MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

# How to Boost the Performance of Your MPI and PGAS Applications with MVAPICH2 Libraries

## A Tutorial at the

## MVAPICH User Group (MUG) Meeting ’18

by

**The MVAPICH Team**

The Ohio State University

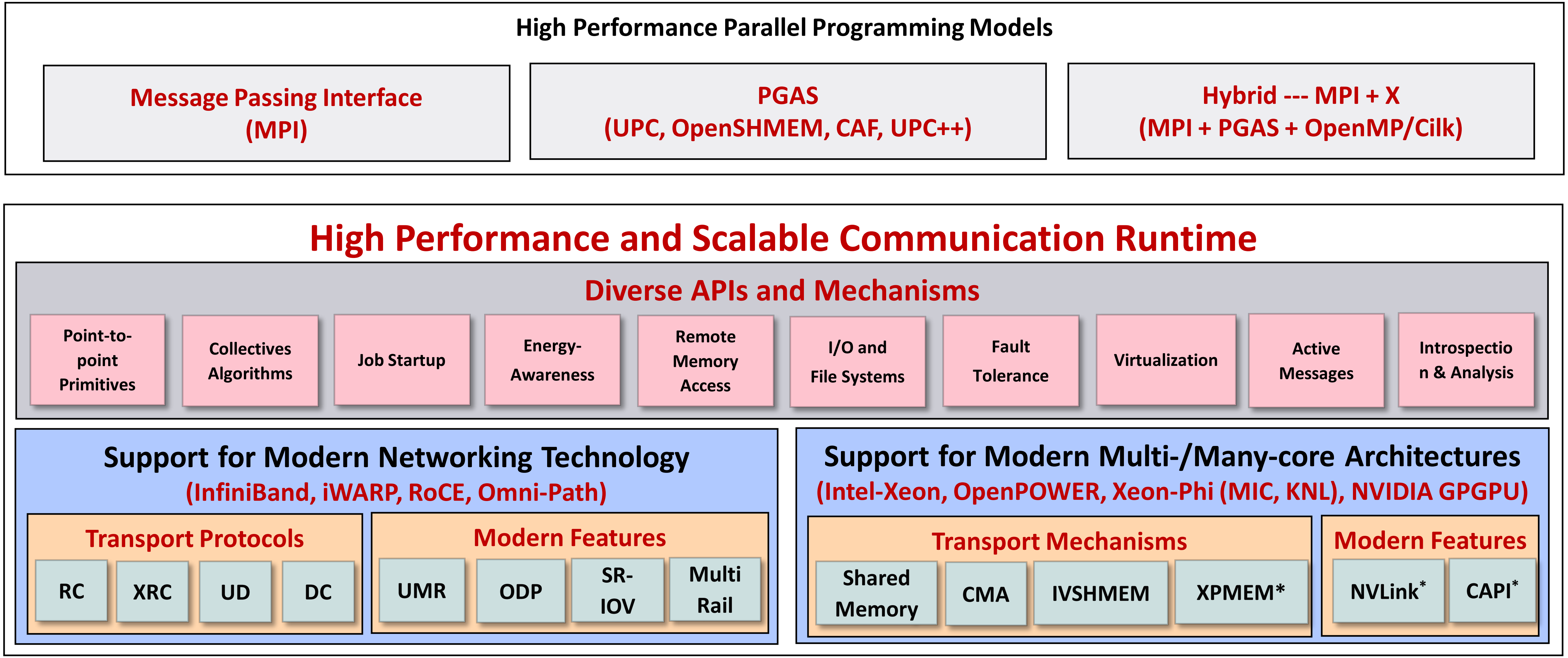
E-mail: panda@cse.ohio-state.edu

http://mvapich.cse.ohio-state.edu/

# Designing (MPI+X) for Exascale

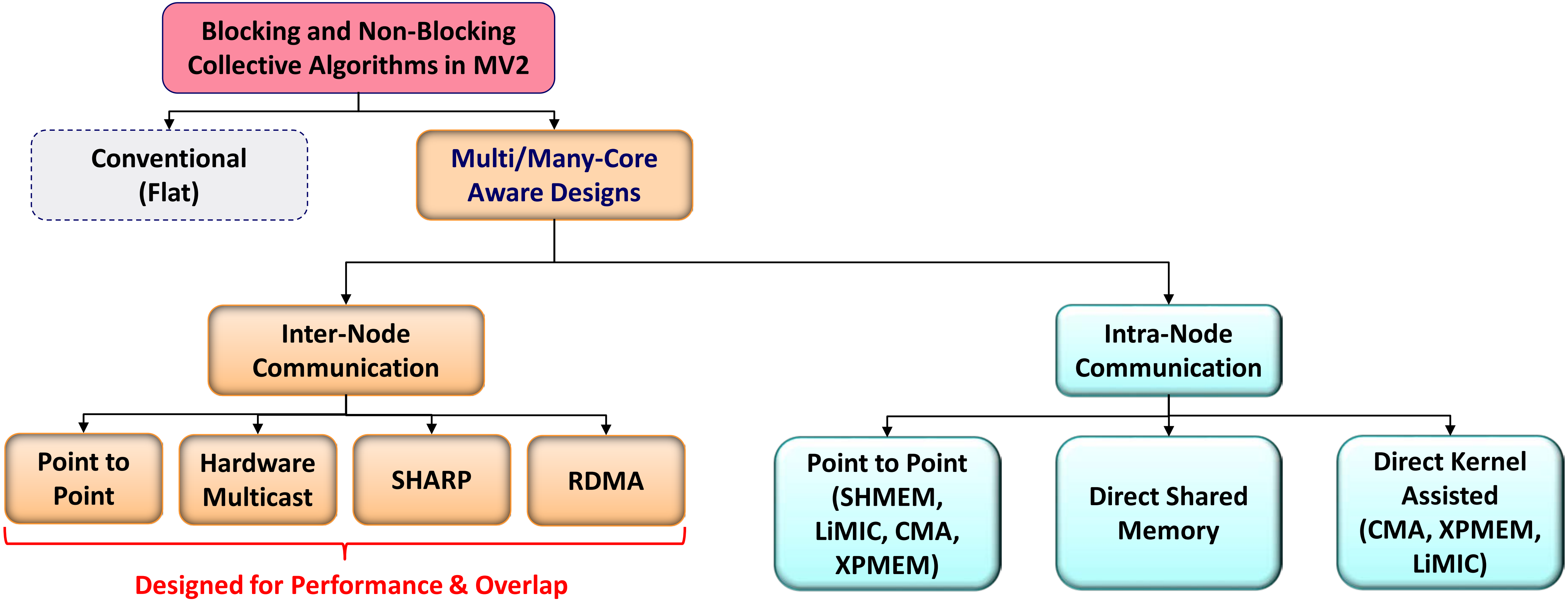
- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
- Scalable Collective communication
  - Offloaded
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
