Open MPI for HPE Cray EX Systems

Howard Pritchard[1], **Thomas Naughton[2]**, Amir Shehata[2] & David Bernholdt[2]
Los Alamos National Laboratory[2], Oak Ridge National Laboratory[2]

Cray User Group Meeting
Helsinki, Finland
May 9, 2023

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725

Introduction

- Goal: Provide a performant alternative MPI
  - Helpful to support different user needs
  - Aid in diagnosing application issues
- Work by ORNL & LANL to port and optimize Open MPI for HPE Cray EX systems
  - *Frontier* @ OLCF
  - *Aurora* @ ALCF
- Highlight key pieces involved in development & testing
  - Changes to Open MPI, Libfabric and OpenPMIx
  - Brief snapshot of current status

ECP OMPI-X Project

Open MPI Overview

- Developed & maintained by collaborators from
  - Academia, Industry and National Laboratories
- Open-source implementation of MPI-3 standard
- Supports resource manager interoperability via **OpenPMIx**
- Supports variety of network fabrics via UCX & **OFI libfabrics**

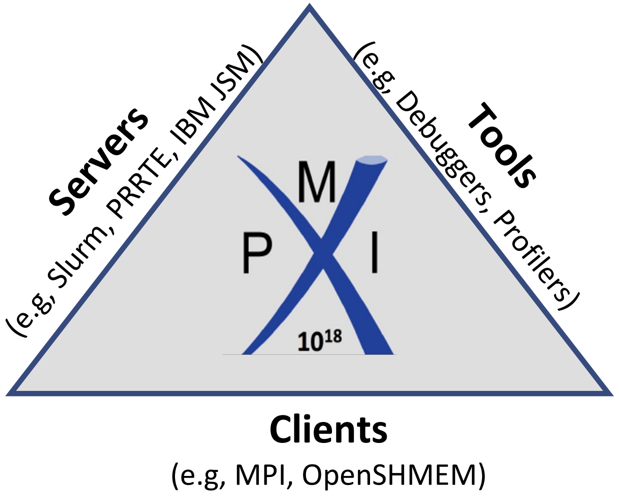
https://Open-MPI.org

Open MPI Project

OpenPMIx & PRRTE

- OpenPMIx: A feature complete implementation of PMIx Standard
  - Provides libraries and programming models portable and well-defined access to commonly available process management services
  - Implemented as C library for connecting PMIx-enabled clients (like Open MPI) with PMIx-enabled Tools (like debuggers) and PMIx-enabled Servers (like PRRTE, SLURM, IBM JSM)
- PRRTE: **P**MIx **R**eference **R**un**T**ime **E**nvironment
  - Portable and feature-rich runtime environment
  - Offers PMIx support even if host environment is not PMIx-enabled
- Open MPI relationship
  - Evolved from Open MPI's ORTE into stand-alone project
  - Next stable release of Open MPI requires PMIx-enabled server
  - Included as 3rd party packages in Open MPI tarballs

https://OpenPMIx.org



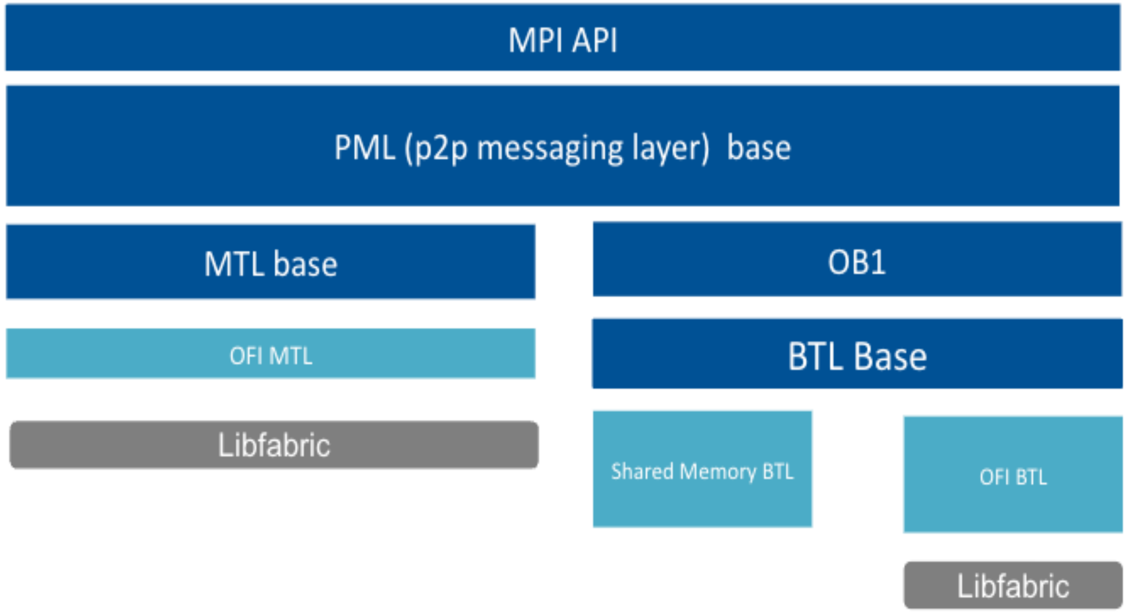
Source: https://PMIx.org

Runtime Details for Slingshot 11 Systems

- Slingshot with VNI enforcement enabled
  - VNI key allows clients to access fabric
  - Must launch PRRTE daemons with system resource manager for VNI key
    - Example: On SLURM based Frontier, use *srun* to start *prted*'s
- Enhancements to the job launch system for *Aurora*
  - PALS launcher support added to PRRTE frameworks
    - PLM: Process Launch Mechanism
    - ESS: Environment Specific Services
  - *Note: On Aurora, not plan to support direct launch of Open MPI because aprun lacks PMIx interface for launch mechanism.*

