Getting Started with Open MPI on NVIDIA Hardware

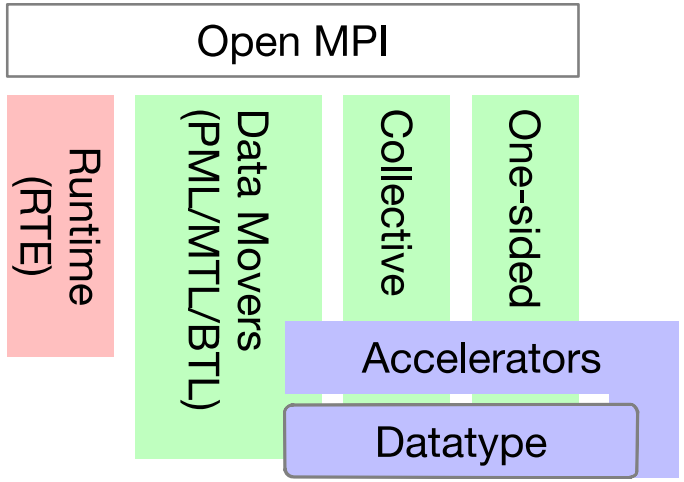
George Bosilca, Nvidia

Rich Graham, Nvidia

June 28, 2024

What is Open MPI ?

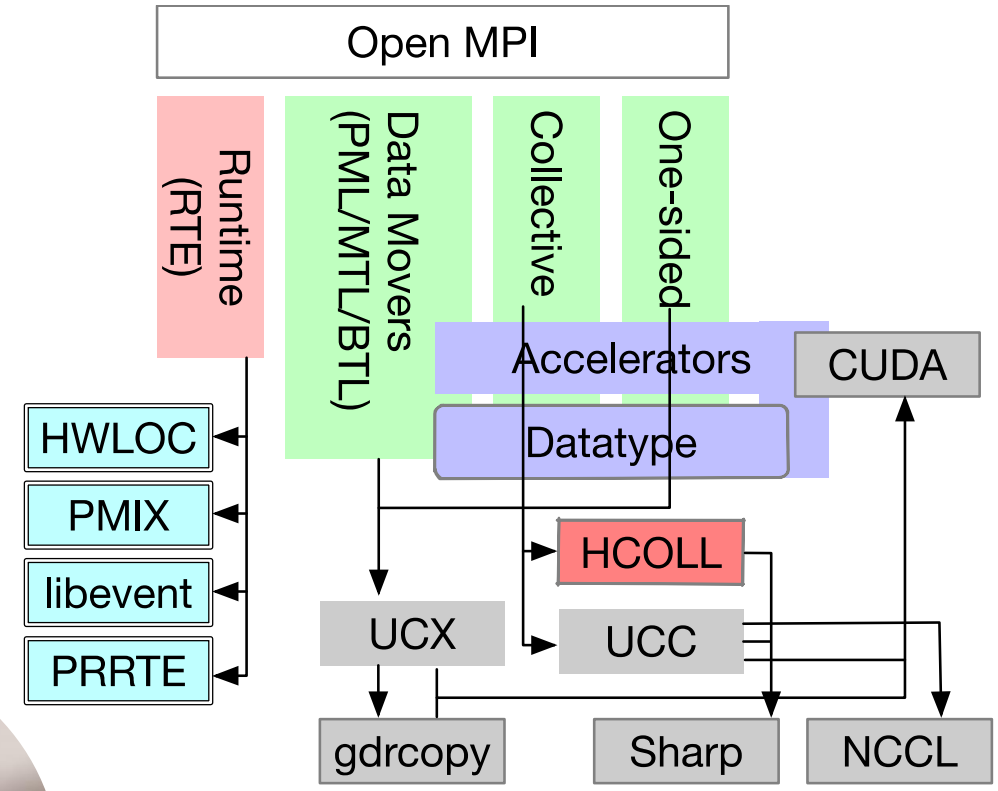
- Open MPI is a community-based open-source implementations of the MPI standard
  - Short list of features
    - Full MPI-4.1 standards conformance
    - Thread safety and concurrency
    - Dynamic process spawning
    - Network and process fault tolerance
    - Run-time instrumentation
    - Many job schedulers supported
    - Accelerators ready (CUDA/ROCM/ZE)
- Composed of dynamic components that can be configured in/out depending on the setup and user needs