  - Multiple end-points per node
- Support for efficient multi-threading
- Integrated Support for GPGPUs and Accelerators
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming
  - MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, MPI + UPC++…
- Virtualization
- Energy-Awareness

# Architecture of MVAPICH2 Software Family




* Upcoming

# Collective Communication in MVAPICH2




Run-time flags:

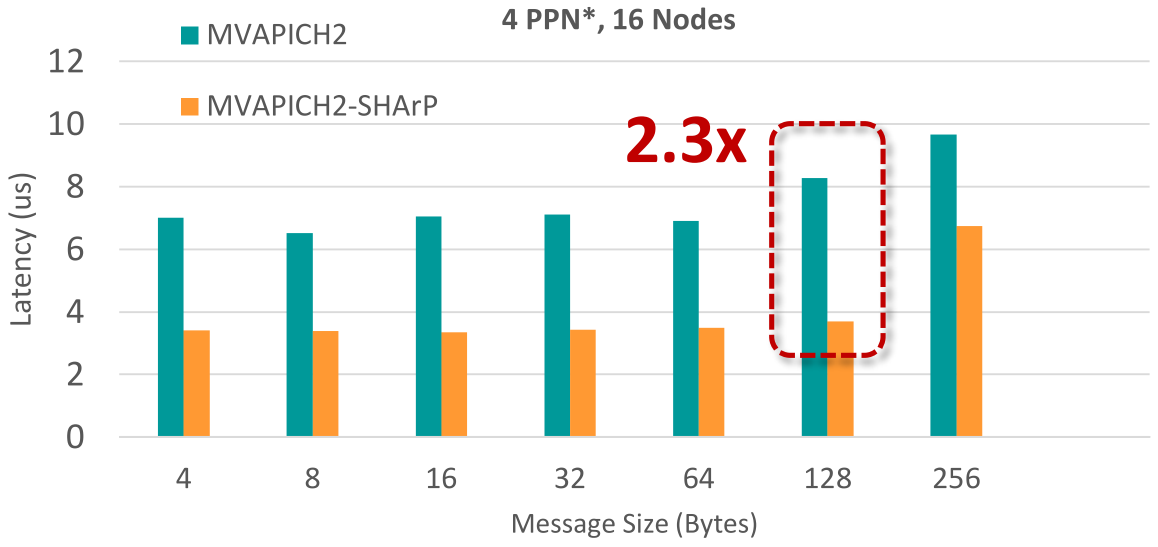
All shared-memory based collectives : MV2_USE_SHMEM_COLL (Default: ON)

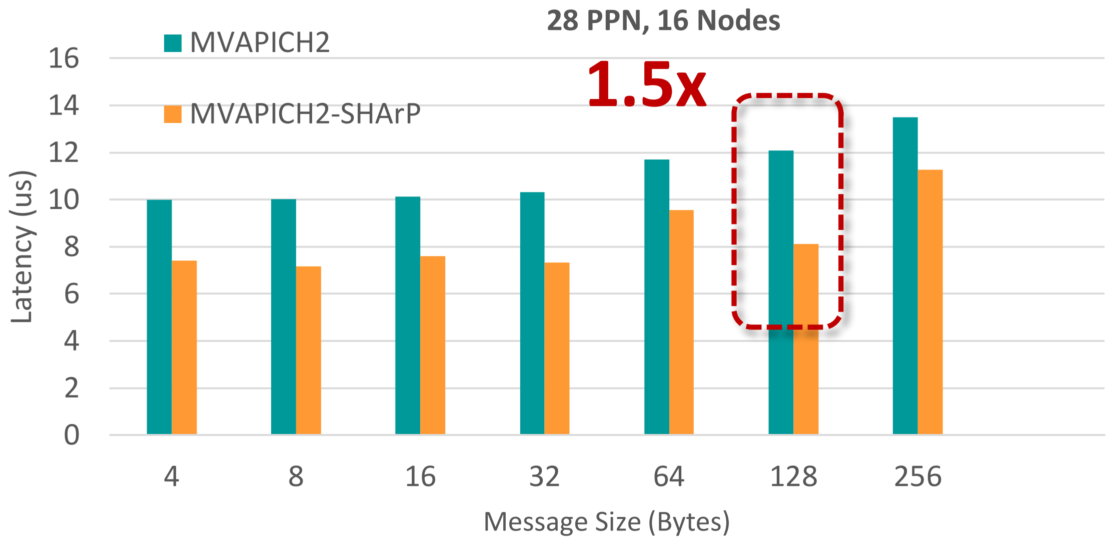
Hardware Mcast-based collectives : MV2_USE_MCAST (Default : OFF)

