Open MPI State of the Union XVI Community Meeting

George Bosilca, NVIDIA
Edgar Gabriel, AMD
Tomislav Janjusic, NVIDIA
Thomas Naughton, ORNL
Luke Robison, Amazon

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

v4.1.x (end of life)

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

- Current release: 4.1.7
  - November 2024
- Likely to be the last release

Open MPI v5.x (Current stable)

Many improvements !

v5.0.x

- Release managers
  - Tomislav Janjusic, NVIDIA

- Current release
  - V5.0.6 November 2024
    - V5.0.7-rc1 to be released ASAP

Open MPI v5.0.x Major Changes

- Replaced launcher ORTE with PRRTE
  - PMIx Reference Runtime Environment (v2.0.x)
  - Requires PMIx v4.1.x
- 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
- Removed MPI C++ bindings

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 v5.0.0 to v5.0.6

- 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 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 v5.0.0 to v5.0.6

- Documentation (https://docs.open-mpi.org/)
  - 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 compile checks
  - Enhanced CI/CD workflows
  - Added macOS testing
  - Improved mpi4py testing support

Open MPI v6.x

Harder, Better, Faster, Stronger

Open MPI v6.0.0 Info

- Release managers
  - TBD
- Release Date
  - Soon

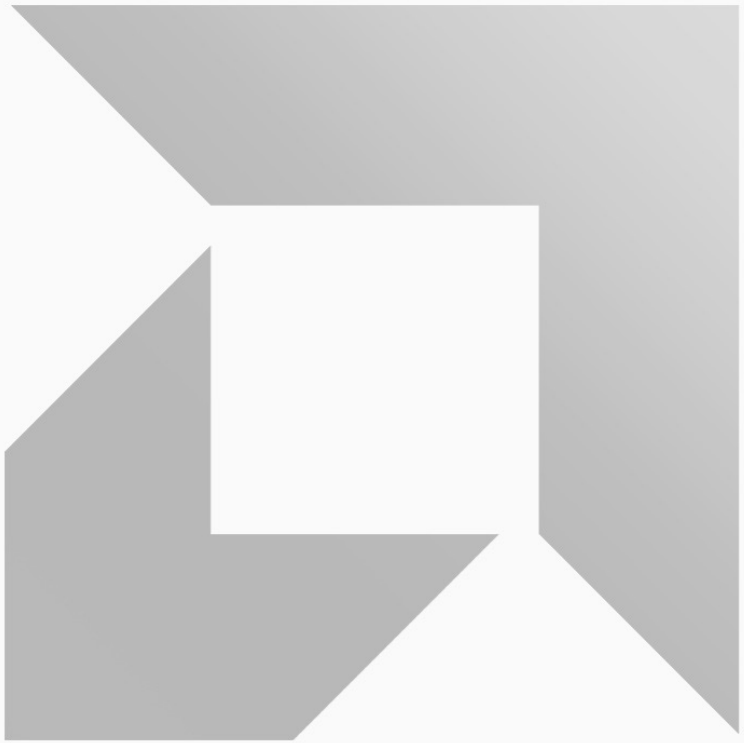
Open MPI v6.0.0 Major Changes

- Extended *accelerator* framework to IPC methods provided by CUDA and RoCM
  - Add MPI_Op support for accelerators
- New collective's components (acoll, xhc)
  - Improve the tuning capabilities by moving to a JSON configuration file (WIP)
- Support for Sessions across the entire code base
- Full support for big count support (MPI_COUNT)
- Support for multiple Fortran compilers on the same install base
- Support for memkind
- ROMIO refresh
- MPI 4.1 compliant (MPI_Event development is lacking manpower)
- MPI ABI (WIP)

Runtime Environment Support

- 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 (MPI features such as spawn/singleton/sessions, or MCA params)
  - Removed support for building Open MPI against an external PRRTE, but an external PRRTE can be used for job launch (using prterun, etc.)
  - Requires PMIx v4.2.0 or later

