

# Design of Collectives and One-Sided Operations in MPI and their Impact on Application-Level Performance and Scalability

Talk at HPC Advisory Council China Workshop

by

**Dhabaleswar K. (DK) Panda**

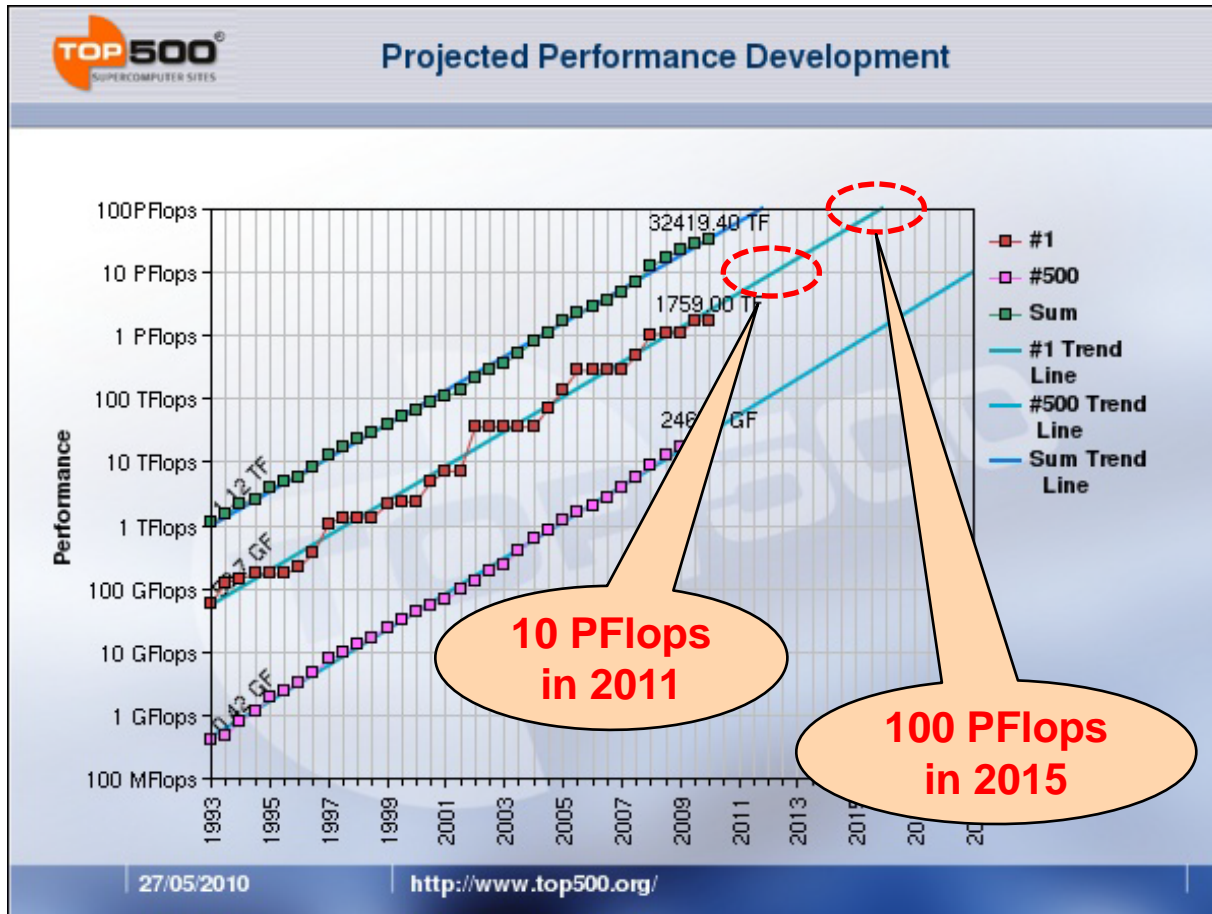
The Ohio State University

E-mail: [panda@cse.ohio-state.edu](mailto:panda@cse.ohio-state.edu)

<http://www.cse.ohio-state.edu/~panda>

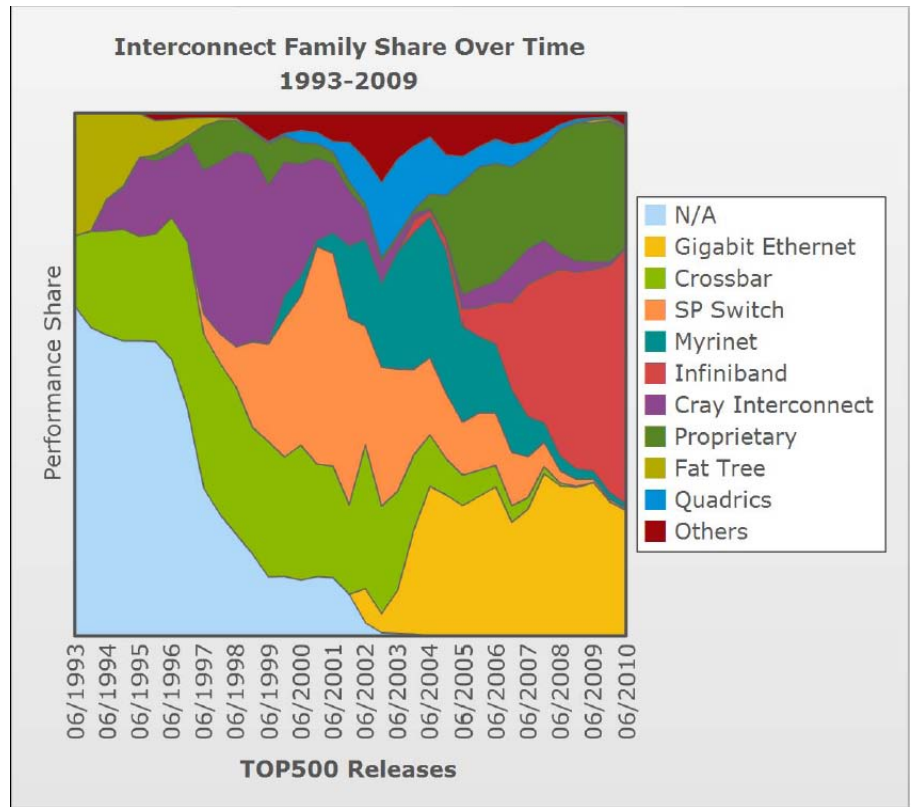
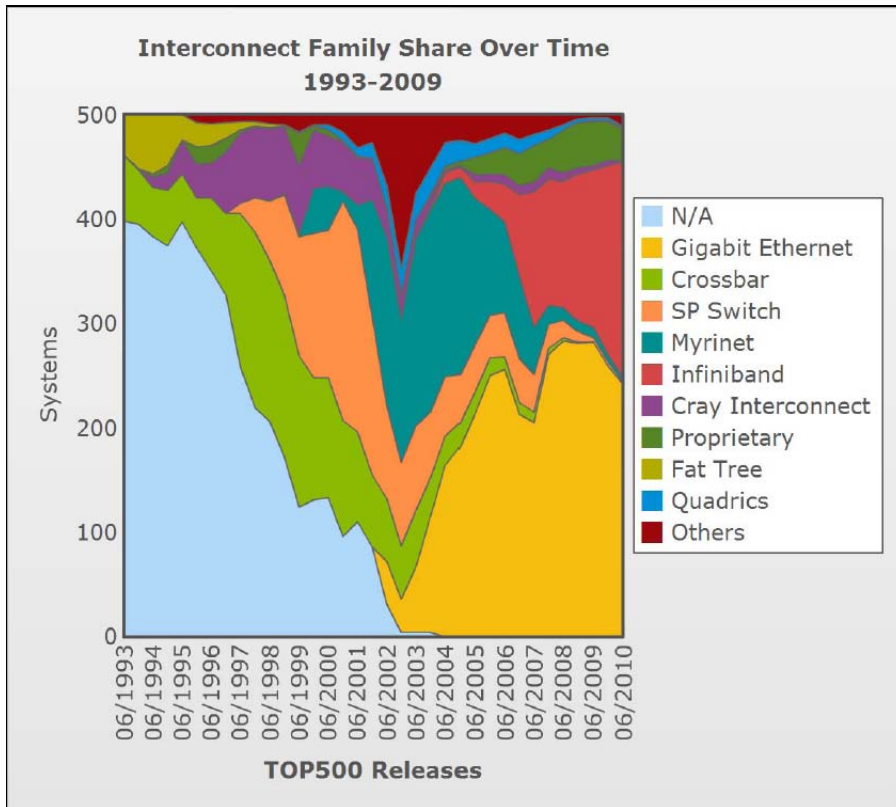


# PetaFlop to ExaFlop Computing



*Expected to have an ExaFlop system in 2018-2019 !*

# InfiniBand in the Top500



Percentage share of InfiniBand is steadily increasing

# Large-scale InfiniBand Installations

- 207 IB Clusters (41.4%) in the Jun '10 Top500 list (<http://www.top500.org>)
- Installations in the Top 30 (15 systems):

120,640 cores (Nebulae) in China (2 <sup>nd</sup> )	26,232 cores (TachyonII) in South Korea (15 <sup>th</sup> )
122,400 cores (RoadRunner) at LANL (3 <sup>rd</sup> )	23,040 cores (Jade) at GENCI (18 <sup>th</sup> )
81,920 cores (Pleiades) at NASA Ames (6 <sup>th</sup> )	33,120 cores (Mole-8.5) in China (19 <sup>th</sup> )
71,680 cores (Tianhe-1) in China (7 <sup>th</sup> )	17,072 cores at JAEA in Japan (22 <sup>nd</sup> )
42,440 cores (Red Sky) at Sandia (10 <sup>th</sup> )	30,720 cores (Dawning) at Shanghai (19 <sup>th</sup> )
62,976 cores (Ranger) at TACC (11 <sup>th</sup> )	24,704 cores by HP (25 <sup>th</sup> )
35,360 cores (Lomonosov) in Russia (13 <sup>th</sup> )	15,360 cores at ERDC (30 <sup>th</sup> )
26,304 cores (Juropa) in Germany (14 <sup>th</sup> )	<i>More are getting installed !</i>

# MVAPICH/MVAPICH2 Software

- High Performance MPI Library for IB and HSE
  - MVAPICH (MPI-1) and MVAPICH2 (MPI-2.2)
  - Used by more than 1,275 organizations in 60 countries
  - More than 45,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 6<sup>th</sup> ranked 81,920-core cluster (Pleiades) at NASA
    - 7<sup>th</sup> ranked 71,680-core cluster (Tianhe-1) in China
    - 11<sup>th</sup> ranked 62,976-core cluster (Ranger) at TACC
  - Available with software stacks of many IB, HSE and server vendors including Open Fabrics Enterprise Distribution (OFED)
  - <http://mvapich.cse.ohio-state.edu>

# Advanced Designs for Collectives, One-Sided and PGAS

- *Collective Communication*
  - *Multi-core Aware*
  - *Topology-aware*
  - *Power-aware*
  - *Exploiting Collective Offload*
    - *Non-blocking collectives*
- *One-sided communication*
  - *Basic inter-node performance*
  - *Synchronization*
  - *Intra-node design*
  - *Application Evaluation*
- *Partitioned Global Address (PGAS) Support*

# Collective Design Framework

Collective Algorithms

Conventional Schemes

Pt2pt  
Inter-node and  
Intra-node

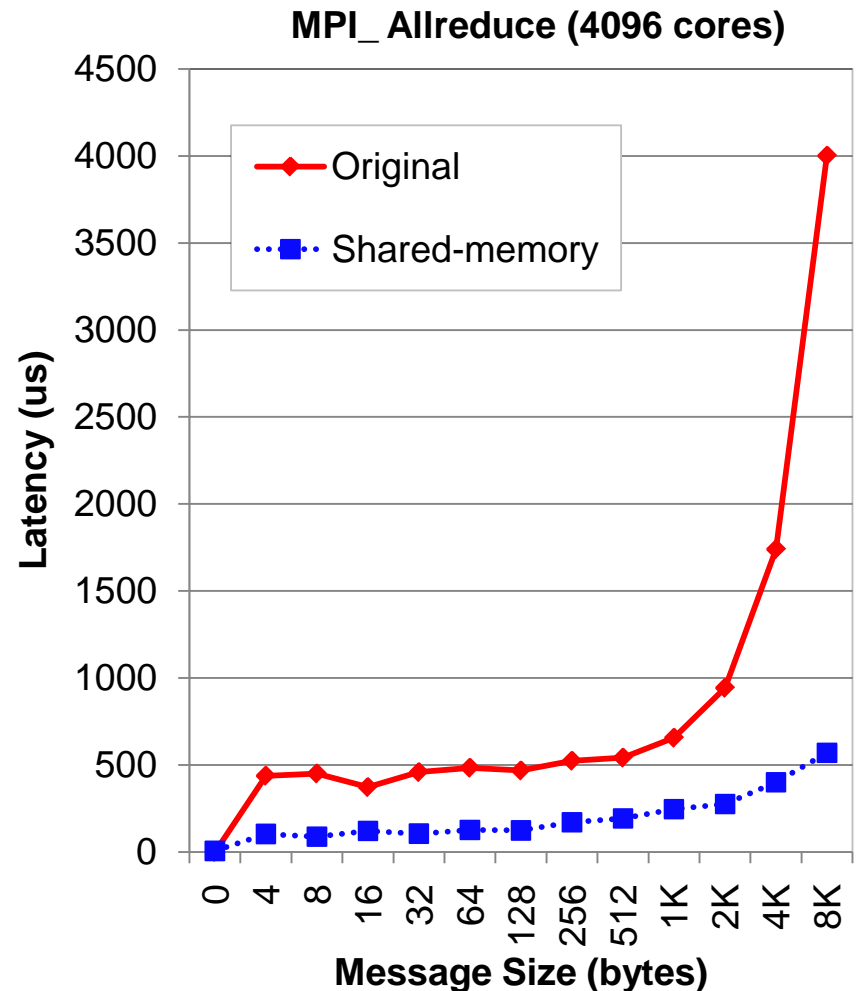
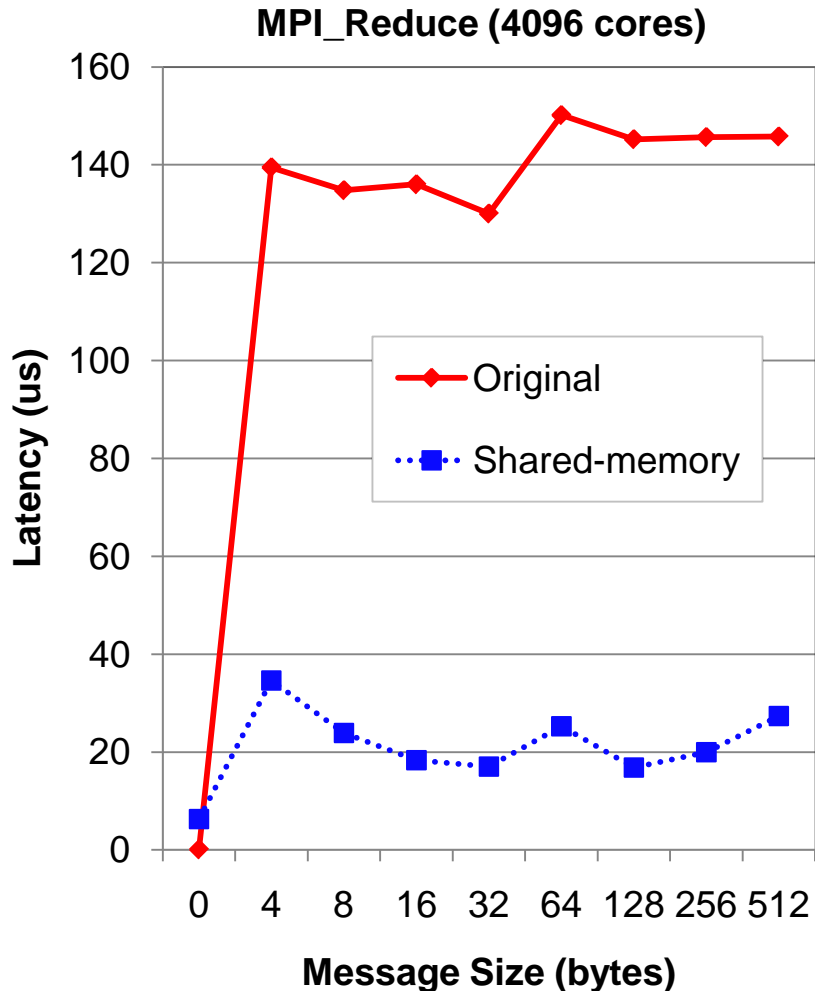
Shared Memory  
Aware Schemes  
for Multi-cores

