Enabling Exascale Co-Design Architecture

Devendar Bureddy

August 2018

Agenda

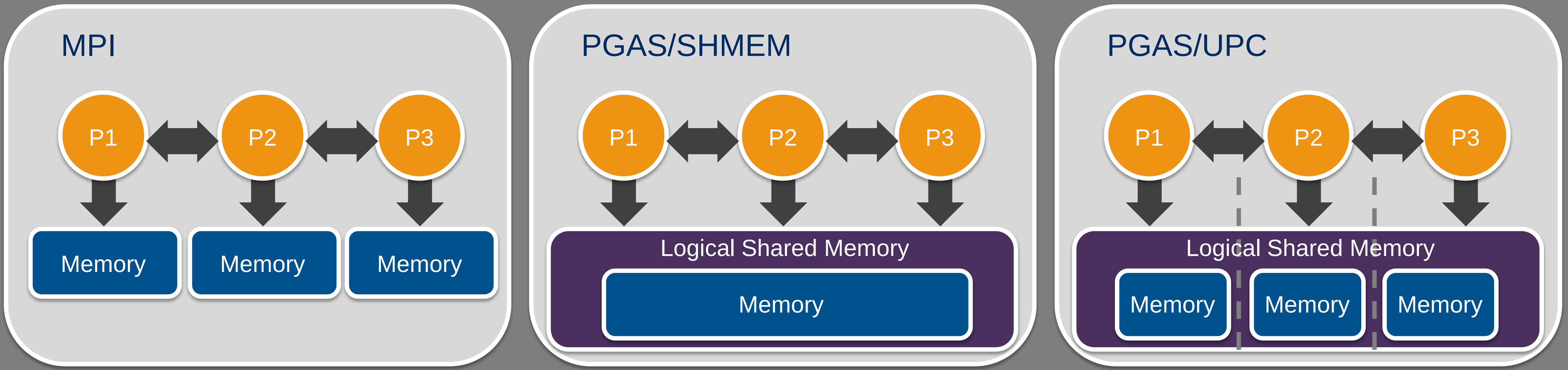
- HPC-X
- HCOLL - Hierarchical Collectives
- SHARP - Scalable Hierarchical Aggregation and Reduction Protocol
  - Architecture
  - Deployment
  - Multi-Channel
  - Multi-Rail
- UCX
  - Introduction
  - Design
  - Features
  - API overview
  - Example

Mellanox HPC-X™ Scalable HPC Software Toolkit

- Complete MPI, PGAS OpenSHMEM and UPC package
- Maximize application performance
- For commercial and open source applications
- Best out of the box experience

Mellanox HPC-X Software Ecosystem

Applications



Point-to-Point: MXM -> UCX

- Reliable Messaging Optimized for Mellanox HCA
- Hybrid Transport Mechanism
- Efficient Memory Registration
- Receive Side Tag Matching

Collective: FCA

- Hardware Acceleration: SHARP, Multicast, CORE-Direct
- Topology Aware
- Separate Virtual Fabric for Collectives

InfiniBand Verbs API

Mellanox HPC-X - Package Contents

HPC-X – Mellanox Scalable HPC Toolkit

- Allow fast and simple deployment of HPC libraries
  - Both Stable & Latest Beta are bundled
  - All libraries are pre-compiled
  - Includes scripts/modulefiles to ease deployment

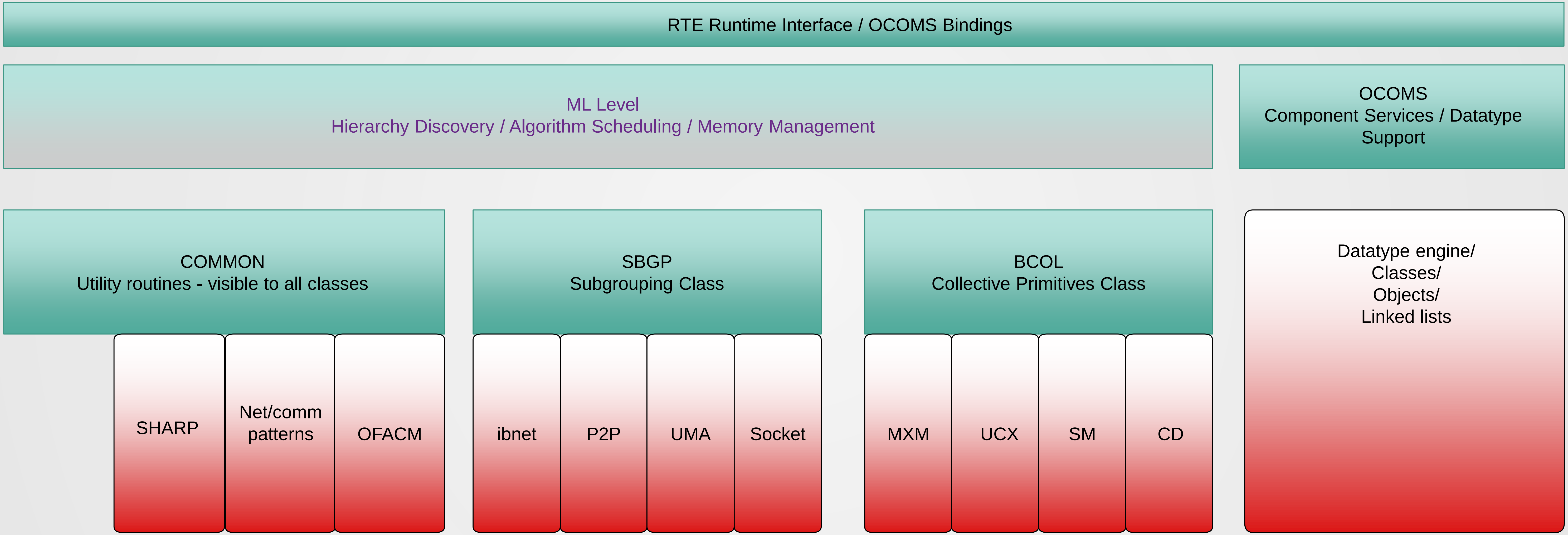
- Package Includes
  - OpenMPI/OpenSHMEM
  - BUPC (Berkeley UPC)
  - UCX
  - MXM (Deprecated)
  - FCA-3.x (HCOLL)
  - KNEM
    - Allows fast intra-node MPI communication for large messages
  - Profiling Tools
    - Libibprof
    - IPM
  - Standard Benchmarks
    - OSU
    - IMB

HCOLL

HCOLL

- Scalable infrastructure: Designed and implemented with current and emerging "*extreme-scale*" systems in mind
  - Scalable communicator creation, memory consumption, runtime interface
  - Asynchronous execution
- Blocking and non-blocking collective routines
- Easily integrated into other packages
  - Successfully integrated into OMPI – "hcoll" component in "coll" framework
  - Successfully integrated in Mellanox OSHMEM
  - Experimental integration in MPICH
- Host level hierarchy awareness
  - Socket groups, UMA groups
- Exposes Mellanox and InfiniBand specific capabilities
  - UCX, CORE-Direct. MCAST, SHARP

HCOLL Software Architecture



Enabling HCOLL

- OpenMPI which ships with MOFED/HPC-X enables HCOLL by default, and its priority is set to highest.

- explicitly enable/disable it
  - `$ mpirun …  -mca coll_hcoll_enable 1 -mca coll_hcoll_np 0 ./osu_barrier`

  - **-mca coll_hcoll_enable {0/1}** : Enable/Disable HCOLL
  - **-mca coll_hcoll_np <comm_size>**: communicator size threshold to use HCOLL collectives

- Selecting HCA device
  - -x HCOLL_MAIN_IB=<dev:port>[,<dev:port>,…]

HCOLL

- HCOLL Supported Collectives
  - MPI_Barrier | MPI_Ibarrier
  - MPI_Allreduce | MPI_Iallreduce
  - MPI_Reduce | MPI_Ireduce
  - MPI_Bcast | MPI_Ibcast
  - MPI_Allgather | MPI_Iallgather
  - MPI_Alltoallv | MPI_Ialltoallv
  - MPI_Alltoall | MPI_Ialltoall
  - MPI_Alltoallv | MPI_Ialltoallv

$\left.\begin{array}{l} \text{MPI\_Barrier} \\ \text{MPI\_Allreduce} \\ \text{MPI\_Reduce} \end{array}\right\}$ SHARP Supported

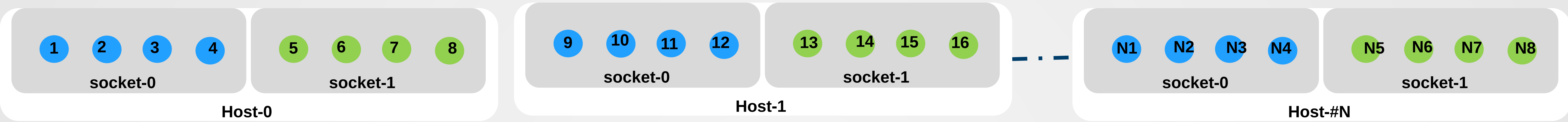
- For non supported collectives, fallback to next high priority collective module (tuned/basic ..)

- Disable specific collective in HCOLL
  - **HCOLL_ML_DISABLE_<COLL_NAME> = 1**
  - Example : Disable Bcast
    - **-x HCOLL_ML_DISABLE_BCAST=1**

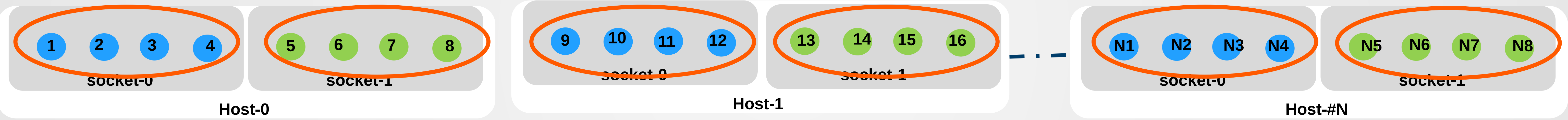
HCOLL: Topology aware sub grouping (SBGP)

HCOLL_SBGP=basesmsocket,basesmuma,p2p

MPI JOB rank layout:

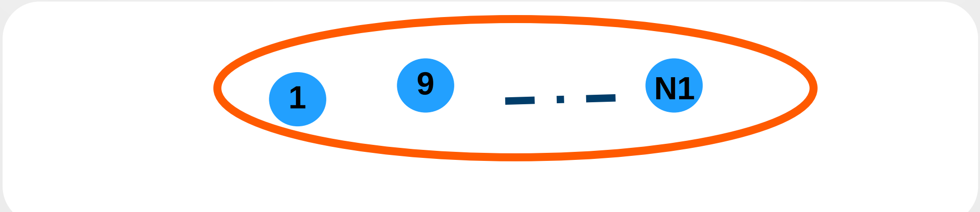


"SOCKET" Subgroup:



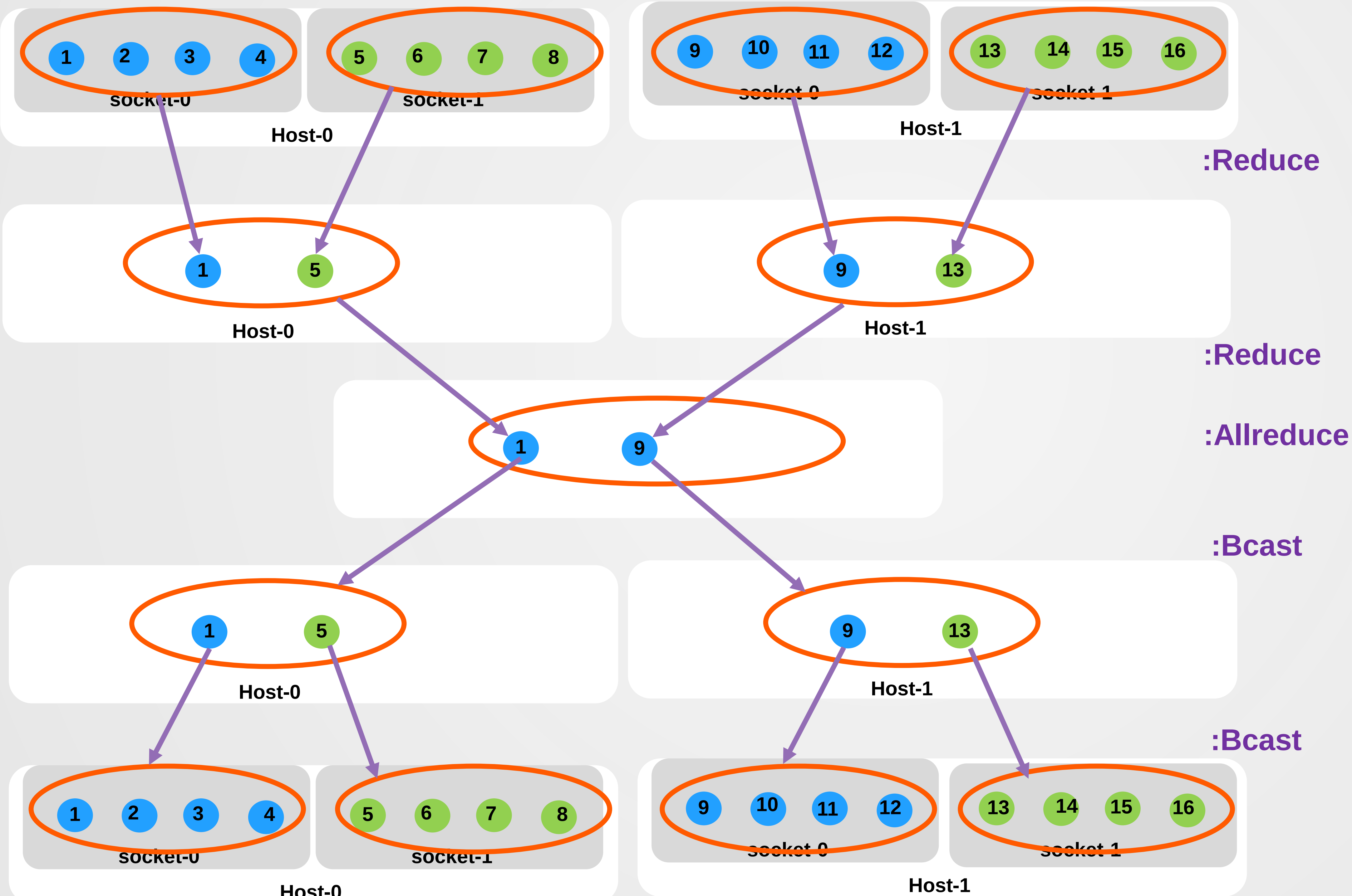
"UMA" Subgroup:

"P2P" Subgroup:
(SHARP group)



HCOLL: Topology aware Collectives

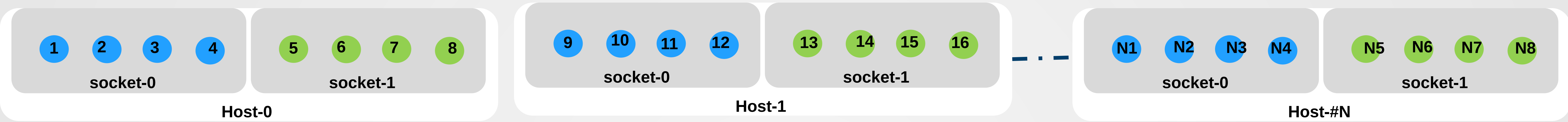
Example: Allreduce



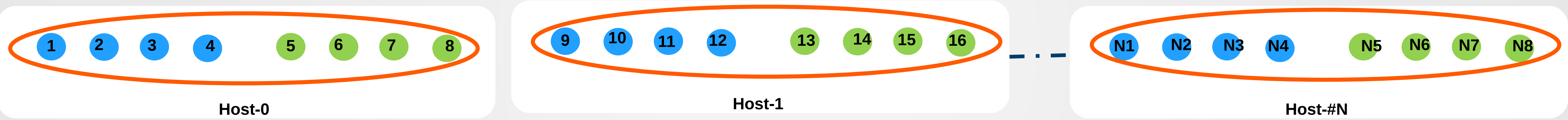
HCOLL: Topology aware sub grouping (SBGP)

HCOLL_SBGP=basesmuma,p2p

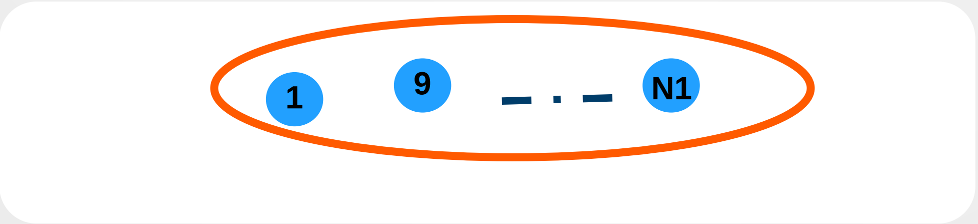
MPI JOB rank layout:



“UMA” Subgroup:



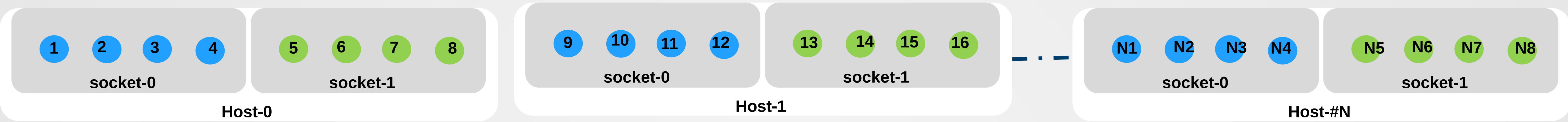
“P2P” Subgroup:



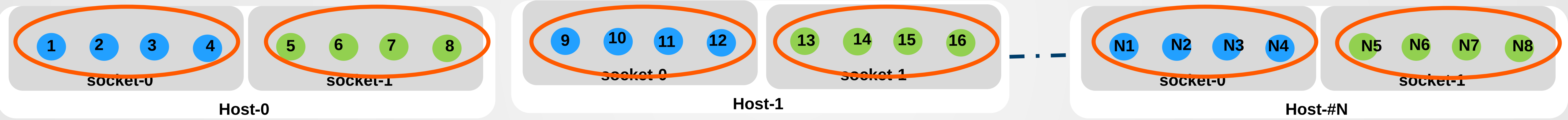
HCOLL: Topology aware sub grouping (SBGP)