- Tentative approach until we figure out a better way



Open MPI Updates from AMD

Edgar Gabriel
Manu Shantharam
Nithya VS

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

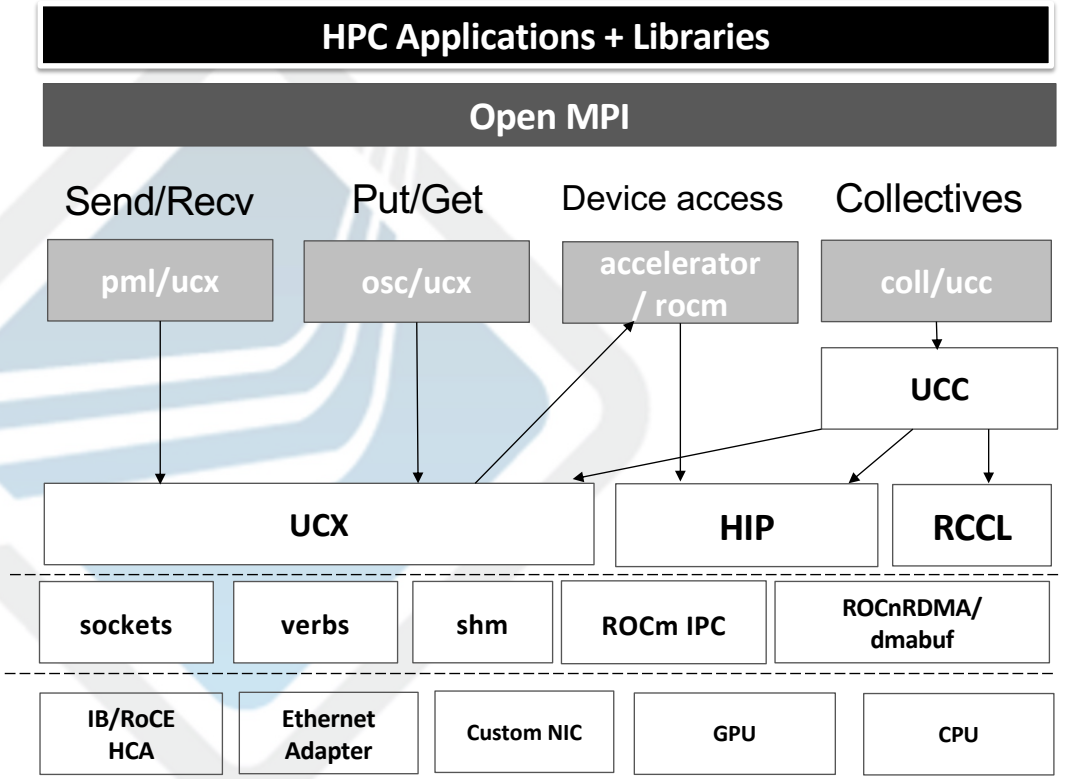
AMD
together we advance_

ROCm Support in Open MPI 5.0

- ROCm support added to Open MPI starting v5.0
  - ROCm memory type detection
  - ROCm device memory allocation
  - Data transfer to/from ROCm device memory
    - Non-contiguous derived datatypes
    - MPI File I/O
  - Enables usage of additional data transfer components (e.g., libfabric, ob1, etc.)
  - Query availability of ROCm support in MPI library (`MPIX_QUERY_ROCM_SUPPORT`)

ROCm Aware Open MPI Software Stack with UCX

- Recommended software stack for InfiniBand and RoCEv2 networks



Shared Memory GPU IPC Support in Open MPI

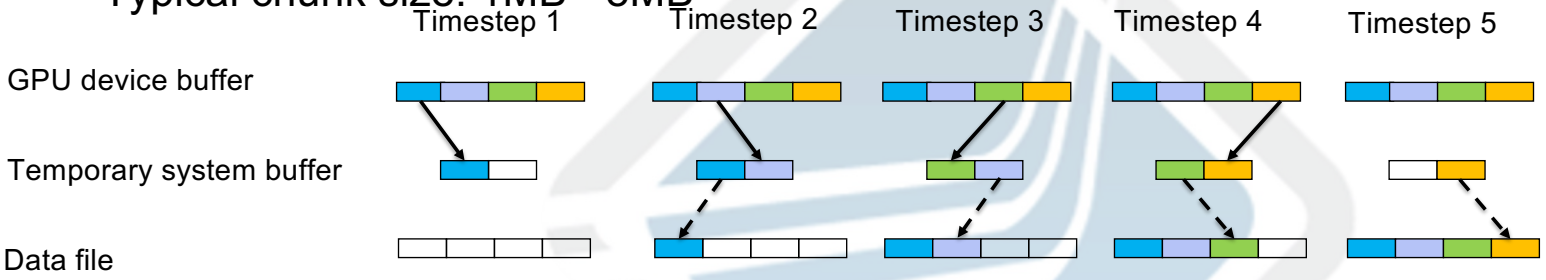
- Extended *accelerator framework* interfaces to support GPU IPC operation
  - Direct GPU-to-GPU data transfer
- Generalized the *btl/smcuda* component to use new API functionality
  - Supports now all GPU types that implement the accelerator framework IPC APIs
- Ongoing work: single-copy GPU IPC component (*smsc/accelerator*) to be used with *btl/sm*
- Beneficial for single node scenarios
- Beneficial for use on network interconnects using libfabric
  - Currently recommended to use the libfabric btl component (*btl/ofi*) instead of *mtl/ofi*

```
mpirun --mca pml ob1 --mca btl_ofi_mode 2 -np 64 ./<exec>
```

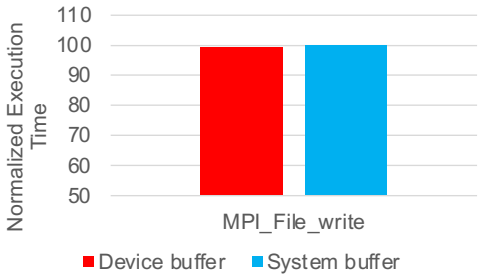
will use *btl/ofi* for inter-node communication, and *btl/sm* or *btl/smcuda* for intra-node communication

MPI File I/O Support for Device Memory: Individual I/O Operations

- Pipeline Protocol used to perform individual File I/O operations from GPU buffers
  - Two temporary buffers in system memory used by the MPI library
  - Double buffering used to alternate between temporary buffers for staging and file I/O
  - Typical chunk size: 1MB - 8MB