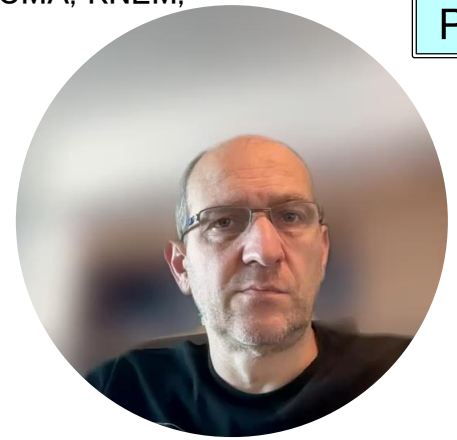


- Official NVIDIA Open MPI build can be found in the HPCX distribution
  - Currently rev 2.19.0 (as of May 5th) based on OMPI 4.1.x
  - Comes in multiple flavors (cuda, MT aka multi-threading, profiling, debug)

Software Stack Dependencies

- Prerequisites Nvidia software used in OMPI
  - CUDA (currently 12.5)
  - Sharp (Scalable Hierarchical Aggregation and Reduction Protocol 3.7.0)
  - gdrcopy (GPUDirect RDMA memory copy 2.4.1)
  - NCCL (NVIDIA Collective Communication Library)
  - HCOLL (supported in the 4.1.x series, capabilities will migrate UCC in future versions)
- Other software dependencies
  - HWLOC, PRRTE, PMIX, libevent can be used either from system packages or from the OMPI distribution (build as internal components).
  - Other dependencies exists (batch schedulers, IO, XPMEM, CMA, KNEM, GPUDirect RDMA driver with MLNX OFED…)





Software Dependency: UCX

- Unified Communication – X Framework (UCX) is an acceleration library, integrated into the Open MPI (PML or BTL layer) and to OpenSHMEM (SPML layer)
  - Supports Infiniband (RC, UC, DC, accelerated verbs)
  - Shared memory (CMA, XPMEM, KNEM)
  - RoCE, TCP, CUDA/gdrcopy
- Hint: `ucx_info -b` to see the configuration of an already installed UCX
- Minimum configure options for a high-performance build `--with-knem --with-xpmem=… --with-cuda=… --with-gdrcopy=…`
  - Optional: `--disable-logging --disable-debug --disable-assertions --disable-params-check`

- How to use in OMPI
  - As a PML `mpirun --mca pml ucx --mca osc ucx …`
  - As a BTL `mpirun --mca pml ob1 --mca btl uct,sm,self …`
  - For OpenSHMEM `mpirun --mca spml ucx --mca atomic ucx …`
- Usual UCX parameters (via environment variables): `UCX_NET_DEVICES (select a device), UCX_TLS (select a transport), UCX_[RNDV|ZCOPY|BCOPY|TM]_THRESH (threshold for different message protocol optimizations), UCX_UNIFIED_MODE (homogeneous environment), UCX_[RC|DC]_MLX5_TM_ENABLE (tag matching)`
  - Extract the default configuration parameters `ucx_info -c` or `ucx_info -d mlx5_0 -ut -e`

Software Dependency: UCC

- Unified Collective Communication (UCC)
  - Support highly efficient collective communication protocols, including some support for GPU and DPU
  - Has support for Nonblocking collective
  - Takes advantage of NVlink topology to select best algorithm for allgather(v) and reducescatter(v)
- Hint: use `ucc_info -b` to see the configuration of an already installed UCC
- Minimum configure options for a high-performance build `--with-sharp=… --with-rdmacm --with-cuda=… --with-nccl=…`
  - Specialize as needed: `--with-tls=cuda,nccl,self,sharp,shm,ucp,mlx5`
- How to use in OMPI
  - Not default in OMPI 4.1 or HPCX. Enable with `mpirun --mca coll_ucc_enable 1 …`
- Usual UCC parameters
  - `UCC_CLS=basic,hier` to enable Nvlink support
  - `UCC_EC_CUDA_USE_COOPERATIVE_LAUNCH=1` enable a single CUDA kernel for CUDA operations in UCC GPU collectives
  - `UCC_TL_MLX5_MCAST_ENABLE=1` enable support for MCAST in broadcast (might need additional parameters to disable sharp and nccl)
  - UCC_TL_MLX5_TUNE