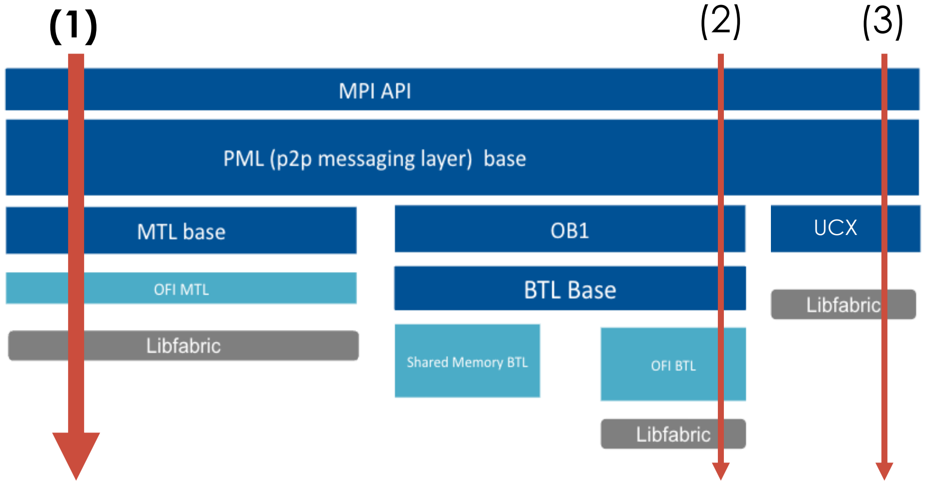
Open MPI Background

- Uses Modular Component Architecture (MCA)
  - Frameworks provide abstract interfaces
  - Frameworks have one or more components
  - Multiple ways to assemble/configure via MCA parameters
- Frameworks of interest
  - PML: Point-to-Point Messaging Layer
  - BTL: Byte Transport Layer
    - *Allows <u>multiple active</u> components*
  - MTL: Message Transport Layer
    - *Allows <u>1 active</u> component*