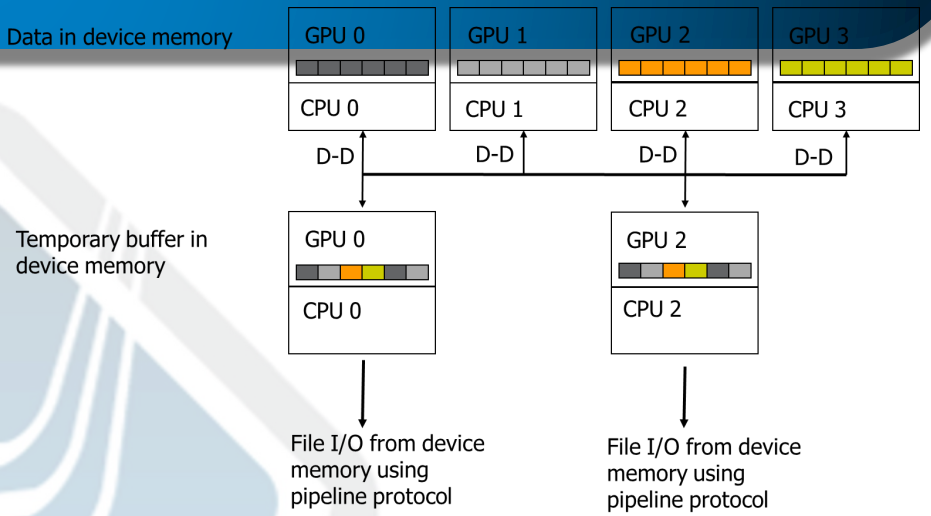


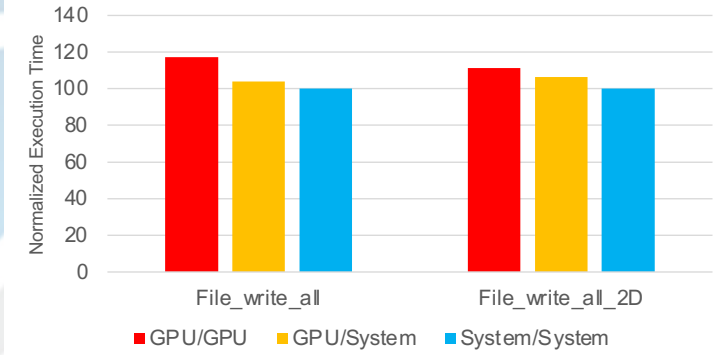
- Preliminary Results
  - AMD EPYC 7A53
  - AMD MI250X GPU
  - Slingshot 11 interconnect
  - Lustre File system



MPI File I/O Support for Device Memory: Collective I/O Operations

- Two options for setting up collective I/O operations
  - Aggregator processes use host memory for data aggregation
    - Communication from Device-to-Host memory
    - File I/O performed from Host/System memory
  - Aggregator processes use device memory for data aggregation
    - Communication from Device-to-Device memory
    - File I/O performed from device memory using pipelined protocol





MPI File I/O Support for Device Memory

	Feature	Open MPI 5.0.x	Open MPI 6.0.x
`MPI_File_write` `MPI_File_read`	Pipeline protocol	✓	✓
`MPI_File_iwrite` `MPI_File_iread`	Pipeline protocol	x	✓
`MPI_File_write_all` `MPI_File_read_all`	Using system buffer for aggregation	✓	✓
`MPI_File_write_all` `MPI_File_read_all`	Using device buffer for aggregation	x	✓
`MPI_File_iread_all` `MPI_File_iwrite_all`	GPU support in general	(not tested)	(not tested)

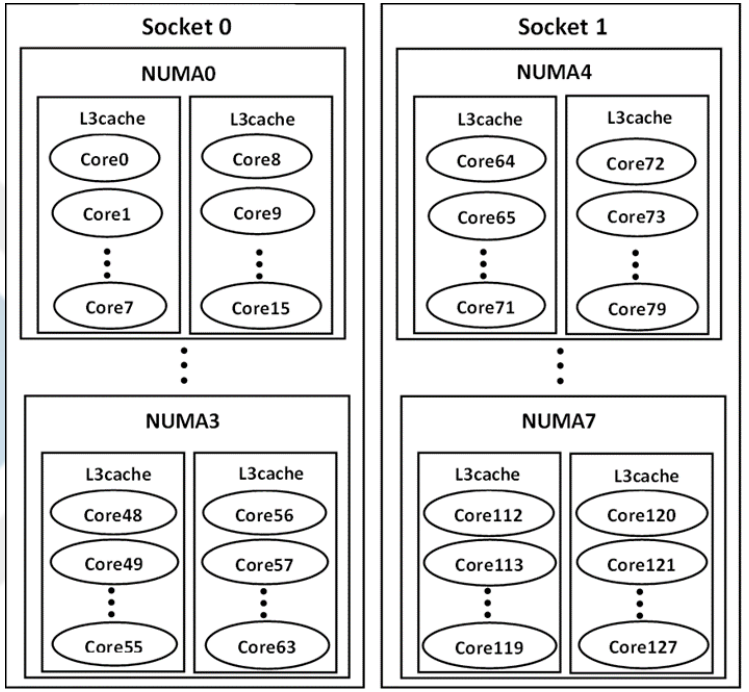
Upcoming in Open MPI 6.0: acoll collective component

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

- “acoll” is optimized for communications within a single node of AMD “Zen”-based processors
- provides the following commonly used collective algorithms: boardcast (MPI_Bcast), allreduce (MPI_Allreduce), reduce (MPI_Reduce), gather (MPI_Gather), allgather (MPI_Allgather), and barrier (MPI_Barrier)