Pt2pt  
Inter-node

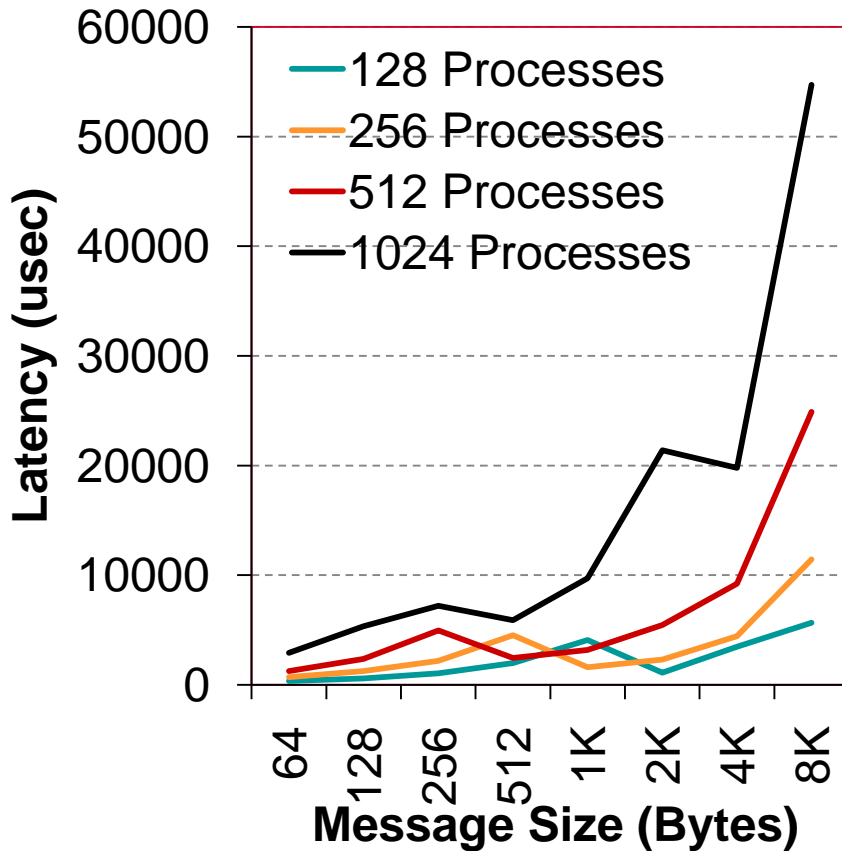
Shared  
memory  
Within the  
node

Available in Both  
MVAPICH and  
MVAPICH2

# Shared-memory Aware Collectives (4K cores on TACC Ranger with MVAPICH2)

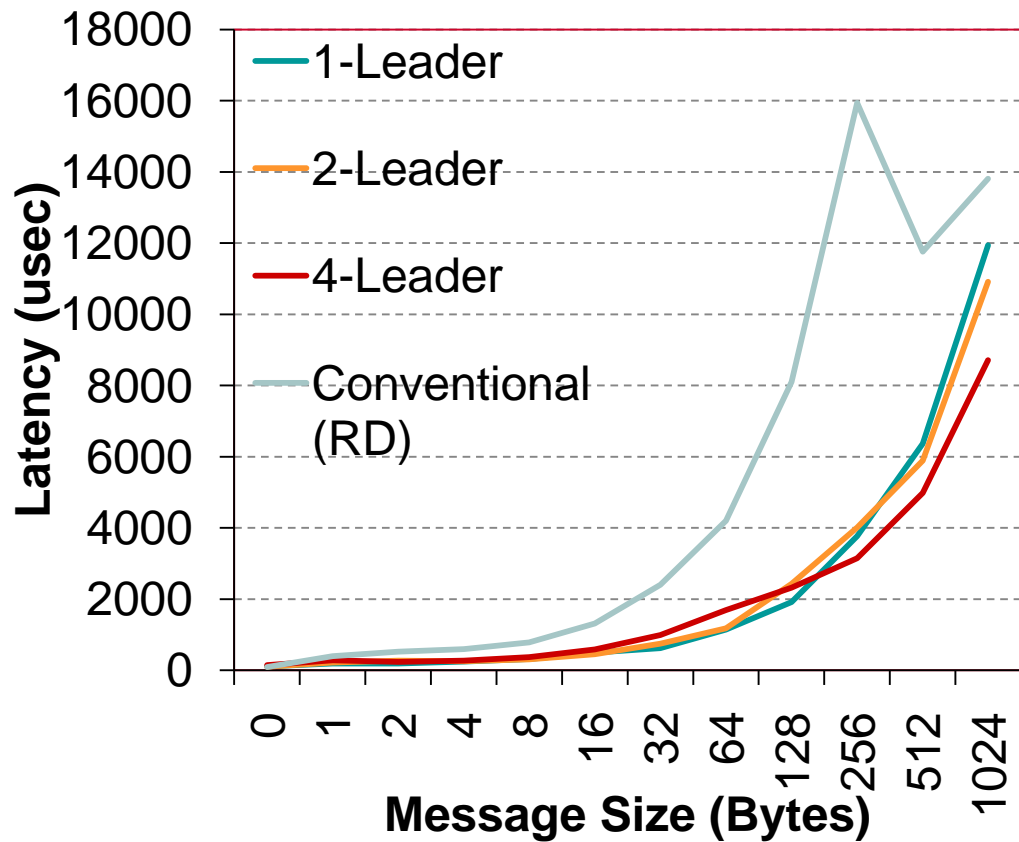


# Multi-Leader Collective Algorithms (MPI\_Allgather)



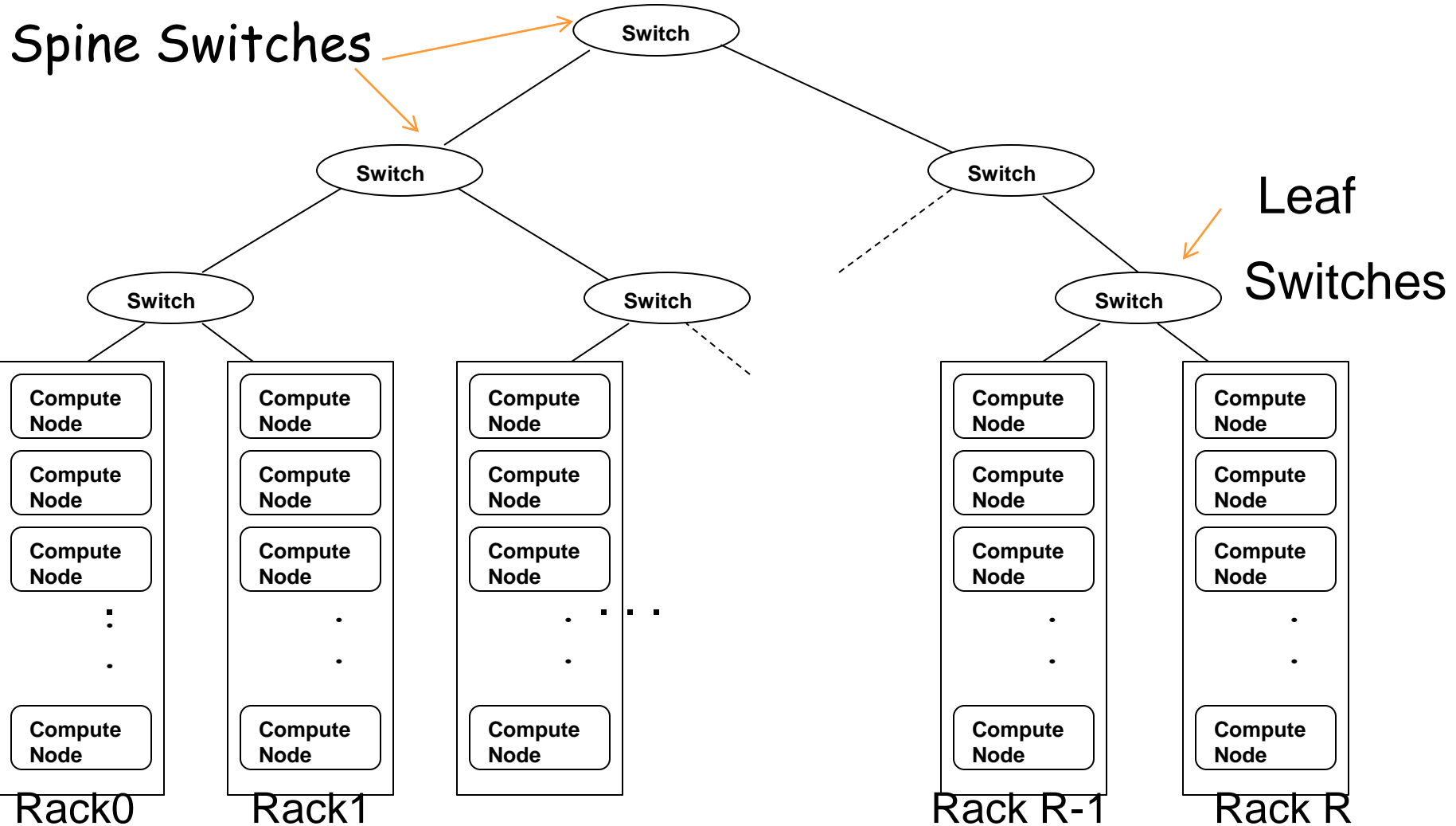
Default Recursive Doubling (RD)  
Scales Badly