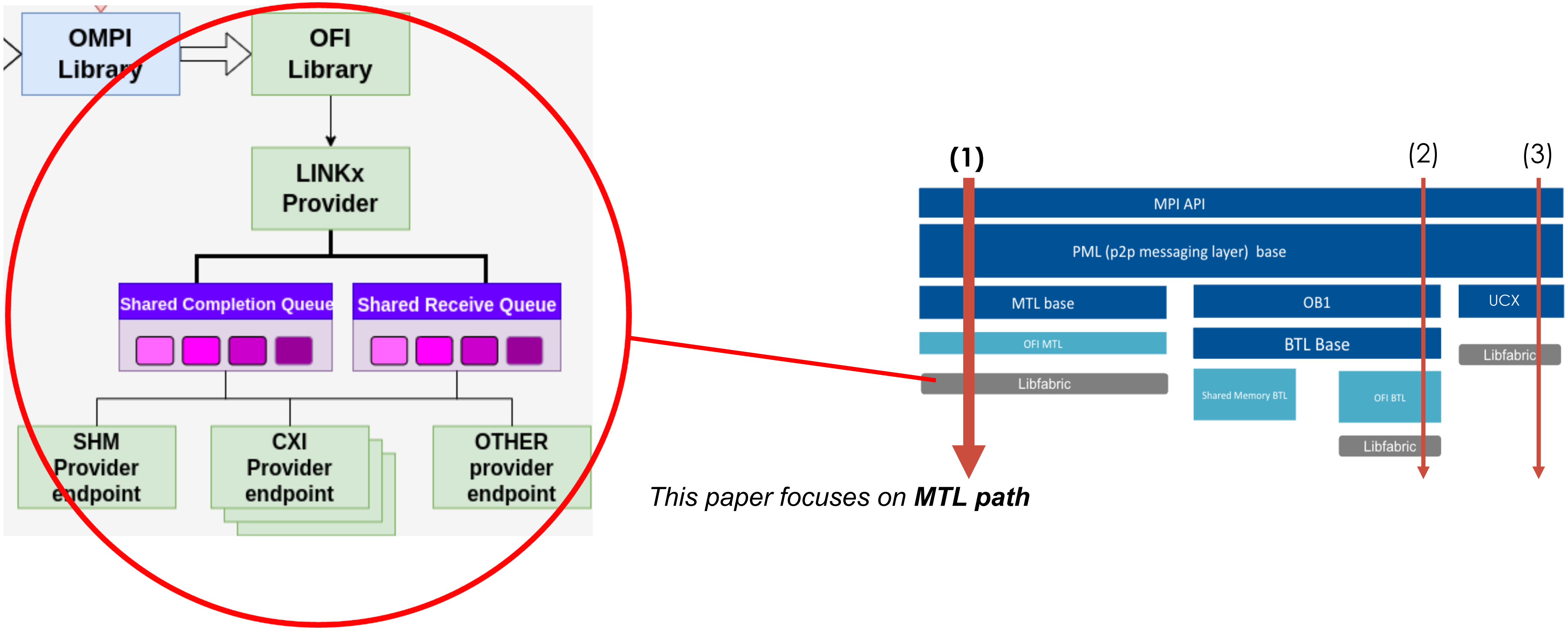
Slingshot 11 & Open MPI

- Cray supports Slingshot 11 via a new CXI libfabric provider
  - Supports communication with both host & device buffers
- CXI not directly support on-node communication
  - Functional but messages egress/ingress node
- Three potential solutions to use CXI provider with Open MPI
  1. **MTL path** – use libfabric tagged message interface
  2. BTL path – use MPI for tag matching & higher level logic, libfabric for byte transfer only
  3. UCX path – use UCX and integrate libfabric under the UCX API



*This paper focuses on **MTL path***

Where LINKx fits in Open MPI Architecture Diagram



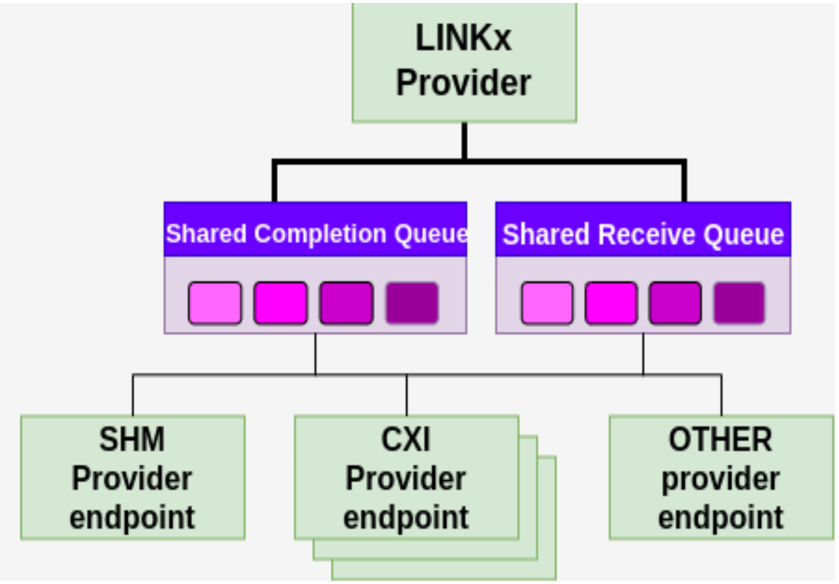
*This paper focuses on **MTL path***

Libfabric: ***LINKx*** provider

- A new OFI libfabric provider to link multiple providers
  - Terminology: LINKx links “core” providers
- Enables Open MPI to use single provider for local & remote communication
  - *Reminder: Open MPI’s MTL limited to 1 active component*
- Chooses endpoint provider based on peer locality
- Shares both its <u>completion queues</u> and <u>receive queues</u> to reduce communication and memory overhead
- Can potentially be expanded to handle multi-rail