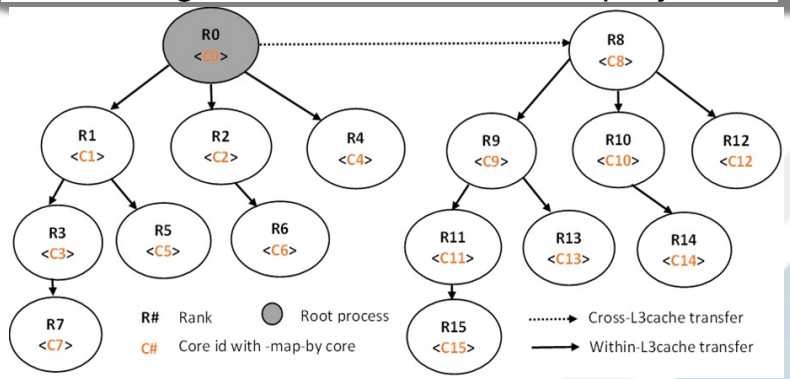
Two stages: Instantiation and Communication

- 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

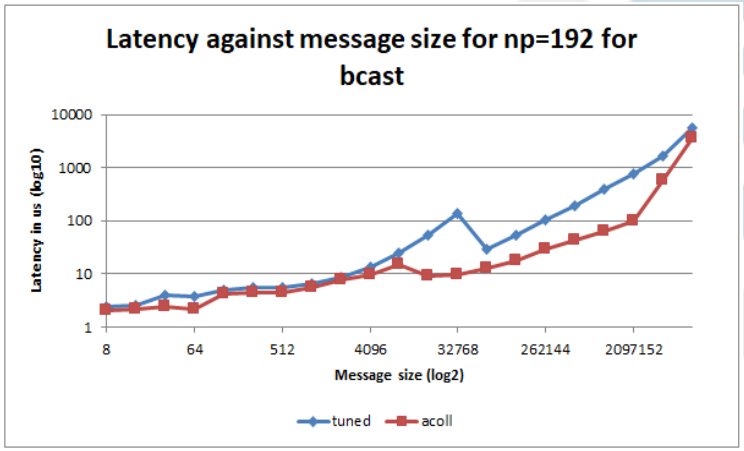


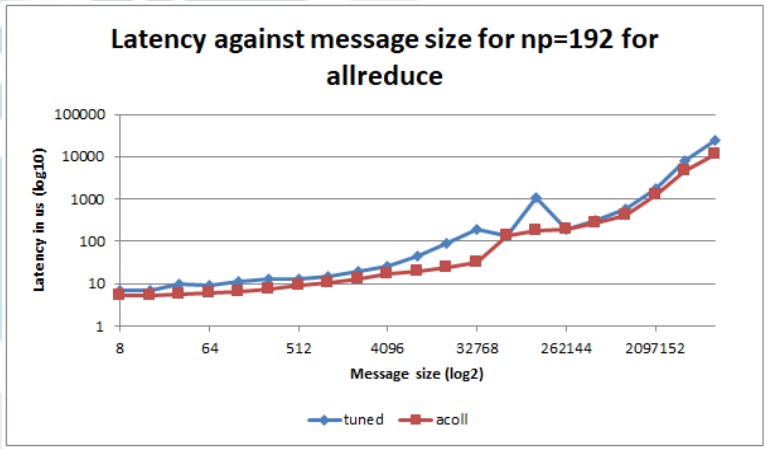
acoll: Example and Preliminary Evaluation

Bcast using acoll with 16 ranks, -map-by core



- System
  - 2-socket
  - AMD EPYC 9654 96-Core Processor per socket
  - 4 NUMA domains per socket
  - UCX (1.15.0), XPMEM (2.3.0)
- Benchmark Suite: OSUv5.9
  - OSU Benchmark Suite, https://mvapich.cse.ohio-state.edu/benchmarks (BSD License)
- OpenMPIv5.0.2 with acoll
  - https://www.open-mpi.org/community/license.php (BSD License)





