# MPI implementation optimization for Slingshot network

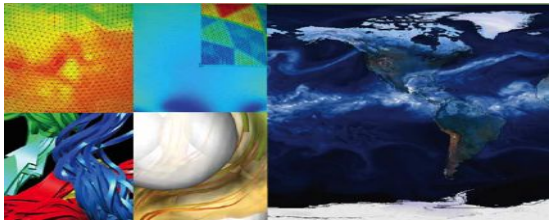
Rahulkumar Gayatri, Afton Geil, Neil Mehta, Adam Lavely and Brandon Cook

CUG 2025

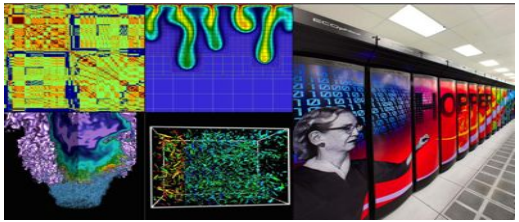
# NERSC: Mission HPC for DOE Office of Science Research

U.S. DEPARTMENT OF ENERGY | Office of Science

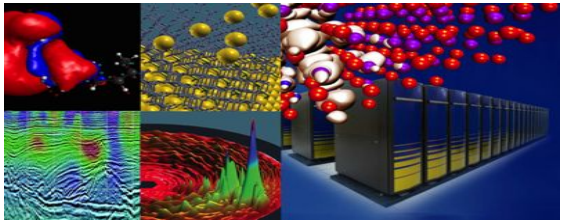
Largest funder of physical science research in the U.S.