HCOLL_SBGP=basesmsocket,p2p

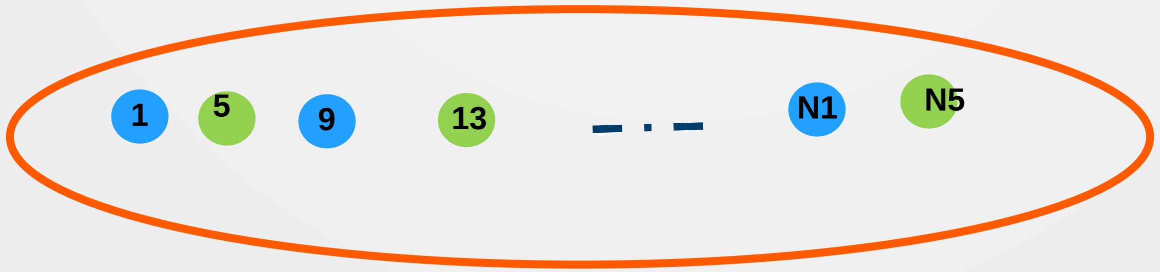
MPI JOB rank layout:



"SOCKET" Subgroup:



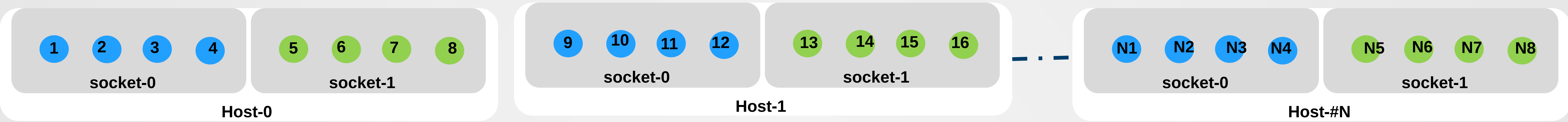
"P2P" Subgroup:



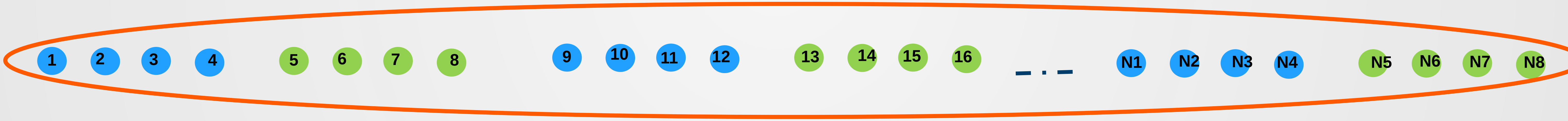
HCOLL: Topology aware sub grouping (SBGP)

HCOLL_SBGP=p2p

MPI JOB rank layout:



"P2P" Subgroup:



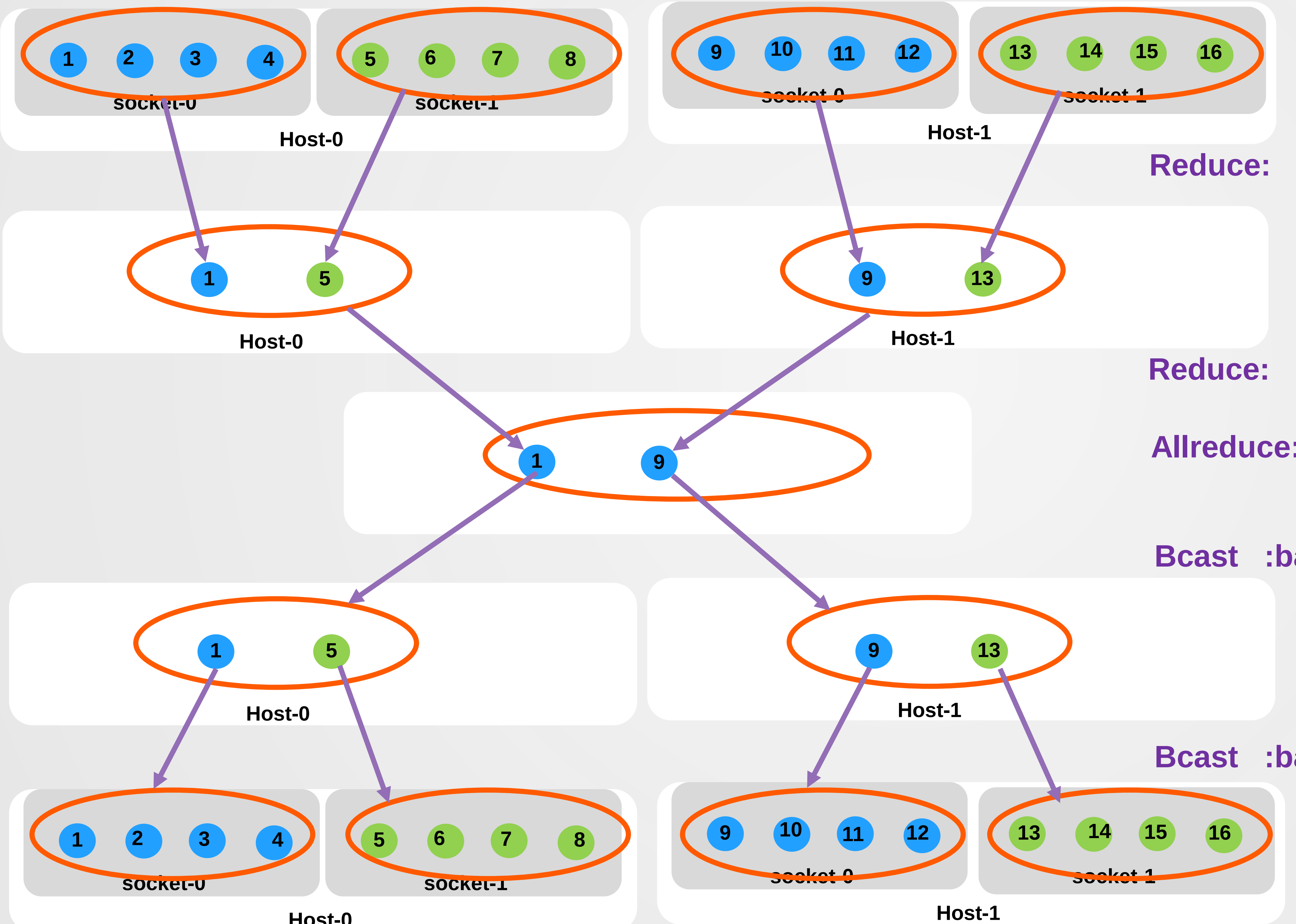
HCOLL

- HCOLL comes with its own set of Communication primitive components, known as bcolls.
  - mlnx_p2p   (MXM)
  - ucx_p2p   (UCX),
  - basesmuma (SHM),
  - cc   (Cross-Channel)
- And its own set of subgroup components known as sbgp
  - Basesmsocket : Socket subgroup
  - Basesmuma   : UMA subgroup
  - P2P   : Network subgroup (SHARP group)

- Choosing BCOLs and SBGPS.
  - Two common env. variables to set are:
    - HCOLL_SBGP and HCOLL_BCOL
    - The above env. variable have a 1:1 mapping, must be the same number.
    - SBGP specifies the subgroup level, BCOL specifies the transport to use at that level

HCOLL: Topology aware Collectives: Allreduce

-x HCOLL_SBGP=basesmsocket,basesmuma,p2p -x HCOLL_BCOL=basesmuma,basesmuma,ucx_p2p



Reduce: basesmuma: SHARED MEMORY

Reduce: basesmuma: SHARED MEMORY

Allreduce: ucx_p2p/SHARP

Bcast :basesmuma: SHARED MEMORY

Bcast :basesmuma: SHARED MEMORY

HCOLL

- Default: 3-level hierarchy is specified with the number of SBGPs (remember, there must be an equal number of bcols)

```
mpirun -np 144 -mca coll_hcoll_enable 1 -mca coll_hcoll_np 0
-x HCOLL_SBGP=basesmsocket,basesmuma,p2p
-x HCOLL_BCOL=basesmuma,basesmuma,ucx_p2p ./osu_barrier
```

- 2-level hierarchy , no socket grouping

```
-x HCOLL_SBGP=basesmuma,p2p -x HCOLL_BCOL=basesmuma,mlnx_p2p
```

- 2-level hierarchy , socket leaders in network group, No intra-socket communication (SHARP Multi-Channel)

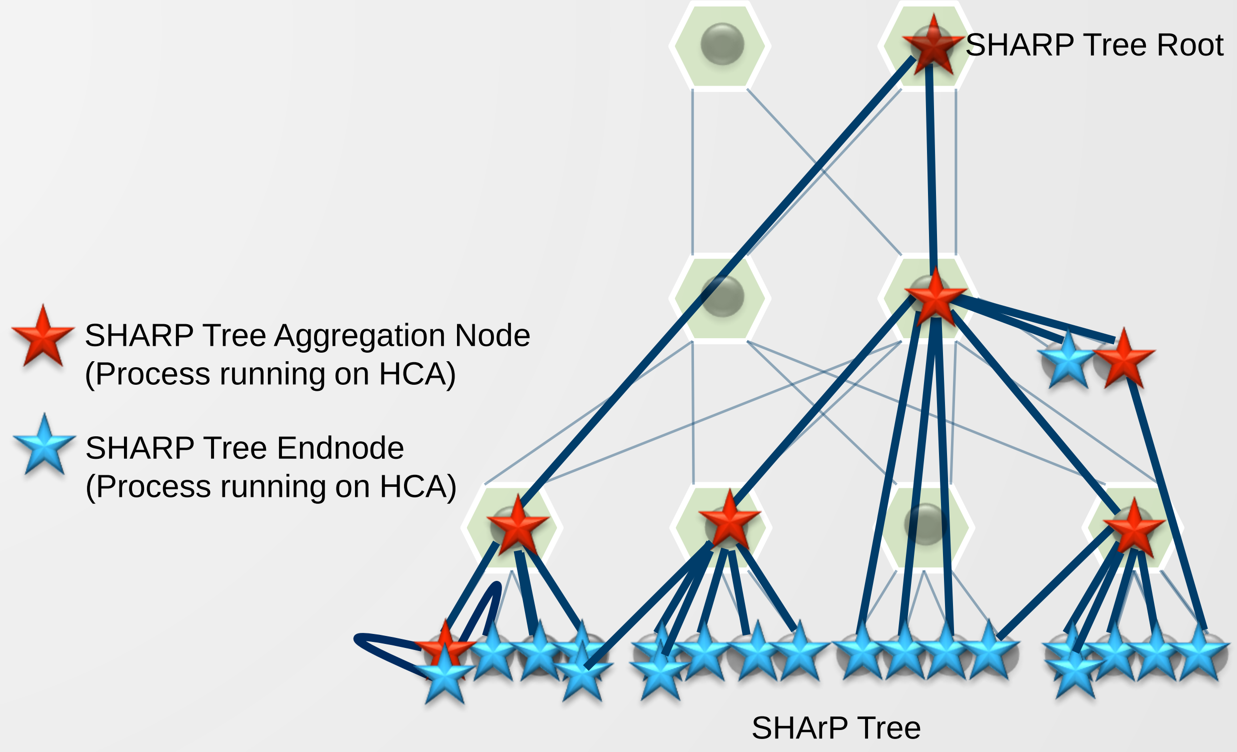
```
-x HCOLL_SBGP=basesmsocket,p2p -x HCOLL_BCOL=basesmuma,mlnx_p2p
```

SHARP

Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)

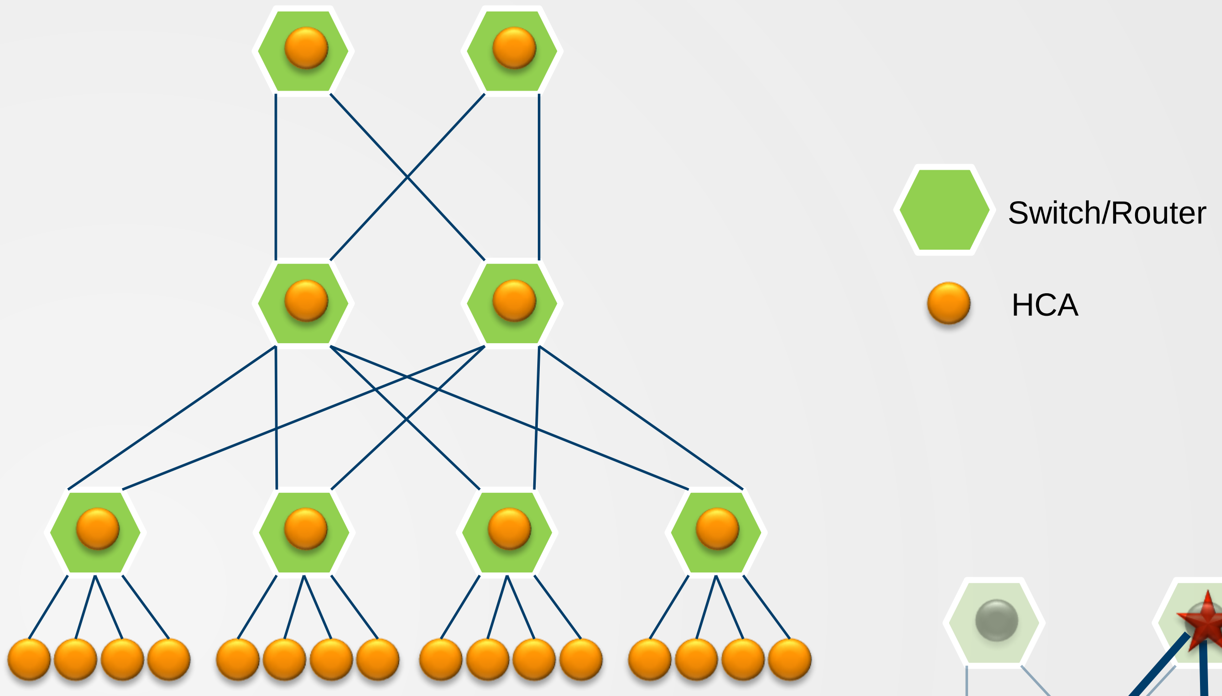
Scalable Hierarchical Aggregation and Reduction Protocol

- Reliable Scalable General Purpose Primitive
  - In-network tree based aggregation mechanism
  - Large number of groups
  - Multiple simultaneous outstanding operations
- Applicable to Multiple Use-cases
  - HPC Applications using MPI / SHMEM
  - Distributed machine learning applications
- Scalable High Performance Collective Offload
  - Barrier, Reduce, All-Reduce, Broadcast and more
  - Sum, Min, Max, Min-loc, max-loc, OR, XOR, AND
  - Integer and Floating-Point, 16/32/64 bits

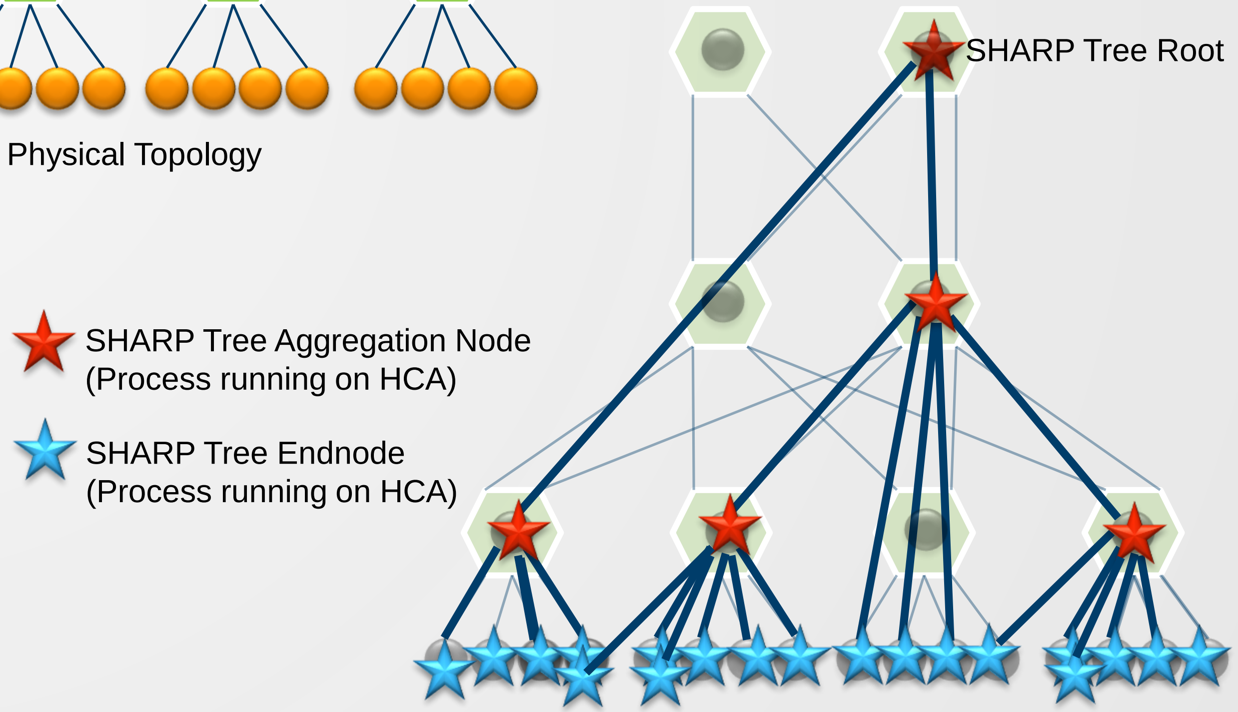


SHARP Tree

- SHARP Operations are executed by a SHARP tree
  - Multiple SHARP trees are supported
  - Each SHARP tree can handle multiple outstanding SHARP operations
  - Within a SHARP tree, each operation is uniquely identified by a SHARP-Tuple



Physical Topology

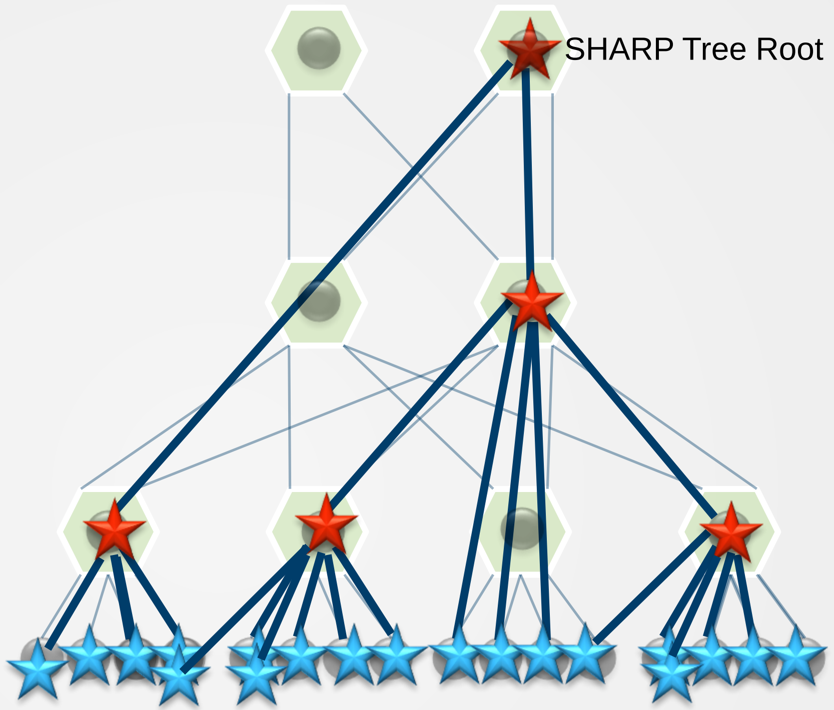


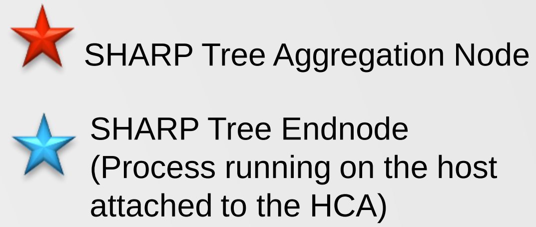
One SHARP Tree

SHARP Tree (cont'd)