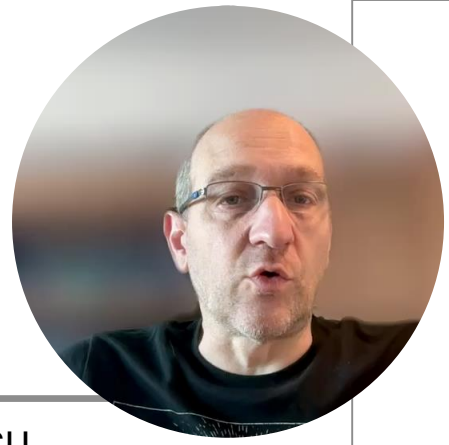
Building OMPI for NVIDIA Hardware

- Hint: use `ompi_info` to see the configuration of an already installed OMPI (search for "Configure command line")
- Minimum configure options for a high-performance build `--with-hcoll=… --with-ucx --with-cuda=… --with-ucc=… --with-xpmem --with-cma --with-knem --with-hwloc=internal`
  - Use internal dependencies: `--with-libevent=internal --with-prrte=internal --with-pmix=internal`
  - Specialize your batch scheduler as needed: `--with-[slurm | sge | pals | moab | lsf | pbs]`
  - Release builds `--with-platform=contrib/platform/mellanox/optimized`
- How to use OMPI
  - `mpirun …`
- To get the best from NVIDIA hardware
  - `mpirun --mca pml ucx --mca coll_ucc_enable 1 …`
  - Or add them into the MCA parameter file at ${HOME}/.openmpi/mca-params.conf as

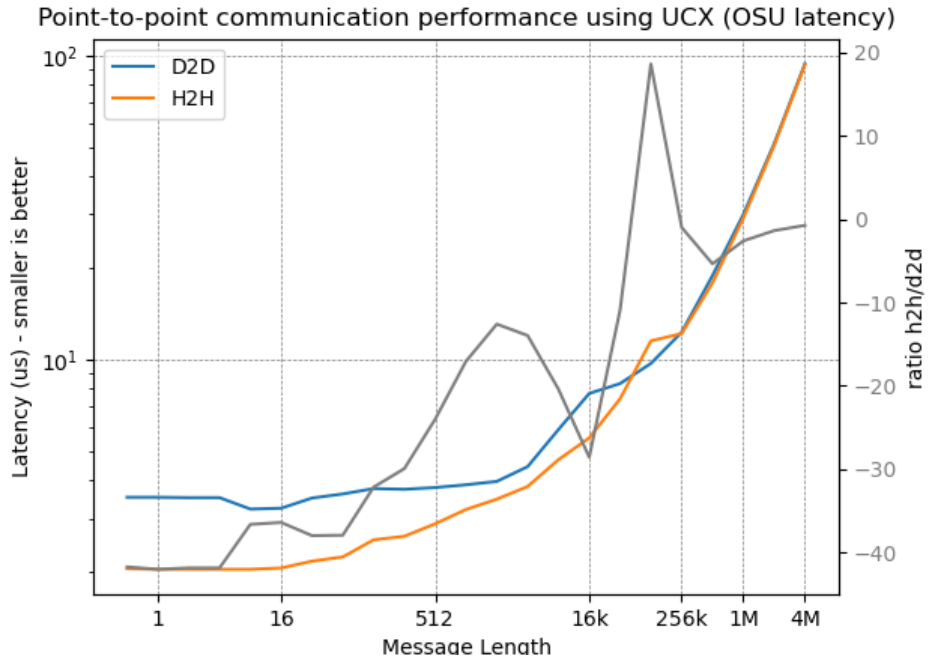
    ```
    pml=ucx
    coll_ucc_enable=1
    ```
  - The stack is optimized for one process per GPU
- Learn more at ReadTheDocs or directly at OMPI Docs