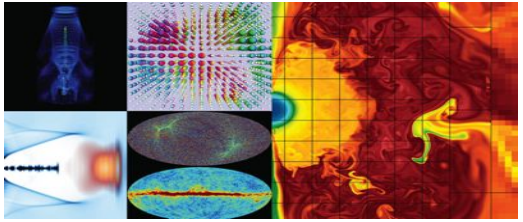
Biological and Environmental Research



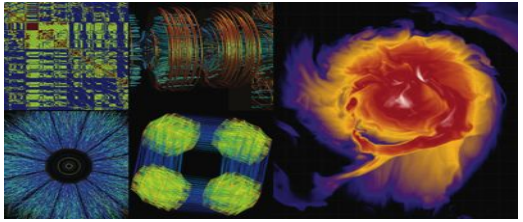
Computing



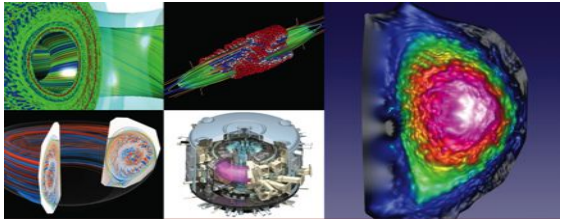
Basic Energy Sciences



High Energy Physics

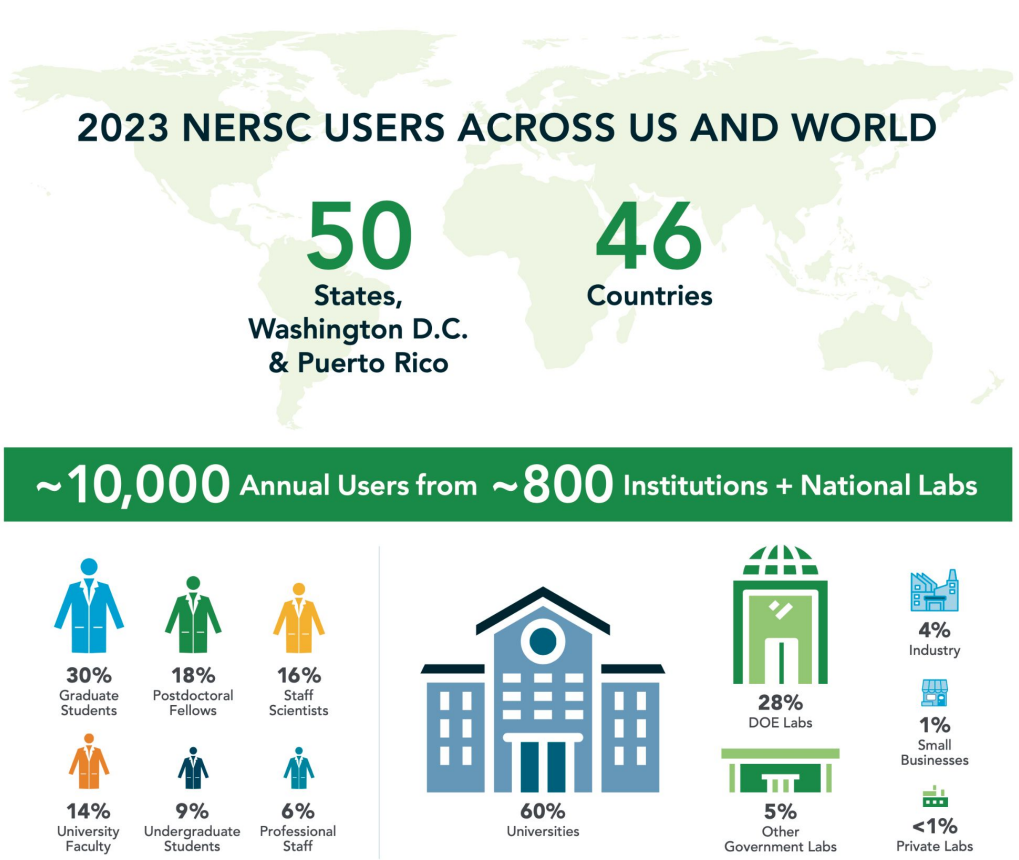


Nuclear Physics

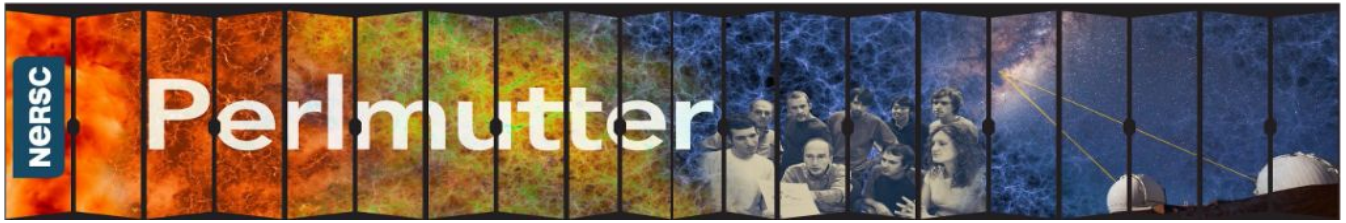


Fusion Energy, Plasma Physics

# NERSC has a very broad user base

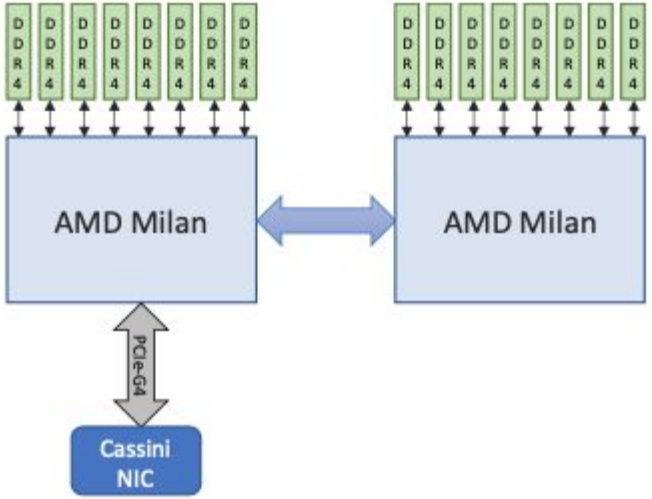


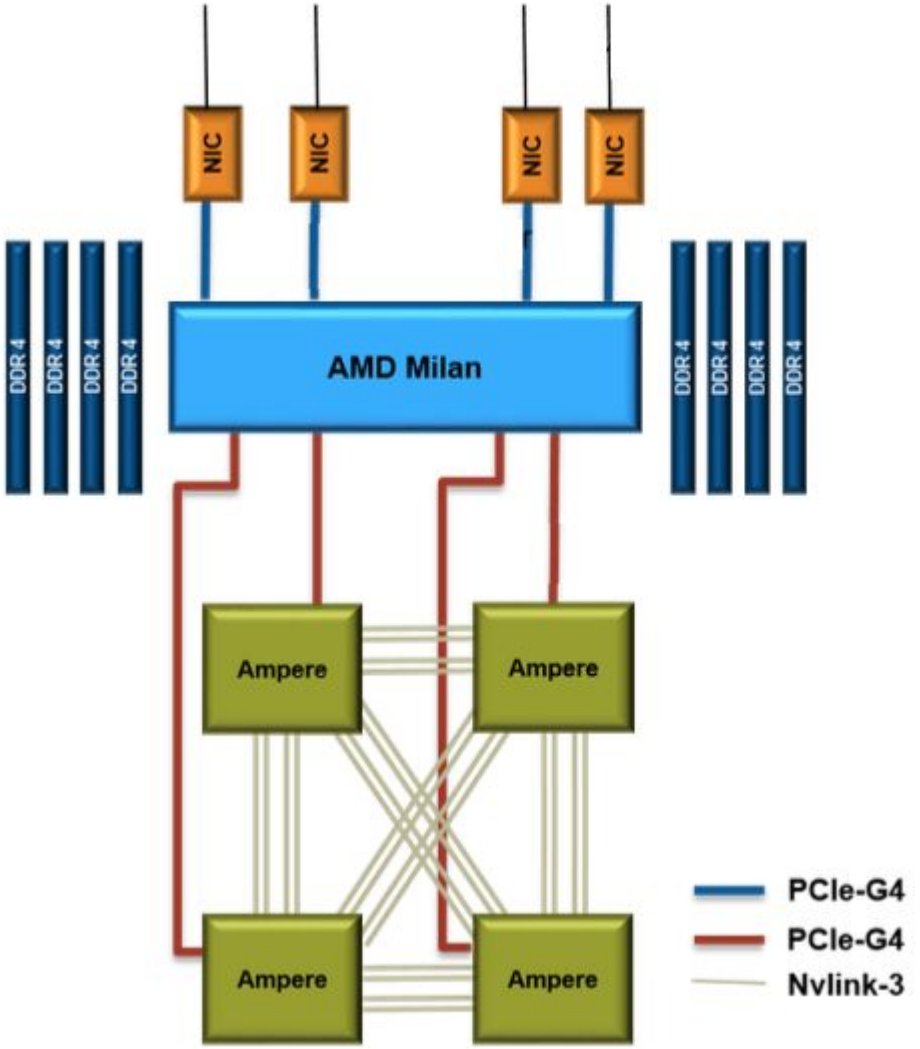

CDC 6600 at LLNL 1974




- CPU Nodes: 3072 nodes with 2x AMD Milan
- GPU Nodes: 1536 Milan + 4x NVIDIA A100
  - 4x HPE Cassini NICs/node
  - 4x 3rd gen NVLink connections per pair of GPUs

# Perlmutter node architecture



# Perlmutter user software

- CPE 24.07
- Cray-mpich 8.1.30
- Libfabric 1.20.1
- XPMEM 2.9.7

# Motivations for 3rd party MPI libraries

- New features & standards MPI 4+
- Containers & portability
- Source access and debugging
- Application specific requirements
- Competition is good

# OSU Benchmarks 7.5.1

**Point-to-point**

**Single root**

- scatter, gather, bcast

**Multi root**

- alltoall

**Multi root with computation**

- allreduce

# Affinity setup

Process to GPU affinity

- `--gpu-bind none` to avoid cgroups interference with IPC
- `export CUDA_VISIBLE_DEVICES=$( SLURM_LOCALID % 4 ))`