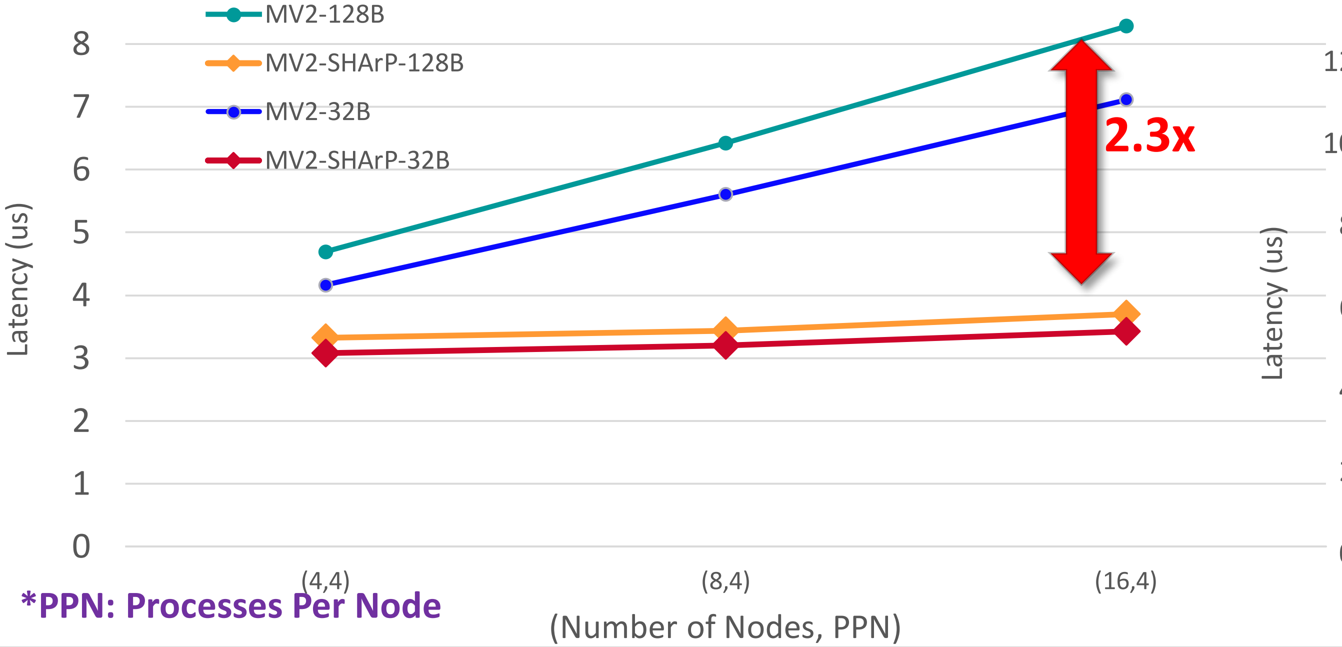
CMA-based collectives : MV2_USE_CMA_COLL (Default : ON)

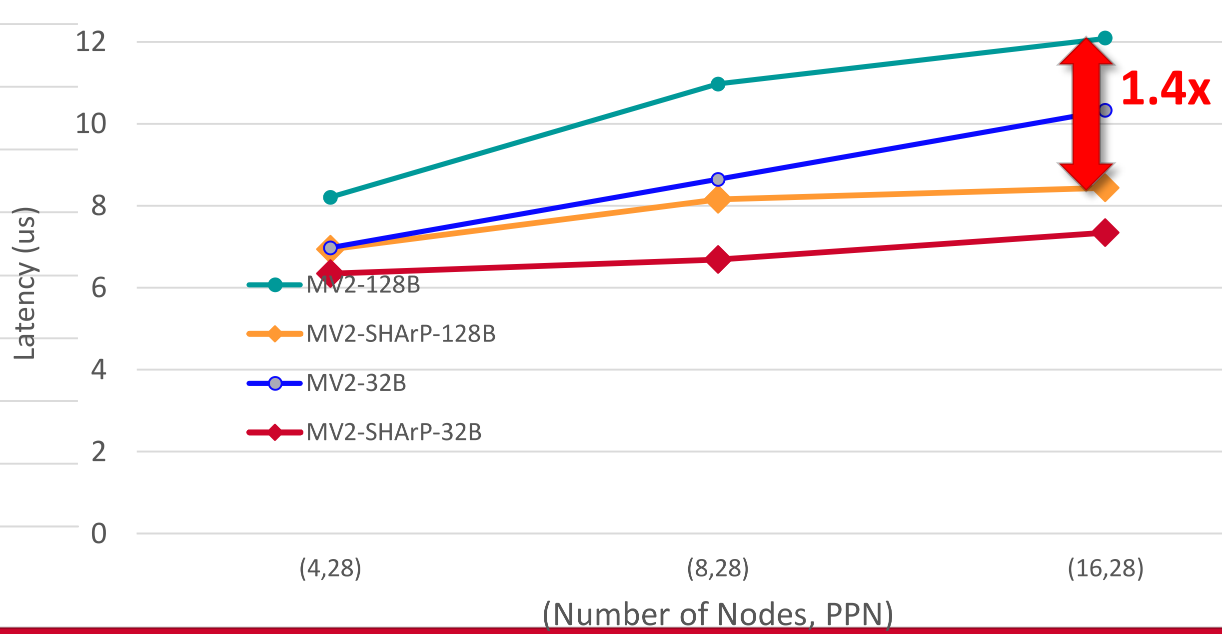
# Advanced Allreduce Collective Designs Using SHArP