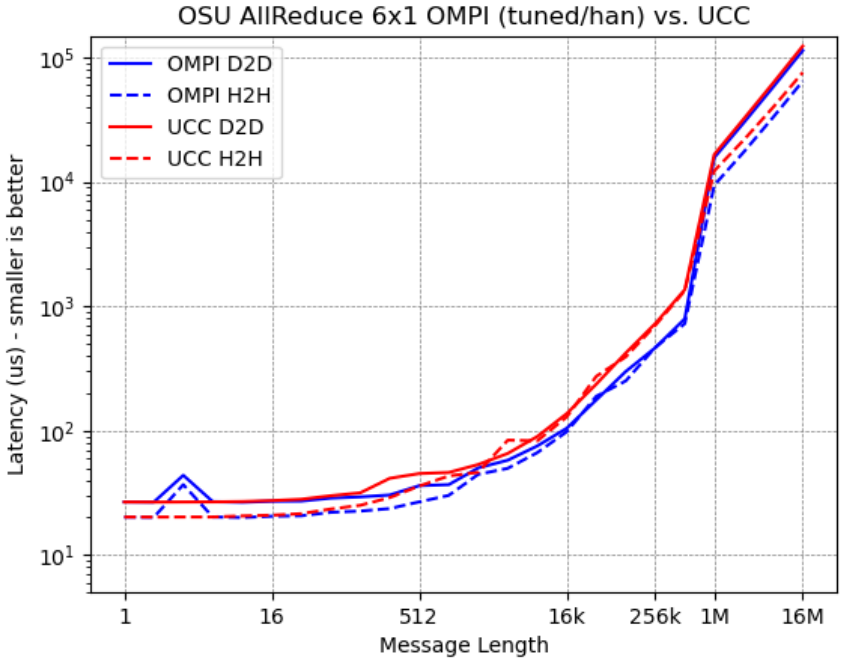
Quick Performance Validation

Grace + GH200



- Use a benchmark depending on what you intend to test (OSU, NetPIPE, …) or use directly your CUDA-enabled application
  - Point-to-point performance
    - CUDA enabled OSU compares the communication performance between GPU to GPU and host to host memory (GH200).
      - D2D device-to-device, H2H host-to-host
    - `mpirun --mca pml ucx -np 2 …`
    - Using device-bound memory has an impact on the latency for small messages, but converges to the same bandwidth as host memory starting from 256k
  - Collective performance
    - CUDA enabled OSU allreduce
    - OMPI: `mpirun --mca pml ucx --map-by ppr:1:node -np 6 …`
    - UCC: `mpirun --mca pml ucx --map-by ppr:1:node --mca coll_ucc_enable 1 -np 6…`
  - Barely any performance difference !!!
    - Let's investigate