Will be available in future  
MVAPICH2 Release

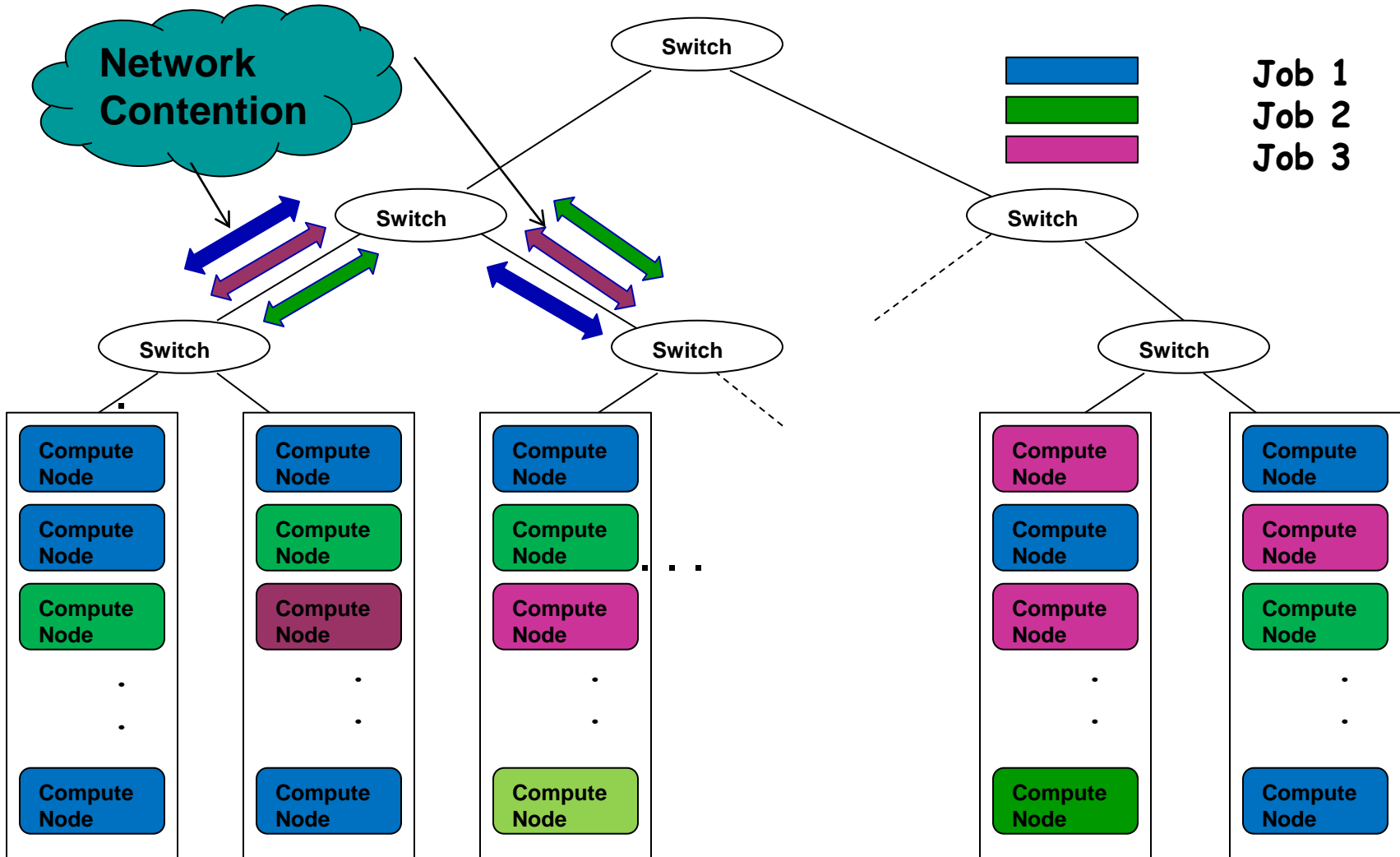


4-Leader Allgather performs  
about 58% better with 1K  
processes

# Typical Topology of Large Scale Clusters



# Network Sharing between Different Jobs

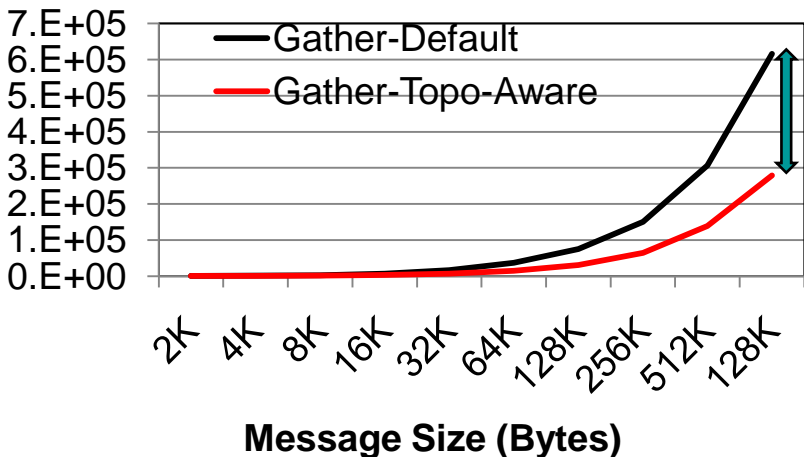


# Need for Topology-Aware Collectives

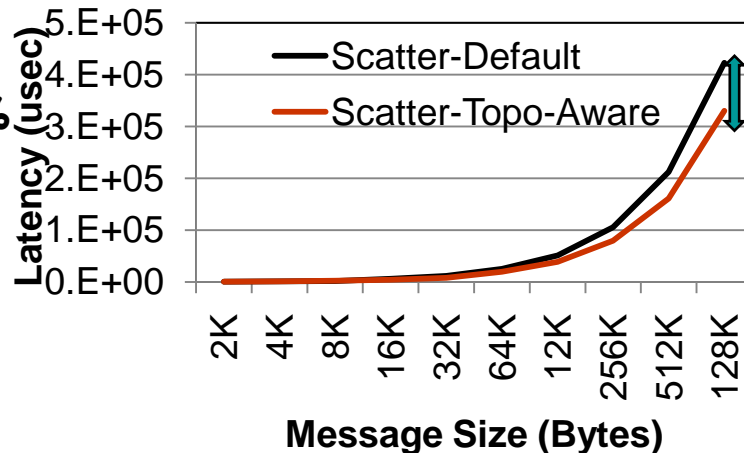
- Can we detect the topology of large-scale InfiniBand clusters efficiently?
- How do we design collective algorithms in a "*Topology-Aware*" manner to minimize the communication costs?

# Topology-Aware Collectives

Latency (usec)

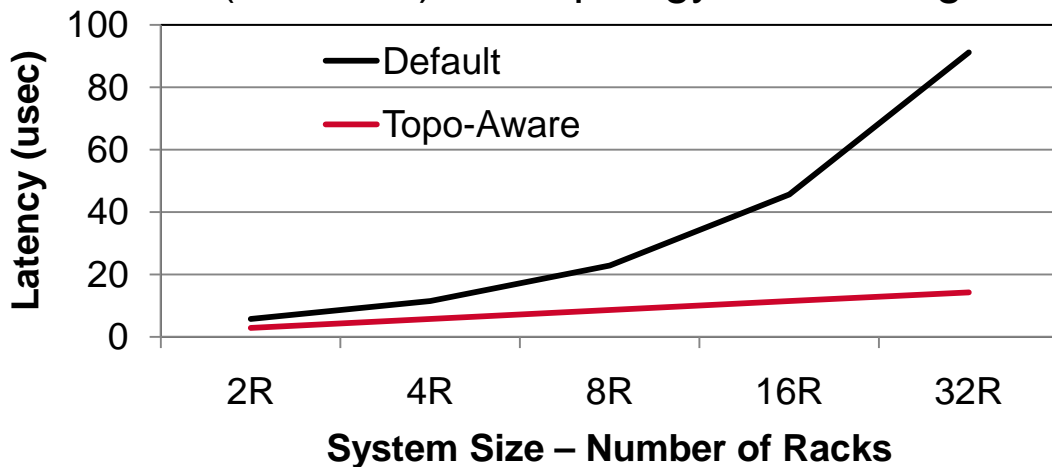


54%



22%

Default (Binomial) Vs Topology-Aware Algorithms with 296 Processes



Estimated Latency Of Default and Topology Aware Algorithms for small messages And Varying System Sizes

# Need for Power-Aware Collective Algorithms

- Collective operations may take significant time on large-scale systems
- Most of the processors are idle polling during the collective operations
- Modern processor architectures allow DVFS and CPU Throttling operations to be performed within a few micro-seconds
- *Can we re-design collective communication algorithms in a power-aware manner taking into consideration the nature of the collective operation?*
- *What is the impact on performance?*

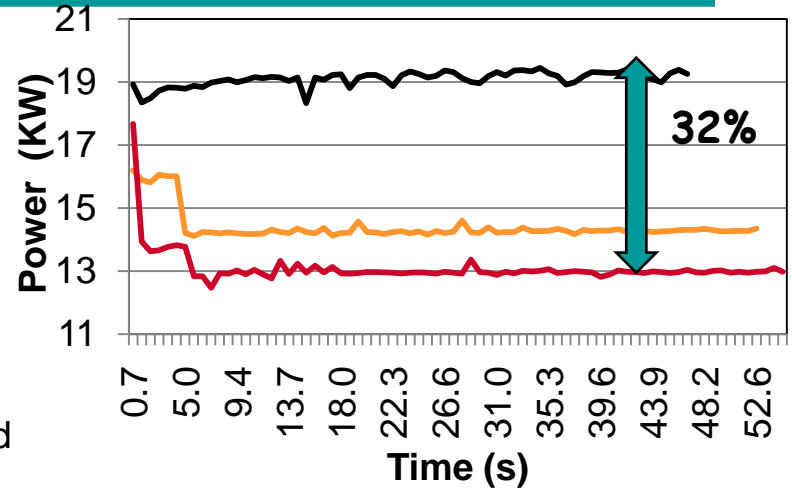
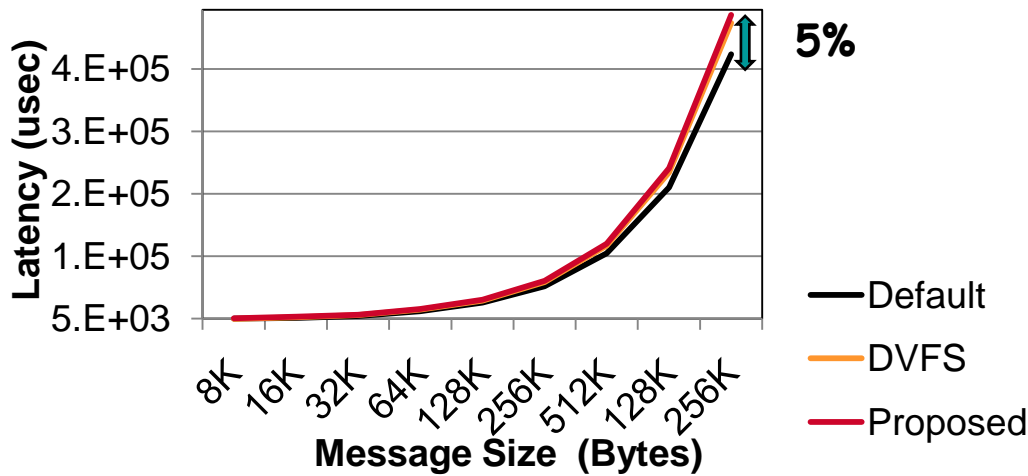
# Design Space of Power-Aware Algorithms

**Default (No Power Savings):** Run each core at peak frequency/throttling state.

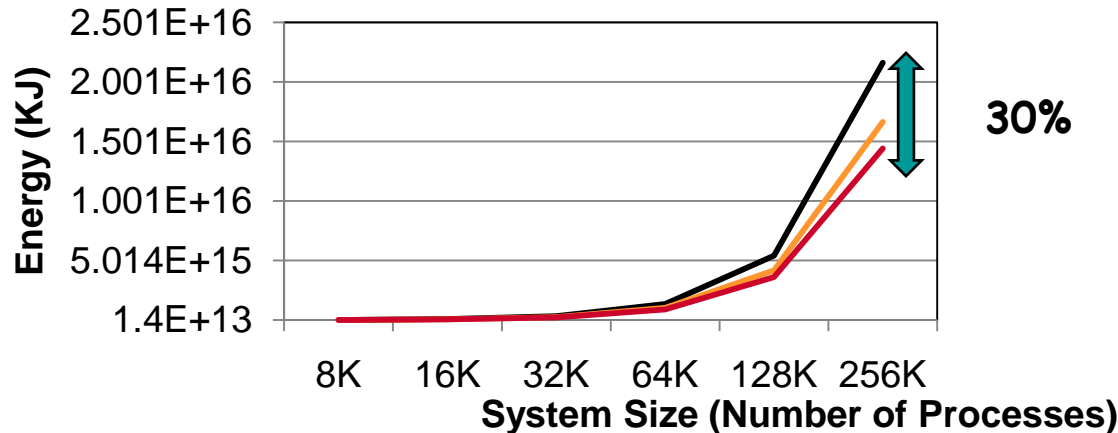
**Frequency Scaling Only:** Dynamically detect communication phases. Treat them as a black-box and scale the CPU frequency

***Proposed:*** Consider the communication characteristics of different collectives, intelligently use both DVFS and CPU Throttling to deliver fine-grained power-savings

# Power-Aware Collectives



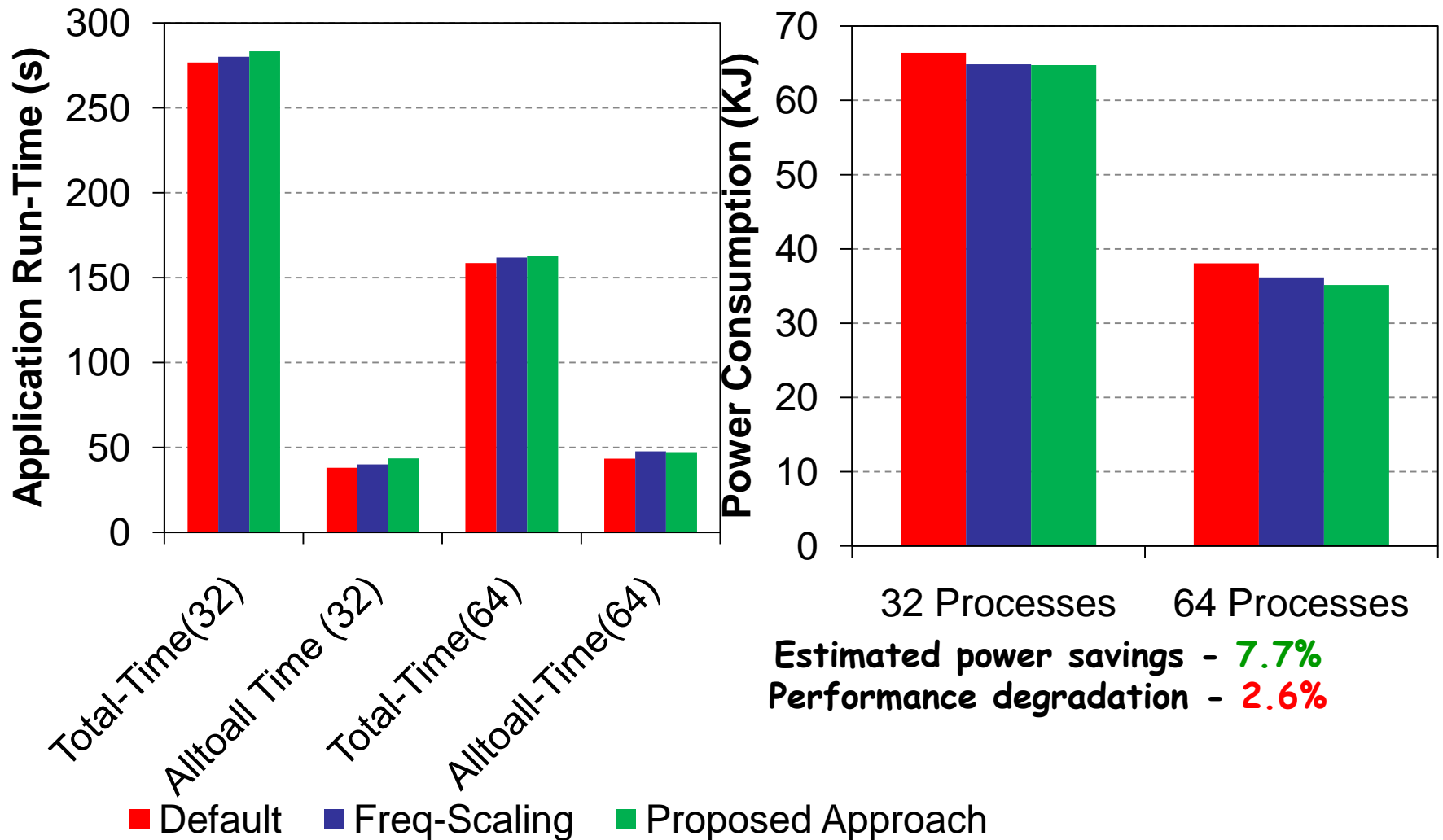
Performance and Power Comparison : MPI\_Alltoall with 64 processes on 8 nodes



Estimated Energy Consumption during an MPI\_Alltoall operation with 128K Message size and Varying System Size

K. Kandalla, E. Mancini, Sayantan Sur and D. K. Panda, "Designing Power Aware Collective Communication Algorithms for Infiniband Clusters, ICPP '10