LINKx Shared Data Structures

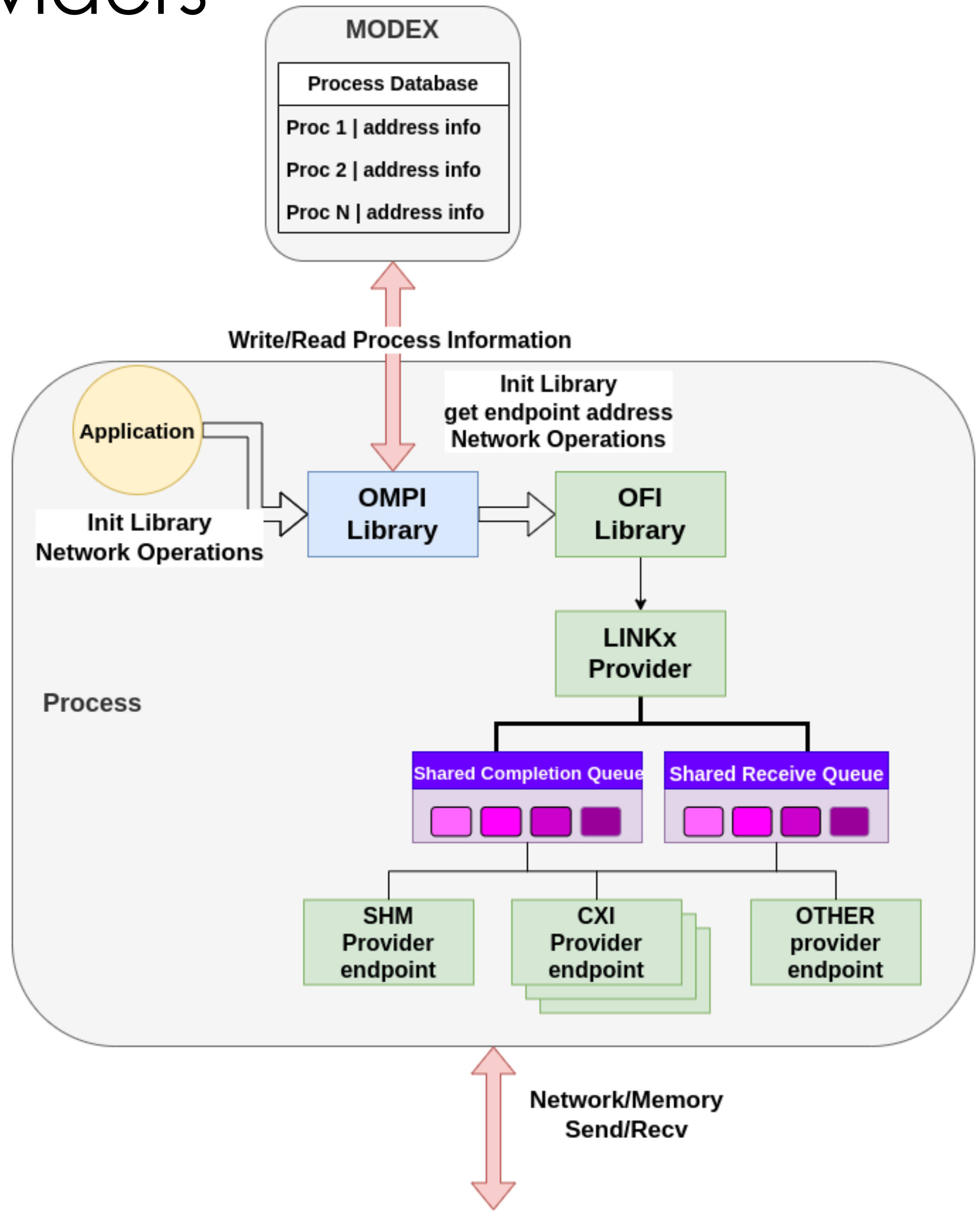
- Unified completion & receive queues
  - LINKx exports queues
  - Core providers in link use exported queues
- Shared queues
  - Avoids each provider needing to maintain separate queues
  - Avoids LINKx needing to search multiple queues
- Matching with LINKx
  - Disable hardware assisted tag matching
  - Use software matching to avoid ambiguity between linked providers



Flow: LINKx joining SHM & CXI providers

Example of Open MPI initialization with LINKx

1. Initialize libfabric to get LINKx provider with SHM+CXI
2. Application does typical libfabric setup for provider
3. LINKx builds structures to track linked providers
4. Open MPI MODEX: Before exchange, LINKx concatenates all addresses in link and publishes
5. Open MPI MODEX: After exchange, Open MPI reads all addresses, LINKx parses & sets up linked providers
6. Open MPI uses libfabric APIs to communicate with peers. At runtime, LINKx examines peer & selects best provider based on locality.



Improvements to Libfabric Shared Memory (SHM)

- New features to SHM provider for MPI use cases
  - Full support for ROCm HSA APIs
  - Add Asynchronous ROCm IPC support
  - Add IPC caching mechanism
  - Add XPMEM support
    - Allows mapping remote process memory space locally; provides efficient method of sharing memory
    - Optimization for H2D case to leverage XPMEM to directly copy into Device memory
    - Support XPMEM export for specific memory regions (instead of entire address space)
  - Add ROCm HIP API support (intended as reference implementation)