Process to NIC affinity

- `MPICH_OFI_NIC_POLICY = BLOCK` (default)
- OpenMPI defaults

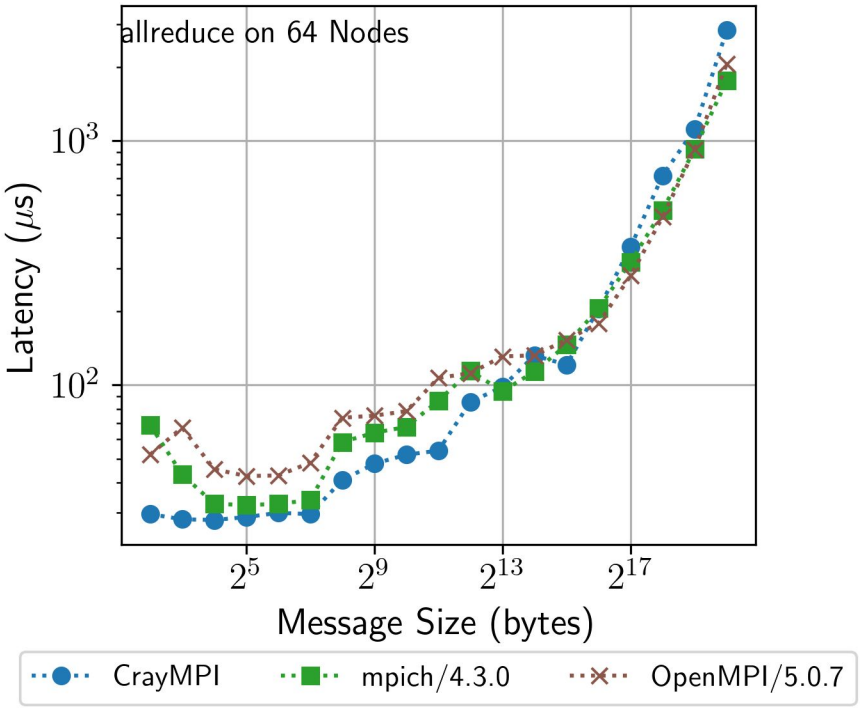
# Benchmark setup and parameters

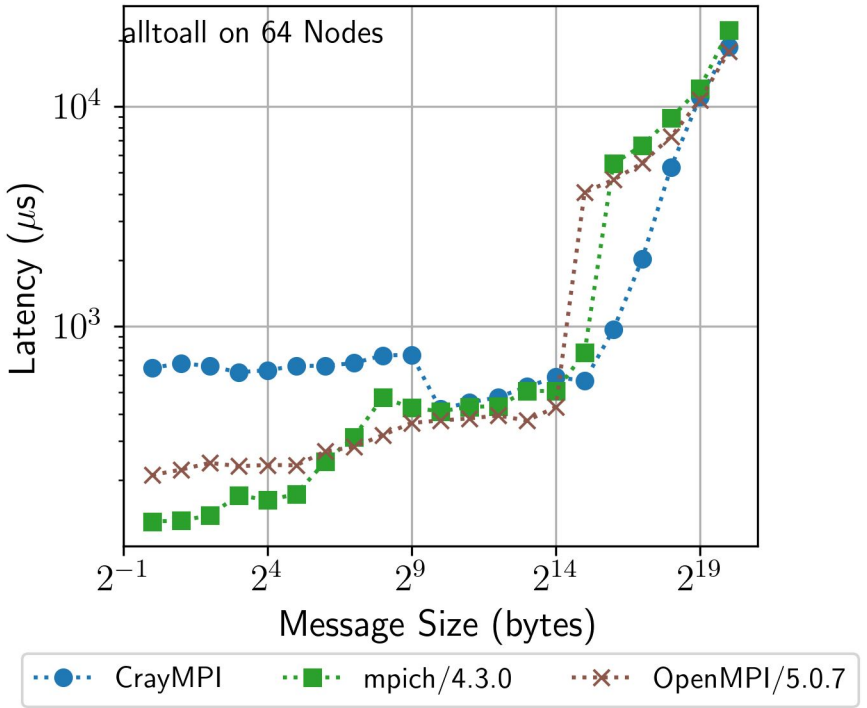
- `cudaMalloc` buffers
- **1**,**2**,...**64**,... nodes
  - **1 & 2 nodes** shown for point-to-point
  - **64** shown for collectives
- `--ntasks-per-node=4`
  - most typical user setup for GPU applications
- MPI datatype = `MPI_CHAR`
  - * only computation is the allreduce