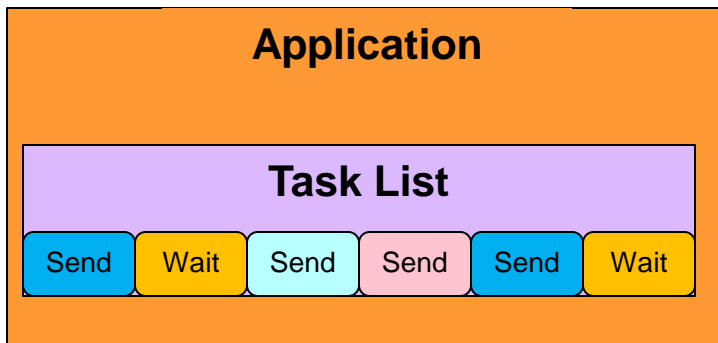
# Power Savings with Applications (CPMD with 32 and 64 Processes)



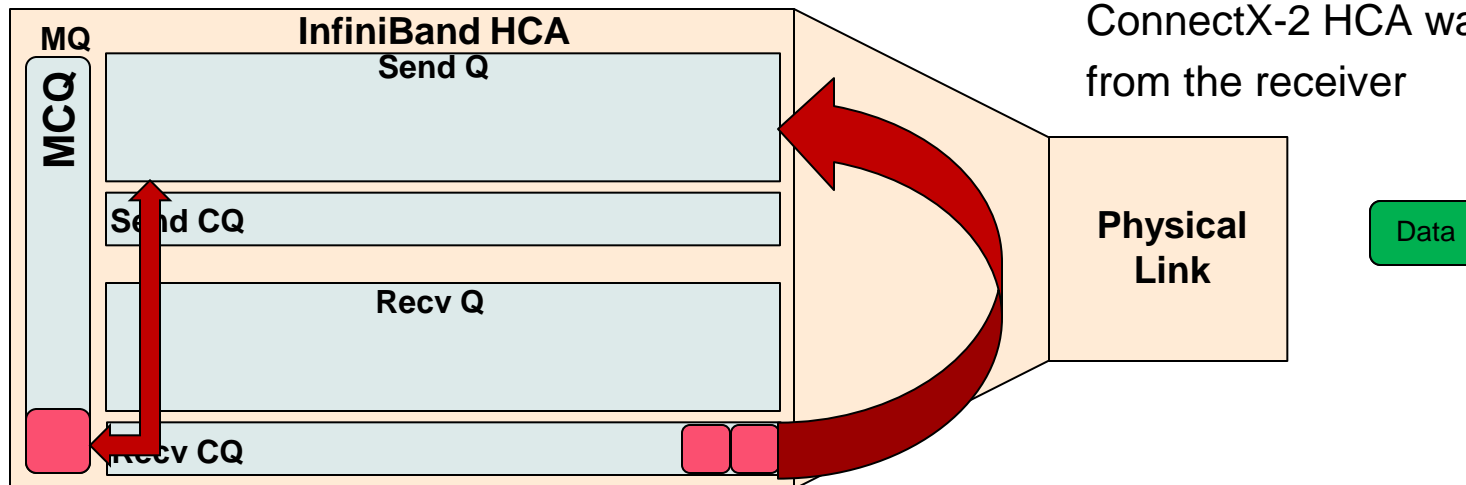
# Collective Offload in ConnectX-2 and Need for Non-Blocking Collectives

- Mellanox's ConnectX-2 adapter features "task-list" offload interface
  - *Extension to existing InfiniBand APIs*
- Collective communication performance is usually a scaling bottleneck
  - Non-blocking Collective Communication in upcoming MPI-3 may provide a solution for Exascale level systems by overlapping time spent in collectives
- Accordingly MPI software stacks need to be re-designed to leverage offload in a comprehensive manner

# Collective Offload Support in ConnectX-2 (Recv followed by Multi-Send)

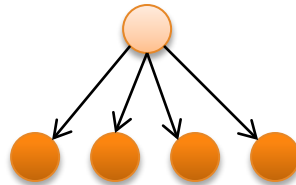


- Sender creates a task-list consisting of only send and wait WQEs
  - One send WQE is created for each registered receiver and is appended to the rear of a singly linked task-list
  - A wait WQE is added to make the ConnectX-2 HCA wait for ACK packet from the receiver

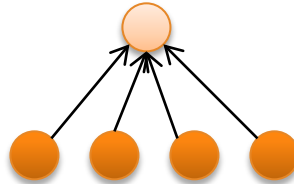


# Collective Communication Primitives

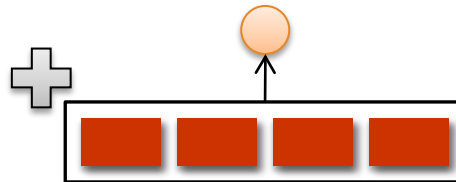
One-to-many  
Multi-send



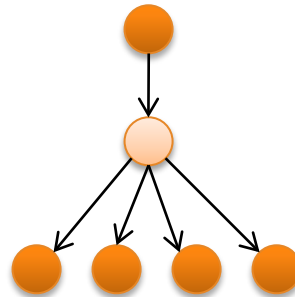
Multi-receive



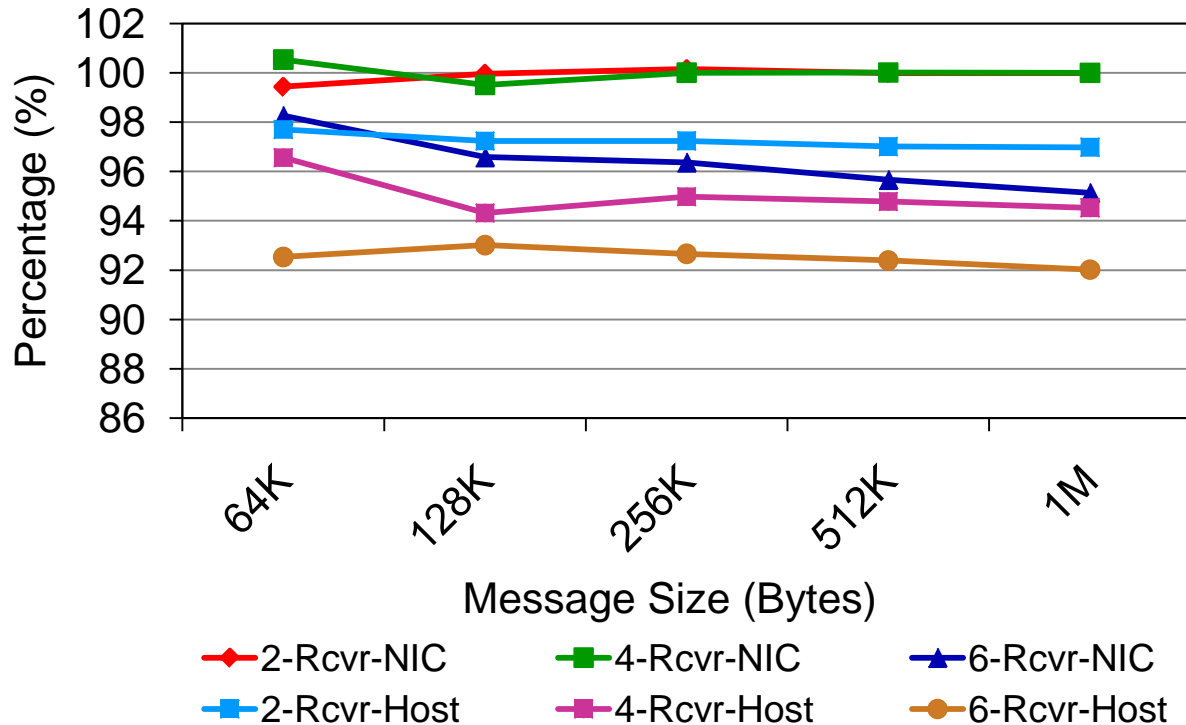
Receive-reduce



Receive-replicate



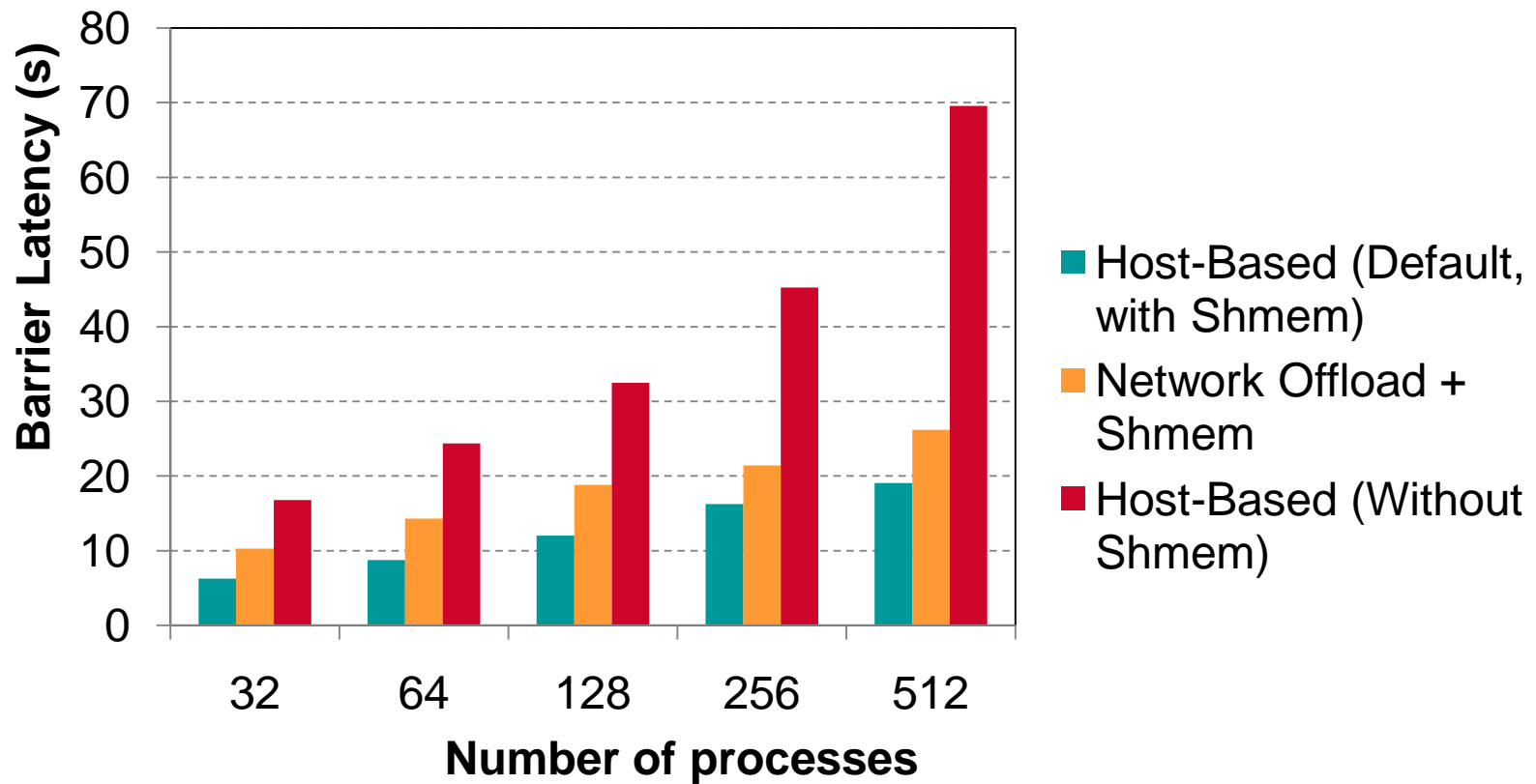
# Multi-Send Overlap Percentage



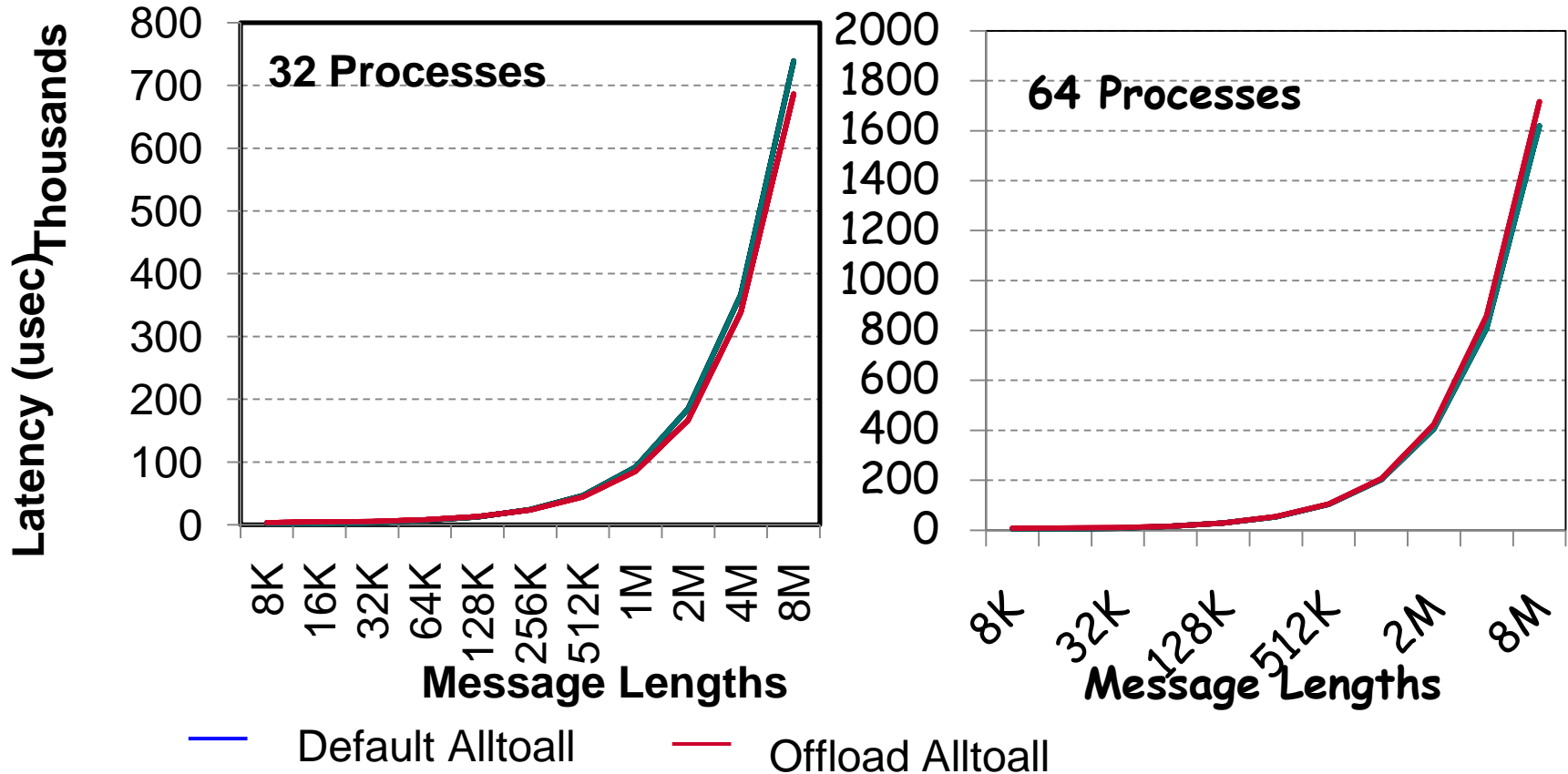
- Offloaded primitives can achieve very good overlap as compared to their host-based counterparts (95% or above)

H. Subramoni, K. Kandalla, S. Sur and D. K. Panda, Design and Evaluation of Generalized Collective Communication Primitives with Overlap using ConnectX-2 Offload Engine , Int'l Symposium on Hot Interconnects (HotI), Aug. 2010.