Improvements for Collectives

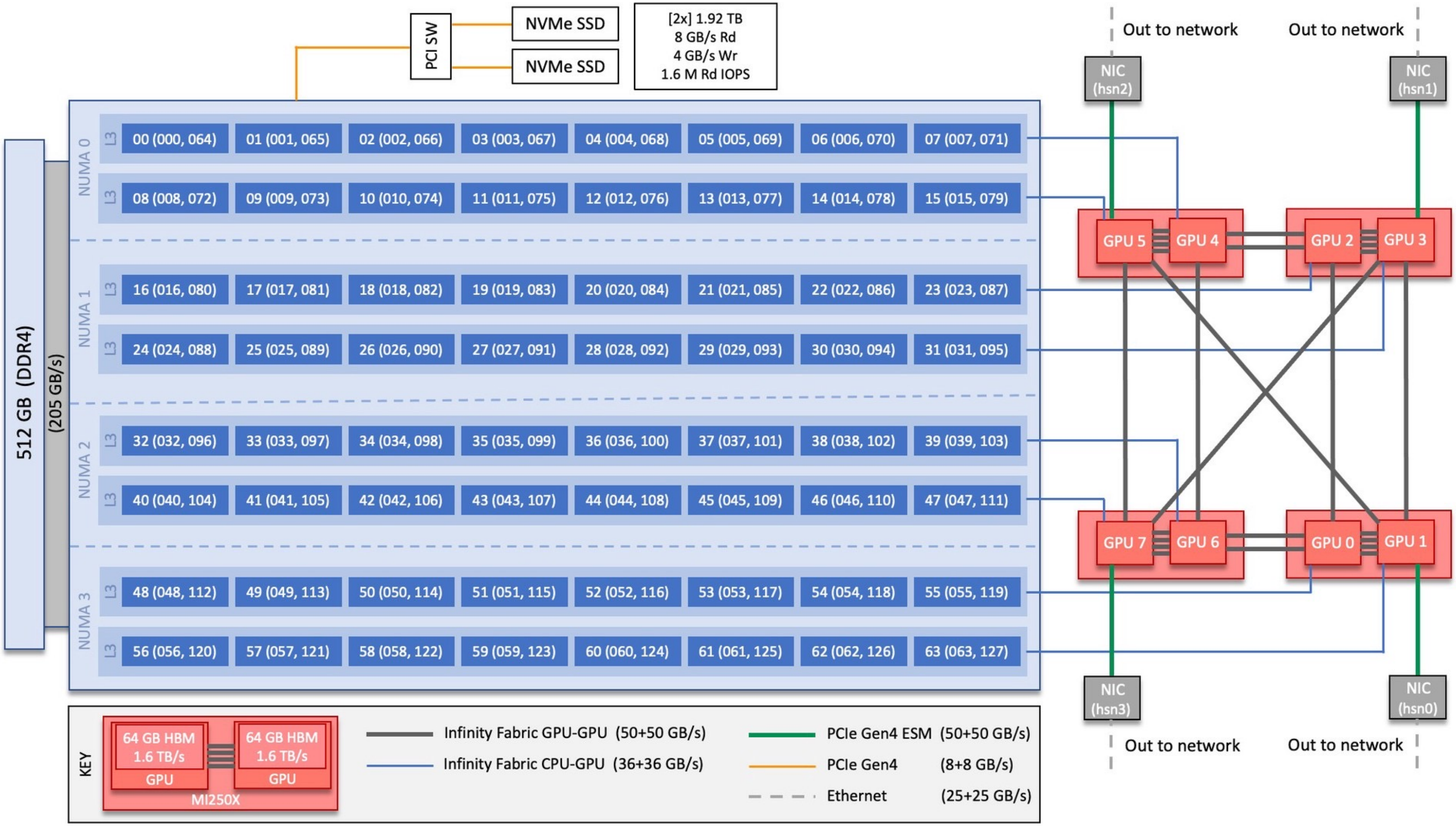
- Key changes to help bring performance closer to Cray MPI
  - Selection of the optimal network interface for a process
  - SHM locking improvements
    - SHM provider locking was very course, causing serialization between processes
    - Moved to more lock free strategy to minimize serialization

Frontier Supercomputer

- HPE Cray EX system
  - 74 cabinets
  - 9,472 AMD EPYC CPUs
  - 37,888 MI250x GPUs
    - Each MI250x GPU has 2 GCDs (Presents as 8 devices/node)
  - Slingshot 11 interconnect

FRONTIER
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY
Hewlett Packard Enterprise
AMD

Frontier Supercomputer - Node Specifications



NUMA 0	L3	00 (000, 064)	01 (001, 065)	02 (002, 066)	03 (003, 067)	04 (004, 068)	05 (005, 069)	06 (006, 070)	07 (007, 071)
	L3	08 (008, 072)	09 (009, 073)	10 (010, 074)	11 (011, 075)	12 (012, 076)	13 (013, 077)	14 (014, 078)	15 (015, 079)
NUMA 1	L3	16 (016, 080)	17 (017, 081)	18 (018, 082)	19 (019, 083)	20 (020, 084)	21 (021, 085)	22 (022, 086)	23 (023, 087)
	L3	24 (024, 088)	25 (025, 089)	26 (026, 090)	27 (027, 091)	28 (028, 092)	29 (029, 093)	30 (030, 094)	31 (031, 095)
NUMA 2	L3	32 (032, 096)	33 (033, 097)	34 (034, 098)	35 (035, 099)	36 (036, 100)	37 (037, 101)	38 (038, 102)	39 (039, 103)
	L3	40 (040, 104)	41 (041, 105)	42 (042, 106)	43 (043, 107)	44 (044, 108)	45 (045, 109)	46 (046, 110)	47 (047, 111)
NUMA 3	L3	48 (048, 112)	49 (049, 113)	50 (050, 114)	51 (051, 115)	52 (052, 116)	53 (053, 117)	54 (054, 118)	55 (055, 119)
	L3	56 (056, 120)	57 (057, 121)	58 (058, 122)	59 (059, 123)	60 (060, 124)	61 (061, 125)	62 (062, 126)	63 (063, 127)

8 GCDs:
4 GPUs x 2 GCD each

4 NICs :
1 per GPU

Experiment Setup

- Data gathered on *Crusher* (smaller, but same HW as *Frontier*)
- System software
  - SUSE Linux 15 SP4 / Linux 5.14.31 (cray-shasta environment)
  - SLURM 22.05.7 with core specialization enabled
  - ROCm v5.3.0, CCE 15.0.0, xpmem v2.5.2
- MPI versions
  - Cray MPI version 8.1.23 with Cray PMI v6.1.8
    - Libfabric: v1.15.2.0
  - Open MPI v5.0.0rc11 with 1 patch*
    - Libfabric: ‘ornl-main’ branch with LINKx & shared memory provider enhancements