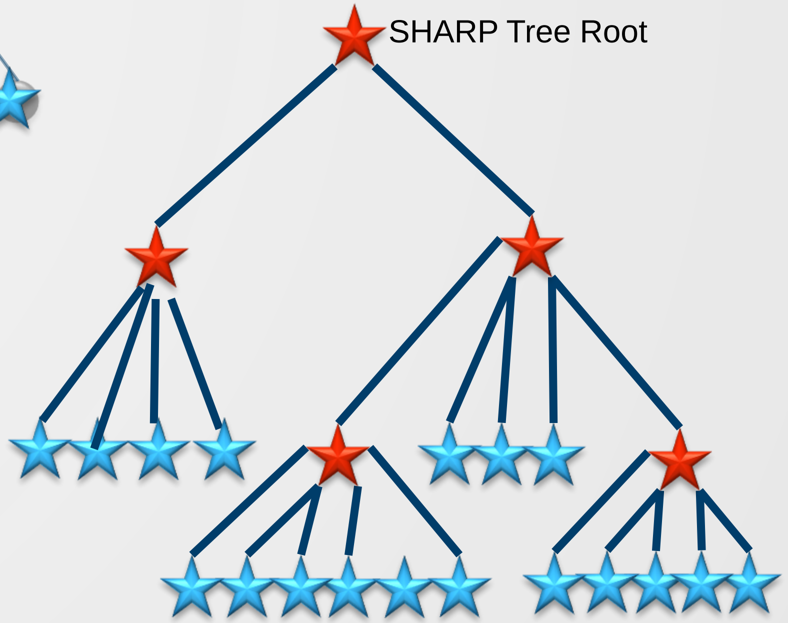
- SHARP tree is a *logical construct*
  - Motivation to loosely follow the underlying physical topology for performance
    - Efficient use of physical link bandwidth
    - Shortest overall path from leaf to root
- SHARP group is a *tree subset*
  - A sub-tree spanning a subset of the tree end-nodes
- SHARP tree nodes are *'Processes'*
  - Multiple SHARP tree "Nodes" can reside on the same physical HCA

SHARP Tree



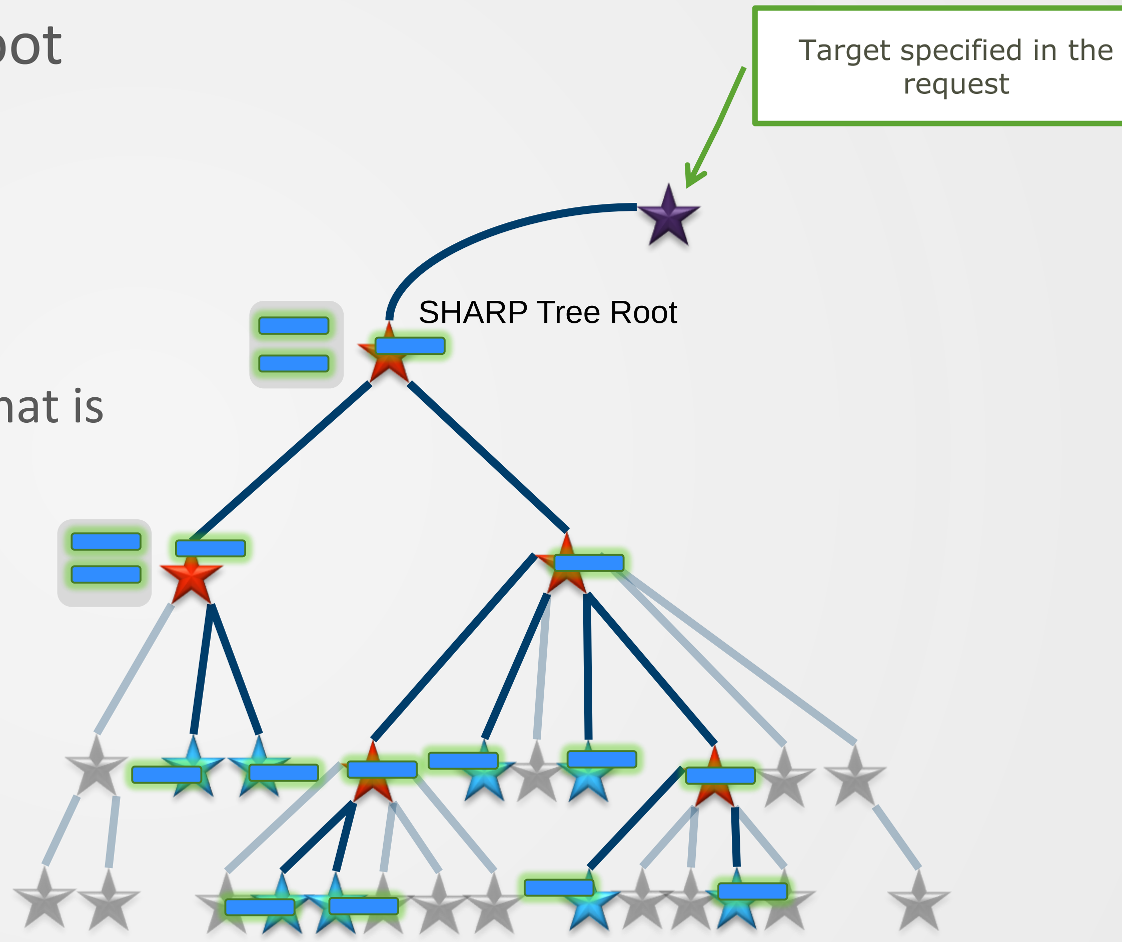


The same SHARP Tree
(Logical View)

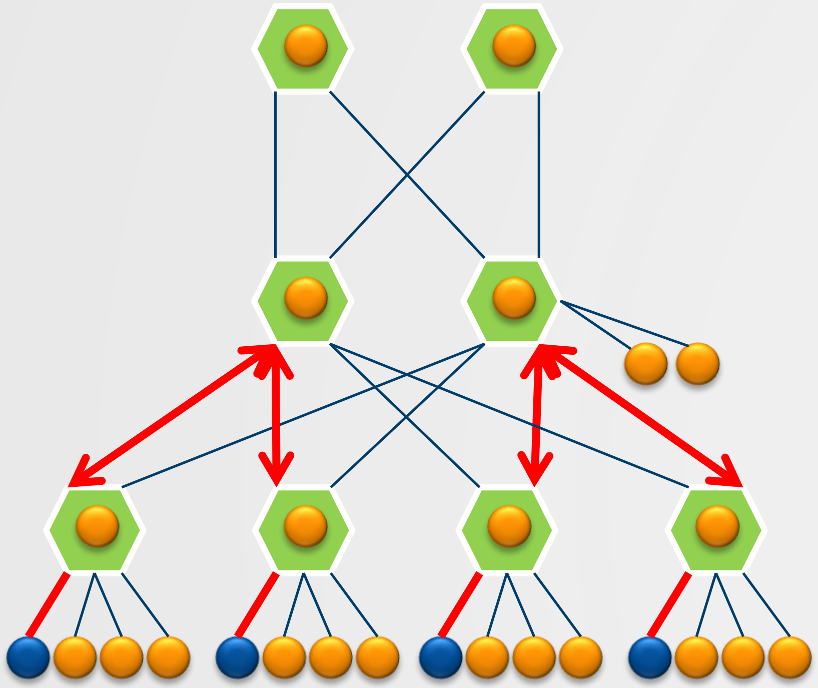


SHARP Principles of Operation

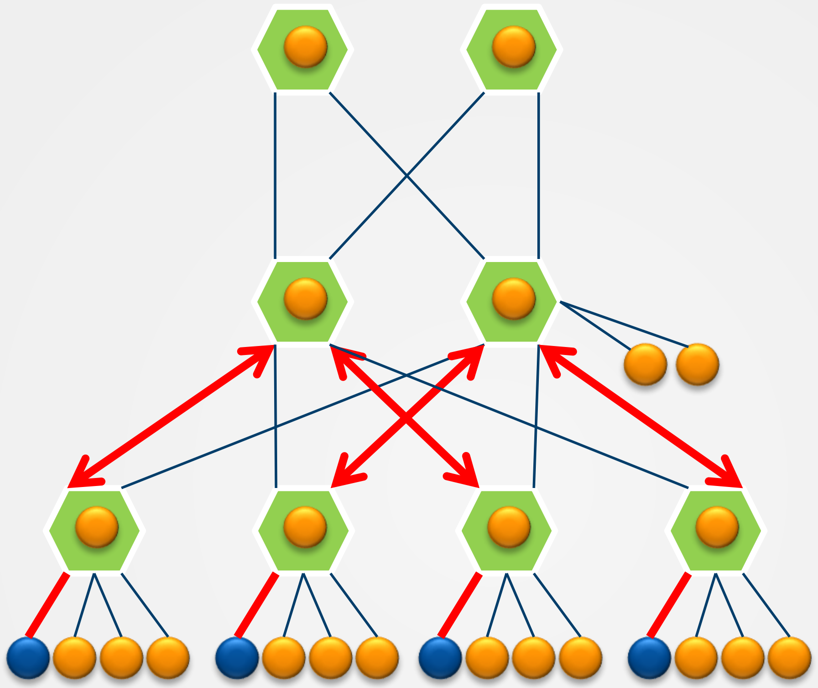
- Aggregation response initiated by the group root
- Aggregation response message
- Upon completion of aggregation operation:
  - Group root (could be tree root) creates a response that is propagated to the hosts
- SHARP tree response
  - Upon receiving a SHARP response packet (from the parent)
  - SHARP response message is sent to each child



HCOLL: SHARP vs No-SHARP

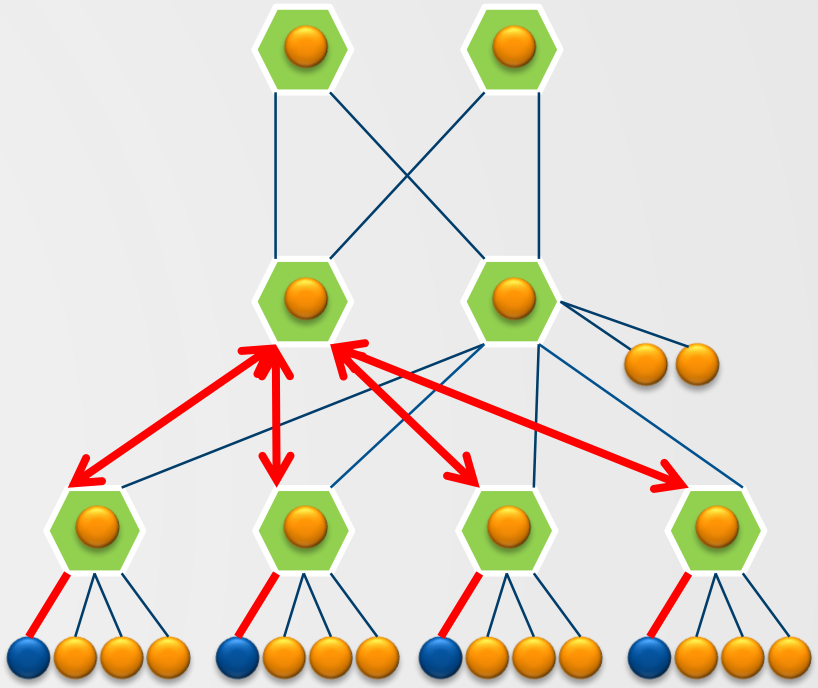


Step 1 `



Step 2 `

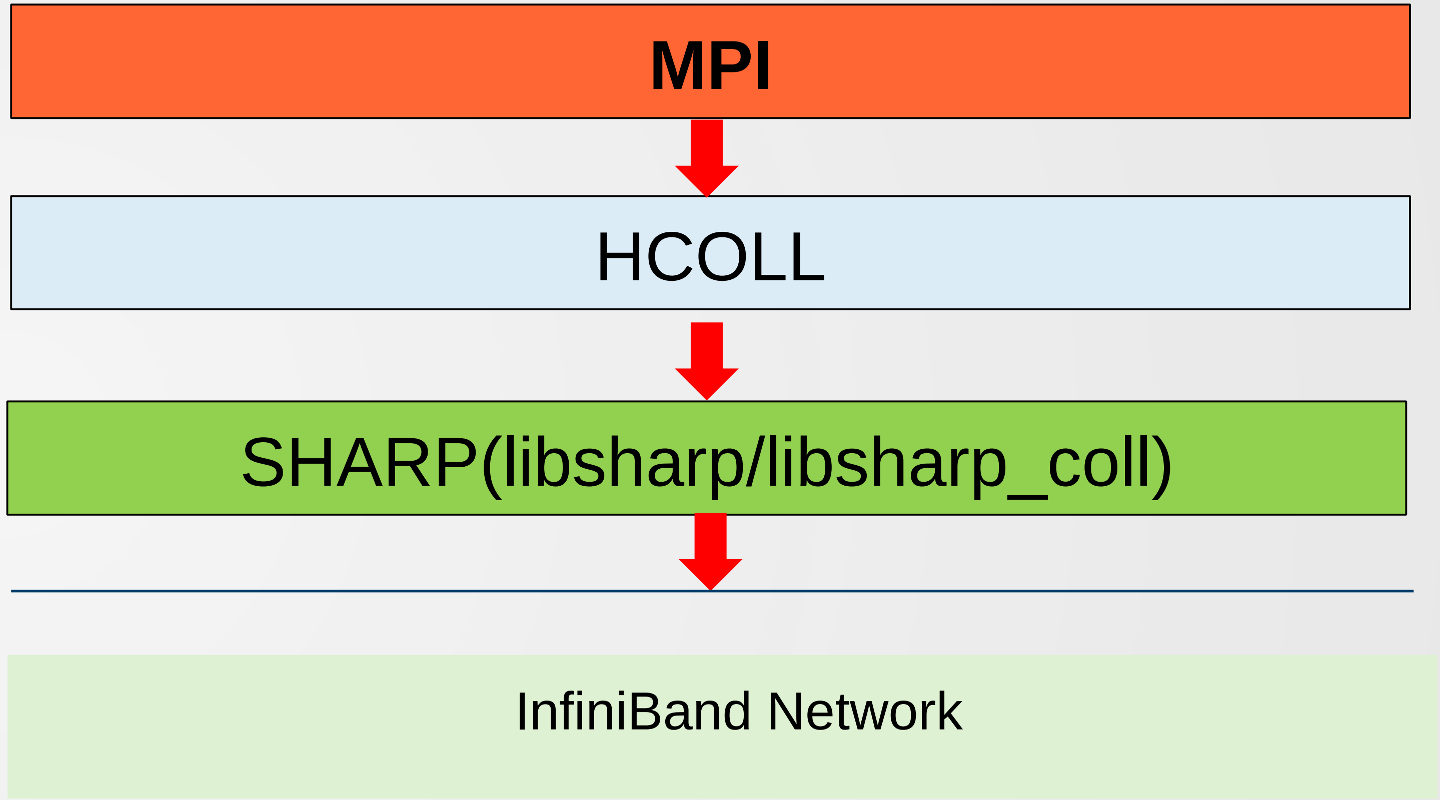
Recursive Doubling `



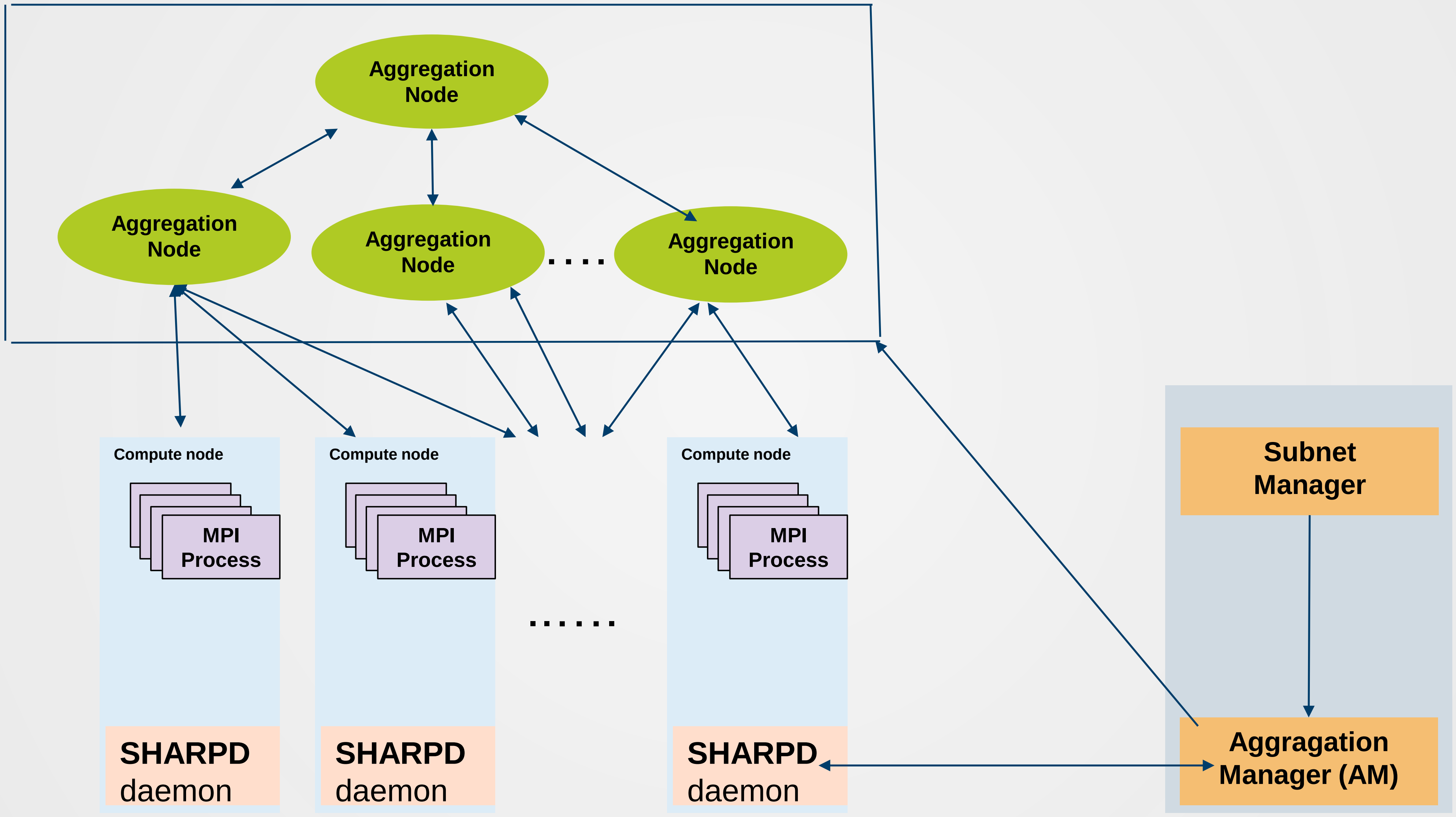
SHARP`

HPCX/SHARP SW architecture

- HCOLL
  - optimized collective library
- Libsharp_coll.so
  - Implementation of high level sharp API for enabling sharp collectives for MPI
  - uses low level libsharp.so API
- Libsharp.so
  - Implementation of low level sharp API
- High level API
  - Easy to use
  - Easy to integrate with multiple MPIs(OpenMPI, MPICH, MVAPICH)



SHARP SW Architecture



SHARP SW components

- SHArP SW components:
  - Libs
    - libsharp.so (low level api)
    - libsharp_coll.so (high level api)
  - Daemons
    - sharpd, sharp_am
  - Scripts
    - sharp_benchmark.sh
    - sharp_daemons_setup.sh
  - Utilities
    - sharp_coll_dump_config
    - sharp_hello
    - sharp_mpi_test
  - public API
    - sharp.h

SHArP: SHArP Daemons

- sharpd: SHArP daemon
  - compute nodes
  - Light wait process
  - Almost 0% cpu usage
  - Only control path

- sharp_am: Aggregation Manager daemon
  - same node as Subnet Manager
  - Resource manager

SHArP: Configuring Subnet Manager

- Edit the opensm.conf file.
- Configure the "routing_engine" parameter.
  - ftree fabric : routing_engine ftree,updn
  - hypercube fabric : routing_engine dor
- Set the parameter “sharp_enabled” to “2”.
- Run OpenSM with the configuration file.
  - % opensm -F <opensm configuration file> -B
- Verify that the Aggregation Nodes were activated by the OpenSM, run "ibnetdiscover".