# MPI Barrier with ConnectX-2 Offload

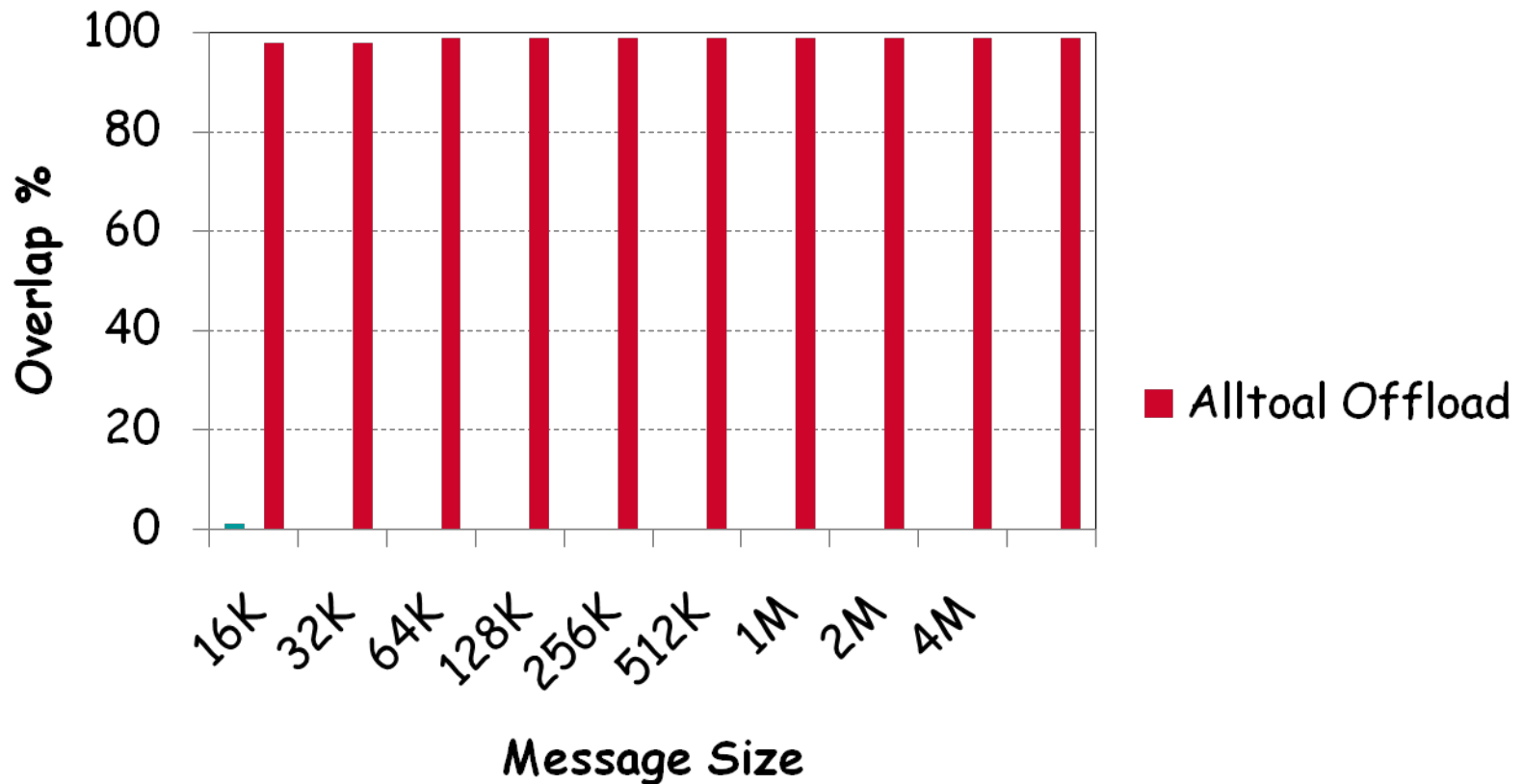


# Alltoall Latency Comparison with ConnectX-2 Collective Offload



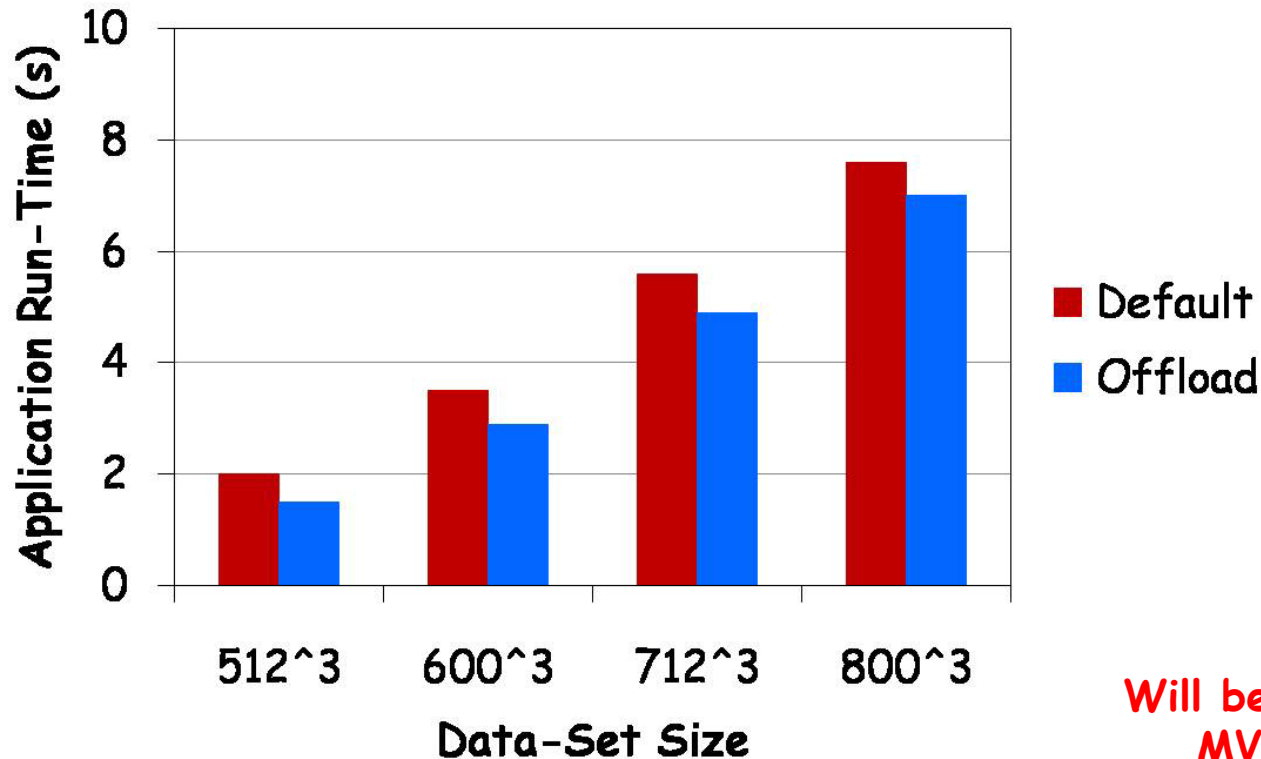
**Offload Alltoall latency is about 10% better than host-based Alltoall latency for 32 Processes.**

# Overlap Capability of Alltoall with ConnectX-2 Offload



Near Perfect Overlap with Network Offload Alltoall for most medium and large messages with 32 Processes

# Redesigned Parallel 3D FFT Library with Collective Offload Alltoall



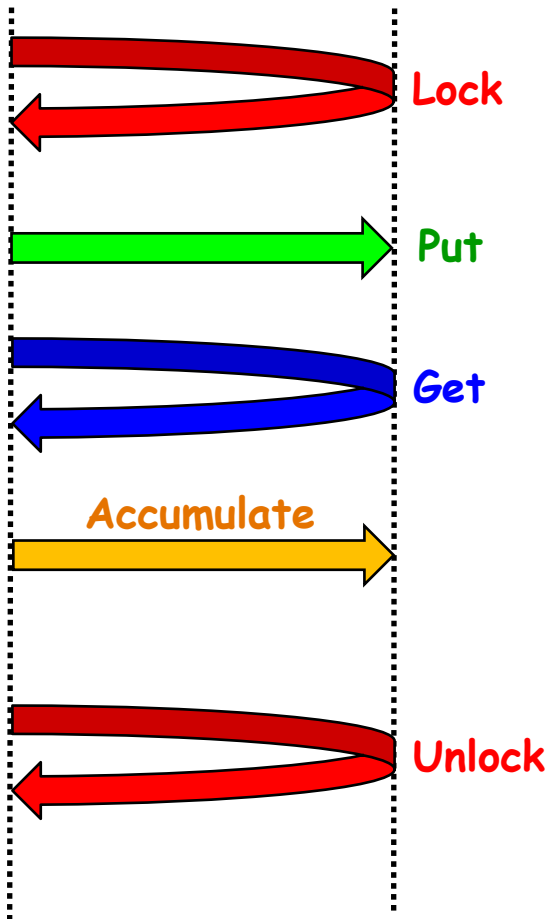
Will be available in Future  
MVAPICH2 Release

Improvement of up to 23% in Application  
Run-Time with Offload Alltoall with 64 Processes

# Advanced Designs for Collectives, One-Sided and PGAS

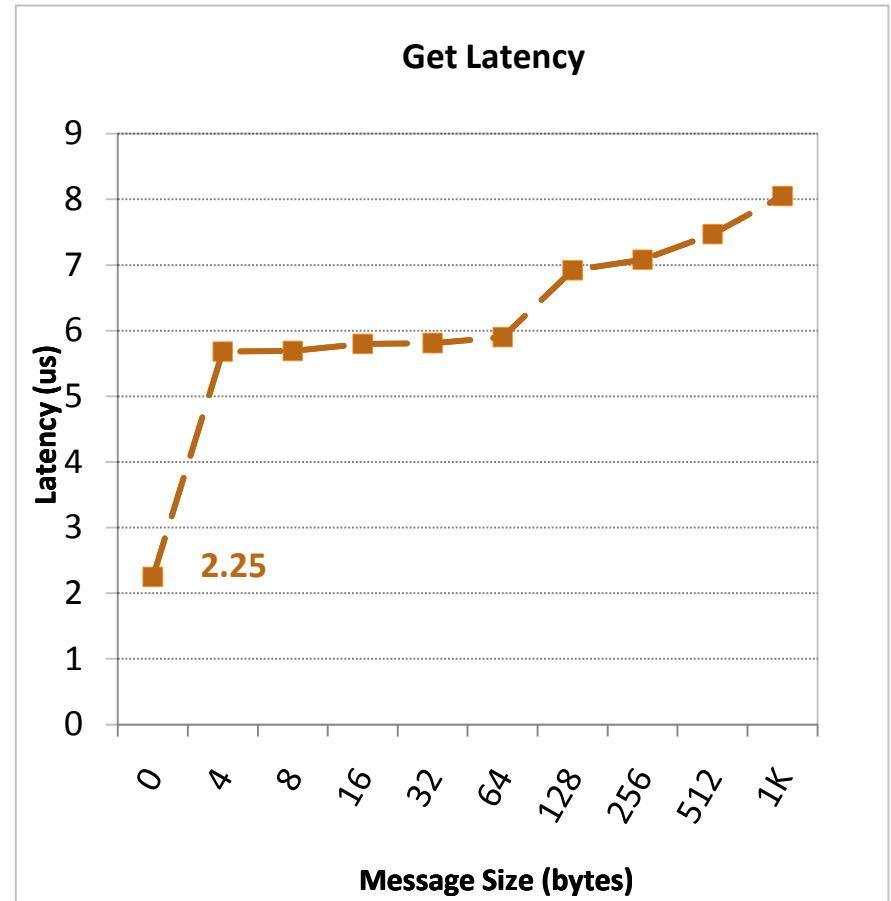
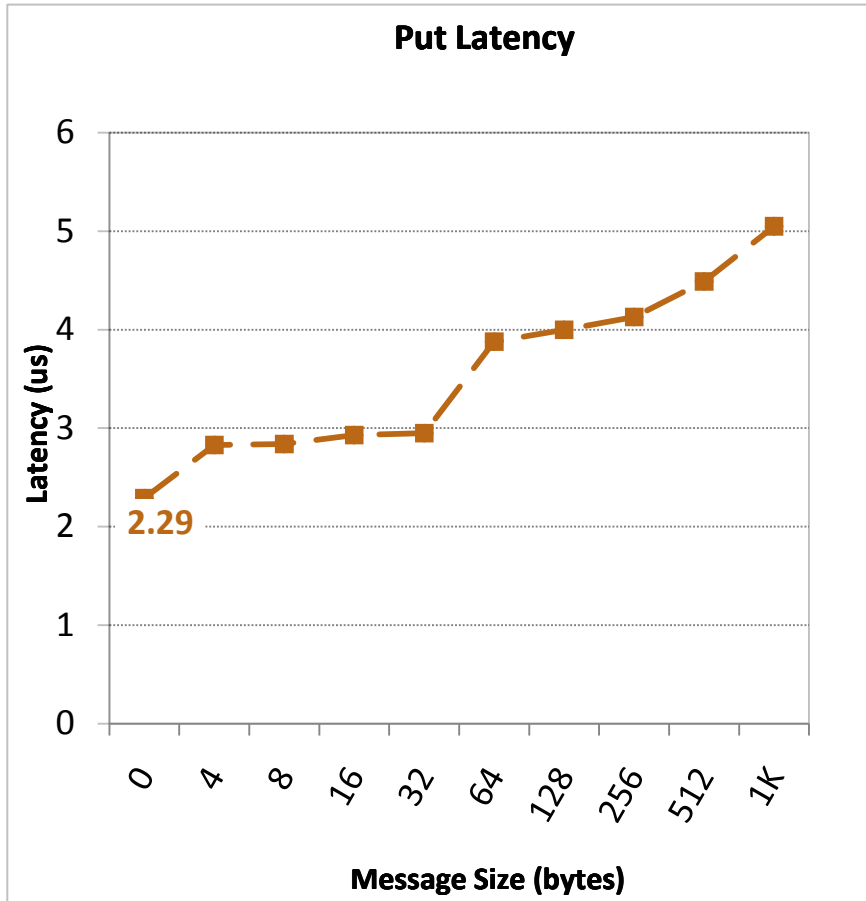
- Collective Communication
  - Multi-core Aware
  - Topology-aware
  - Power-aware
  - Exploiting Collective Offload
    - Non-blocking collectives
- One-sided communication
  - Basic inter-node performance
  - Synchronization
  - Intra-node design
  - Application Evaluation
- Partitioned Global Address (PGAS) Support