## osu_allreduce (OSU Micro Benchmark) using MVAPICH2 2.3b



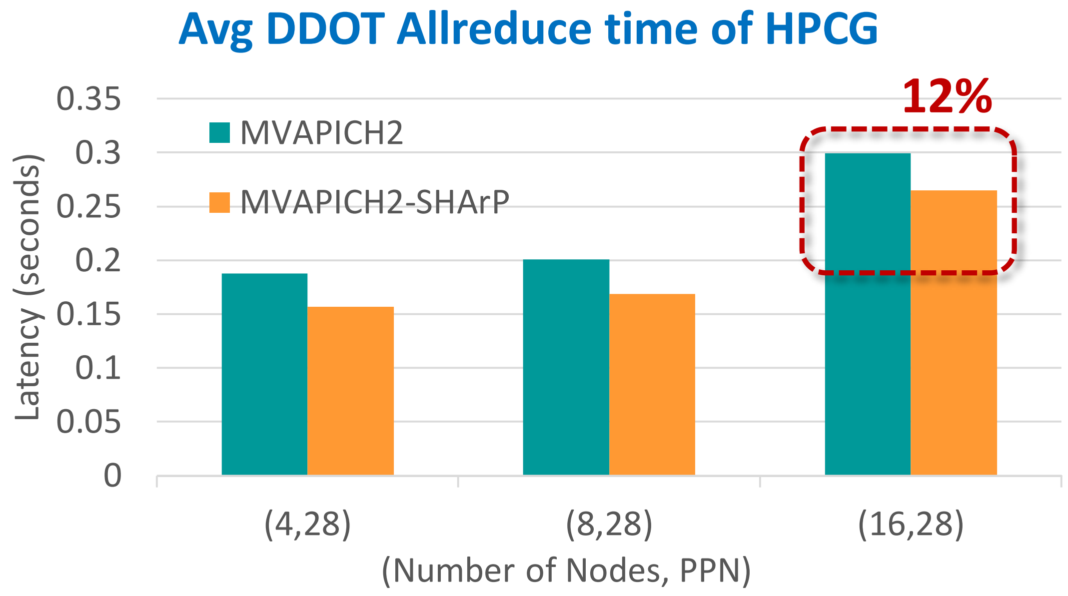



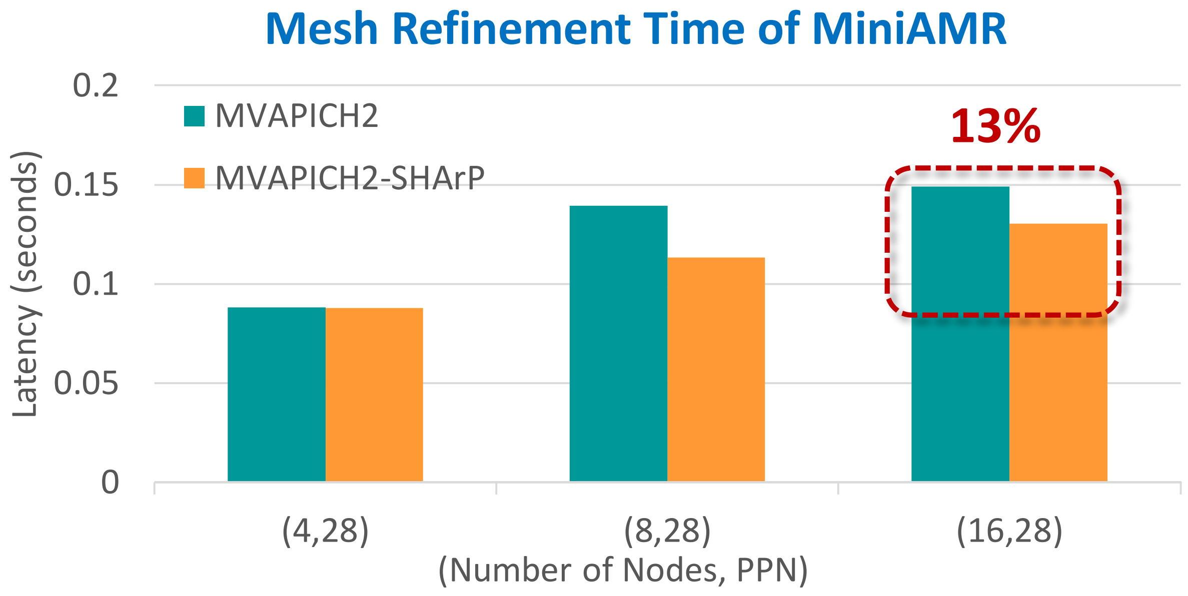




*PPN: Processes Per Node

# Benefits of SHARP at Application Level



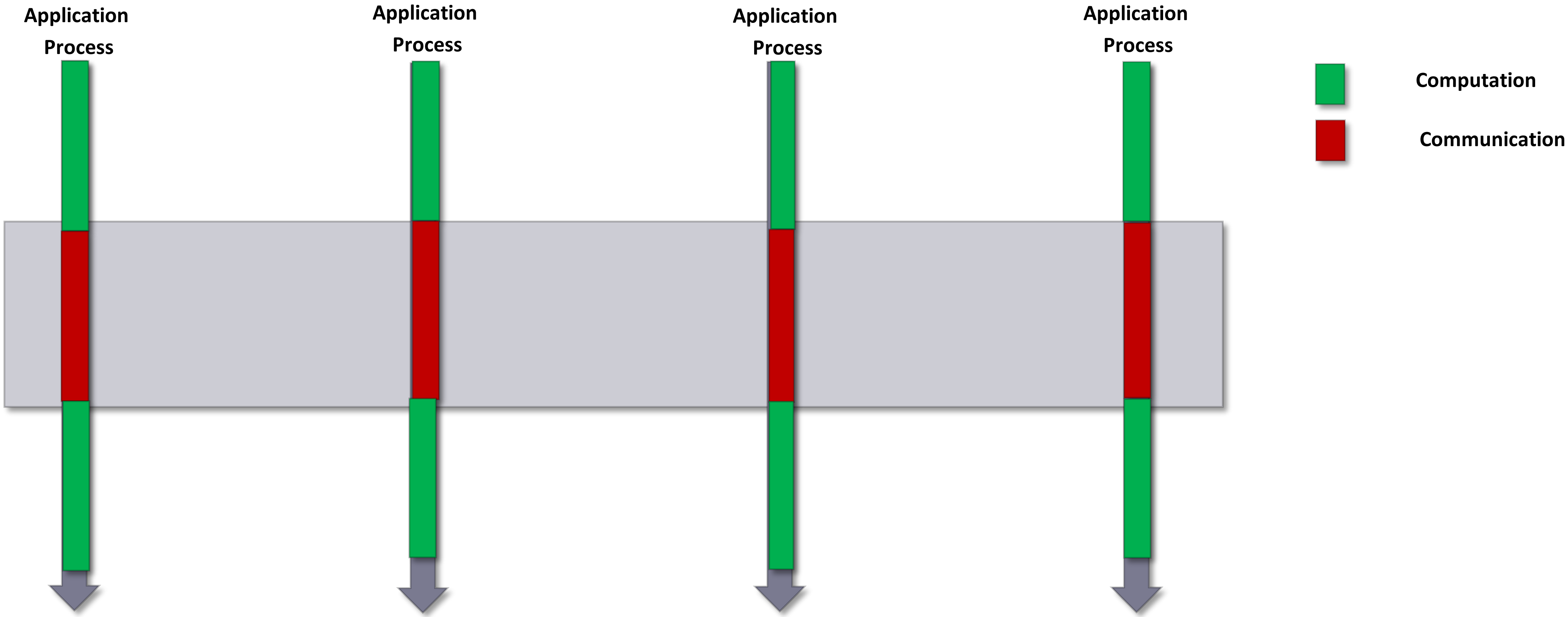




**SHARP support available since MVAPICH2 2.3a**

| Parameter | Description | Default |