For example:

```
vendid=0x0
devid=0xcf09
sysimgguid=0x7cfe900300a5a2a0
caguid=0x7cfe900300a5a2a8
Ca      1 "H-7cfe900300a5a2a8"       # "Mellanox Technologies Aggregation Node"
[1](7cfe900300a5a2a8)  "S-7cfe900300a5a2a0"[37]    # lid 256 lmc 0 "MF0;sharp2:MSB7800/U1" lid 512 4xFDR
```

SHARP: Configuring Aggregation Manager

- Using OpensSM 4.9 or later does not require any special configuration in the AM.
- Configure AM with OpenSM v4.7-4.8:
  - Create a configuration directory for the future SHArP configuration file.
    - % mkdir $HPCX_SHARP_DIR/conf
  - Create root GUIDs file.
    - Copy the root_guids.conf file if used for configuration of Subnet Manager to $HPCX_SHARP_DIR/conf/root_guid.cfg (or)
    - Identify the root switches of the fabric and create a file with the node GUIDs of the root switches of the fabric.
    - For example : if there are two root switches files contains

      0x0002c90000000001

      0x0002c90000000008
  - Create sharp_am.conf file

```
% cat > $HPCX_SHARP_DIR/conf/sharp_am.conf << EOF
root_guids_file $HPCX_SHARP_DIR/conf/root_guid.cfg
ib_port_guid <PortGUID of the relevant HCA port or 0x0>
EOF
```

SHARP: Running SHARP Daemons

- Setup the daemons
  - $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh

- Usage
  - Usage: sharp_daemons_setup.sh <-s> <-r> [-p SHArP location dir] <-d daemon> <-m>

    -s - Setup SHArP daemon

    -r - Remove SHArP daemon

    -p - Path to alternative SHArP location dir

    -d - Daemon name (sharpd or sharp_am)

    -m - Add monit capability for daemon control

  - $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharp_am
  - $service sharp_am start

SHARP: Running SHARP Daemons

- sharp_am
  - %$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharp_am
  - %service sharp_am start
  - Log : /var/log/sharp_am.log

- Sharpd
  - conf file: $HPCX_SHARP_DIR/conf/sharpd.conf
    - ib_dev <relevant_hca:port>
    - sharpd_log_level 2
  - %pdsh -w <hostlist> $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s $HPCX_SHARP_DIR -d sharpd
  - %pdsh -w jupiter[001-032] service sharpd start
  - Log : /var/log/sharpd.log

HCOLL: Enable SHARP

- HCOLL_ENABLE_SHARP

  0 – Disable SHARP (Default)

  1 - probe SHARP availability and use it (recommended if SHARP is not must)

  2 - Force to use SHARP (recommended if SHARP is must)

  3 - Force to use SHARP for all MPI communicators (Benchmarking)

  4 - Force to use SHARP for all MPI communicators and for all supported collectives

- SHARP_COLL_LOG_LEVEL
  - SHARP debug log

    0 – fatal

    1 – error

    2 – warn (Default)

    3 – info (minimal progress log, Recommended to track SHARP is being used or not)

    4 – debug

    5 – trace

HCOLL_ENABLE_SHARP

- **DISABLE SHARP (default)**

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 **-x HCOLL_ENABLE_SHARP=0** -x SHARP_COLL_LOG_LEVEL=3   ./osu_barrier

OSU MPI Barrier Latency Test v5.3.2

Avg Latency(us)

3.72

- **HCOLL_ENABLE_SHARP=1, Setup status: correct**

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 **-x HCOLL_ENABLE_SHARP=1** -x SHARP_COLL_LOG_LEVEL=3   ./osu_barrier

[nemo01:0:21559 - context.c:485] INFO job (ID: 327680001) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:21559 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)

[nemo01:0:21559 - comm.c:417] INFO [group#:0] group id: 7 tree idx:0 rail_idx:0 group size:8 quota: (osts:2 user_data_per_ost:128) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x4c590e000007) mlid:c005

OSU MPI Barrier Latency Test v5.3.2

Avg Latency(us)

3.03

Understanding SHARP Log

- [nemo01:0:21559 - context.c:485] INFO job (ID: 327680001) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
  - SHARP job quota request

- [nemo01:0:21559 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)
  - SHARP JOB request succeeded
  - JOB Quota information, default 10% quota

- [nemo01:0:21559 - comm.c:417] INFO [group#:0] group id: 7 tree idx:0 rail_idx:0 group size:8 quota: (osts:2 user_data_per_ost:128) mgid: (subnet prefix: 0xff12a01bfe800000
  - MPI Communicator create
  - SHARP group create for NODE leaders (HCOLL P2P group)
    - Group size.
    - QUOTA / group info

HCOLL_ENABLE_SHARP

- **HCOLL_ENABLE_SHARP=2, Setup status: correct, #outstanding communicators in job > #sharp groups**

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 **-x HCOLL_ENABLE_SHARP=2** -x SHARP_COLL_LOG_LEVEL=3 -x **SHARP_COLL_JOB_QUOTA_MAX_GROUPS=1** $HPCX_MPI_TESTS_DIR/imb/IMB-MPI1 Barrier **-npmin 4**

[nemo01:0:28628 - context.c:485] INFO job (ID: 1759248385) resource request quota: (osts:0 user_data_per_ost:128 max_groups:1 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:28628 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 **max_groups:1** max_qps:1 max_group_channels:1)

[nemo01:0:28628 - comm.c:417] INFO **[group#:0]** group id: 0 tree idx:0 rail_idx:0 group size:8 quota: (osts:51 user_data_per_ost:128) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x59590e000000) mlid:c005

[nemo01:28628:0][common_sharp.c:360:comm_sharp_coll_comm_init] SHArP: sharp group create failed:SHArP Group alloc error(-4)

[nemo01:0:28628 - comm.c:242] WARN sharp_alloc_groups_info failed: No available groups(-11)

[nemo01:0:28628 unique id 0] WARN No available groups in sharp_alloc_groups_info.

[nemo01:28628:0][common_sharp.c:369:comm_sharp_coll_comm_init] SHArP: continuing without sharp on this communicator..

```
#---------------------------------------------------
# Benchmarking Barrier
# #processes = 4
# ( 4 additional processes waiting in MPI_Barrier)
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         2.16         2.16         2.16
#---------------------------------------------------
# Benchmarking Barrier
# #processes = 8
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         2.26         2.26         2.26
```

HCOLL_ENABLE_SHARP

- $ **HCOLL_ENABLE_SHARP=3, Setup status: correct, #outstanding communicators in job > #sharp groups**

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 **-x HCOLL_ENABLE_SHARP=3** -x SHARP_COLL_LOG_LEVEL=3 **-x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=1** $HPCX_MPI_TESTS_DIR/imb/IMB-MPI1 Barrier -npmin 4

[nemo01:0:28791 - context.c:485] INFO job (ID: 1742602241) resource request quota: (osts:0 user_data_per_ost:128 max_groups:1 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:28791 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 **max_groups:1** max_qps:1 max_group_channels:1)

[nemo01:0:28791 - comm.c:417] **INFO [group#:0]** group id: 0 tree idx:0 rail_idx:0 group size:8 quota: (osts:51 user_data_per_ost:128) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x5b590e000000) mlid:c005

Barrier

[nemo01:28791:0][common_sharp.c:360:comm_sharp_coll_comm_init] SHArP: sharp group create failed:SHArP Group alloc error(-4)

[nemo01:28791:0][common_sharp.c:365:comm_sharp_coll_comm_init] SHArP: Fallback disabled, exiting..

[nemo04:25599:3][common_sharp.c:365:comm_sharp_coll_comm_init] SHArP: Fallback disabled, exiting..

[nemo03:25083:2][common_sharp.c:365:comm_sharp_coll_comm_init] SHArP: Fallback disabled, exiting..

[nemo02:30524:1][common_sharp.c:365:comm_sharp_coll_comm_init] SHArP: Fallback disabled, exiting..

[nemo01:0:28791 unique id 0] WARN No available groups in sharp_alloc_groups_info.

[nemo01:0:28791 - comm.c:242] WARN sharp_alloc_groups_info failed: No available groups(-11)

\---

Primary job terminated normally, but 1 process returned

a non-zero exit code. Per user-direction, the job has been aborted.

SHARP Group size threshold

- HCOLL_SHARP_NP (default: 2)

  Number of nodes(node leaders, HCOLL P2P) threshold in communicator to create SHArP group and use SHArP collectives

  Useful for using SHARP resources effectively.

```
$ mpirun -map-by node  -np 8  -x HCOLL_MAIN_IB=mlx5_1:1  -x HCOLL_ENABLE_SHARP=3  -x SHARP_COLL_LOG_LEVEL=3   $HPCX_MPI_TESTS_DIR/imb/IMB-MPI1 Barrier -npmin 4
[nemo01:0:872 - context.c:485] INFO job (ID: 313327617) resource request quota: ( osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[nemo01:0:872 - context.c:628] INFO tree_info: tree idx:0 quota: ( osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)
[nemo01:0:872 - comm.c:417] INFO [group#:0] group id: 1 tree idx:0 rail_idx:0 group size:8 quota: ( osts:2 user_data_per_ost:128 ) mgid: ( subnet prefix: 0xff12a01bfe800000 interface id: 0x5d590e000001 ) mlid:c005
# Barrier
[nemo01:0:872 - comm.c:417] INFO [group#:0] group id: 2 tree idx:0 rail_idx:0 group size:4 quota: ( osts:2 user_data_per_ost:128 ) mgid: ( subnet prefix: 0xff12a01bfe800000 interface id: 0x5d590e000002 ) mlid:c005
#----------------------------------------------------
# Benchmarking Barrier
# #processes = 4
# ( 4 additional processes waiting in MPI_Barrier)
#----------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         1.96         1.96         1.96
#----------------------------------------------------
# Benchmarking Barrier
# #processes = 8
#----------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         2.31         2.31         2.31


# All processes entering MPI_Finalize
```

SHARP Group size threshold cont..

- with Group size threshold

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=3 **-x HCOLL_SHARP_NP=6** -x SHARP_COLL_LOG_LEVEL=3 $HPCX_MPI_TESTS_DIR/imb/IMB-MPI1 Barrier -npmin 4

[nemo01:0:1400 - context.c:485] INFO job (ID: 1768488961) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:1400 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)

[nemo01:0:1400 - comm.c:417] INFO [group#:0] group id: 3 tree idx:0 rail_idx:0 **group size:8** quota: (osts:2 user_data_per_ost:128) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x5f590e000003) mlid:c00

```
# List of Benchmarks to run:

# Barrier
#---------------------------------------------------
# Benchmarking Barrier
# #processes = 4
# ( 4 additional processes waiting in MPI_Barrier)
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         2.53         2.53         2.53
#---------------------------------------------------
# Benchmarking Barrier
# #processes = 8
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
        1000         2.26         2.26         2.26

# All processes entering MPI_Finalize
```

SHARP Allreduce threshold

- HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX
  - Maximum allreduce size run through SHArP
  - Default : 256B
  - Increase threshold with increased quota resources.
  - Size > HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX uses default HCOLL non-sharp based algorithms
  - For range [Max_SHARP_Payload .. HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX] uses sharp fragmentation
    - Each fragment requires an OST
    - **Fragmentation performance depends on #OSTS assigned to the group**
    - Too much fragmentation may not be optimal compared to non-sharp based solution

SHARP Job Quota Resources

- Resources (per tree quota)
  - Groups
  - OSTs (Out Standing Transactions)
  - Payload for OSTs
- Options
  - SHARP_COLL_JOB_QUOTA_MAX_GROUPS
    - Maximum no.of groups(communicator) quota request. value 0 mean allocate default value.
    - #communicators
  - SHARP_COLL_JOB_QUOTA_OSTS
    - Maximum job (per tree) OST quota request. value 0 mean allocate default quota.
    - Parallelism on communicator
  - SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST
    - Maximum payload per OST quota request. value 0 mean allocate default value.
    - Maximum payload per packet
    - Max: 256B

SHARP Job Quota Resources cont ..

- Default Resources
  - Job requests 0 osts(default), 0 groups(default)
  - SHARP tree get default quota. Default: AM configurable (10 % of Max resources)
  - Minimun #OSTs/group = 2

```
$ mpirun -map-by node  -np 8  -x HCOLL_MAIN_IB=mlx5_1:1  -x HCOLL_ENABLE_SHARP=2  -x SHARP_COLL_LOG_LEVEL=3   ./osu_barrier

[nemo01:0:29912 - context.c:485] INFO job (ID: 1796276225) resource request quota: ( osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[nemo01:0:29912 - context.c:628] INFO tree_info: tree idx:0 quota: ( osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)
[nemo01:0:29912 - comm.c:417] INFO [group#:0] group id: 4 tree idx:0 rail_idx:0 group size:8 quota: ( osts:2 user_data_per_ost:128 ) mgid: ( subnet prefix: 0xff12a01bfe800000 interface id: 0x61590e000004 ) mlid:c005
# OSU MPI Barrier Latency Test v5.3.2
# Avg Latency(us)
         3.01
```

- Group quota
  - #OSTS/group = Max(2 , #OSTS/#max_groups)
  - Above example , Max (2, 51/51) = 2

SHARP Job Quota Resources cont ..

Requesting specific Resources

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 -x **SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=8 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256** ./osu_barrier

[nemo01:0:30882 - context.c:485] INFO job (ID: 1978073089) resource request quota: (osts:128 user_data_per_ost:256 max_groups:8 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:30882 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:128 user_data_per_ost:256 max_groups:8 max_qps:1 max_group_channels:1)

[nemo01:0:30882 - comm.c:417] INFO [group#:0] group id: 6 tree idx:0 rail_idx:0 group size:8 quota: (osts:16 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x65590e000006) mlid:c005

OSU MPI Barrier Latency Test v5.3.2

Avg Latency(us)

2.80

- GROUP Quota OSTS = Max (2, 128/8) = **16**

SHARP Fragmentation, Pipelining

- HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048
  - #Fragments = Message size / Payload_per_OST
  - 16 Fragments If Payload per OST is 128B
  - 8 Fragments if Payload per OST is 256B (Max payload)
- Pipelining performance depends on Group quota resources.
  - Group_quota_osts = #OSTS/#max_groups
  - Pipeline depth = Min (#fragments, groups_quota_osts)
- Group quota can be increased by
  - Increasing #OSTs
  - Decreasing #max_groups

SHARP Fragmentation, Pipelining cont..

- With Default quota

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 **-x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048** ./osu_allreduce -m :2048

[nemo01:0:5917 - context.c:485] INFO job (ID: 538443777) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:5917 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)

[nemo01:0:5917 - comm.c:417] INFO [group#:0] group id: 3 tree idx:0 rail_idx:0 group size:8 quota**: (osts:2 user_data_per_ost:128**) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x73590e000003) mlid:c005

OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	2.49
8	2.54
16	2.52
32	2.58
64	2.60
128	2.72
256	3.62
512	6.60
1024	12.70
2048	**24.36 <== #fragments=16, pipeline depth = 2**

SHARP Fragmentation, Pipelining cont ..