# Methodology

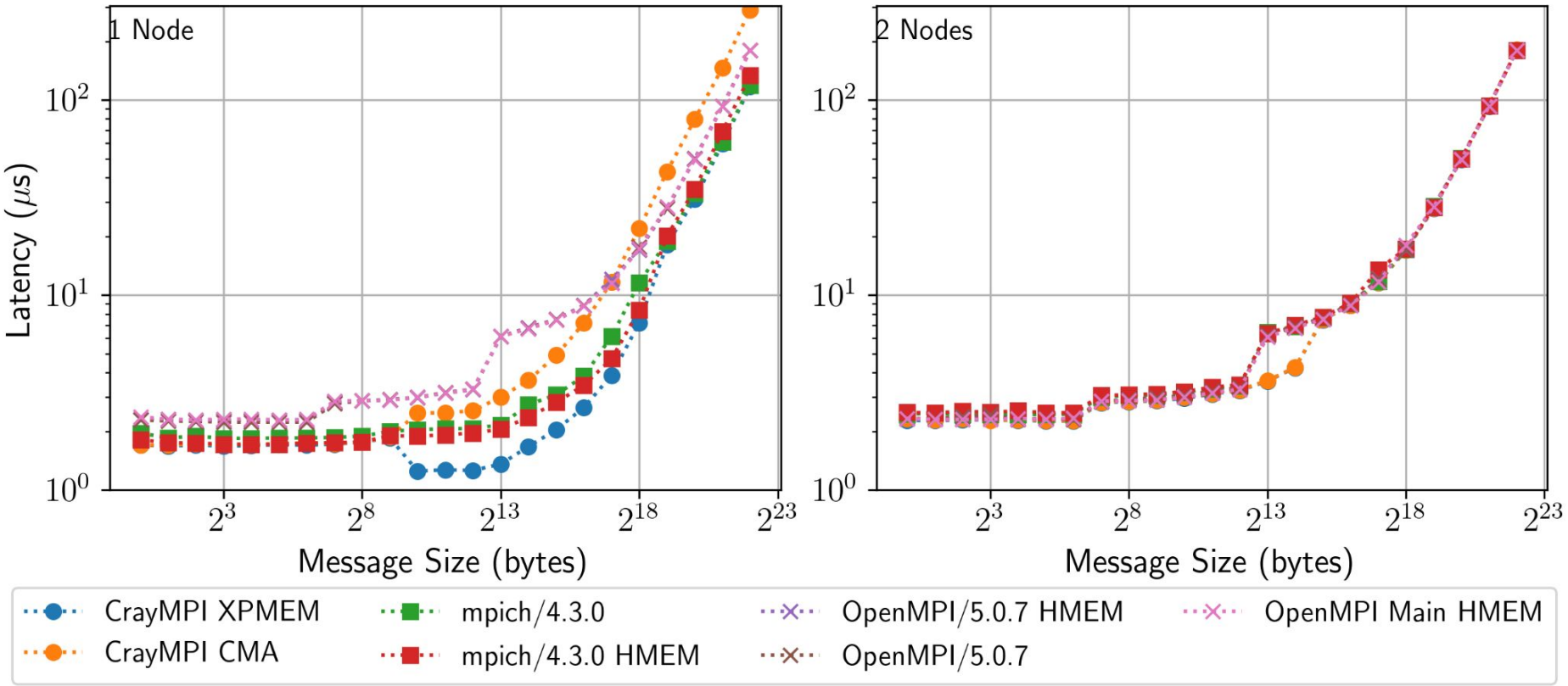
- Compare average latencies/bandwidths for each of the benchmarks using Cray-MPI, MPICH, and Open MPI
- Small messages results use average of 1000 runs after 10 warm-up runs
- For messages > 8192 B, we use average of 100 runs after 10 warm-up runs
- While averages can obscure variation between runs, it is acceptable for a high-level comparison between different MPI implementations