# MPI One-sided Communication



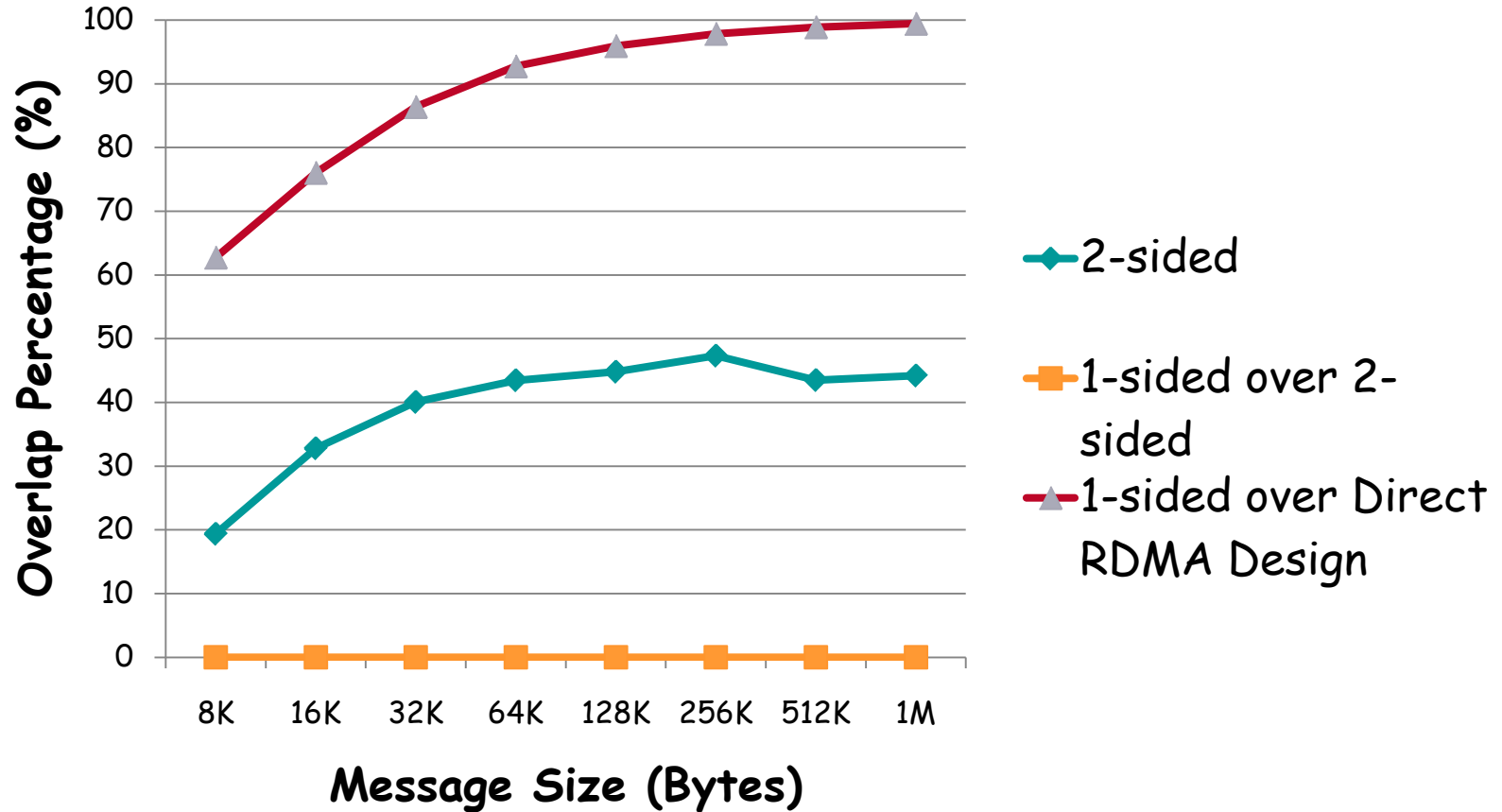
- Specified by the MPI-2 standard
- Data movement operations
  - MPI\_Put
  - MPI\_Get
  - MPI\_Accumulate
- Synchronization operations
  - MPI\_Lock/MPI\_Unlock
  - MPI\_Win\_fence
  - MPI\_Win\_post, MPI\_Win\_start, MPI\_Win\_complete, MPI\_Win\_wait
- **Design Challenges**
  - Improved Communication Performance (two-sided vs. direct one-sided)
  - Reduced Synchronization Overhead
  - Better overlap capability

# Inter-node Put and Get Latency (ConnectX-QDR): Direct One-Sided

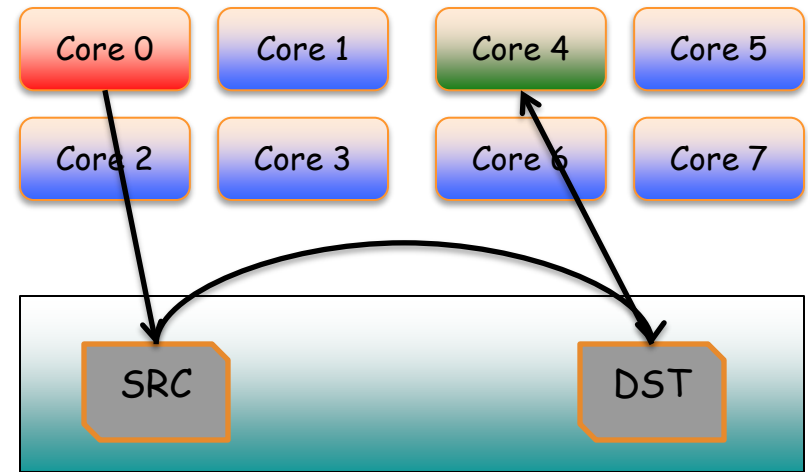
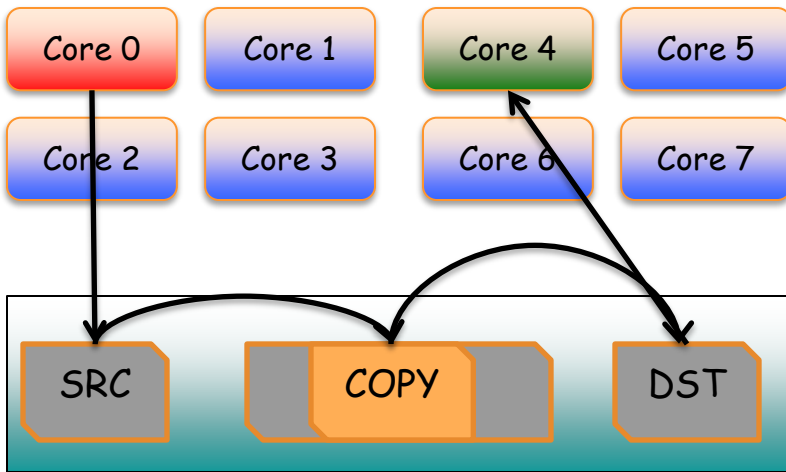


All numbers taken on 2.4 GHz Quad-core (Nehalem) Intel with IB switch

# Inter-node Overlap Performance

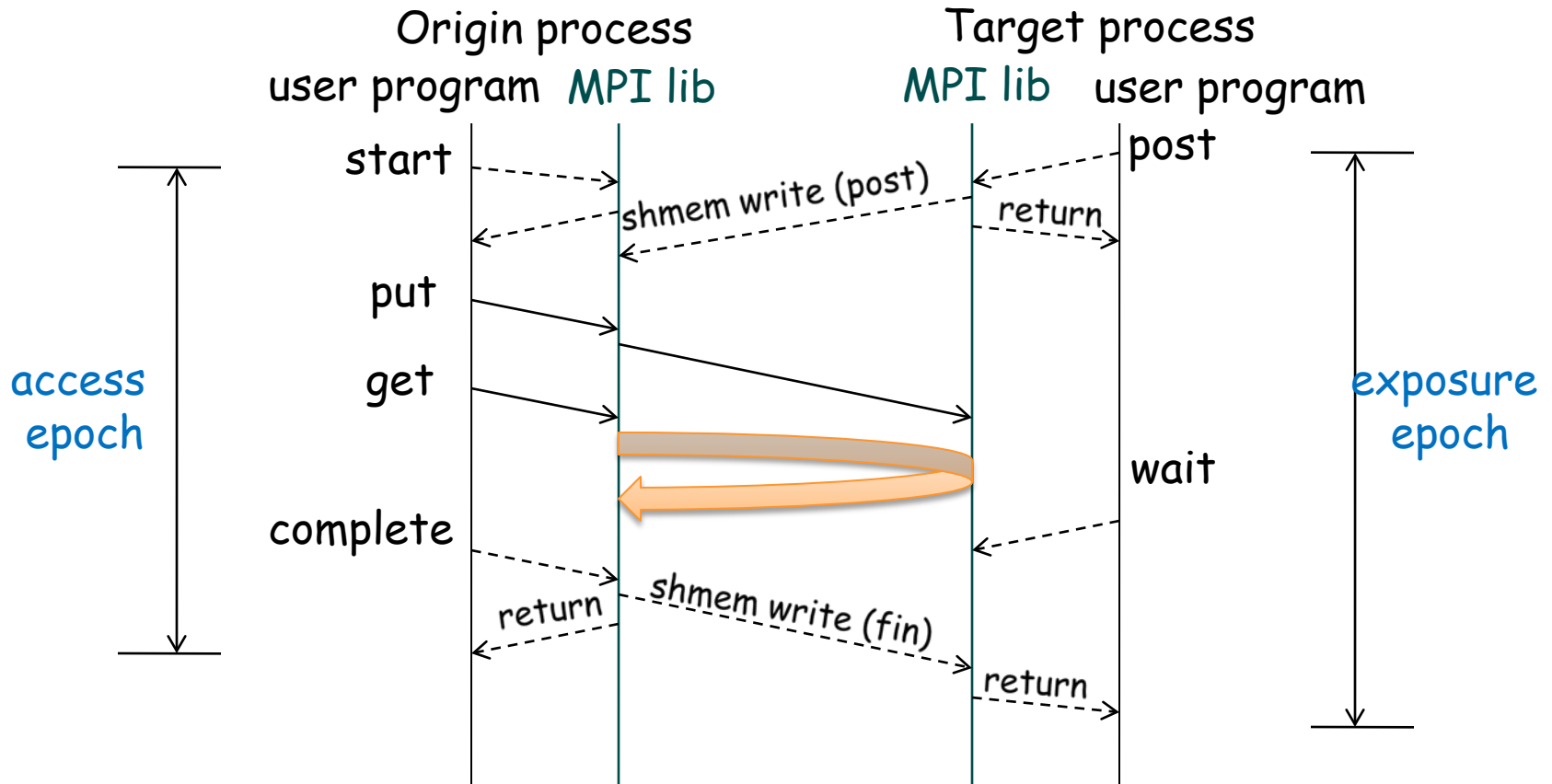


# Intra-node One-sided Communication



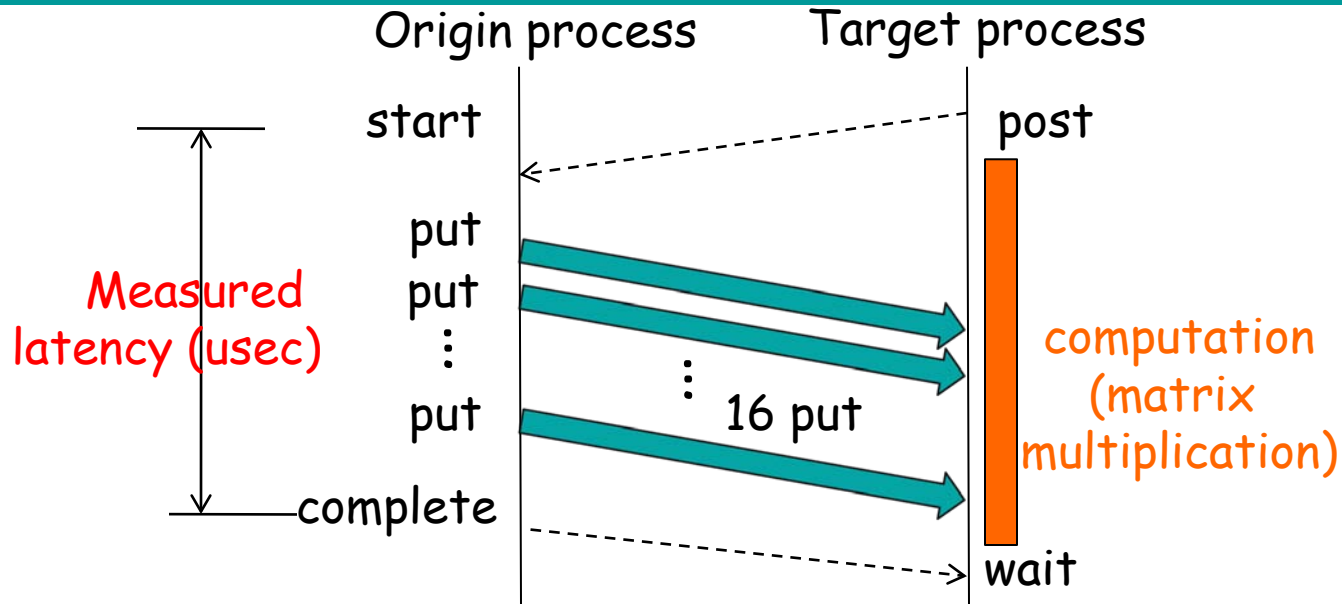
- User-level shared memory techniques lead to two copies
- One copy methods
  - Kernel based (LiMIC2, KNEM)
  - On-board DMA engines, such as Intel I/OAT