- With quota resource tuning (128 OSTS, 16 groups, 256B payload)

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=2 -x **SHARP_COLL_LOG_LEVEL=3 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048 -x SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=16 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256** ./osu_allreduce -m :2048

[nemo01:0:10485 - context.c:485] INFO job (ID: 371916801) resource request quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:10485 - context.c:628] INFO tree_info: tree idx:0 quota: (**osts:128 user_data_per_ost:256 max_groups:16** max_qps:1 max_group_channels:1)

[nemo01:0:10485 - comm.c:417] INFO [group#:0] group id: 8 tree idx:0 rail_idx:0 group size:8 quota: (**osts:8 user_data_per_ost:256**) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x7d590e000008) mlid:c005

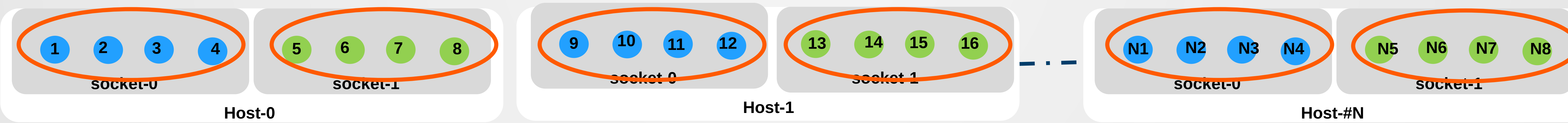
OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	2.44
8	2.54
16	2.48
32	2.53
64	2.54
128	2.74
256	3.40
512	**4.23**
1024	**5.58**
2048	**7.83 <=== #fragments =8 , Pipeline depth = 8**

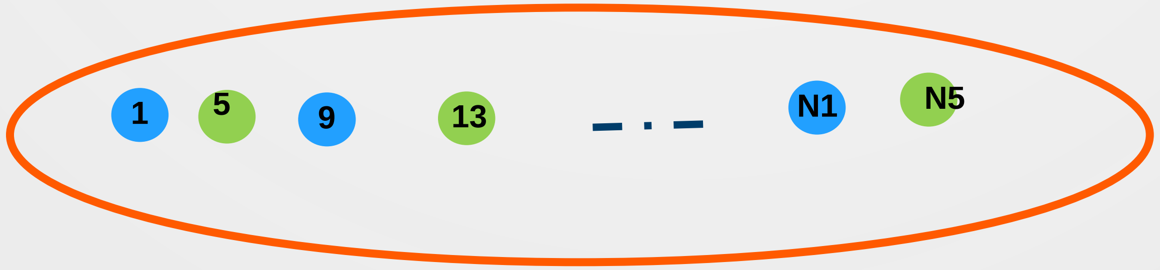
SHARP Multichannel

- w/o Multichannel
  - Only one rank per node participates in the network communicator.
  - In HCOLL, it is node leader process in communicator. i.e P2P subgroup
- With Multichannel
  - More than one rank per node participates in the network communication.
  - Best suits with HCOLL Subgrouping mechanism
  - **-x HCOLL_SBGP=basesmsocket,p2p -x HCOLL_BCOL=basesmuma,ucx_p2p**
    - Socket leaders form P2P Subgroup
  - Avoid inter-socket communication
  - Best for small messages
  - For Large messages, Shared memory channel is less overhead than network overhead

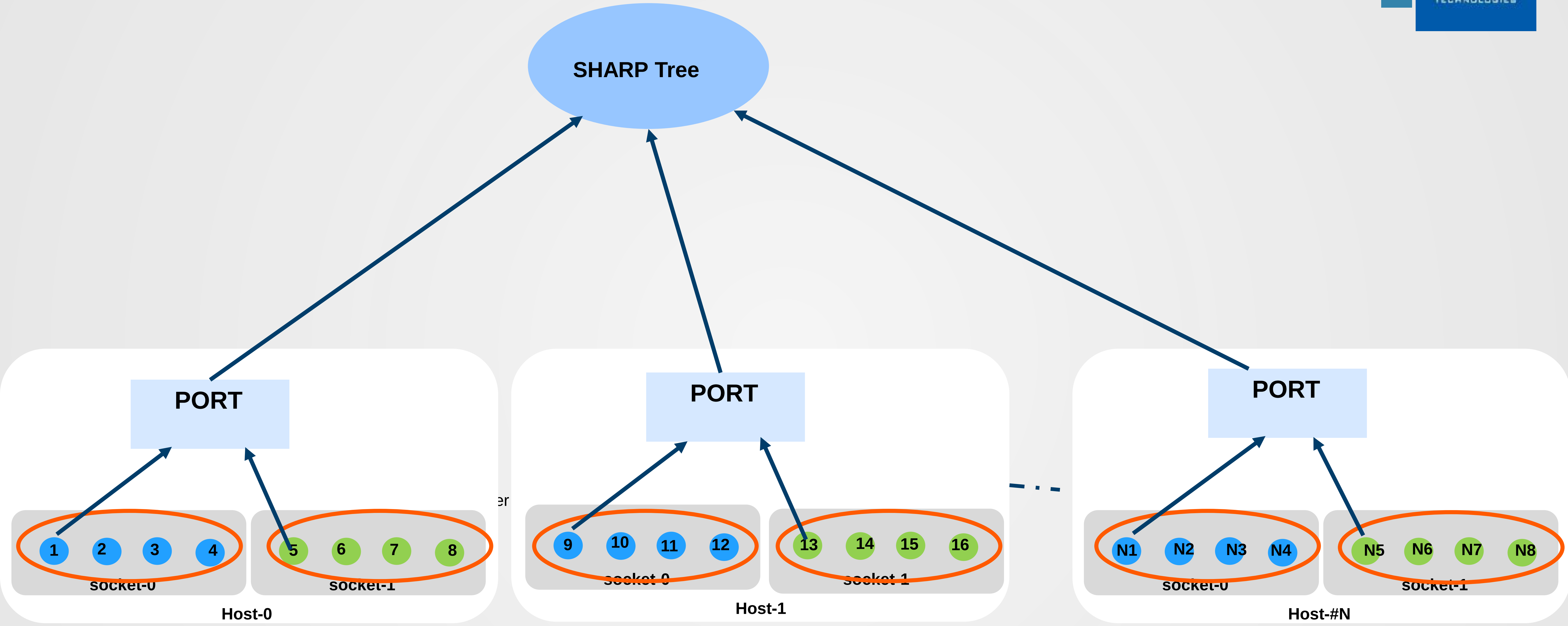
"SOCKET" Subgroup:



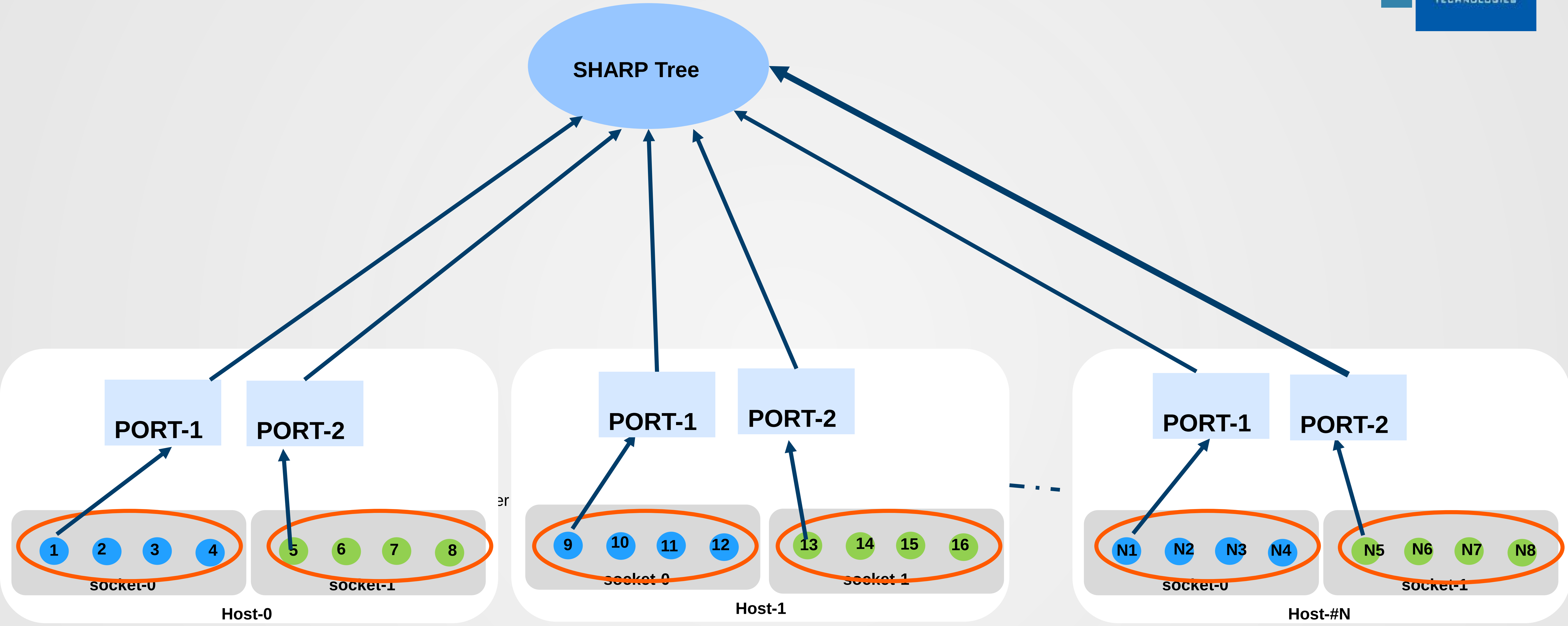
"P2P" Subgroup:



Multichannel, single port



Multichannel, two ports, single tree



SHARP Multichannel cont..

- w/o Multichannel

mpirun -map-by node -np 64 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048 -x SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=16 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256 ./osu_allreduce -m :2048

[nemo01:0:18786 - context.c:485] INFO job (ID: 1158217729) resource request quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 **max_group_channels:1,** num_trees:1)

[nemo01:0:18786 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 max_group_channels:1)

[nemo01:0:18786 - comm.c:417] INFO [group#:0] group id: 8 tree idx:0 rail_idx:0 group size:4 quota: (osts:8 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x96590e000008) mlid:c005

OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	3.58
8	3.65
16	3.61
32	3.77
64	3.81
128	4.09
256	4.74
512	5.84
1024	7.89
2048	12.05

SHARP Multichannel cont..

- with Multichannel on single port
  - Both socket leaders connected to same port

$ mpirun -map-by node -np 64 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048 -x SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=16 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256 **-x HCOLL_SBGP=basesmsocket,p2p -x HCOLL_BCOL=basesmuma,ucx_p2p** ./osu_allreduce -m :2048

[nemo01:0:20014 - context.c:485] INFO job (ID: 866254849) resource request quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 **max_group_channels:2**, num_trees:1)

[nemo01:0:20014 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 **max_group_channels:2**)

[nemo01:0:20014 - comm.c:417] INFO [group#:0] group id: c tree idx:0 rail_idx:0 group size:8 quota: (osts:8 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x9d590e00000c) mlid:c005

OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	2.97
8	2.97
16	3.00
32	3.08
64	3.12
128	3.41
256	4.08
512	5.04
1024	6.79
2048	10.17

SHARP Multichannel cont..

- with Multichannel on two ports
  - Each socket leader connected to different port
  - Both ports are connected to same SHARP tree

$ $ mpirun -map-by node -np 64 **-x HCOLL_MAIN_IB=mlx5_2:1,mlx5_1:1** -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=2048 -x SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=16 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256 -x HCOLL_SBGP=basesmsocket,p2p -x HCOLL_BCOL=basesmuma,ucx_p2p ./osu_allreduce -m :2048

[nemo01:0:21245 - context.c:485] INFO job (ID: 101646337) resource request quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 **max_group_channels:2**, num_trees:1)

[nemo01:0:21245 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 **max_group_channels[_per_port]:1**)

[nemo01:0:21245 - comm.c:417] INFO [group#:0] group id: 1 tree idx:0 rail_idx:0 group size:8 quota: (osts:8 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0xa7590e000001) mlid:c005

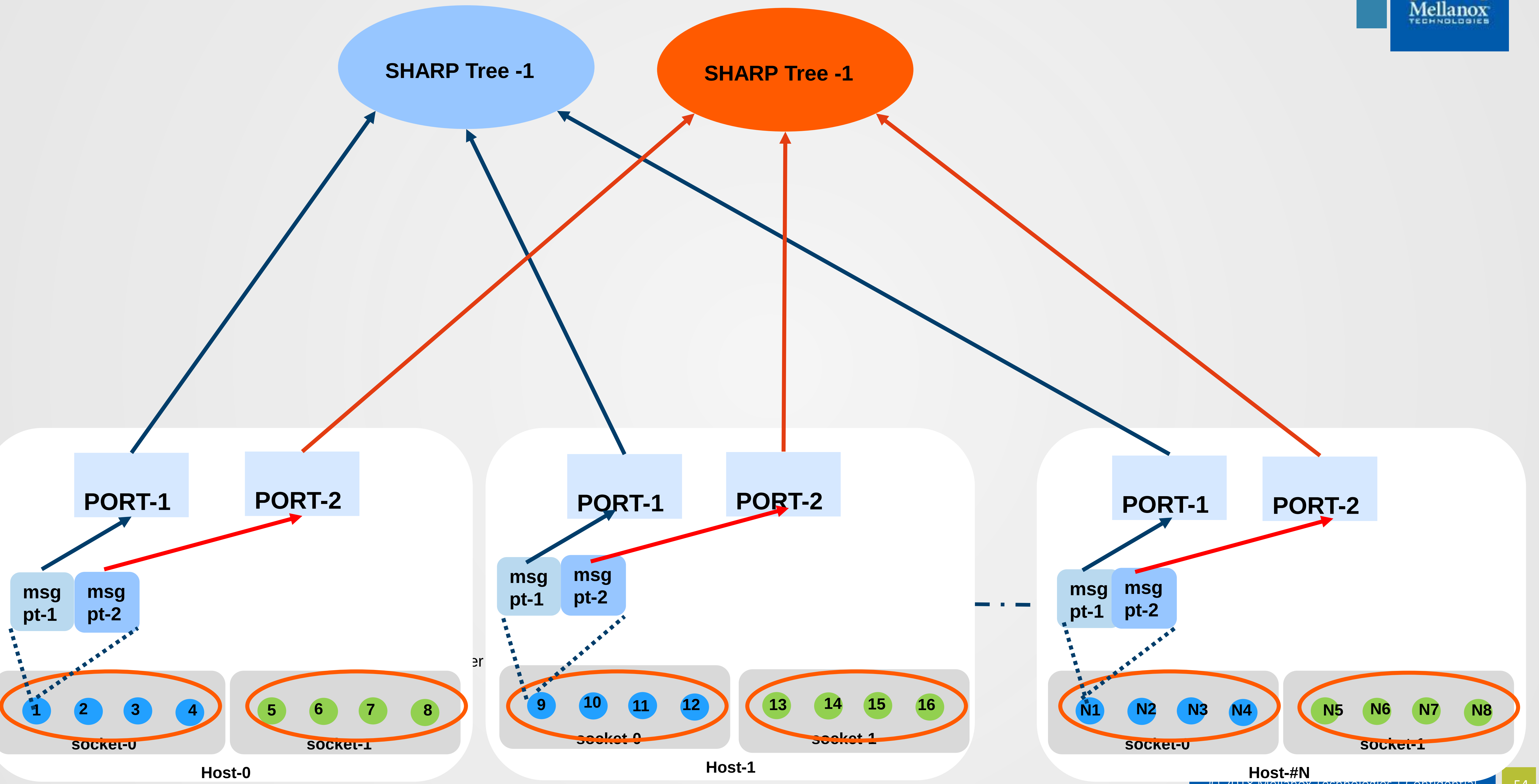
OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	3.18
8	3.69
16	3.71
32	3.92
64	3.89
128	4.18
256	4.83
512	6.09
1024	7.48
2048	10.49

SHARP Multi-rail

- Multiple SHARP trees, tree per port
- More resources
- SHARP group per tree
- Multiple SHARP groups per MPI Communicator
- For large messages
  - Stripe messages on both groups
- Small message (< OST payload size)
  - On Single group
  - Round-Robin
- SHARP_COLL_GROUPS_PER_COMM (default: 1)

Multi-Rail, 2 ports, 2 trees. 2 groups



SHARP Multi-rail cont..

$ mpirun -map-by node -np 64 -H nemo01,nemo02,nemo03,nemo04 -x HCOLL_MAIN_IB=mlx5_1:1,mlx5_2:1 -x HCOLL_ENABLE_SHARP=2 -x SHARP_COLL_LOG_LEVEL=3 -x HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX=4096 -x SHARP_COLL_JOB_QUOTA_OSTS=128 -x SHARP_COLL_JOB_QUOTA_MAX_GROUPS=16 -x SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=256 -x SHARP_COLL_GROUPS_PER_COMM=2 ./osu_allreduce -m :2048

[nemo01:0:23439 - context.c:485] INFO job (ID: 893648897) resource request quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 max_group_channels:1, **num_trees:2**)

[nemo01:0:23439 - context.c:628] **INFO tree_info: tree idx:0** quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 max_group_channels:1)

[nemo01:0:23439 - context.c:628] **INFO tree_info: tree idx:1** quota: (osts:128 user_data_per_ost:256 max_groups:16 max_qps:16 max_group_channels:1)

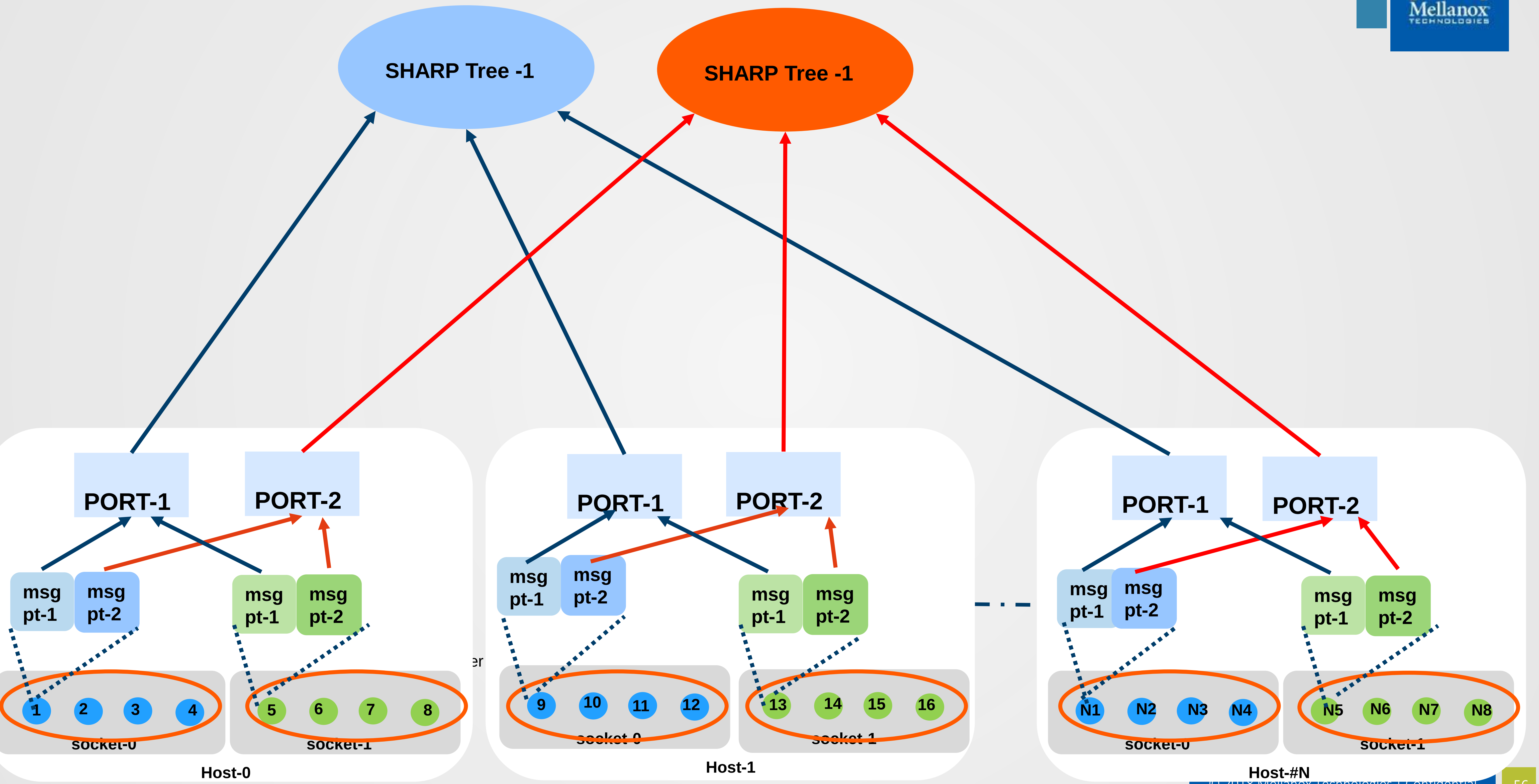
[nemo01:0:23439 - comm.c:417] INFO [**group#:0**] group id: 2 tree idx:0 rail_idx:0 group size:4 quota: (osts:8 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0xaa590e000002) mlid:c005