|---|---|---|
| MV2_ENABLE_SHARP=1 | Enables SHARP-based collectives | Disabled |
| --enable-sharp | Configure flag to enable SHARP | Disabled |

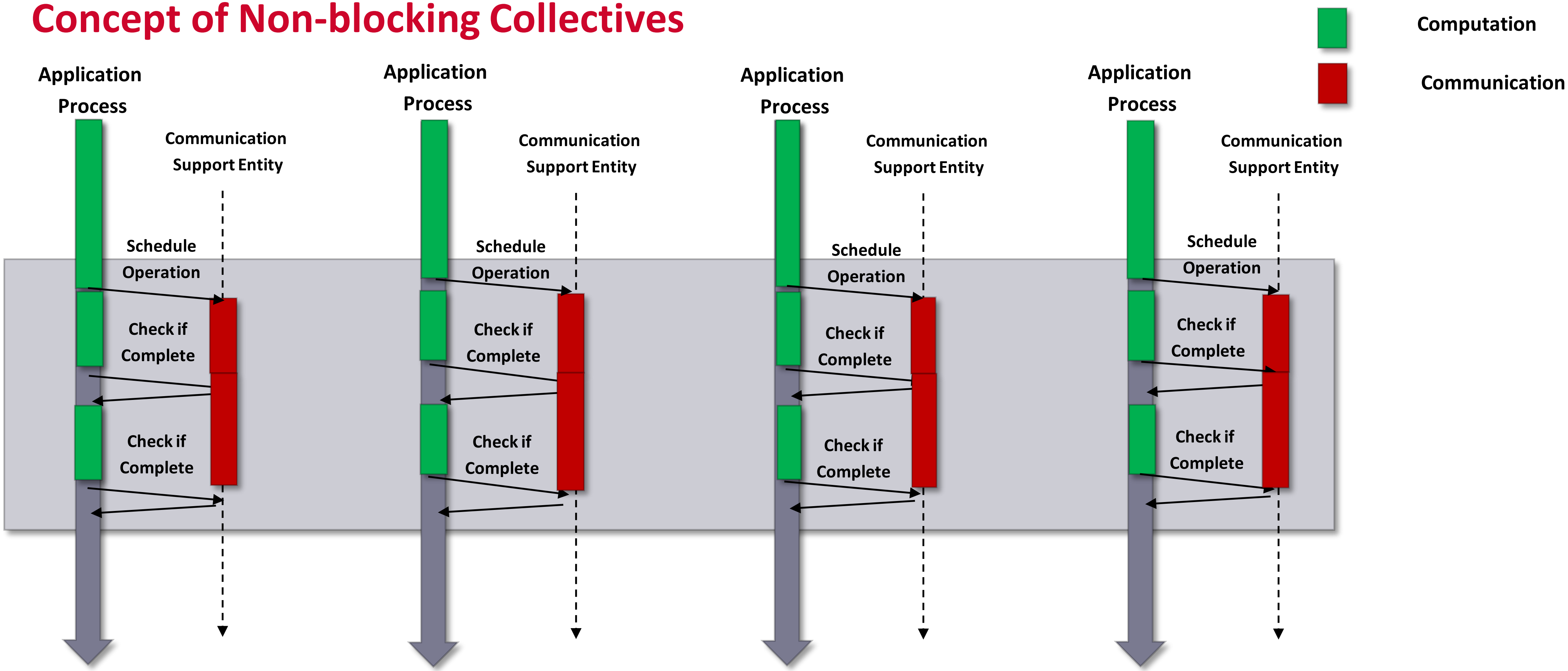
- **Refer to Running Collectives with Hardware based SHArP support section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3b-userguide.html#x1-990006.26**