Lower is better

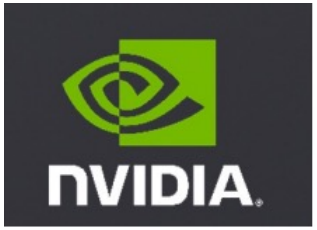
DISCLAIMER

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

OMPI v5.0.x NVIDIA Updates

Tomislav Janjusic, Joshua Ladd



ompi/coll component

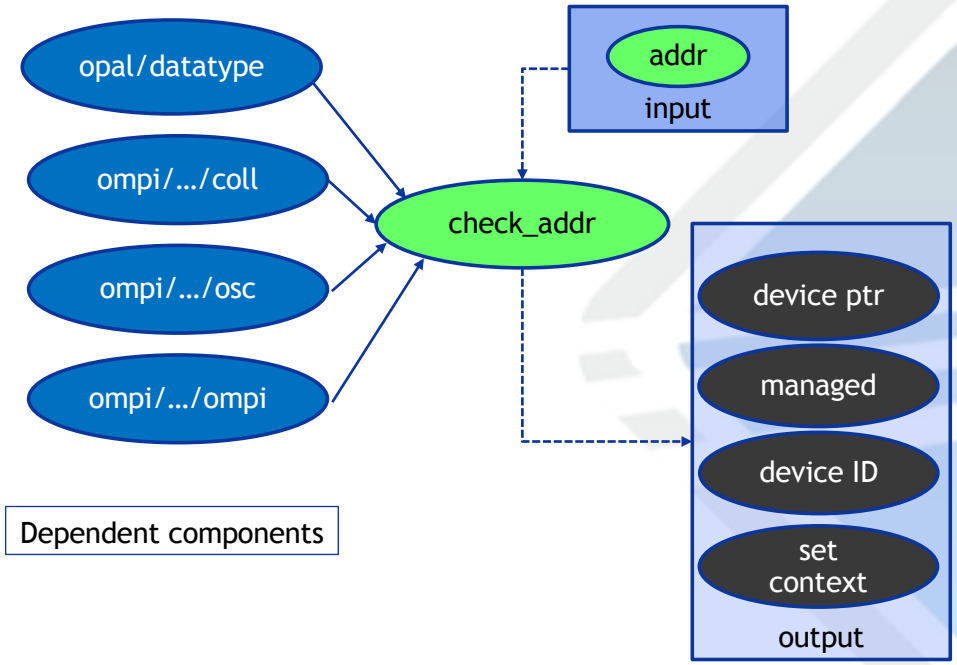
- Improve coll/tuned tuning guide in the docs
  - Add documentation to the user guide for the coll/tuned component
  - Summarize the key information needed to effectively use the tuning features
- Teach the coll tuned dynamic rules file reader to look for the alltoall_algorithm_max_requests tuning parameter
  - Set the max requests control variable through the dynamic rules file mechanism for the linear_sync N algorithm

CUDA

- cuMemCreate/cuMemAllocAsync support

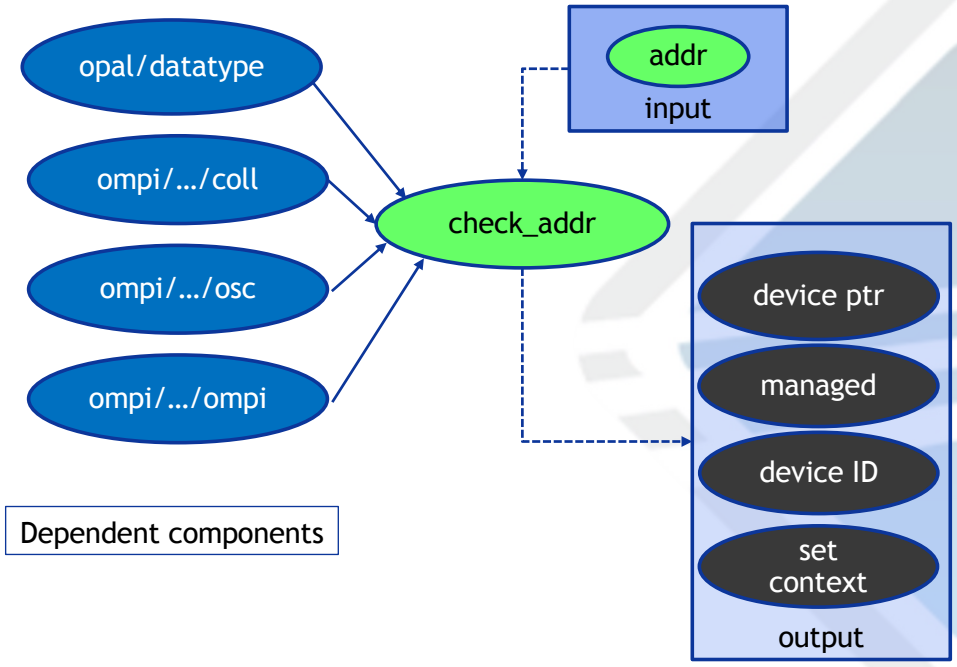
- Supported in OMPI v4.1.7 and v5.0.6 and everything after
- To be integrated with MPI memkind support

CUDA



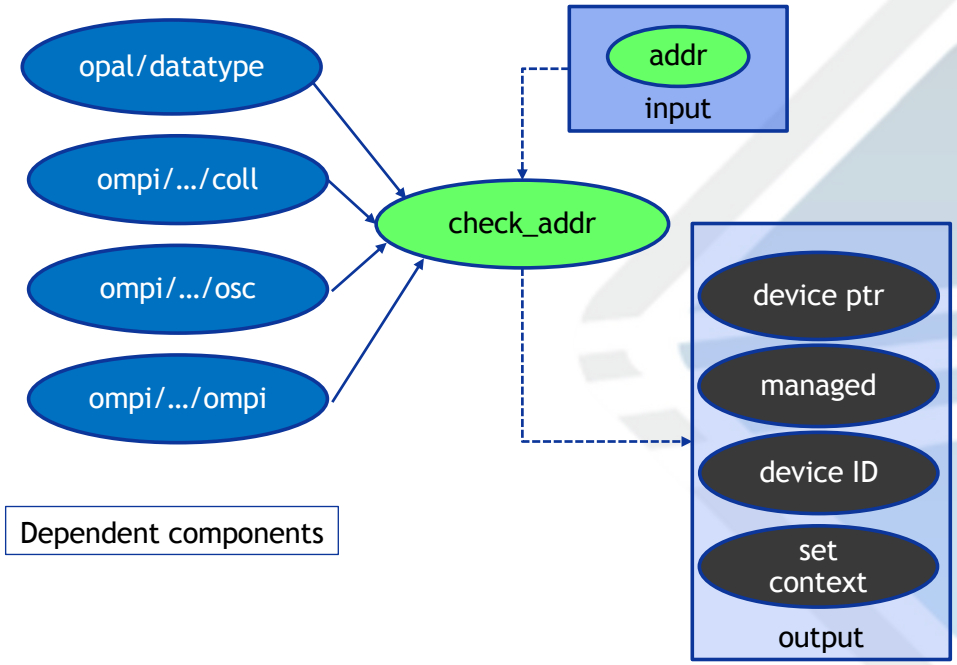
- OMPI and OPAL components check if a given CUDA pointer is host accessible by querying if “opal/cuda” or “accelerator” component can identify the pointer
- *check_addr()* function returns several attributes for dependent components to make decisions if it is recognizable