[nemo01:0:23439 - comm.c:417] INFO [**group#:1**] group id: 3 tree idx:1 rail_idx:1 group size:4 quota: (osts:8 user_data_per_ost:256) mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0xab590e000003) mlid:c006

OSU MPI Allreduce Latency Test v5.3.2

# Size	Avg Latency(us)
4	3.66
8	3.71
16	3.71
32	3.82
64	3.93
128	4.25
256	4.99
512	5.89
1024	8.02
2048	12.90

Multichannel-Multi-rail: 4 ports, 2 trees, 2 channels/tree



SHARP Multicast result distribution

- SHARP aggregation results from root of the sharp group tree is distributed by
  - UD Multicast
  - Reliable RC
- MPI process receives duplicates results and discards late arrival
- Requires IPoIB setup for MCAST group creation [ONLY].
- -x SHARP_COLL_ENABLE_MCAST_TARGET=1 (default enabled)

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=1 -x SHARP_COLL_LOG_LEVEL=3   ./osu_barrier

[nemo01:0:21559 - context.c:485] INFO job (ID: 327680001) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:21559 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)

[nemo01:0:21559 - comm.c:417] INFO [group#:0] group id: 7 tree idx:0 rail_idx:0 group size:8 quota: (osts:2 user_data_per_ost:128) **mgid: (subnet prefix: 0xff12a01bfe800000 interface id: 0x4c590e000007) mlid:c005**

$ mpirun -map-by node -np 8 -x HCOLL_MAIN_IB=mlx5_1:1 -x HCOLL_ENABLE_SHARP=1 -x SHARP_COLL_LOG_LEVEL=3 -x SHARP_COLL_ENABLE_MCAST_TARGET=0 ./osu_barrier

[nemo01:0:24720 - context.c:485] INFO job (ID: 122224641) resource request quota: (osts:0 user_data_per_ost:128 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)

[nemo01:0:24720 - context.c:628] INFO tree_info: tree idx:0 quota: (osts:51 user_data_per_ost:128 max_groups:51 max_qps:1 max_group_channels:1)

[nemo01:0:24720 - comm.c:417] INFO [group#:0] group id: 7 tree idx:0 rail_idx:0 group size:8 quota: (osts:2 user_data_per_ost:128) **mgid: (subnet prefix: 0x0 interface id: 0x0)** mlid:0

SHARP Config utility

- $ /opt/mellanox/sharp/bin/sharp_coll_dump_config **(-f for FULL description)**

  SHARP_COLL_LOG_LEVEL=2
  SHARP_COLL_BUFFER_POOL_SIZE=4096
  SHARP_COLL_MAX_PAYLOAD_SIZE=256
  SHARP_COLL_MAX_INLINE_SIZE=256
  SHARP_COLL_NUM_SHARP_COLL_REQ=128
  SHARP_COLL_MAX_SEND_WR=64
  SHARP_COLL_MAX_RECV_WR=256
  SHARP_COLL_MAX_RX_FILL_SIZE=256
  SHARP_COLL_RX_PREPOST_THRESHOLD=64
  SHARP_COLL_RX_PREPOST_THRESHOLD_SOFT=255
  SHARP_COLL_ENABLE_MCAST_TARGET=1
  SHARP_COLL_GROUP_RESOURCE_POLICY=1
  SHARP_COLL_USER_GROUP_QUOTA_PERCENT=0
  SHARP_COLL_GROUPS_PER_COMM=1
  SHARP_COLL_JOB_QUOTA_OSTS=0
  SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST=128
  SHARP_COLL_JOB_QUOTA_MAX_GROUPS=0
  SHARP_COLL_JOB_QUOTA_MAX_QPS_PER_PORT=0
  SHARP_COLL_JOB_NUM_TREES=0
  SHARP_COLL_CHECK_GROUP_LEAVE_ERRORS=0
  SHARP_COLL_PIPELINE_DEPTH=64
  SHARP_COLL_ENABLE_GLOBAL_ERROR_SYNC=1
  SHARP_COLL_JOB_CREATE_TIMEOUT=10000
  SHARP_COLL_JOB_CREATE_POLLING_DELAY=1
  SHARP_COLL_ERROR_CHECK_INTERVAL=180000
  SHARP_COLL_POLL_BATCH=4

UCX

HPC-X – UCX - Unified Communication – X Framework

- Open source library - Collaboration between industry, laboratories, and academy.
- Replaced MXM in HPC-X
- All of MXM’s features and more…
- $ mpirun –mca pml ucx …
- Already integrated into OpenMPI, MPICH, non MPI applications.
- $HPCX_UCX_DIR/bin/ucx_info –f

UCX Framework Mission

- Collaboration between industry, laboratories, and academia
- To create open-source production grade communication framework for data centric and HPC applications
- To enable the highest performance through co-design of software-hardware interfaces

Background

MXM

- Developed by Mellanox Technologies
- HPC communication library for InfiniBand devices and shared memory
- Primary focus: MPI, PGAS

UCCS

- Developed by ORNL, UH, UTK
- Originally based on Open MPI BTL and OPAL layers
- HPC communication library for InfiniBand, Cray Gemini/Aries, and shared memory
- Primary focus: OpenSHMEM, PGAS
- Also supports: MPI

PAMI

- Developed by IBM on BG/Q, PERCS, IB VERBS
- Network devices and shared memory
- MPI, OpenSHMEM, PGAS, CHARM++, X10
- C++ components
- Aggressive multi-threading with contexts
- Active Messages
- Non-blocking collectives with hw accleration support

UCX is an Integration of Industry and Users Design Efforts

UCX Goals

API
Exposes broad semantics that target data centric and HPC programming models and applications

Performance oriented
Optimization for low-software overheads in communication path allows near native-level performance

Production quality
Developed, maintained, tested, and used by industry and researcher community

Community driven
Collaboration between industry, laboratories, and academia

Research
The framework concepts and ideas are driven by research in academia, laboratories, and industry

Cross platform
Support for Infiniband, Cray, various shared memory (x86-64 and Power), GPUs

Co-design of Exascale Network APIs

What’s new about UCX?

- **Simple and consistent API**
- Choosing between low-level and high-level API allows easy integration with a wide range of applications and middleware.
- Protocols and transports are selected by capabilities and performance estimations, rather than hard-coded definitions.
- Support thread contexts and dedicated resources, as well as fine-grained and coarse-grained locking.
- Accelerators are represented as a transport, driven by a generic “glue” layer, which will work with all communication networks.

The UCX Framework

<u>UC-S for Services</u>

This framework provides basic infrastructure for component based programming, memory management, and useful system utilities

<u>Functionality</u>:
Platform abstractions, data structures, debug facilities.

<u>UC-T for Transport</u>

Low-level API that expose basic network operations supported by underlying hardware. Reliable, out-of-order delivery.

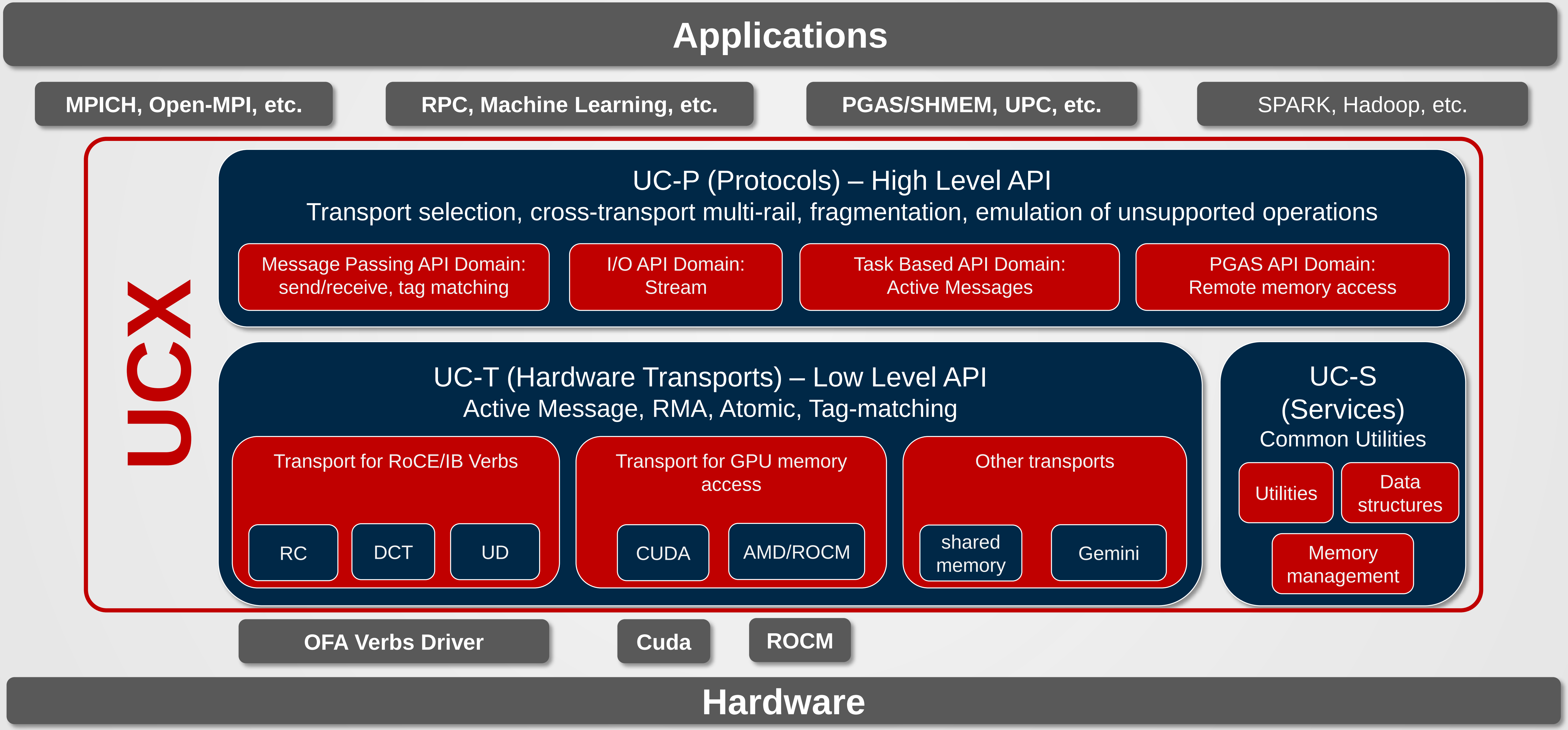
<u>Functionality</u>:
Setup and instantiation of communication operations.

<u>UC-P for Protocols</u>

High-level API uses UCT framework to construct protocols commonly found in applications

<u>Functionality</u>:
Multi-rail, device selection, pending queue, rendezvous, tag-matching, software-atomics, etc.

High-level Overview



UCX: Support matrix

- Transports:
  - Infiniband/RoCE
    - RC, DC, UD
  - Shared memory
    - Posix, SysV, knem, cma, xpmem
  - Cray/uGNI

- Platforms:
  - Linux native
  - SRIOV
  - Containers (Singularity, ..)

- Architectures:
  - x86_64
  - ARM (64-bit)
  - PowerPC

UCX: Clarifications

- UCX is not a device driver
- UCX is a framework for communications
- Close-to-hardware API layer
- Providing an access to hardware’s capabilities
- UCX relies on drivers supplied by vendors

UCX: Roadmap (1)

- Version v1.2 (October 2017)
  - Remote memory access, atomics, tag-matching
  - Connection establishment using out-of-band exchange
  - Performance and scalability tunings
  - Multi-thread support
  - IB transports: RC, UD, DC, Accelerated verbs
  - Shared memory: SysV, Posix, KNEM, CMA, XPMEM
  - Hardware tag matching
  - Out-of-order RDMA for adaptive routing

- Version v1.3 (February 2018)
  - Multi-rail protocols
  - Client/server connection establishment
  - Stream-based API for data-driven applications
  - On-demand paging with global key support
  - Low memory footprint optimization
  - Reporting connection errors

UCX: Roadmap (2)

- Version v1.4 (July 2018)
  - GPU-direct protocols support
  - Bitwise atomics
  - Optimizations for ConnectX-5

- Future versions:
  - Support extended upstream verbs API
  - Stream API with zero-copy
  - Strided data-types (UMR offload)
  - Fault tolerance and keep-alive

Management and Licensing

- Code review
  - API changes are approved by all maintainers + documentation owner
  - Maintainer per component, e.g vendor-specific transport
- License
  - BSD 3 Clause license
  - Contributor License Agreement – BSD 3 based
- Bi-weekly conference calls
- Latest release

  https://github.com/openucx/ucx/releases/tag/v1.4.0
- Mailing list

  https://elist.ornl.gov/mailman/listinfo/ucx-group
- Wiki

  https://github.com/openucx/ucx/wiki

UCX FEATURES

HPC-X – UCX - Unified Communication – X Framework

- Socket Direct

  - Brief introduction:
    - ❑ Representing one physical port as two separate HCAs.

  - Usage in UCX:
    - ❑ Enable global addressing mode:
      -x UCX_IB_ADDR_TYPE=ib_global

  - Reduces the latency to far CPU socket

  - Enable the multi-lane feature to get full wire-speed

HPC-X – UCX - Unified Communication – X Framework

SRIOV

- Brief introduction:
  - ❑ Single Root Input/Output Virtualization, useful in HPC apps launched in containers
- Usage in UCX:
  - ❑ Enable global addressing mode:
    -x UCX_IB_ADDR_TYPE=ib_global
- Hardware/software limitations and/or requirements:
  - ❑ dc and dc_x transports are not supported

HPC-X – UCX - Unified Communication – X Framework

Hardware Tag-Matching

- Brief introduction:
  - ❑ **Tag Matching ("TM") is a technology available in Mellanox ConnectX-5 HCAs that allows offloading the processing of point-to-point MPI messages from the host machine onto the network card.** TM enables zero copy of MPI messages, i.e., messages are written directly to the user's buffer without intermediate buffering and copies. It also provides a complete rendezvous progress by Mellanox devices. **Such overlap capability enables the CPU to proceed with computation while the remote data is gathered by the adapter.**

- Usage in UCX:
  - ❑ TM offload is disabled by default. To enable it either UCX_RC_TM_ENABLE or UCX_DC_TM_ENABLE should be set to 'y' (for all rc and dc transports correspondingly).

- Expected performance gain:
  - ❑ ~30% with 1k-32k message latency (osu_latency) (2 processes)
  - ❑ Orders of magnitude with mpi_overhead, IMB-NBC (benchmarks which estimate MPI CPU utilization)
  - ❑ ~3% with UMT and ~21% with AMG applications with 512 processes.

HPC-X – UCX - Unified Communication – X Framework

Hardware Tag-Matching

- Known issues and/or trouble shooting tips:
  - ❑ Will not show performance improvement for all applications and in some cases, there may be degradations.
  - ❑ There are several TM offload related variables that may be helpful in tuning a particular application that does not show any improvement with offload enabled:
    - UCX_TM_THRESH: This variable defines the threshold for using TM offload. Messages smaller than this value will be handled in SW. The default value is 1024b, because using TM offload implies noticeable performance overhead (which is better to avoid with small messages).
- Hardware/software limitations and/or requirements:
  - ❑ HW TM is supported with IB RC and DC transports starting from ConnectX-5.
  - ❑ MLNX_OFED_LINUX-4.1-4.1.1.0 or higher.

More information about HW TM:
https://wikinox.mellanox.com/display/MAR/Tag+Matching

HPC-X – UCX - Unified Communication – X Framework

- Adaptive Routing

  - Brief introduction:
    - ❑ Enable out-of-order RDMA data placement. Disabled by default.
  - Usage in UCX:
    - ❑ -x UCX_IB_OOO_RW=y
  - Improves bandwidth performance for the Random-Ring benchmark
  - Hardware/software limitations and/or requirements:
    - ❑ Configure OpenSM and the fabric with AR support
    - ❑ ConnectX-5 and above

HPC-X – UCX - Unified Communication – X Framework

RoCE and RoCE v2

- Brief introduction:
  - ❑ UCX supports RoCE out-of-box. The user may want to specify the HCA to use with the Ethernet port with
    -x UCX_NET_DEVICES=mlx5_0:1
- Usage in UCX:
  - ❑ -x UCX_IB_TRAFFIC_CLASS=106
  - ❑ RoCE v2: -x UCX_IB_GID_INDEX=1 (use show_gids)
- Additional relevant information:
  - ❑ Not supported on dc and dc_x
- Hardware/software limitations and/or requirements:
  - ❑ Lossless fabric needs to be configured - https://community.mellanox.com/docs/DOC-2855

HPC-X – UCX - Unified Communication – X Framework

Multi-Rail

- Brief introduction:
  - ❑ Multi-Rail feature allows to use up to 3 ports (lanes) on the host to improve communication bandwidth

- Usage in UCX:
  - ❑ Set variables:

    UCX_MAX_RNDV_LANES=<number> (large messages)

    UCX_MAX_EAGER_LANES=<number> (small messages)

- Expected performance gain:
  - ❑ Simple performance benchmarks (osu_bw) demonstrated close to linear bandwidth growing on ConnectX-5 devices on the rc_x transport

HPC-X – UCX - Unified Communication – X Framework

Multi-Rail

- Hardware/software limitations and/or requirements:
  - Limited by 3 ports
- Additional relevant information:
  - This optimization is relevant for large messages (RNDV proto), on small/mid size messages, the optimization has much less effect.