* PR #11565 “ofi: NIC selection”
https://github.com/open-mpi/ompi/pull/11565

Experiment Setup

- Testing tool sets `HIP_VISIBLE_DEVICES` to best setting for Crusher
- 8 MPI processes per node (one per GCD)
  - Mapping: 1 process per L3cache (--map-by ppr:1:l3cache)
    - Except for P2P inter-node tests, map 1 process per node (--map-by ppr:1:node)
  - Bind processes to core

Example run lines

Open MPI with LINKx enhanced libfabric

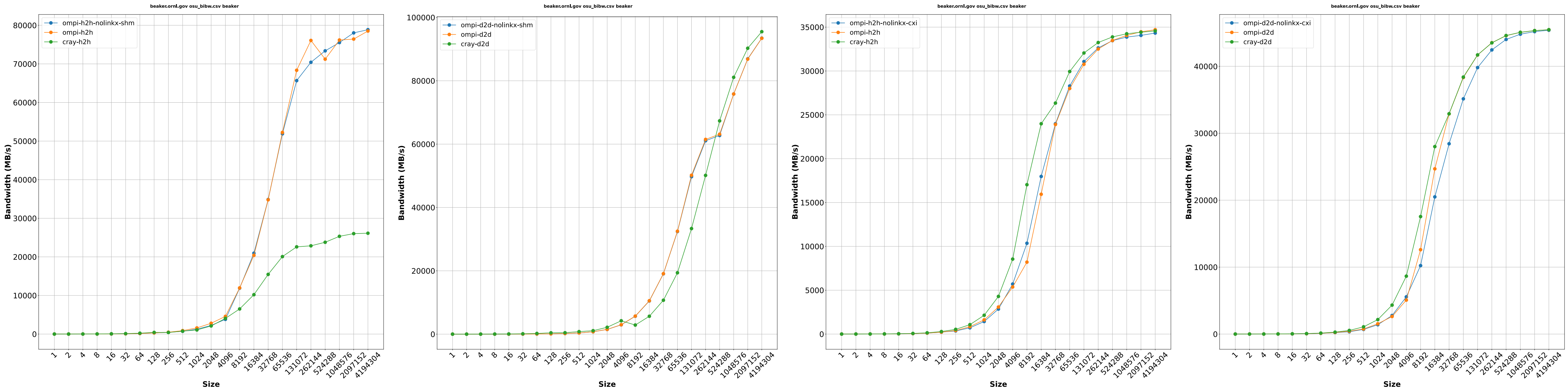
```
mpirun \
-x FI_USE_XPMEM -x LD_LIBRARY_PATH \
--mca btl ^tcp,ofi,vader,openib \
--mca pml ^ucx --mca mtl ofi \
--mca opal_common_ofi_provider_include "shm+cxi:linkx" \
 --map-by ppr:1:l3cache  --bind-to core \
--display mapping,bindings --np 512 \
<osu-exe> H H
# -- or --
<osu-exe> -d rocm D D
```

** Note: Most of these parameters set via modulefile*

CrayMPICH with system libfabric

```
srun \
--cpu-bind=v,cores \
--ntasks 512 \
--ntasks-per-node 8 \
-N 64 \
-t 10000 \
<osu-exe> H H
 # --or--
<osu-exe> -d rocm D D
```

Point-to-Point Bi-directional Bandwidth (osu_bibw)



(a) H2H intra-node

(b) D2D intra-node

(c) H2H inter-node

(d) D2D inter-node

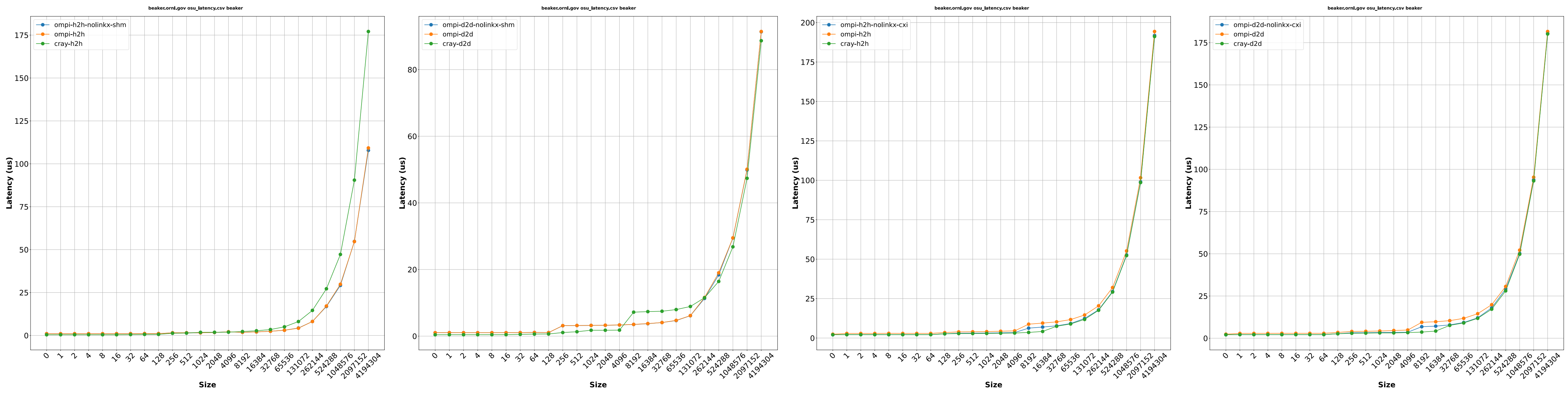
Fig. 4: Point-to-Point Bidirectional Bandwidth `osu_bibw`