CUDA



- Until version v4.1.6, v5.0.5 CUDA accelerator component could identify cudaMalloc, cudaMallocManaged
- Newer API: **cuMemCreate()** and **cudaMallocAsync()** can create device or host backed memory and support **multi-node NVLINK capabilities**
- OMPI v4.1.7 and v5.0.6 can support allocations from the newer API.

CUDA



- Using the new API and UCX 1.18 users can take advantage of IPC capabilities to realize multi-node NVLINK transfers
- Open MPI and UCX will extend multi-node NVLINK support to Blackwell systems when the officially supported CUDA driver is released.

OpenSHMEM

- MCS implementation of SHMEM 1.5 LOCKS
- Lock refresher:
  - *shmem_set_lock();*
    mutual exclusion lock, wait for free lock before acquiring lock.
  - *shmem_test_lock*();
    similar to shmem_set_lock sets a mutual exclusion lock
    only if it is currently cleared
  - *shmem_clear_lock*();
    Clears lock previously set by *shmem_[set|test]_lock()*

OpenSHMEM

- MCS implementation of SHMEM LOCKS
  - Adding MCS algorithm-based implementation for *shmem_locks*
  - Improves performance for large scale SHMEM applications
  - New default algorithm which can be disabled by an mca parameter

    --mca oshmem_enable_mcs_lock 0 (fall-back to ticket lock)

OpenSHMEM

Aspect	Ticket Lock	MCS Lock
Fairness	Ensures FIFO order	Ensures FIFO order
Scalability	Spinning generates cache traffic	Each thread spins on local memory
Cache Coherence Traffic	Threads spin on shared current_ticket	Threads spin on private nodes
Memory Overhead	Two shared counters (next_ticket, current_ticket)	Each thread requires its own lock memory
Blocking Behavior	Spins on shared variable, increasing contention	Spins locally, reducing contention
Implementation Complexity	Simple and easy to implement	More complex due to linked list management
Performance Under High Contention	Poorer performance due to spinning on shared memory	Performs well; minimal contention on shared resources

Acknowledgements

Many thanks to the individuals and teams across various groups for their contribution to Open MPI, including:

Akshay Vekatesh, Vishwanath Venkatesan, Burlen Loring, George Bosilca

Performance Improvements for Real-World Applications

Luke Robison

aws

AWS for HPC

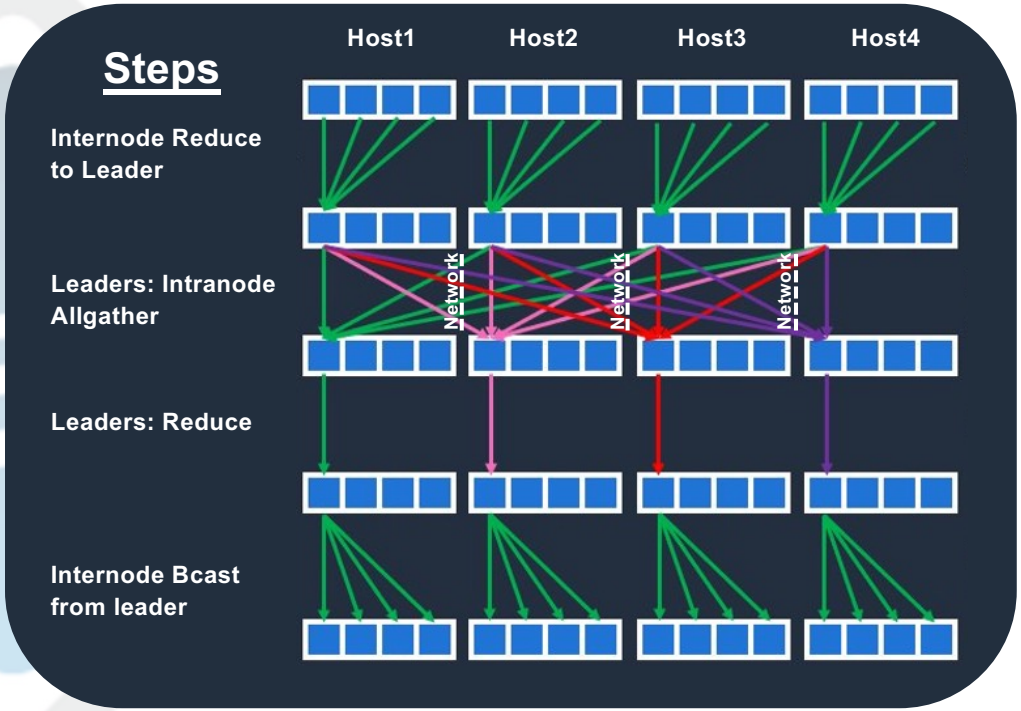
- Customers use AWS because the variety of compute offerings we provide.
- F1 Racing and Metro Weather run their CFD and weather forecasting workloads on AWS. Customers also run Life Sciences, Energy, and a variety of other workloads.
- At AWS we're optimizing the full software and hardware stack, including Graviton CPUs, EFA network, MPI runtimes, and applications themselves.