# Problems with Blocking Collective Operations




- Communication time cannot be used for compute
  - No overlap of computation and communication
  - Inefficient

# Concept of Non-blocking Collectives




- Application processes schedule collective operation
- Check periodically if operation is complete
- **Overlap of computation and communication => Better Performance**
- *Catch: Who will progress communication*

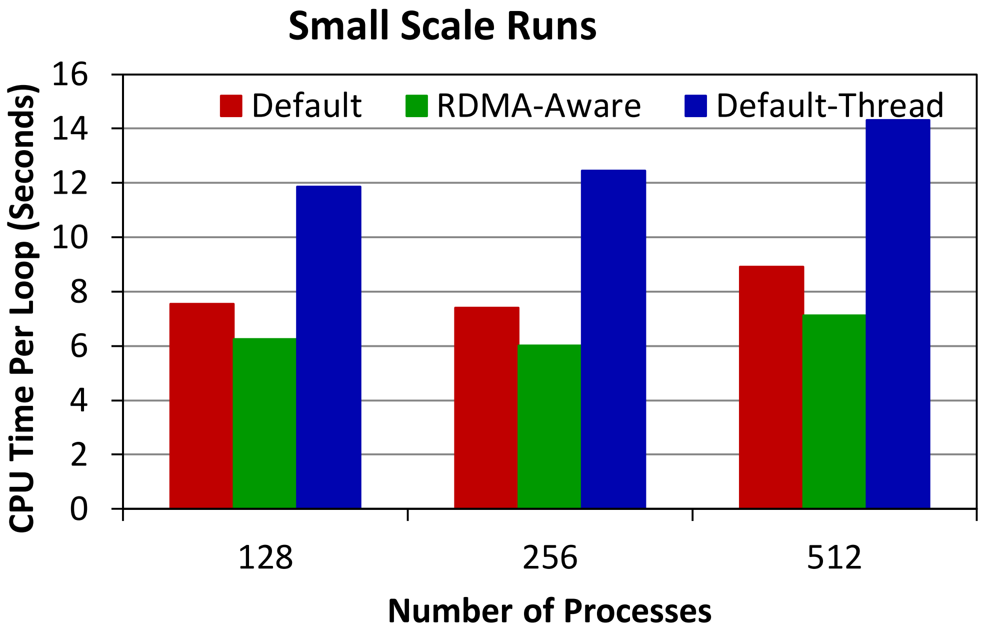
# Non-blocking Collective (NBC) Operations