# Broadly similar performance on CPUs

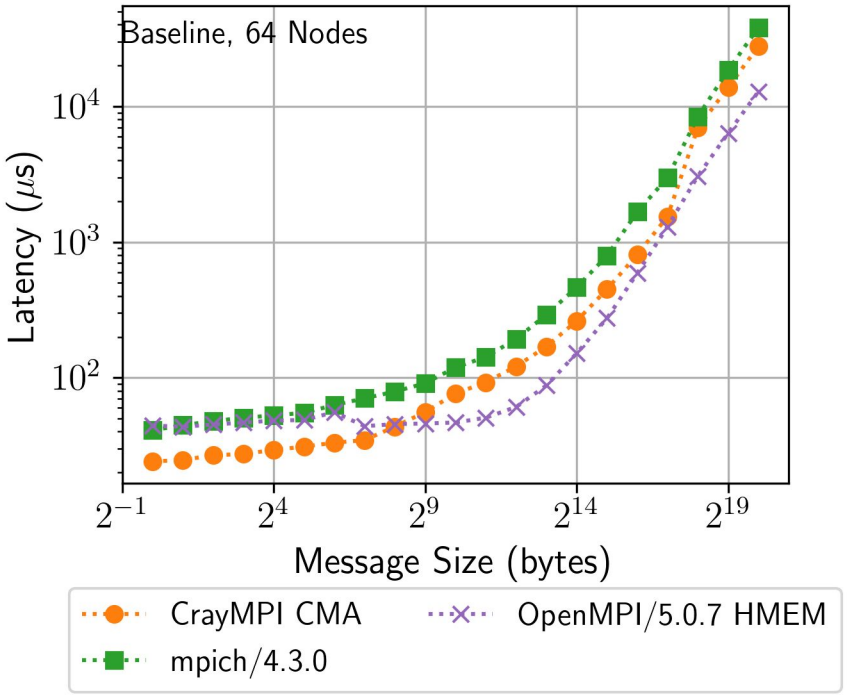


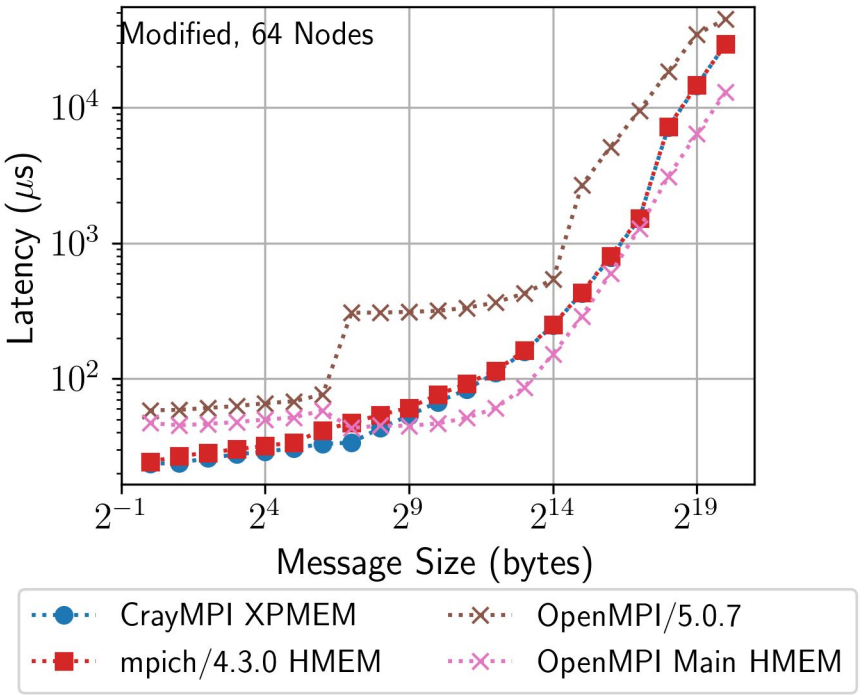



# GPU Point-to-point



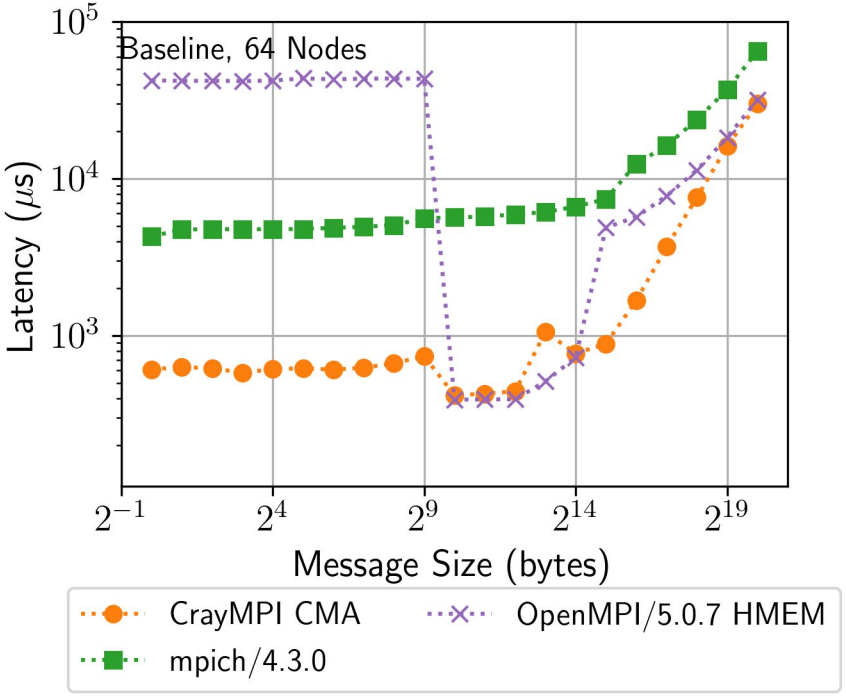
# GPU MPI_Scatter - mpich RDMA off by default