Optimizing Collectives

- We profiled a variety of applications across different customer domains
  - OpenFOAM, Ansys Fluent, GROMACS, LAMMPS, WRF, Palace, Code Saturne
- From the profile data we observed a few collectives which were frequently used and have benefited from additional optimizations:
  - MPI_Allreduce (Fluent, Code Saturne, OpenFOAM)
  - MPI_Alltoall (Palace, Code Saturne)
  - MPI_Alltoallv (Palace, LAMMPS, Code Saturne)

Reductions and Gathers

- Allreduce in coll/han was already implemented using a series of Reduce, Allgather, and Broadcast collectives as illustrated (since 2020)
- In April we added support for configurable “k” fanout in both Reduce, and Allgather, thus improving performance of all three collectives
- Using these tunings, we observe latencies to improve by 2x in microbenchmarks (8k vCPUs, small message sizes)
- OpenFOAM showed improvement of up to 35% in iterations / minute
- Code Saturne showed improvement of up to 50% in elapsed time

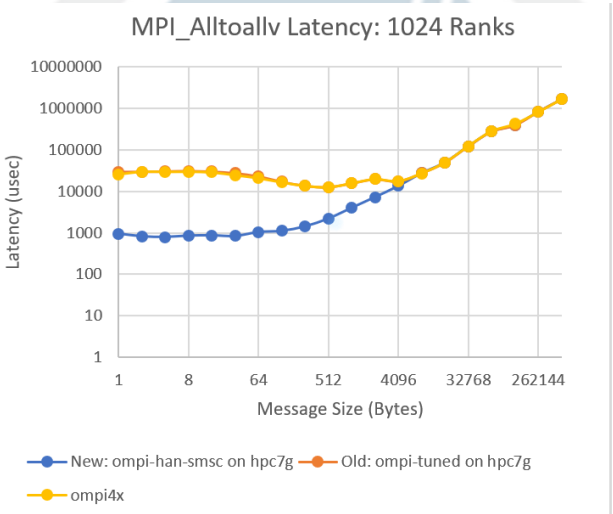


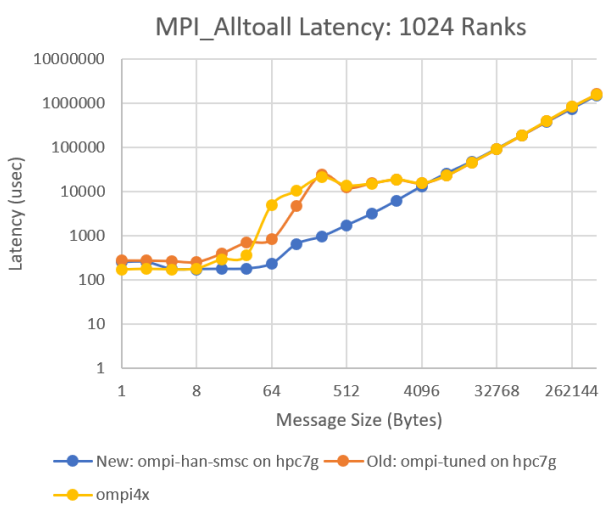
Full blog post: [AWS Blog: Open MPI Allreduce improvement]

Alltoall and Alltoallv

- Previously Alltoall was implemented without knowledge of topology, using either Bruck's method or a direct-send approach. Alltoallv was limited to only the direct-send.
- We implemented a novel algorithm using a hierarchical method and Shared Memory mapping

- Substantial improvements in microbenchmarks (20x or more), but difficult to realize in applications (imbalance, or application-driven data packing)