- Take-away: Trend is following CrayMPICH

(Unexplained problem w/ CrayMPICH H2H)

Key:
Open MPI no LINKx (cxi or shm)
Open MPI with LINKx
CrayMPICH

Point-to-Point Latency (osu_latency)



(a) H2H intra-node (b) D2D intra-node (c) H2H inter-node (d) D2D inter-node

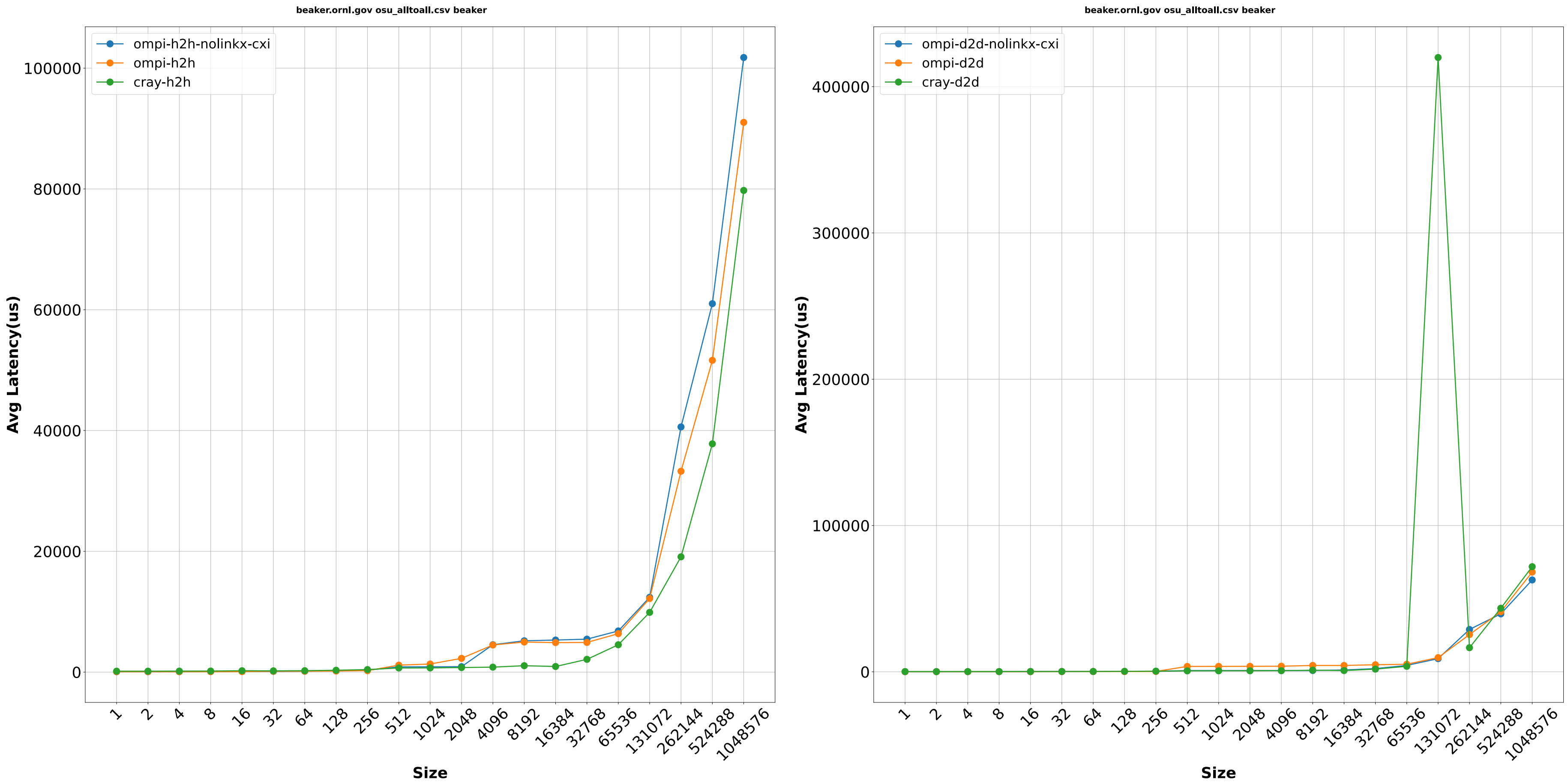
Fig. 5: Point-to-Point Latency `osu_latency`

- Take-away: Trend is following CrayMPICH

(Unexplained problem w/ CrayMPICH H2H)