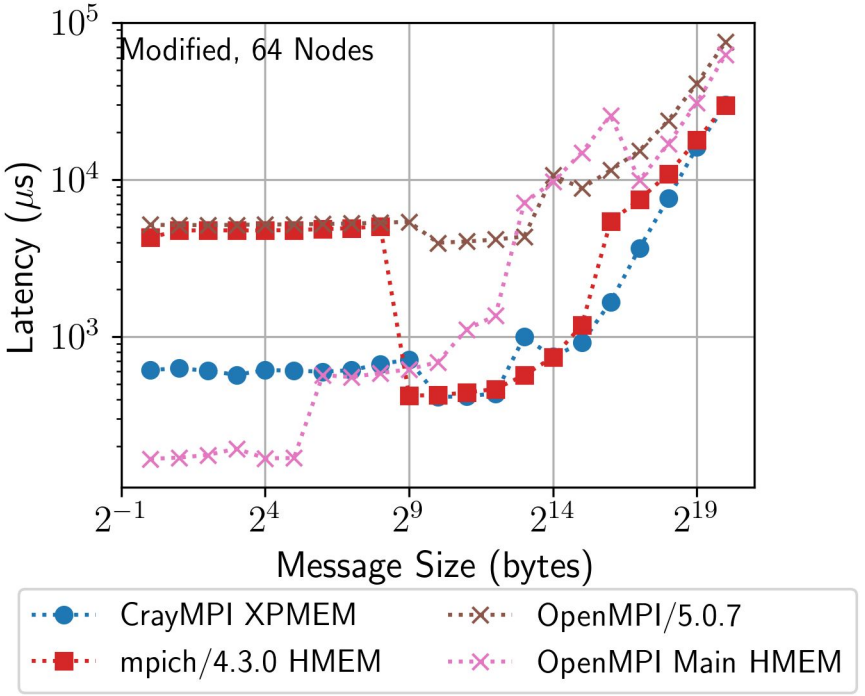



`MPIR_CVAR_CH4_OFI_ENABLE_HMEM`

# MPI_Alltoall



# Software regressions and fixes

MPICH

- Still some work to do with small immediate/ eager messages (issue #7375 on github for mpich)

OpenMPI Performance was highly variable

- Turning off RDMA makes variability go away by making everything slow!
- **Greatly improved** since da14c12 on the main branch with new implementations of alltoall, allgather, and several other collectives

# Aside - git bisect to find the commit

```bash
#!/bin/bash
set -euxo pipefail

git clone --recursive https://github.com/open-mpi/ompi.git

configure_build_install() {
  local commit_sha="$1"
  local install_prefix="${INSTALL_BASE}/${commit_sha}"
  ./configure CC=gcc CXX=g++ --other-options
  make && make install
}

run_mpi_test() {
  # Run a test and see if this is a good commit.

  # Check for failure
  if grep -qi "failure" "${benchmark_log}"; then
    git -C "${SRC_DIR}" bisect good
    return 1
  else
    git -C "${SRC_DIR}" bisect bad
    return 0
  fi
}

# Build and test the latest commit as initial reference
initial_commit_sha=$(git rev-parse --short HEAD)
configure_build_install "${initial_commit_sha}"
run_mpi_test "${initial_commit_sha}" || true  # continue regardless of
result

# Start git bisect
cd "${SRC_DIR}"
git bisect start
git bisect good "${initial_commit_sha}"
git bisect bad v5.0.7

# Automated bisect loop
while true; do
  current_commit_sha=$(git rev-parse --short HEAD)
  configure_build_install "${current_commit_sha}"
  bisect_output=run_mpi_test
  if $bisect_output; then
    echo "Continuing bisect: GOOD commit found."
  else
    echo "Continuing bisect: BAD commit found."
  fi

  # Check if bisect has finished
  if echo "${bisect_output}" | grep -q "is the first good commit"; then
    echo "Bisect complete!"
    git bisect reset
    echo "${bisect_output}"
    break
  fi
done
```

Configure build OpenMPI

Test the MPI build

Start bisect
main commit - good
v5.0.7 - bad

Continue until we find the first good commit

# Aside - git bisect to find the commit

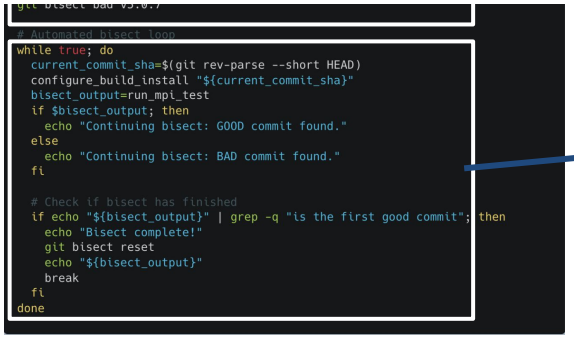

Configure build OpenMPI

The bisect script pointed to a commit which updated the alltoall, allgather and a few other implementations that are the reason for this performance boost in alltoall.