# Design Goals for Intra-node RMA Communication



- Realize true one-sided synchronization and data transfer

# Reduced Process Skew



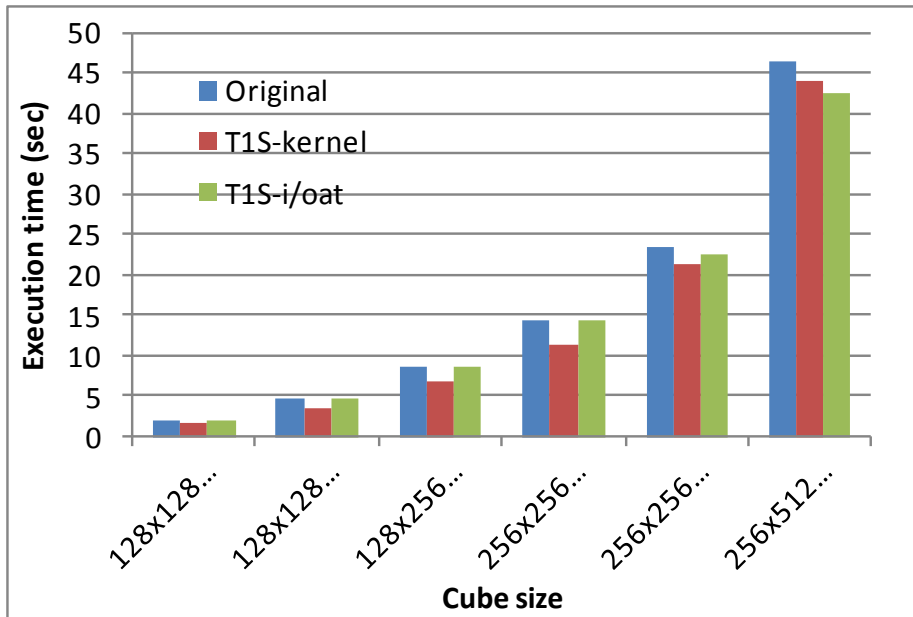
Latency (usec) of 16 put with increasing process skew (message size = 256KB)

Matrix size	no comp	32x32	64x64	128x128	256x256
Original	3404	3780	6126	27023	194467
T1S-kernel	3365	3333	3398	3390	3572
T1S-i/oat	2291	2298	2310	2331	2389

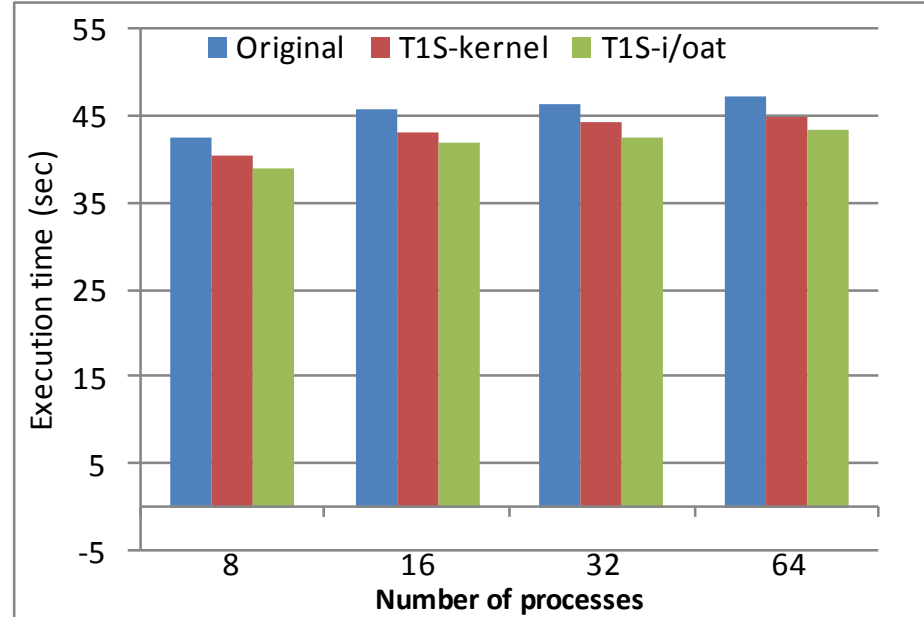
- New designs remove dependency, more robust to process skew

# Application-Level Performance Benefits

Performance with varying data sets (32 processes)



Weak scaling performance (128x128x128 elements per process)

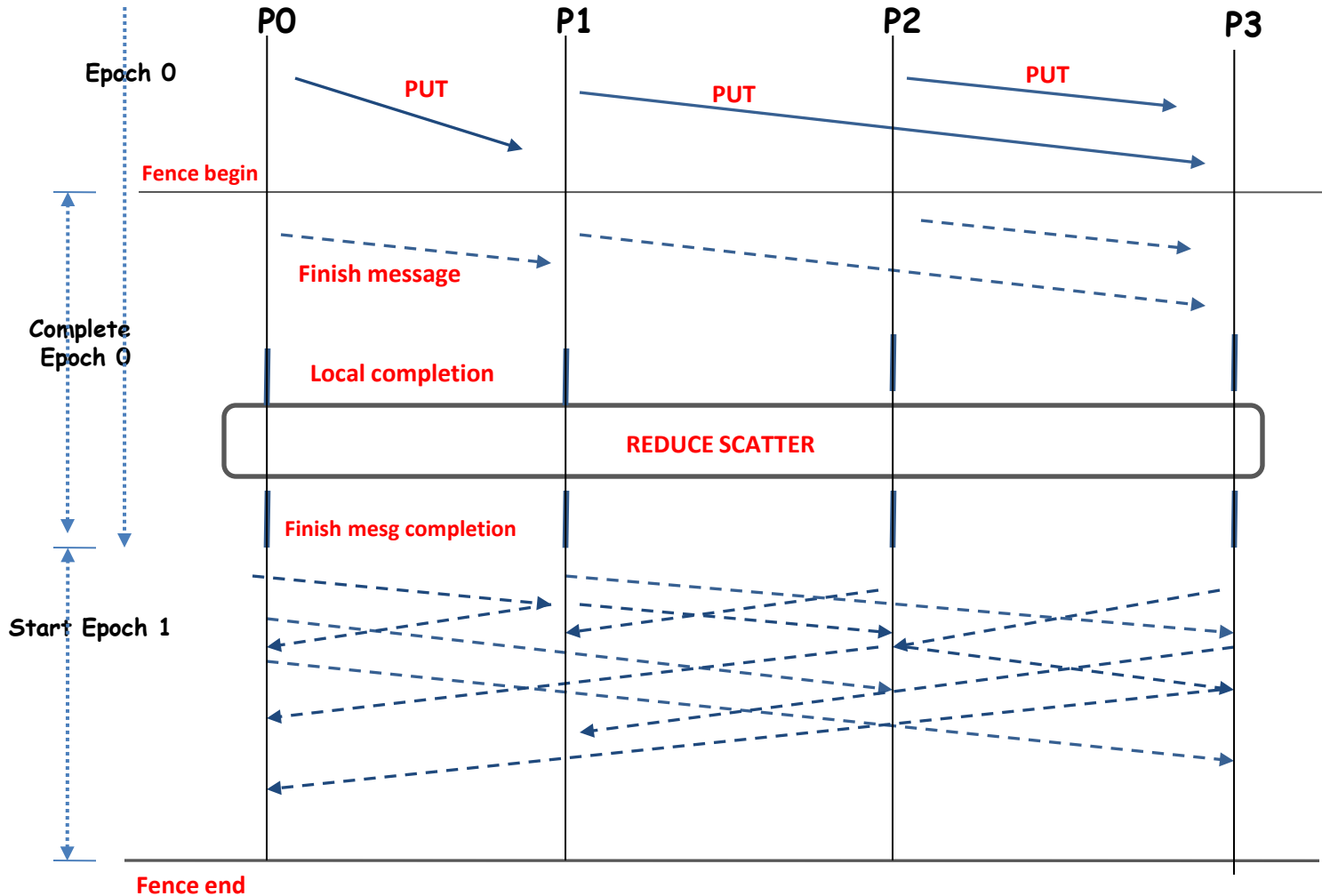


- AWM-Olsen: stencil-based earthquake simulation application
  - Nearest-neighbor communication; performs on 3-dimensional data set
  - Modified it to use MPI-2 one-sided semantics
- New designs show 10% improvement for larger problem sizes

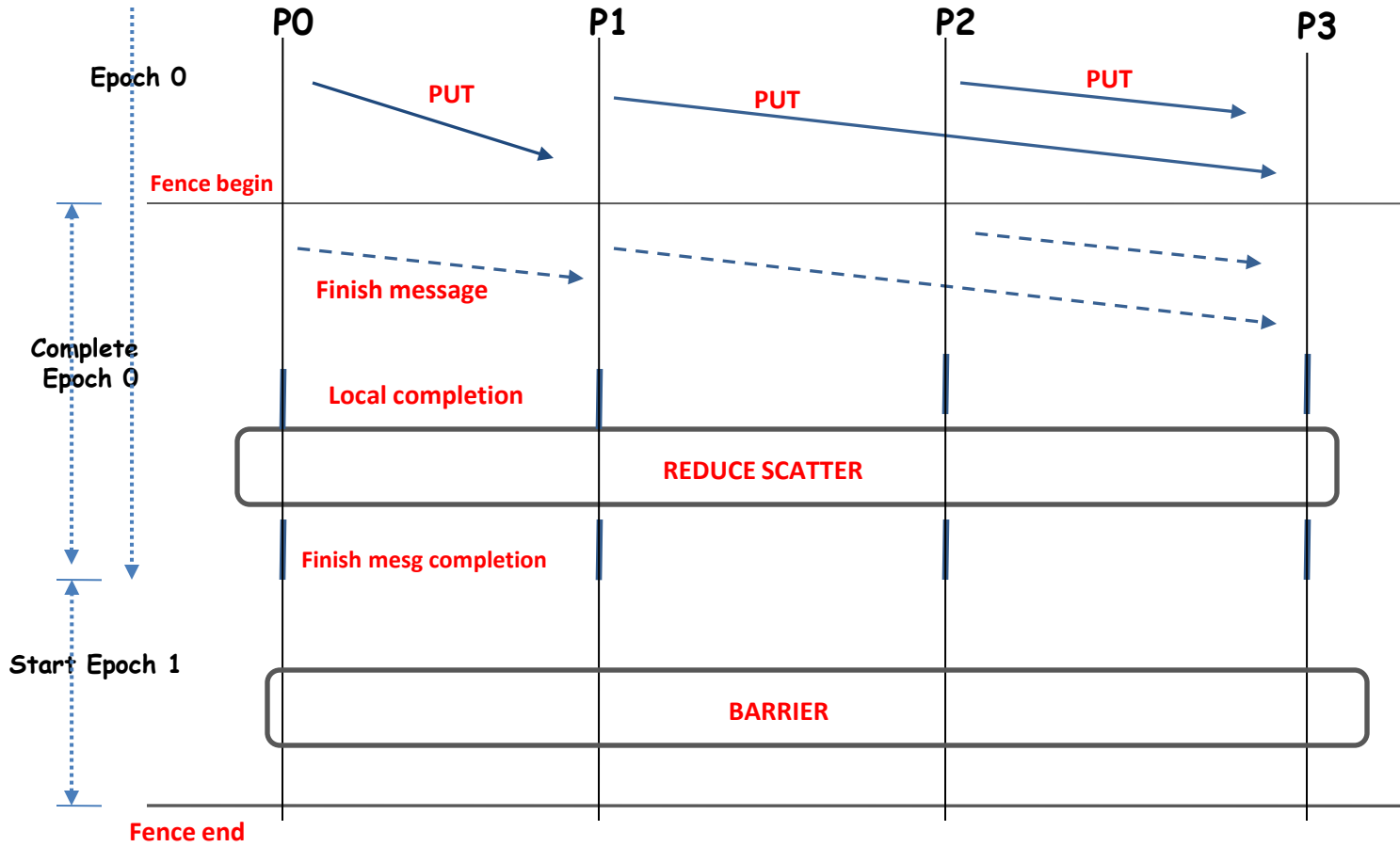
Will be available in the  
MVAPICH2 1.6 Release

P. Lai, S. Sur and D. K. Panda, Designing Truly One-Sided MPI-2 RMA Intra-node Communication on Multi-core Systems, International Computing Conference (ISC'10), May-June 2010. Best Paper Award at ISC '10.