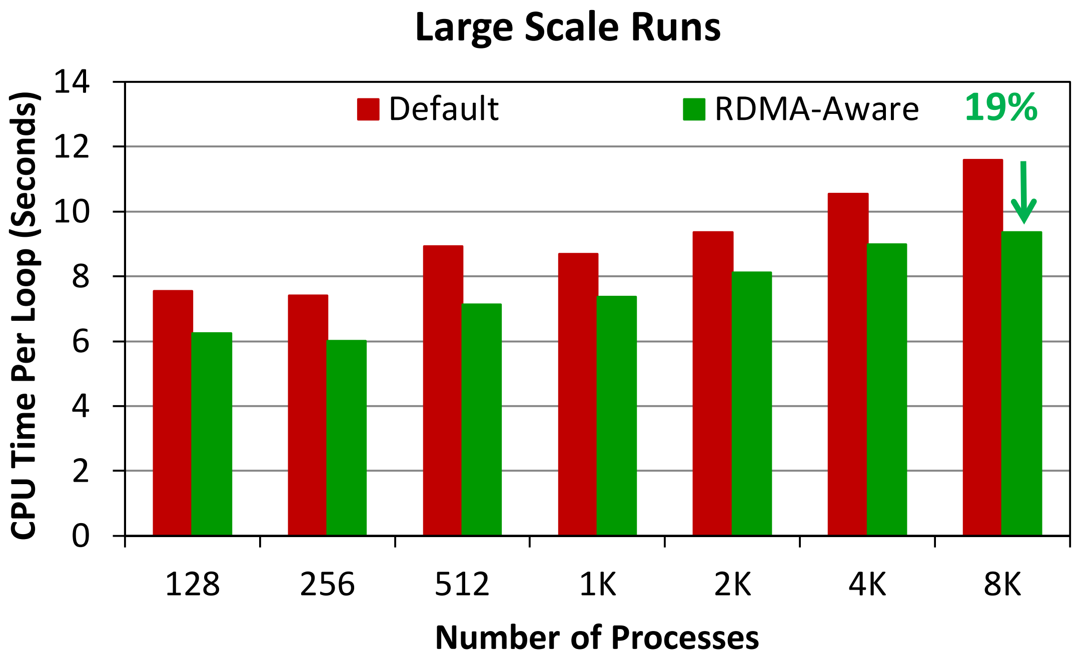
- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI  may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap
- A process calling a NBC operation
  - Schedules collective operation and immediately returns
  - Executes application computation code
  - Waits for the end of the collective
- The communication progress by
  - Application code through MPI_Test
  - Network adapter (HCA) with hardware support
  - Dedicated processes / thread in MPI library
- There is a non-blocking equivalent for each blocking operation
  - Has an “I” in the name
    - MPI_Bcast -> MPI_Ibcast; MPI_Reduce  -> MPI_Ireduce

# How do I write applications with NBC?

```
void main()
{
    MPI_Init()
    …..
    MPI_Ialltoall(…)
    Computation that does not depend on result of Alltoall
    MPI_Test(for Ialltoall) /* Check if complete (non-blocking) */
    Computation that does not depend on result of Alltoall
    MPI_Wait(for Ialltoall) /* Wait till complete (Blocking) */
    …
    MPI_Finalize()
}
```

# P3DFFT Performance with Non-Blocking Alltoall using RDMA Primitives



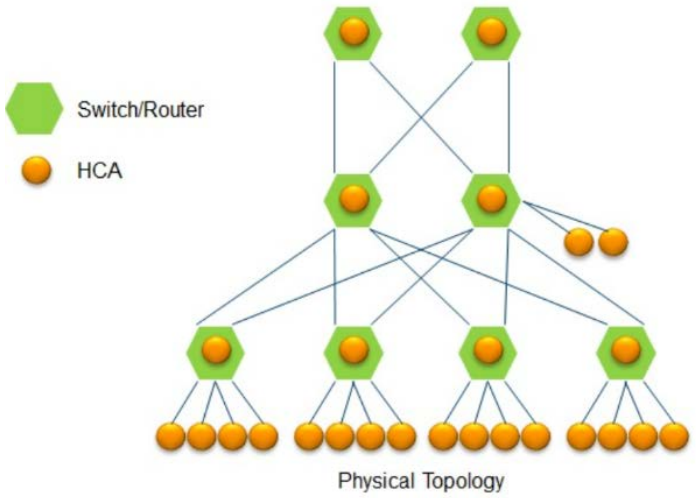




- Weak scaling experiments; problem size increases with job size
- RDMA-Aware delivers 19% improvement over Default @ 8,192 procs
- Default-Thread exhibits worst performance
  - Possibly because threads steal CPU cycles from P3DFFT
  - Do not consider for large scale experiments

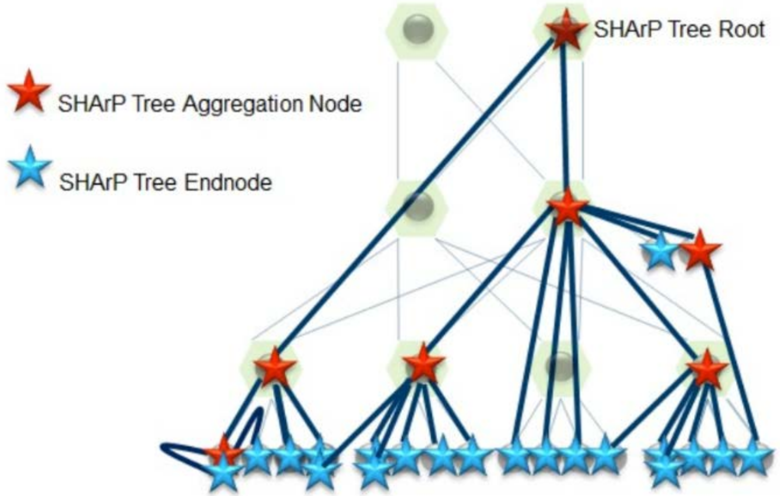
**Will be available in future**

# Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

- Management and execution of MPI operations in the network by using SHArP
  - Manipulation of data while it is being transferred in the switch network
- SHArP provides an abstraction to realize the reduction operation
  - Defines Aggregation Nodes (AN), Aggregation Tree, and Aggregation Groups
  - AN logic is implemented as an InfiniBand Target Channel Adapter (TCA) integrated into the switch ASIC *
  - Uses RC for communication between ANs and between AN and hosts in the Aggregation Tree *