Key:
Open MPI no LINKx (cxi or shm)
Open MPI with LINKx
CrayMPICH

Collective Alltoall (osu_alltoall)



(a) H2H

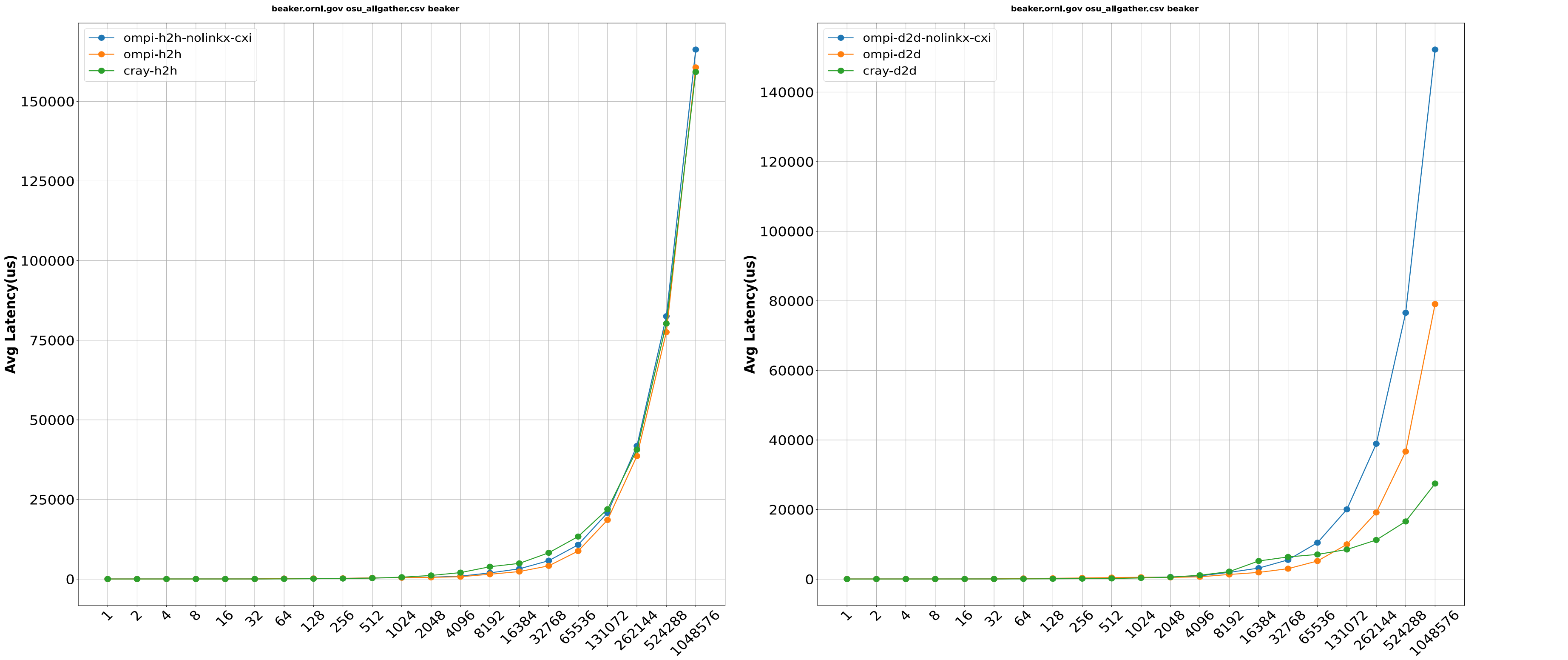
(b) D2D

Fig. 7: Collective Alltoall `osu_alltoall`

- Take-away: Trend is following CrayMPICH

Key:
Open MPI no LINKx (cxi or shm)
Open MPI with LINKx
CrayMPICH

Collective Allgather (osu_allgather)



(a) H2H

(b) D2D

Fig. 9: Collective Allgather `osu_allgather`

- Take-away: D2D needs work as size grows

Key:
Open MPI no LINKx (cxi or shm)
Open MPI with LINKx
CrayMPICH

Future Work

- MPI
  - Performance improvements (small messages, collectives)
  - Finalize open issue for MPI one-sided support
- Libfabric
  - Complete LINKx support for all libfabric APIs
  - Productize LINKx and test linking multiple providers
  - Upstream changes
  - Multi-rail support via LINKx
- Other
  - Support Intel GPUs on Aurora

Summary

- Outlined key challenges addressed to bring Open MPI up on Frontier with good performance
  - Still improvements to be made, function and performance shows good trends compared to vendor’s well tuned CrayMPICH
  - Experiments highlight importance of process affinity to GPU/Network
- Presented status of Open MPI with Slingshot 11
  - Summary of new LINKx provider (joins SHM & CXI provider)
  - LINKx used for Open MPI’s MTL/OFI framework
  - Highlighted improvements (SHM provider, ROCm & XPMEM support)

Questions?

Acknowledgement: This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Howard Pritchard acknowledges support by the National Nuclear Security Administration. Los Alamos National Laboratory is operated by Triad National Security, LLC for the U.S. Department of Energy under contract 89233218CNA000001. This research was partially supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.