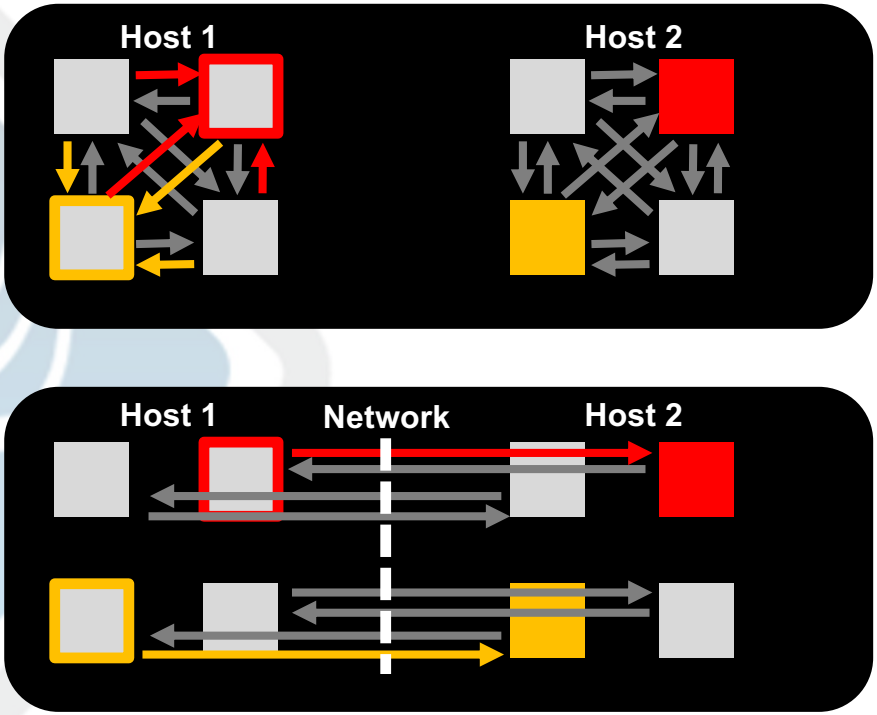


Alltoall and Alltoallv

Steps

- Exchange pointers, and mmap data from ranks on the same host
- Select a remote host, and assign a local “intermediary” for each rank on the remote
- memcpy data to appropriate intermediaries
- Send data from intermediaries to remotes
- Recv data from remote intermediaries
- Repeat for next host

Initial setup-cost of mapping host-local memory is recouped during repeated parallel access with minimal ordering and without shm bounce-buffer



Next Steps

- Try these for yourself!
  - We recommend hpc7g instances powered by Graviton3
- These improvements are available in mainline branch today
- They will ship with Open MPI 6.0.0 release
- We continue to seek out more optimizations!

- Getting Started with Graviton:
  - https://github.com/aws/aws-graviton-getting-started
- Instance Overviews
  - Hpc7g: https://aws.amazon.com/ec2/instance-types/hpc7g/
  - C8g: https://aws.amazon.com/ec2/instance-types/c8g/
- More Information on Annapurna Labs:
  - https://www.aboutamazon.com/news/aws/take-a-look-inside-the-lab-where-aws-makes-custom-chips
  - We are Hiring: 124 Open Postings as of Monday
    - https://www.amazon.jobs/content/en/teams/amazon-web-services/annapurna-labs
- HPC Customer success stories on Graviton and across AWS
  - https://aws.amazon.com/hpc/customers/

Open MPI Activities at DOE – ORNL/LANL

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

Los Alamos – Current Work

- Infrastructure for supporting
  - Big count (targeting 6.0.x releases)
  - MPI-4.2 ABI proposal (targeting later release)
- Fork of PRRTE to be used for the 6.0.x releases
- Spack support (review PRs, add new versions, etc.)
- Bug fixes (particularly for HPE SS11)

Fork of PRRTE

- Decision was made to use a fork of the PRRTE project for the 6.0.x release
- Focus on stability rather than features.
- Working with students at Univ. of Louisville on changes to make the PRRTE fork easier (transparent) to use for the Open MPI community

CASS & S4PST

CASS: Consortium for the Advancement of Scientific Software

- US DOE supported stewardship of current/future scientific computing software
- Support for community software tools/packages post Exascale Computing Project
- Foster collaboration across collection of software stewardship organizations (SSOs)
- 6 Currently funded core member SSO projects

- **S4PST: Stewardship for Programming Systems and Tools (Lead: K. Teranishi)**
  - Includes Open MPI & OpenPMIx tasks (Rep: T. Naughton)

https://s4pst.org

https://cass.community

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

Libfabric Updates for Slingshot 11

- HPE public release of CXI provider
- ORNL upstreamed LINKx (lnx) “linking” provider
  - Couple new patches coming (fixes from latest testing)
- See: https://github.com/ofiwg/libfabric

prov/lnx - https://github.com/ofiwg/libfabric/pull/10550
xpmem - https://github.com/ofiwg/libfabric/pull/10551

SS11+LINKx Libfabric Build Notes

Modules & Configuration

login05:$ module **unload cray-mpich libfabric craype-network-ofi**
login05:$ module **load craype-accel-amd-gfx90a rocm xpmem**
*login05:$ module **load json-c***
login05: $ **./configure **
**--prefix=$MY_PREFIX **
**--enable-cxi **
**--enable-xpmem=$(pkg-config --variable=prefix cray-xpmem) **
**--with-rocr=$ROCM_PATH **
CC=cc && \
make –j 8 && make install

Environment Settings

login05: $ **env | grep shm+cxi**
FI_LNX_PROV_LINKS=**shm+cxi**
OMPI_MCA_opal_common_ofi_provider_include=**shm+cxi:lnx**

login05: $ **fi_info | grep shm+cxi**
provider: **shm+cxi:lnx**
domain: **shm+cxi0:ofi_lnx_domain**

PreReqs

PreReq: libcxi
pkg-config --variable=prefix libcxi

RPMs providing header/library:
login05: $ **rpm -qf /usr/include/libcxi/libcxi.h**
cray-libcxi-devel-0.9-SSHOT2.2.1_20240604140029_39a4254e6b4a.x86_64
login05: $ **rpm -qf /usr/lib64/libcxi.so.1.5.0**
cray-libcxi-0.9-SSHOT2.2.1_20240604140029_39a4254e6b4a.x86_64

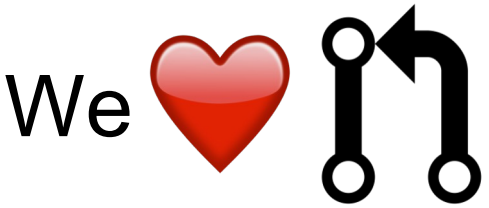
Getting Started on Frontier

```
module unload cray-mpich cray-pmi
module use ums ums024
module load openmpi
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!