[da14c12]

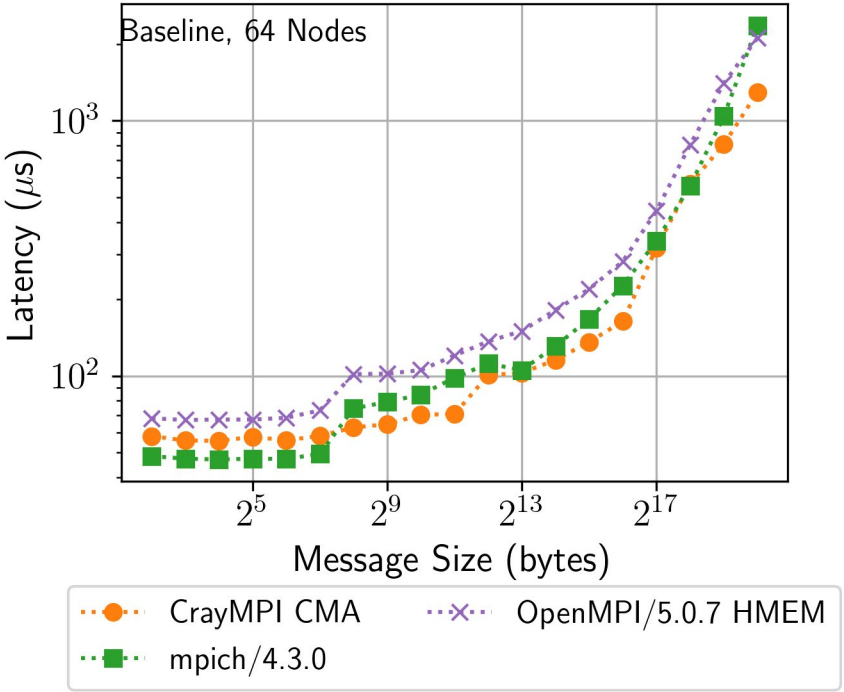
the MPI build

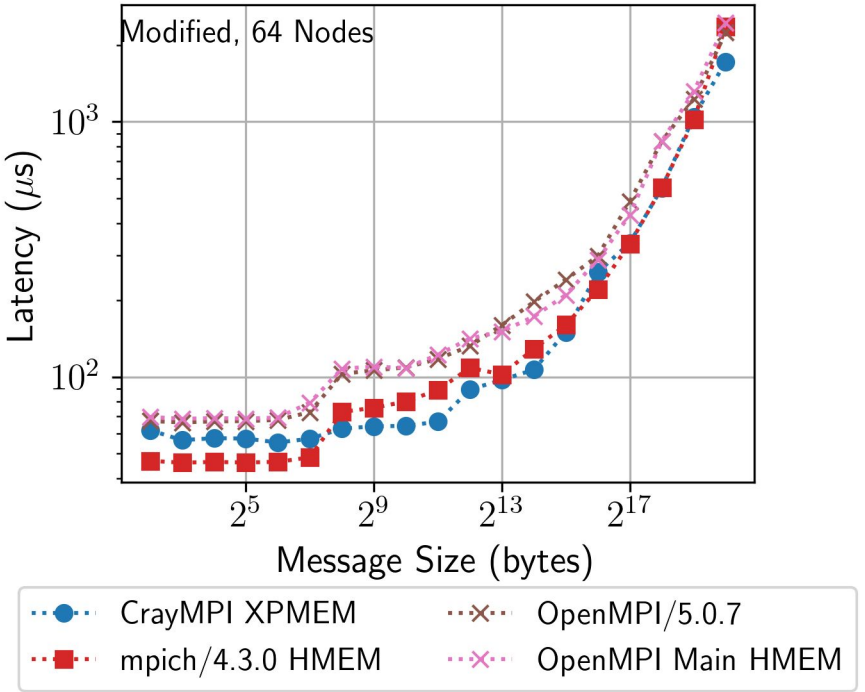
bisect
commit - good
7 - bad




Continue until we find the first good commit

# GPU MPI_Allreduce - now with computation



# turn off OpenMPI autotuned collectives

In addition to enabling RDMA and moving to main..

HAN (Hierarchical Autotuned collectives for Networks) does not appear to take GPU buffers into account leading to poor performance

# Key findings and Recommendations

- Verify your defaults!
- Run benchmarks at scale (not just 2 nodes)
  - run a wide variety of message sizes
- check variability
- GPU RDMA is good
- Current released OpenMPI has issues
  - HAN doesn't seem to know about GPUs
- git bisect is cool
- MPICH and OpenMPI can be close and *sometimes better* than Cray MPI

# Acknowledgements


- Howard Pritchard for help with OpenMPI questions
- Hui Zhou for help with MPICH questions

**This research used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility**

# Questions?
# Comments?