HPC-X – UCX - Unified Communication – X Framework

MEMIC – Memory In Chip

- Brief introduction:
  - ❑ Usage of on-device memory to send messages
- Usage in UCX (Command line and results) :
  - ❑ No special actions required, enabled by default
- Expected performance gain:
  - ❑ Benchmarks are demonstrated 0.2 usec improvements on ConnectX-5 for short (up to 2k) messages
- Hardware/software limitations and/or requirements:
  - ❑ Works on ConnectX-5 rc_x/dc_x transports only
- Additional relevant information:
  - ❑on-board memory which is shared across all launched processes
  - ❑ If there isn't' enough memory – fallback mode will be enabled automatically
  - ❑ Every UCX worker (thread) will try to allocate own portion of memory

CUDA-Aware UCX

- Standard UCX, MPI interface for data movement from GPU
- High performance RDMA based point-to-point communication
- Off-loaded data movement from GPU with GPUDirect RDMA technologies

UCX-CUDA: features

- GPU Direct RDMA
- Protocols using GDR COPY
- CUDA IPC
- Pipelining
- CUDA support in STREAM API

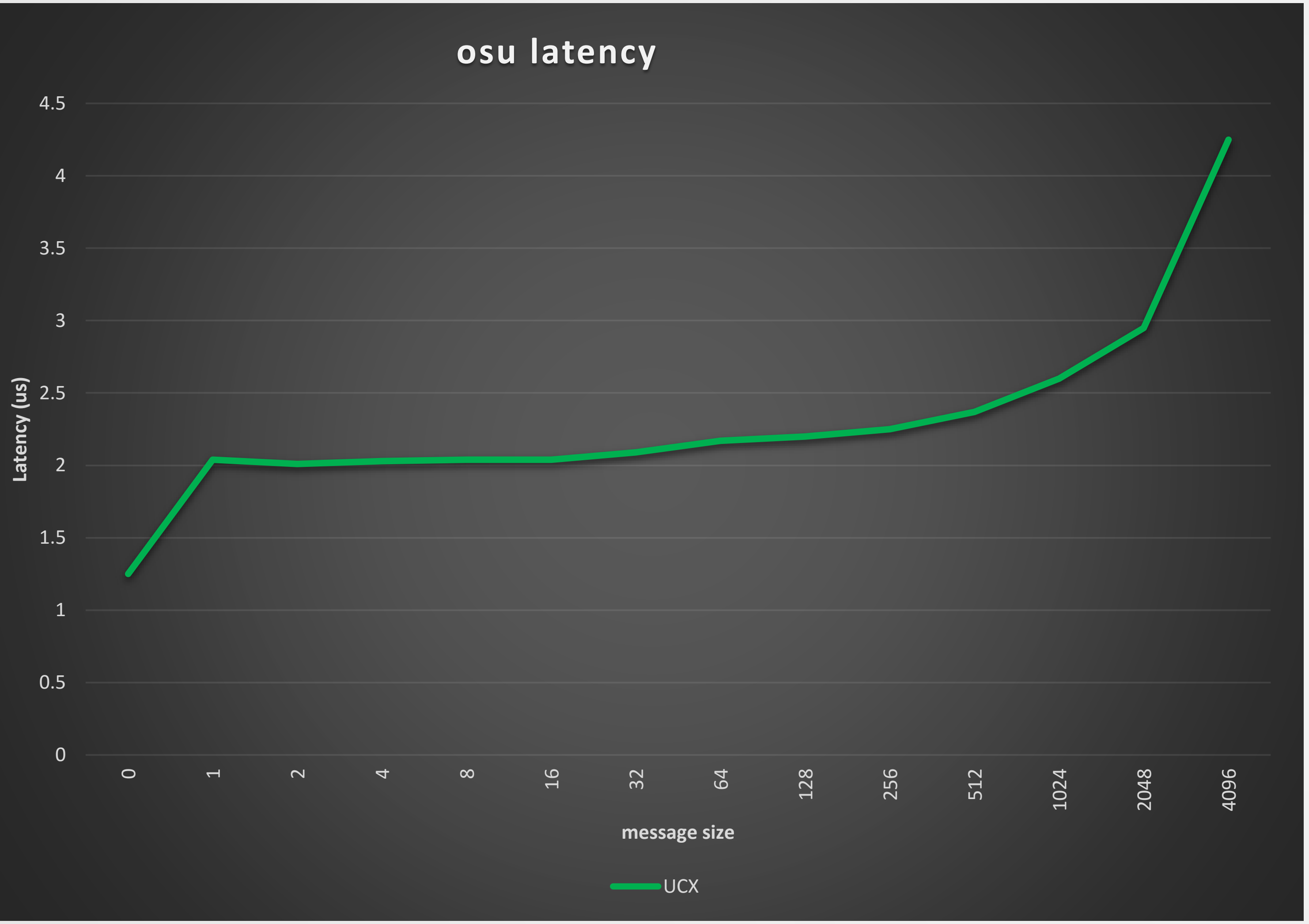
System Requirements

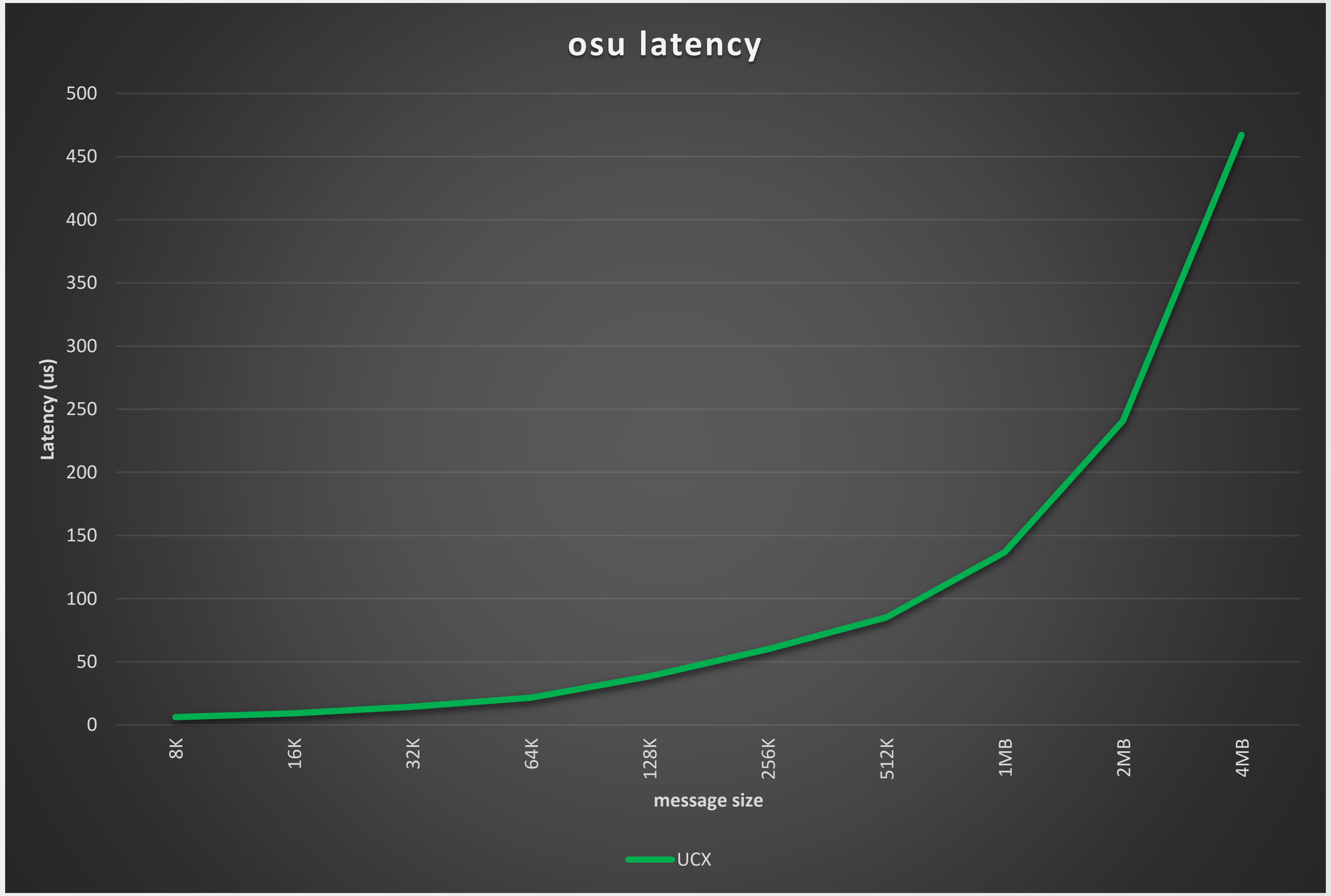
- **CUDA v8.0 or higher - refer to NVIDIA documents for CUDA Toolkit cuda installation**
- **Mellanox OFED GPUDirect RDMA plugin module**
  - Mellanox OFED - refer to MLNX_OFED webpage
  - GPUDirect RDMA - refer to Mellanox OFED GPUDirect RDMA webpage
  - Once the NVIDIA software components are installed, it is important to verify that the GPUDirect RDMA kernel module is properly loaded on each of the compute systems where you plan to run the job that requires the GPUDirect RDMA.
    - To check whether the GPUDirect RDMA module is loaded, run:
      - **service nv_peer_mem {status/start/stop/restart}**
    - To run this verification on other Linux flavors
      - **lsmod | grep nv_peer_mem**
- **GDR COPY plugin module**
  - GDR COPY is a fast copy library from NVIDIA, used to transfer between HOST and GPU. For information on how to install GDR COPY, refer to its GitHub webpage
  - Once GDR COPY is installed, it is important to verify that the gdrcopy kernel module is properly loaded on each of the compute systems where you plan to run the job that requires the GDR COPY.
    - /etc/init.d/gdrcopy {start|stop|restart}
  - To check whether the GDR COPY module is loaded, run:
    - **lsmod | grep gdrdr**

Installing UCX-CUDA

- UCX can be downloaded from
  - https://github.com/openucx/ucx
- Prerequisites
  - CUDA
  - OFED
  - GPUDirectRDMA
  - GDR Copy
- Supported Architectures
  - CPU architecture: x86 , power pc
  - NVIDIA GPU architectures: Tesla, Kepler, ascal, Volta
- Configure
  - --with-cuda --with-gdrcopy
- Runtime flags
  - -x UCX_TLS=rc_x,cuda_copy,gdr_copy (UCX v1.3, no CUDA -IPC)
  - OOB (UCXv1.4)
- Also distributed with HPC-X
  - hpcx-cuda module

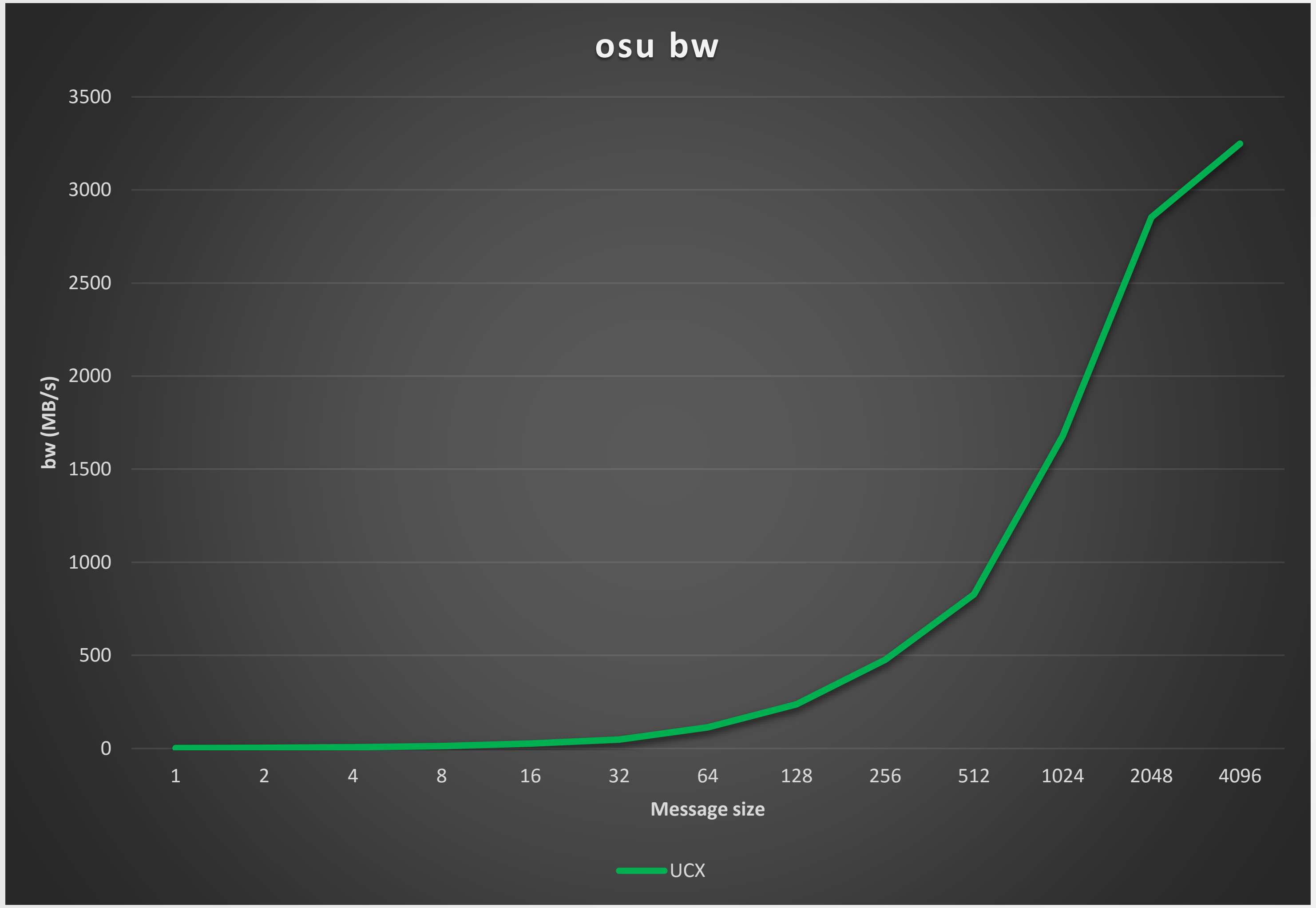
Performance of ucx-cuda: latency

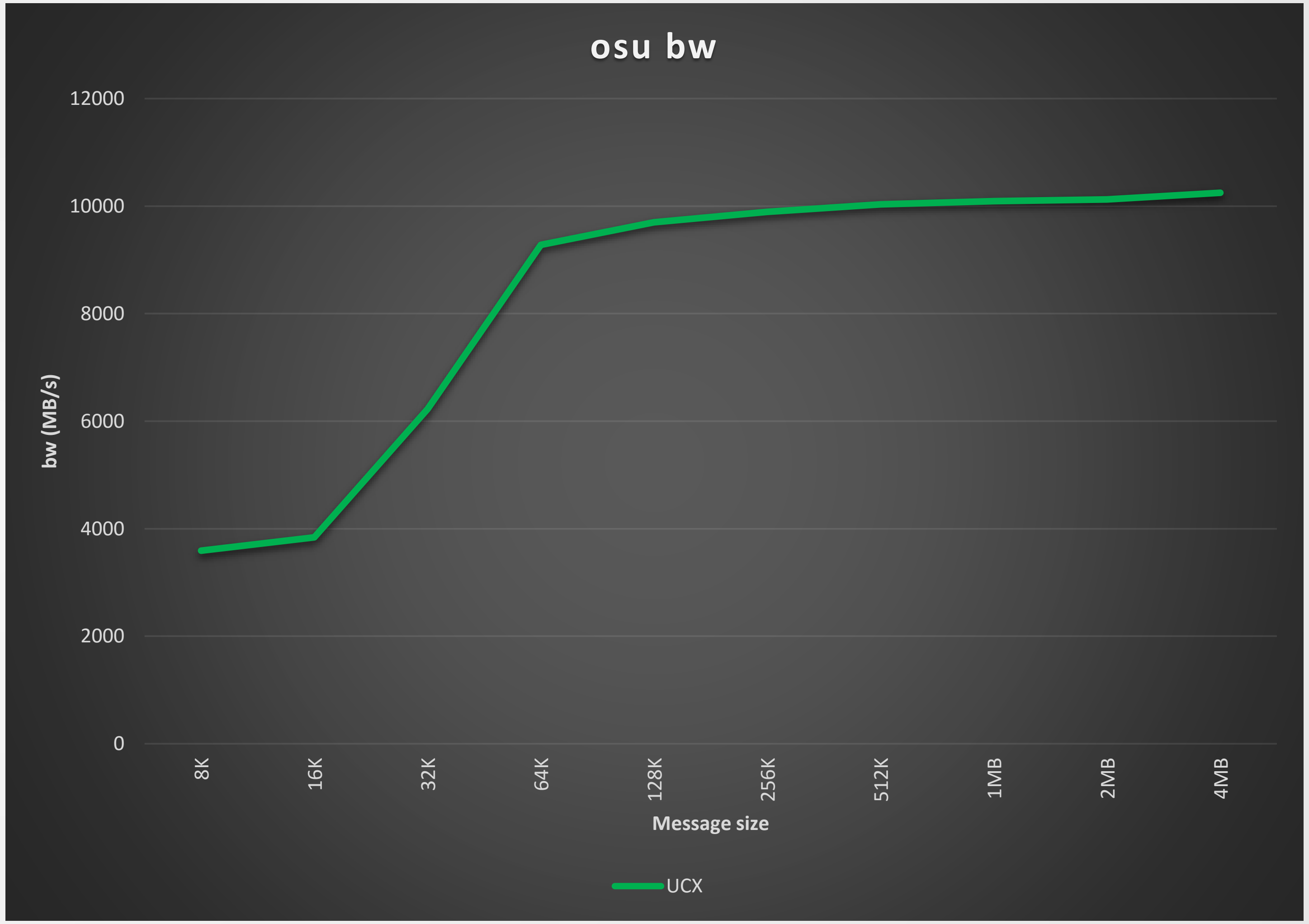




- Setup:
  - CPU: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz GPU: Tesla P100
  - * UCX: master, ompi-3.x