Physical Network Topology*

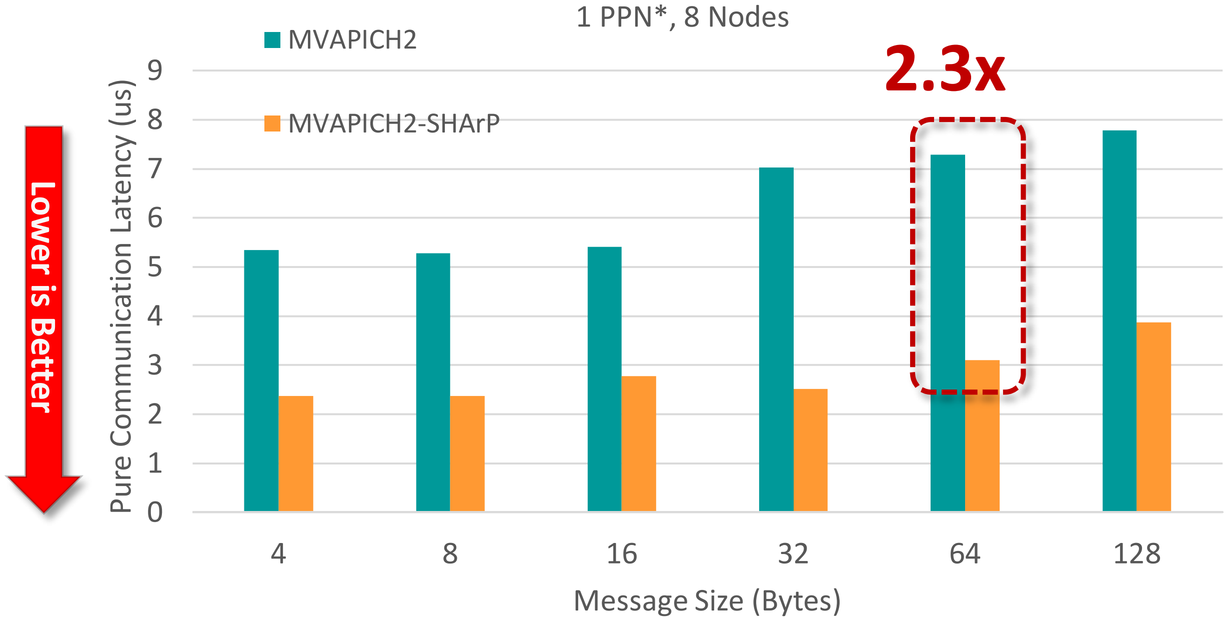


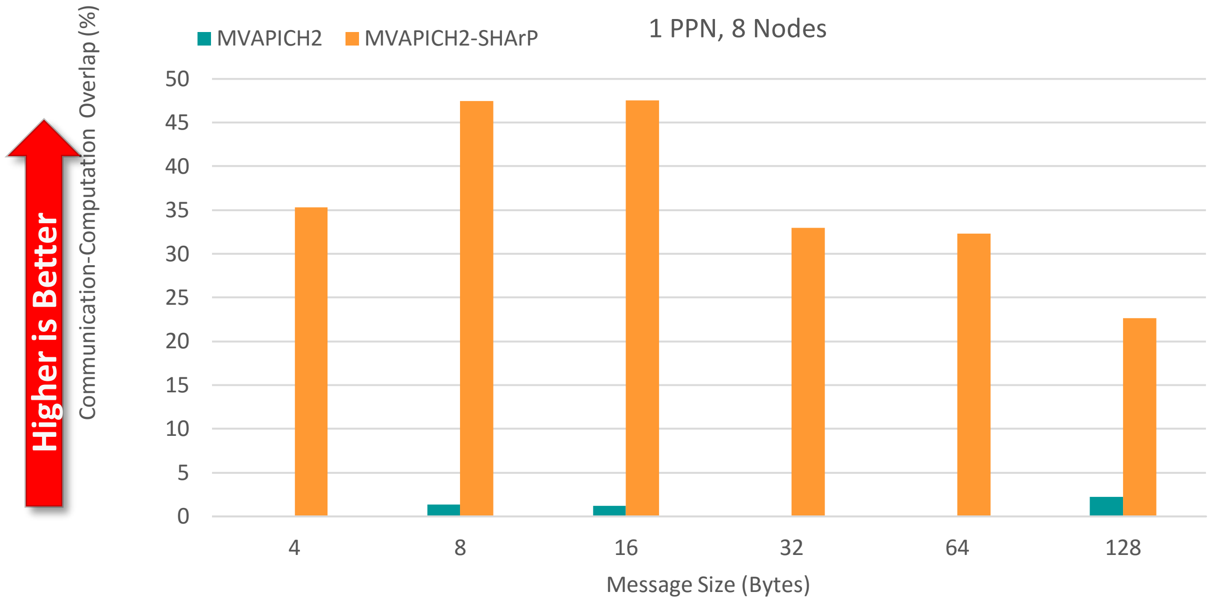

Logical SHArP Tree*

* Bloch et al. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction

# Evaluation of SHArP based Non Blocking Allreduce

## MPI_Iallreduce Benchmark




- Complete offload of Allreduce collective operation to Switch helps to have much higher overlap of communication and computation

**Available since MVAPICH2 2.3a**

*PPN: Processes Per Node

# MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF …)
  - MPI + Task*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching*
  - Adapter Memory*
- Enhanced communication schemes for upcoming architectures
  - NVLINK*
  - CAPI*
- Extended topology-aware collectives
- Extended Energy-aware designs and Virtualization Support
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended FT support
- Support for * features will be available in future MVAPICH2 Releases

# Thank You!

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project

http://mvapich.cse.ohio-state.edu/