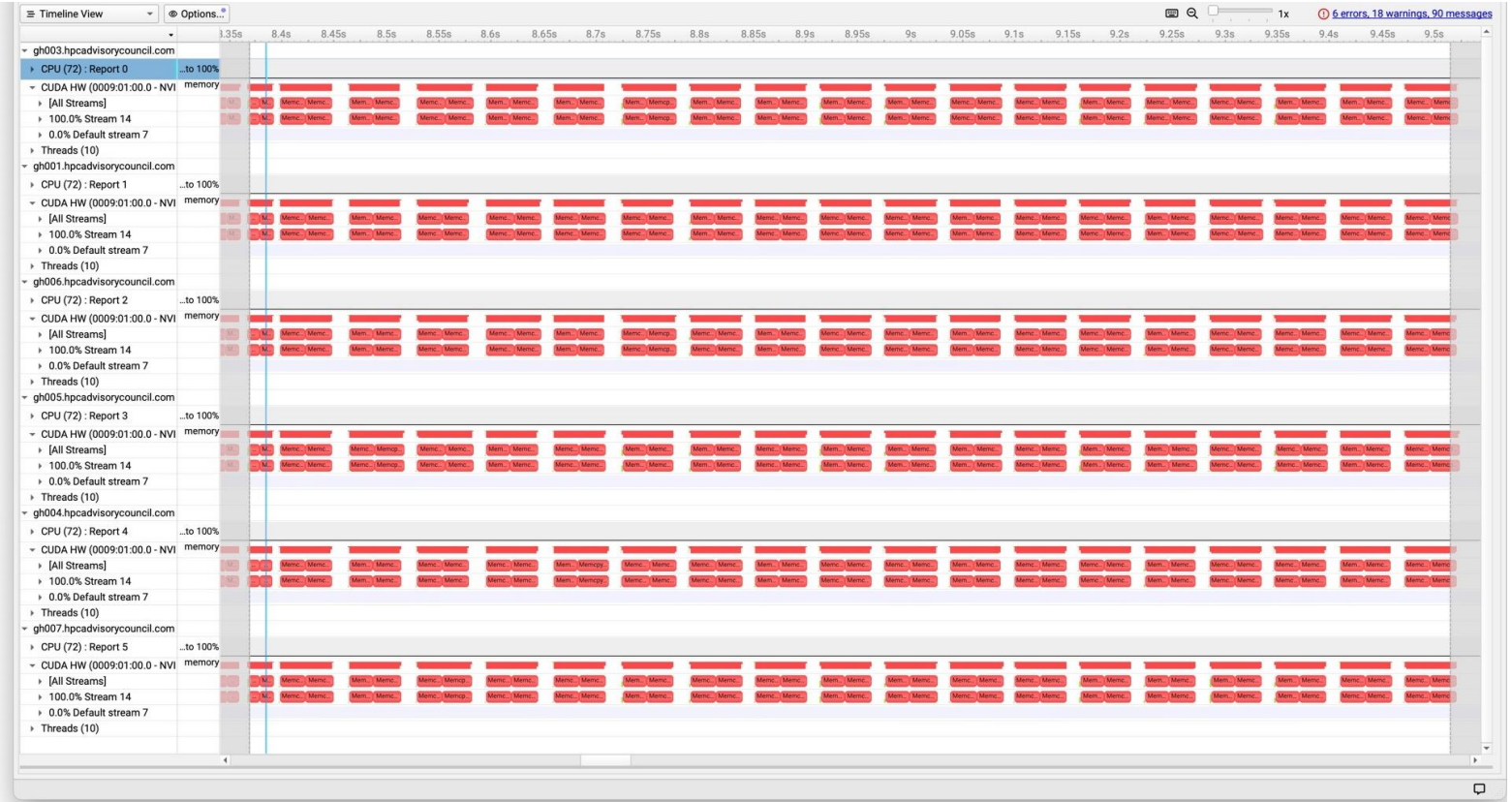




Understanding the Performance Issues

Using nsight to understand the performance

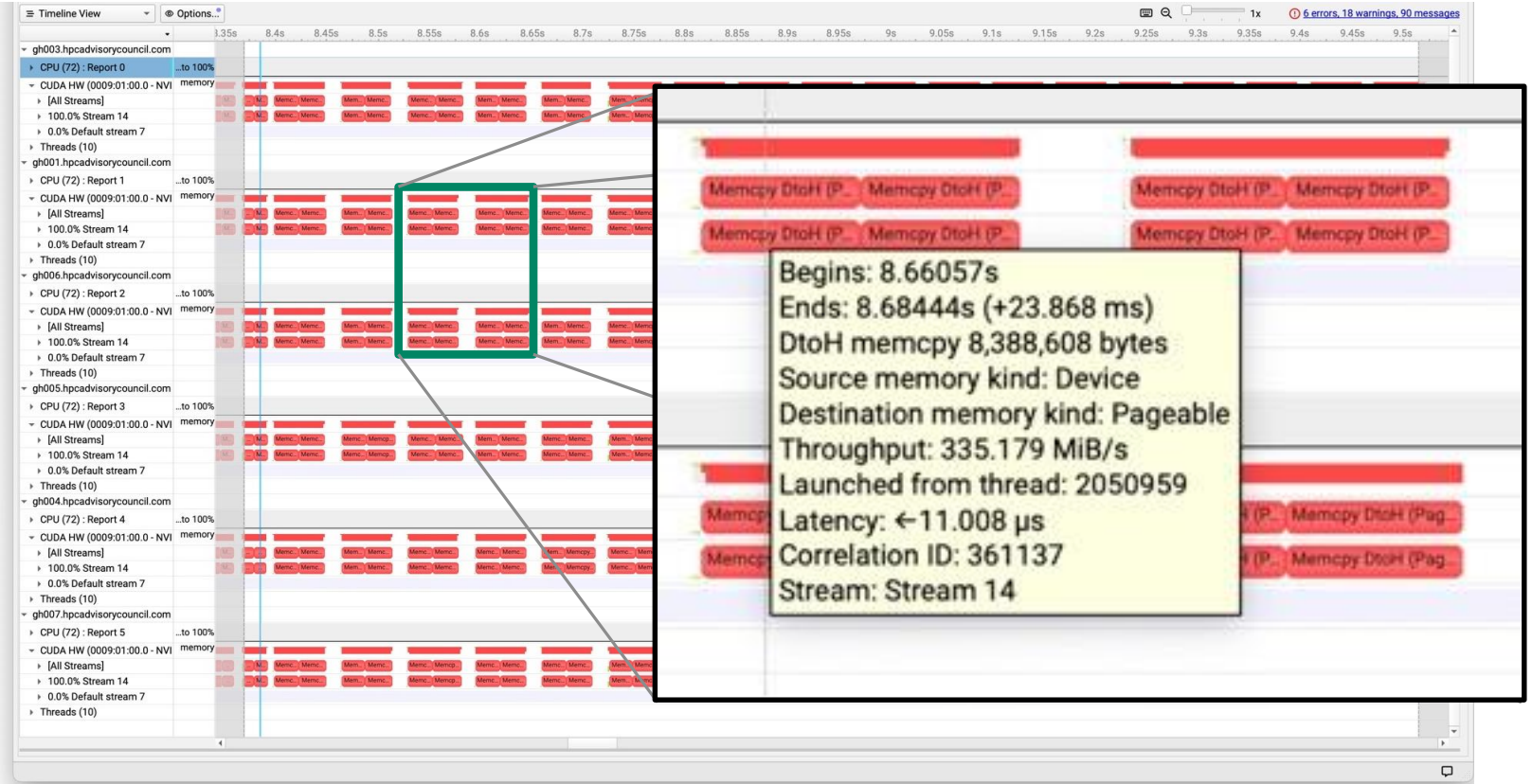
- NVIDIA Nsight enable developers to profile and optimize software that utilizes the latest accelerated computing hardware
  - Here we are mostly interested in understanding the interactions between the CUDA enabled MPI and the CUDA hardware
- mpirun --mca pml ucx --map-by ppr:1:node --mca coll_ucc_enable 1 -np 6 nsys profile ⋯