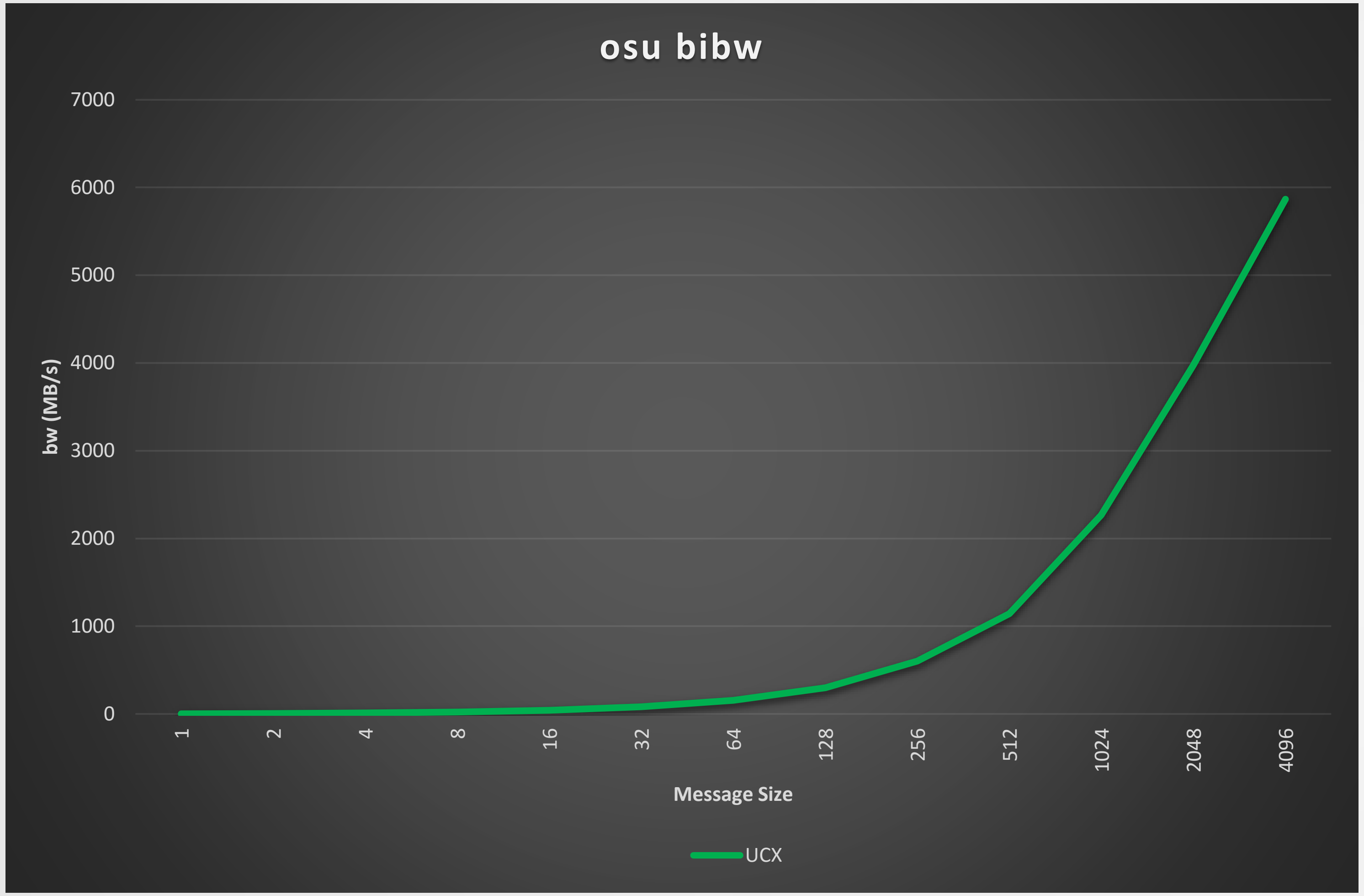
Performance of ucx-cuda: Bandwidth

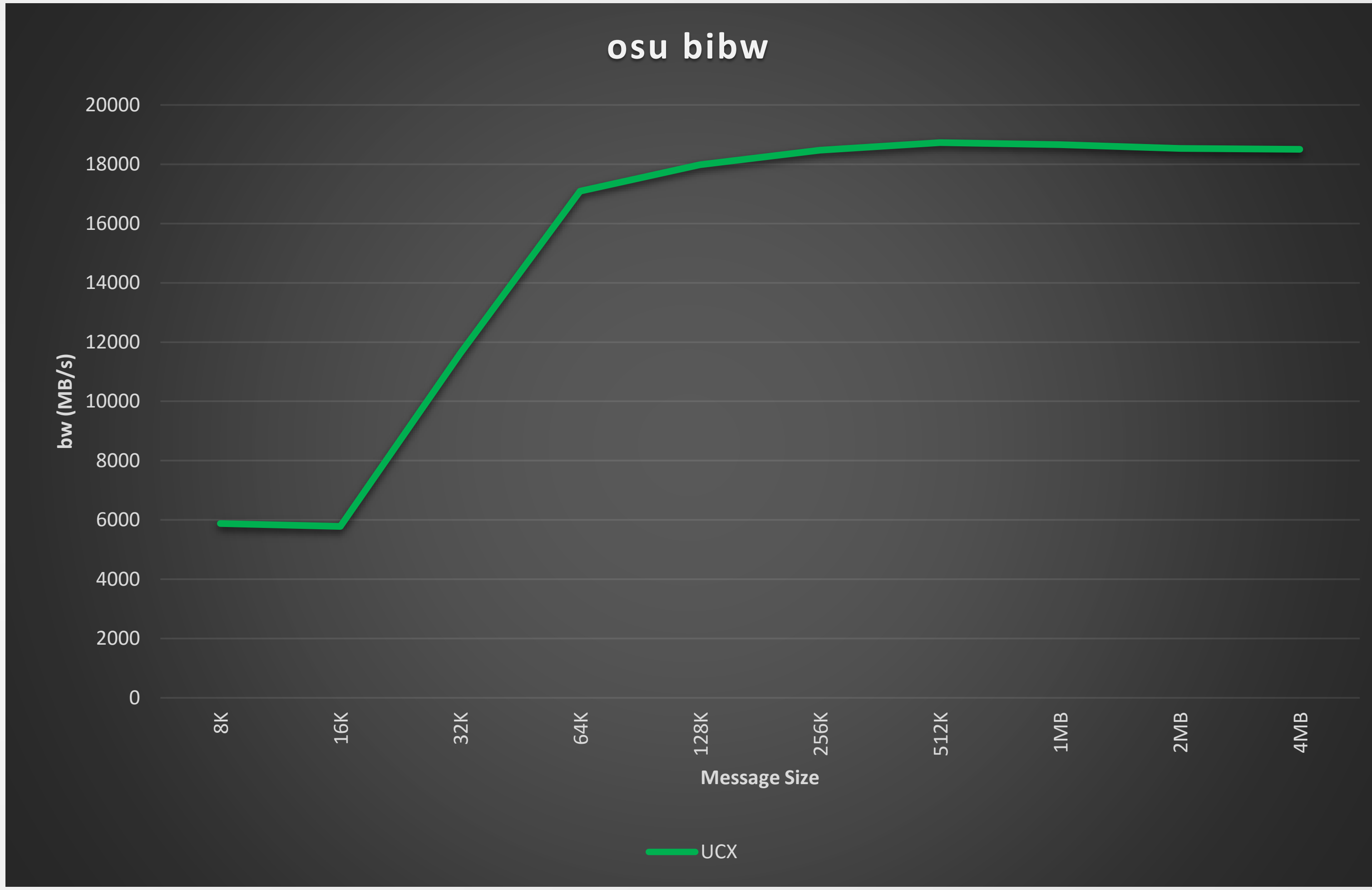




- Setup:
  - CPU: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz GPU: Tesla P100
  - * UCX: master, ompi-3.x

Performance of ucx-cuda: bi-directional bandwidth





- Setup:
  - CPU: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz GPU: Tesla P100
  - * UCX: master, ompi-3.x

UCX API Overview

UCP (Protocol Layer)

- Combine transports, devices, and operations, for optimal performance
  - Query transport capabilities and performance estimations
  - Select devices/transports according to reachability
  - Select best protocols for each data transfer type
  - For example: eager vs. rendezvous, bcopy vs. zcopy
- Unified transport infrastructure
  - Fragmentation (transport has limited MTU)
  - Emulate unsupported operations
  - Expose one-sided connection establishment
- Aggregate multiple devices to single logical connection
- Software tag matching

UCP contexts

- ucp_context_h

  Top-level context for the application.

- ucp_worker_h

  Communication resources and progress engine context. A possible usage is to create one worker per thread or per CPU core.

- ucp_ep_h

  Connection to a remote worker, used to send data to remote peer. Contains handles for all transport-level connections in use to the remote peer.

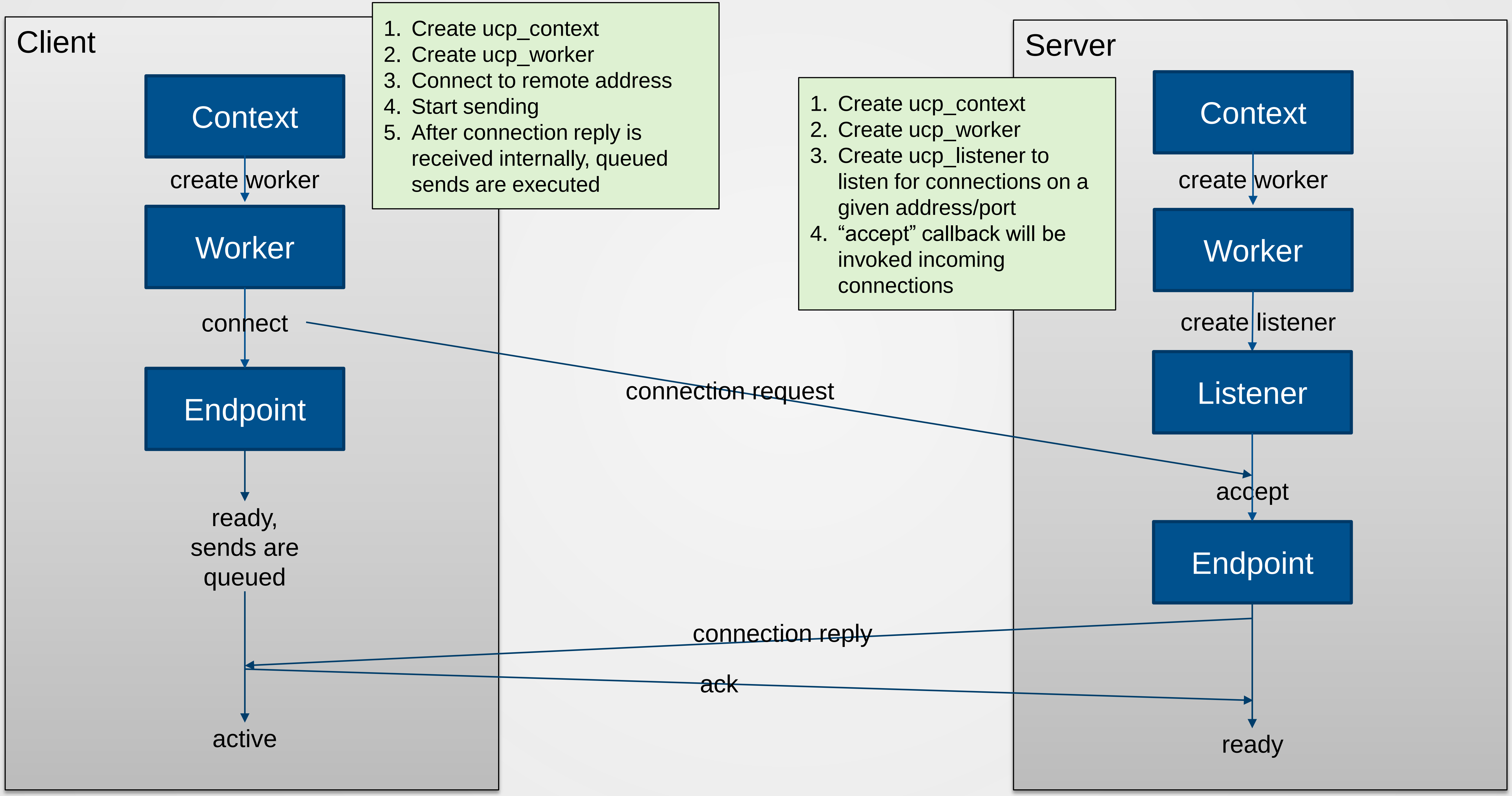
- ucp_mem_h

  Handle to memory allocated or registered in the local process. Contains an array of uct_mem_h's for currently active transports.

- ucp_rkey_h

  Remote memory handle. Allows access to remote memory for one-sided operations and atomics.

UCP connection establishment – client/server model



1. Create ucp_context
2. Create ucp_worker
3. Connect to remote address
4. Start sending
5. After connection reply is received internally, queued sends are executed

1. Create ucp_context
2. Create ucp_worker
3. Create ucp_listener to listen for connections on a given address/port
4. “accept” callback will be invoked incoming connections

UCP API – stream-based send/receive

Use cases:

- Drop-in replacement for TCP sockets use cases (cloud, storage, …)
- In-place processing of streaming data

- Non-blocking send:

```
ucs_status_ptr_t ucp_stream_send_nb(ucp_ep_h ep, const void *buffer, size_t count,
                                    ucp_datatype_t datatype, ucp_send_callback_t cb,
                                    unsigned flags)
```

- Non-blocking receive:

```
ucs_status_ptr_t ucp_stream_recv_nb(ucp_ep_h ep, void *buffer,
                                    size_t count, ucp_datatype_t datatype,
                                    ucp_stream_recv_callback_t cb,
                                    size_t *length, unsigned flags)
```

- Fetch next data fragment from stream:

```
ucs_status_ptr_t ucp_stream_recv_data_nb(ucp_ep_h ep, size_t *length)
```

UCP API – tag-based send/receive

Use cases:

- Implementing MPI-1 standard (OpenMPI, MPICH)

- Non-blocking send:

```
ucs_status_ptr_t ucp_tag_send_nb(ucp_ep_h ep, const void *buffer,
                                 size_t count, ucp_datatype_t datatype,
                                 ucp_tag_t tag, ucp_send_callback_t cb)
```

- Non-blocking receive:

```
ucs_status_ptr_t ucp_tag_recv_nb(ucp_worker_h worker, void *buffer,
                                 size_t count, ucp_datatype_t datatype,
                                 ucp_tag_t tag, ucp_tag_t tag_mask,
                                 ucp_tag_recv_callback_t cb)
```

UCP API – remote memory access

Use cases:

- Implementing MPI-RMA operations (OpenMPI, MPICH)
- Implementing PGAS/SHMEM stack (OpenSHMEM)
- Bulk data transfer after synchronizing with tagged control messages

Write to remote memory:

```
ucs_status_t ucp_put_nb(ucp_ep_h ep, const void *buffer, size_t length,
                        uint64_t remote_addr, ucp_rkey_h rkey,
                        ucp_send_callback_t cb)
```

- Implicit non-blocking variant: **`ucp_put_nbi`**

Read from remote memory:

```
ucs_status_t ucp_get_nb(ucp_ep_h ep, void *buffer, size_t length,
                        uint64_t remote_addr, ucp_rkey_h rkey,
                        ucp_send_callback_t cb)
```

- Implicit non-blocking variant: **`ucp_get_nbi`**

UCP API – Atomic operations

Use cases:

- MPI-RMA
- PGAS/SHMEM

- Perform atomic operation on remote memory:

```
ucs_status_t ucp_atomic_fetch_nb(ucp_ep_h ep, ucp_atomic_fetch_op_t opcode,
                                 uint64_t value, void *result, size_t op_size,
                                 uint64_t remote_addr, ucp_rkey_h rkey,
                                 ucp_send_callback_t cb)
```

  - Operations:
    - Fetch-and-add
    - Swap
    - Compare-and-swap
    - Bitwise: AND, OR, XOR
  - Size: 32 or 64 bit

UCP API – request management

- Check request status:

```
ucs_status_t ucp_request_test(void *request, ucp_tag_recv_info_t *info)
struct ucp_tag_recv_info {
    ucp_tag_t          sender_tag;
    size_t             length;
}
```

- Release completed request:

```
void ucp_request_free(void *request)
```

  - Must be called for every request to release it back to UCP

UCP API – progress

- **Progress communications on a worker:**

```
void ucp_worker_progress(ucp_worker_h worker)
```

  - Only API function which progresses communications
  - Calls non-blocking request callbacks
  - Must avoid calling it recursively

- **Wait for communications without consuming CPU:**

```
ucs_status_t ucp_worker_wait(ucp_worker_h worker)
```

  - Returns when something may have happened on the worker
  - Need to call `ucp_worker_progress` after `ucp_worker_wait` returns.

- **Get file descriptor for event loop:**

```
ucs_status_t ucp_worker_get_efd(ucp_worker_h worker, int *fd)
```

  - The file descriptor would be signaled in case of any event on the worker
  - Could be used in select/poll/epoll_wait

UCP API - datatypes

- Describe the memory layout of data buffers, to avoid extra memory movements and use HW offloads when possible

- Contiguous datatype:

```
ucp_dt_make_contig(_elem_size)
```

  - Sequence of elements of fixed size

- Scatter-gather list (IOV):

```
typedef struct ucp_dt_iov {
    void    *buffer;
    size_t  length;
} ucp_dt_iov_t;

ucp_dt_make_iov()
```

- Generic type – user provided pack/unpack callbacks

```
ucs_status_t ucp_dt_create_generic(const ucp_generic_dt_ops_t *ops, void *context,
                                   ucp_datatype_t *datatype_p)
```

Usage example (1) - init

(Full example code in: test/examples/ucp_client_server.c)

```
/* Create UCP context */
ucp_context_h ucp_context;
ucp_params_t ucp_params;

ucp_params.field_mask   = UCP_PARAM_FIELD_FEATURES;
ucp_params.features     = UCP_FEATURE_STREAM;
ucp_init(&ucp_params, NULL, &ucp_context);

/* Create UCP worker */
ucp_worker_h ucp_worker;
ucp_worker_params_t worker_params;

worker_params.field_mask  = UCP_WORKER_PARAM_FIELD_THREAD_MODE;
worker_params.thread_mode = UCS_THREAD_MODE_SINGLE;
ucp_worker_create(ucp_context, &worker_params, &ucp_worker);
```


Usage example (2) - connect

```
struct sockaddr_in connect_addr;

/* Initialize destination address */
connect_addr.sin_family     = AF_INET;
connect_addr.sin_addr       = inet_addr("1.2.3.4");
connect_addr.sin_port       = htons(1234);

/* Create UCP endpoint */
ucp_ep_params_t ep_params;
ucp_ep_h ucp_ep;

ep_params.field_mask        = UCP_EP_PARAM_FIELD_FLAGS |
                              UCP_EP_PARAM_FIELD_SOCK_ADDR;
ep_params.flags             = UCP_EP_PARAMS_FLAGS_CLIENT_SERVER;
ep_params.sockaddr.addr     = &connect_addr;
ep_params.sockaddr.addrlen = sizeof(connect_addr);
ucp_ep_create(ucp_worker, &ep_params, &ucp_ep);
```

Usage example (3) - send

```
size_t message_length;
char *message;
void *request;

/* Send stream data */
request = ucp_stream_send_nb(ucp_ep, message, message_length,
                             ucp_dt_make_contig(sizeof(char)),
                             send_completion_callback, 0 /* flags */);
if (UCS_PTR_IS_ERR(request)) {
    fprintf(stderr, "unable to send: %s\n“,
            ucs_status_string(UCS_PTR_STATUS(request));
} else if (UCS_PTR_STATUS(request) == UCS_OK) {
    /* Completed */
} else {
    /* Uncompleted, need to wait for it */
    while (ucp_request_test(request, NULL) == UCS_INPROGRESS) {
        ucp_worker_progress(ucp_worker);
    }
    ucp_request_release(request);
}
```

Usage example (4) - receive

```
size_t message_length, recv_length;
char *message;
void *request;

/* Receive stream data into a buffer */
request = ucp_stream_recv_nb(ucp_worker, message, message_length,
                             ucp_dt_make_contig(sizeof(char)),
                             recv_completion_callback, &recv_length, 0 /* flags */);
if (UCS_PTR_IS_ERR(request)) {
    fprintf(stderr, "unable to receive: %s\n",
            ucs_status_string(UCS_PTR_STATUS(request)));
} else if (UCS_PTR_STATUS(request) == UCS_OK) {
    /* Completed */
} else {
    /* Uncompleted, need to wait for it */
    while (ucp_request_test(request, NULL) == UCS_INPROGRESS) {
        ucp_worker_progress(ucp_worker);
    }
    ucp_request_release(request);
}
```

Thank You