rail setups

Usage with Open MPI

- OFI provider to 'lnx'

  OMPI_MCA_opal_common_ofi_provider_include=**lnx** **# OFI 2.x change**

- Runtime settings for Slurm

  PRTE_MCA_ras_slurm_use_entire_allocation=1 **# Give mpirun all CPUs**

  PRTE_MCA_ras_base_launch_orted_on_hn=1 **# Launch prted via srun**
  # to setup Slingshot VNI

- Versions

Open MPI	>= 5.0.7
Libfabric	>= 2.2.0
Slingshot	Libcxi 1.0.x (SHS 12.0.1)

Usage with Open MPI (2)

- PCI location of LNX fi_info from 1st device in link
  - Example: Frontier's 4 SS11 devices (cxi0,1,2,3)

FI_LNX_PROV_LINKS=**shm+cxi:cxi0|shm+cxi:cxi1|shm+cxi:cxi2|shm+cxi:cxi3**

Closest to **CPUs 48-63 NUMA 3** | **CPUs 16-31 NUMA 1** | **CPUs 0-15 NUMA 0** | **CPUs 32-47 NUMA 2**

FI_LNX_PROV_LINKS=**shm+cxi:cxi0,cxi1,cxi2,cxi3**

Closest to **CPUs 48-63 NUMA 3**

Usage with Open MPI (2)

- PCI location of LNX fi_info from 1st device in link
  - Example: Frontier's 4 SS11 devices (cxi0,1,2,3)

FI_LNX_PROV_LINKS=**shm+cxi:cxi0|shm+cxi:cxi1|shm+cxi:cxi2|shm+cxi:cxi3**

Closest to — CPUs 48-63 NUMA 3 | CPUs 16-31 NUMA 1 | CPUs 0-15 NUMA 0 | CPUs 32-47 NUMA 2

~~FI_LNX_PROV_LINKS=**shm+cxi:cxi0,cxi1,cxi2,cxi3**~~

~~Closest to — CPUs 48-63 NUMA 3~~

On Frontier, **NOT** what you want

Full Listing: Environment Settings

```
PRTE_MCA_ras_slurm_use_entire_allocation=1
PRTE_MCA_ras_base_launch_orted_on_hn=1
OMPI_MCA_opal_common_ofi_provider_include=lnx
OMPI_MCA_mtl=ofi
OMPI_MCA_mtl_ofi_av=table

FI_SHM_USE_XPMEM=1
FI_LNX_PROV_LINKS=shm+cxi:cxi0|shm+cxi:cxi1|shm+cxi:cxi2|shm+cxi:cxi3
```

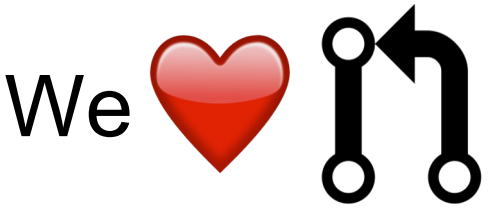
Getting Started on Frontier

```
module unload cray-mpich cray-pmi

module use   ums ums024
module load  ums ums024
module load  openmpi/5.0.9

mpirun --bind-to core --map-by ppr:1:l3cache \
       --np $SLURM_NTASKS \
       gpuwrapper.sh ./app
```

Questions?

Where Do We Need Help?

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



Come join us!