Understanding the Performance Issues

Use nsight to understand the performance

- NVIDIA Nsight enable developers to profile and optimize software that utilizes the latest accelerated computing hardware
  - Here we are mostly interested in understanding the interactions between the CUDA enabled MPI and the CUDA hardware
- mpirun --mca pml ucx --map-by ppr:1:node --mca coll_ucc_enable 1 -np 6 nsys profile …

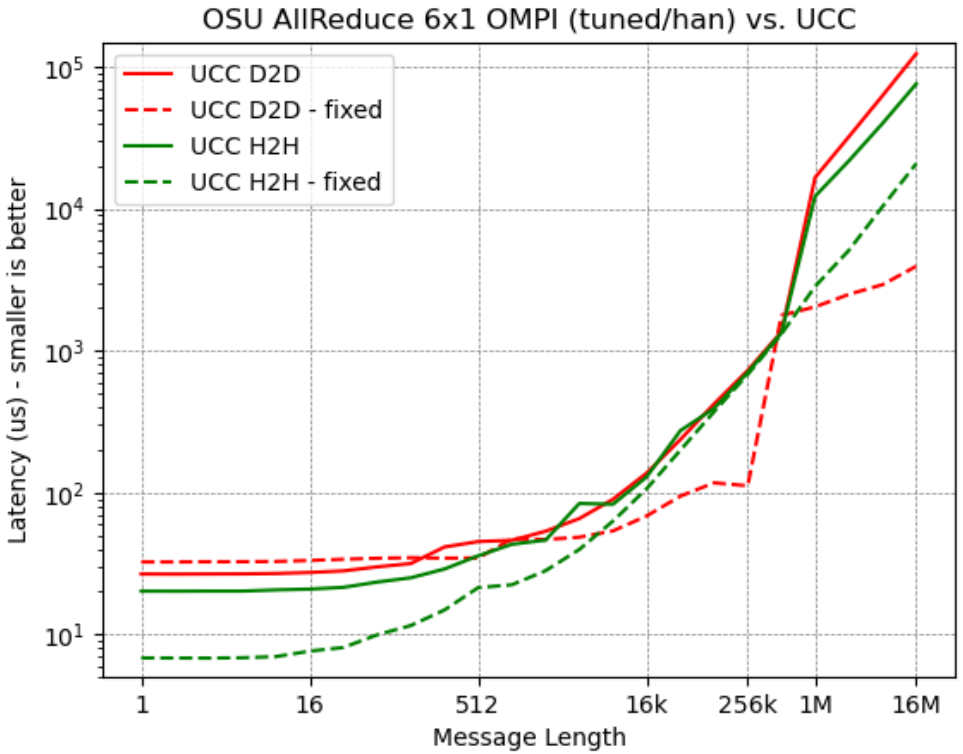


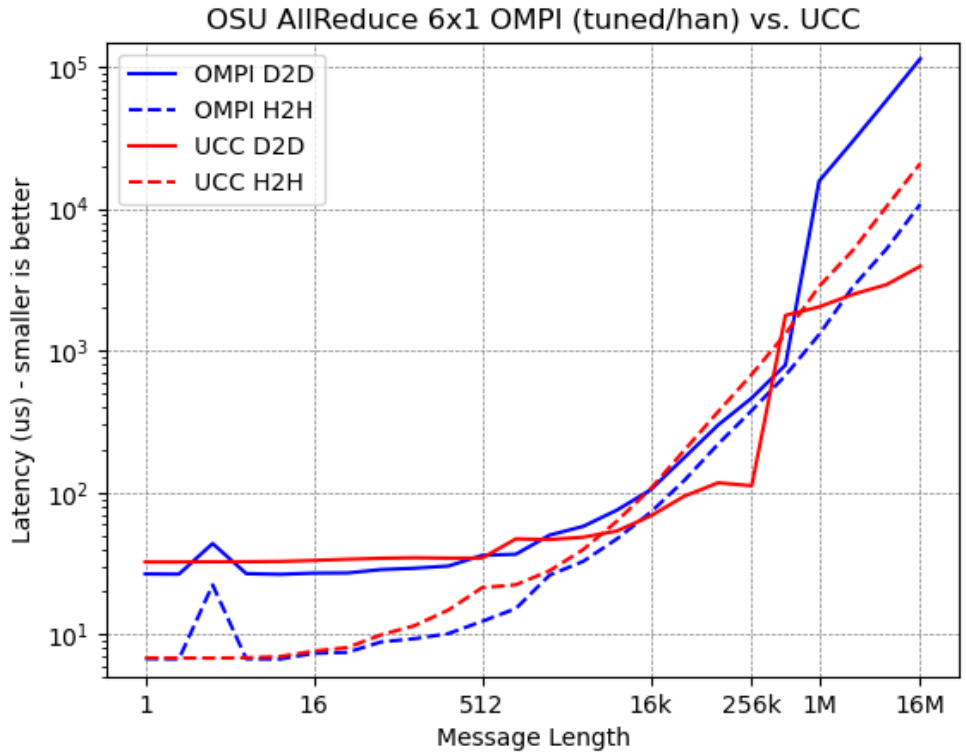
- There are two DtoH before each operation signaling the data is first moved onto the host and the reduction is executed on the host
  - There is no need for two DtoH because MPI_Allreduce only has one single input and one single output buffer
  - And there is no need DtoH if the collective module already supports CUDA memory (such as in the case of UCC)
- These identify two issues in OMPI code that are now fixed in main and will make their way into OMPI 5.0.4

Performance Validation

Grace + GH200

- Collective performance
  - CUDA enabled OSU allreduce
  - OMPI: `mpirun --mca pml ucx --map-by ppr:1:node -np 6 ⋯`
  - UCC: `mpirun --mca pml ucx --map-by ppr:1:node --mca coll_ucc_enable 1 --mca coll ^accelerator -np 6⋯`
- Full list of tuning parameters is difficult to explore, but the community strive to define sane defaults for most cases. For everything else `ompi_info --params coll all -l 9` and `ucc_info ⋯` are your first helpers





Conclusion

- List of known “features” that are to be fixed
- Read the Docs !
- Ask the community !
  - Mailing list (users, devel)
  - Github Issues
- Join the community github.com
  - We need help with addressing open issues, performance validations as well as improving the documentation (both OMPI and MPI).

- NVIDIA contributors to OMPI are active on all these platforms.


NVIDIA

Questions and Answers

- Q1) Which version of Open MPI, UCX, and UCC are used in the presentation?
  Are these combinations recommended now?
  - A1) Latest stable releases for all projects. 5.0.4 for OMPI, 17.1 for UCX and 1.3.0 for UCC.
    However, the HEADS of their git repo are regularly tests, so all recent combinations of these libraries should work.

- Q2) Does the open-source version of Open MPI support the nsys profile option?
  - A2) Yes. It is not as powerful as other more MPI-specialized tools, but it is the best when one wants to see the interactions with the GPU.

- Q3) Open MPI 5.0.x series version is also distributed. How do you think which series of Open MPI is recommended to use, version 4.1.x or version 5.0x?
  - A3) For any new install, I would not install the 4.1 anymore, the official stable version is now the 5.0.
    If one looks for a smooth update path, installing a newer 4.1 could make sense, but even then I would strongly suggest to update to the 5.x as soon as possible.