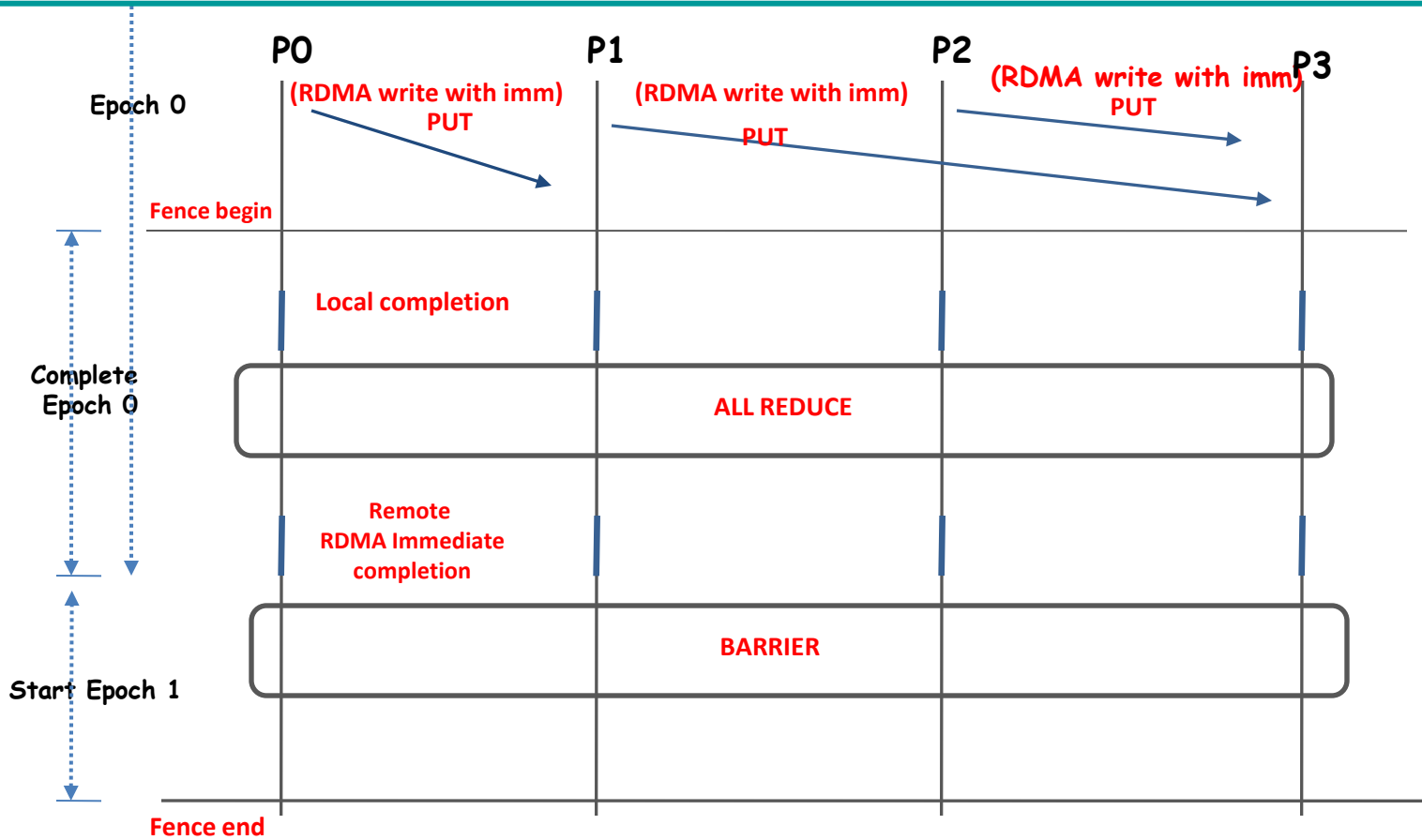
# Fence-Naive Design (Fence-1S)



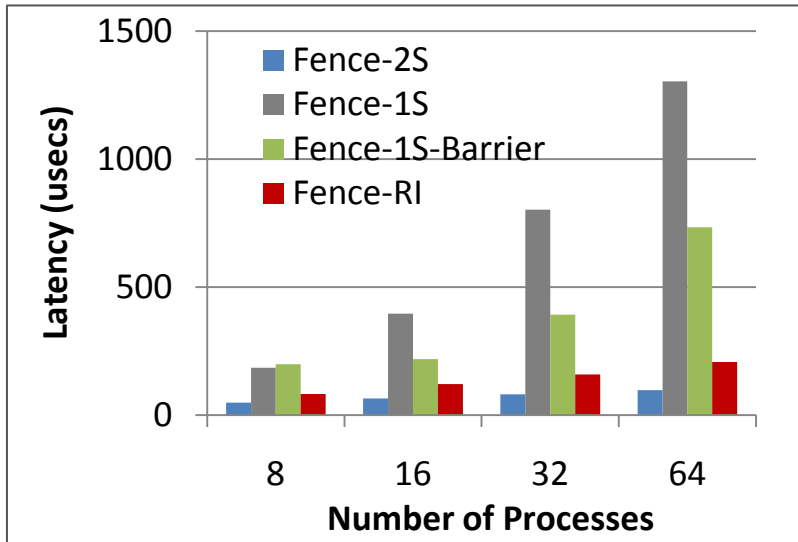
# Fence-Enhanced Design with Barrier (Fence-1S-Barrier)



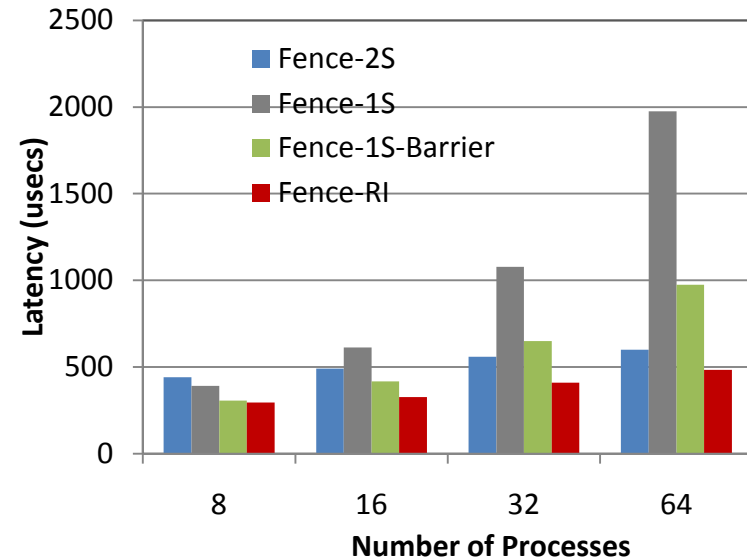
# Fence Design with RDMA Write (Fence-RI)



# Performance of Put Operations and Fence

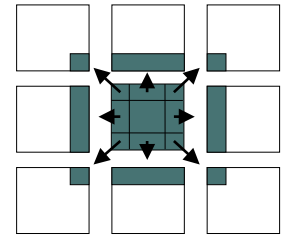


## Halo Communication Performance



- Fence-RI performs better than Fence-1S barrier
- Fence-2S still performs the best
  - However poor overlap capability

- Mimics halo or Ghost cell update
- The Fence-RI scheme performs the best



G. Santhanaraman, T. Gangadharappa, S. Narravula, A. Mamidala and D. K. Panda, Design Alternatives for Implementing Fence Synchronization in MPI-2 One-sided Communication on InfiniBand Clusters, IEEE Cluster 2009, September 2009.

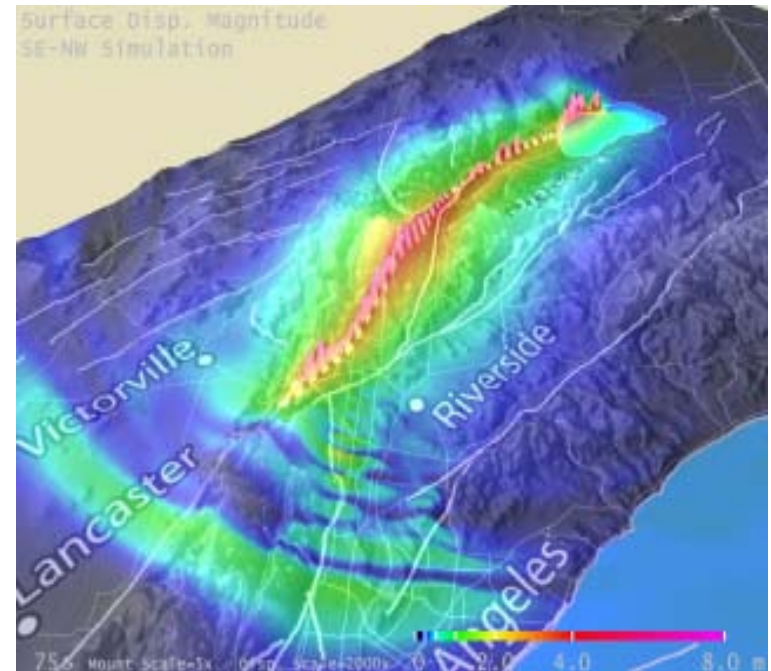
Will be available in Future MVAPICH2 Release

# RMA Usage in Real World Applications

- The important questions are:
  - *Given a new MPI feature, how do I modify my application to effectively leverage the feature?*
  - *Are these changes simple to perform? Or do they force me to inspect the application algorithm closely?*
- The ground reality is that systems, platforms are gradually getting more and more complex
- We are combining the knowledge of application scientists, MPI library designers and system deployment experts to re-design some representative applications to leverage MPI-2 features

# First study with AWP-ODC Seismic Simulation Application

- AWM-ODC, a widely used seismic modeling application
- Runs on 100s of thousands of cores
- Consumes millions of CPU hours every year on the TeraGrid
- Uses MPI-1, spends up to 30% of time in communication progress
- Shows potential for improvement through overlap



Shakeout Earthquake Simulation  
*Visualization credits:* Amit Chourasia,  
Visualization Services, SDSC  
*Simulation credits:* Kim Olsen et. al. SCEC,  
Yifeng Cui et. al., SDSC

# AWP-ODC: Application Pseudo-code

```
MAIN LOOP IN AWP-ODC
do i = timestep0, timestepN
  Compute Velocities - Tv
  Swap Velocity data with neighboring sub-grids - Tcv
  Compute Stresses - Ts
  Swap Stress data with neighboring sub-grids Tcs
enddo
```

```
SWAP VELOCITY DATA
North and South Exchange
s2n(u1,north-mpirank, south-mpirank)
  recv from south, send to north
n2s(u1, south-mpirank, north-mpirank)
  send to south, recv from north
. . . repeat for velocity components v1,w1
East and West Exchange
w2e(u1,west-mpirank, east-mpirank)
  recv from west, send to east
e2w(u1, east-mpirank, west-mpirank)
  send to west, recv from east
. . . repeat for velocity components v1,w1
Up and Down Exchange
```

```
...
S2N
Copy 2 planes of data from variable to send buffer
copy north boundary excluding the ghost cells
MPI_Isend(sendbuffer, north-mpirank)
MPI_Irecv(recvbuffer, south-mpirank)
MPI_Waitall()
Copy 2 planes of data from recvbuffer to variable
copy into south ghost cells
```

main loop takes most of the time

swap velocity and stress  
for each dimension,  
**blocking for completion to  
each neighbor**

data needs to be copied into bounce  
buffers to send multi-dimensional  
data

# Overlap using MPI-2 RMA

```
MPI_Win_post(group, 0, window)
```

```
! pre-posting the window to all neighbors
```

## MAIN LOOP IN AWP-ODC

```
  Compute velocity component u  
  Start exchanging velocity component u  
  Compute velocity component v  
  Start exchanging velocity component v  
  Compute velocity component w  
  Start exchanging velocity component w  
  Complete Exchanges of u,v and w  
  MPI_Win_post(group, 0, window)  
  ! For the next iteration
```

## START EXCHANGE

```
  MPI_Win_start(group, 0, window)  
  s2n(u1,north-mpirank, south-mpirank)  
  ! recv from south, send to north  
  n2s(u1, south-mpirank, north-mpirank)  
  ! send to south, recv from north  
  . . . repeat for east-west and up-down
```

## COMPLETE EXCHANGE

```
  MPI_Win_complete(window)  
  MPI_Win_wait(window)  
  s2nfill(u1, window buffer, south-mpirank)  
  n2sfill(u1, window buffer, north-mpirank)  
  . . . repeat for east-west and up-down
```

## S2N

```
  Copy 2 planes of data from variable to sendbuffer  
  ! copy north boundary excluding ghost cells  
  MPI_Put(sendbuffer, north-mpirank)
```

## S2NFILL

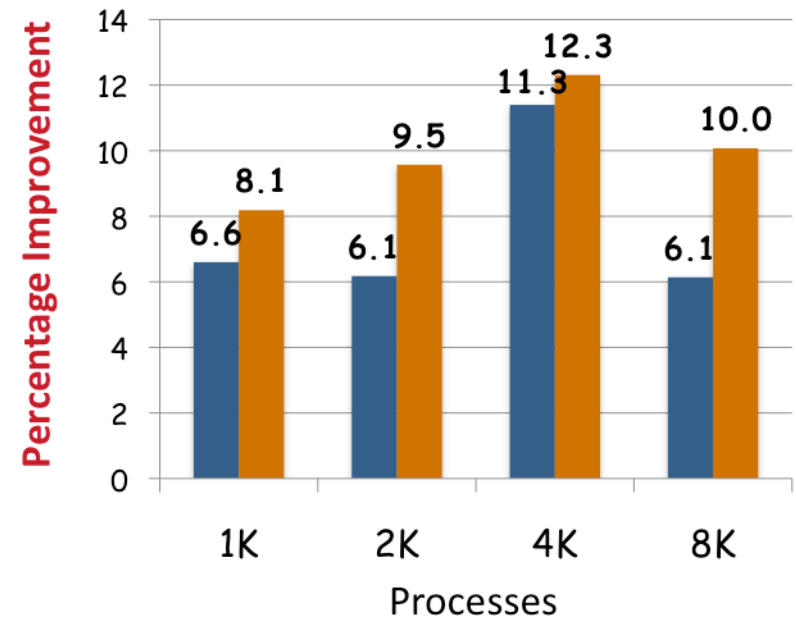
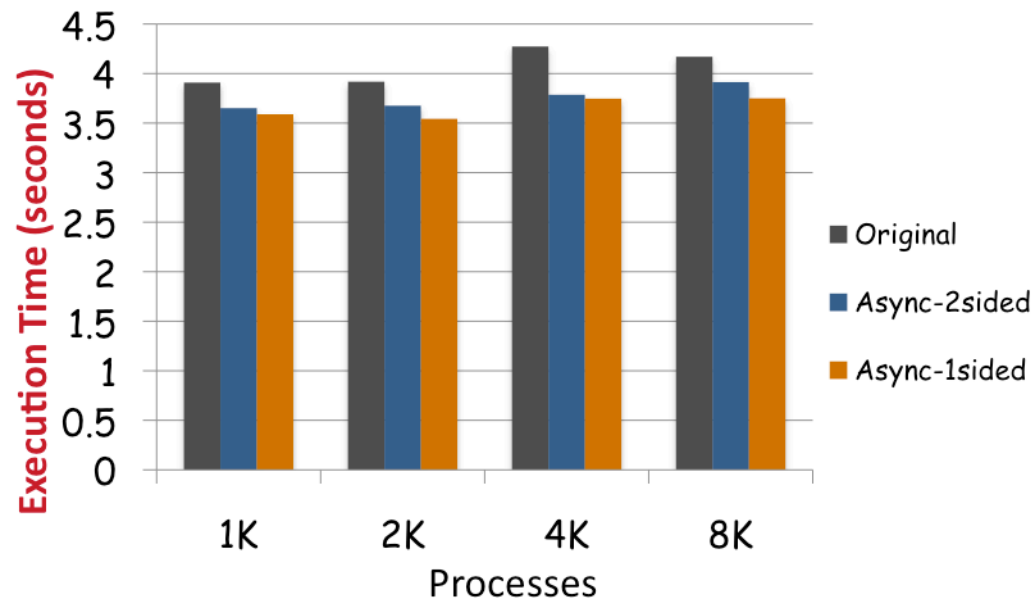
```
  Copy 2 planes of data from window buffer to variable  
  ! copy into south ghost cells
```

pre-post window (combined: u, v, w)

post starts and issue non-blocking  
MPI\_Put

issue completes and waits to finish

# Experimental Results with One-Sided Design and Overlap



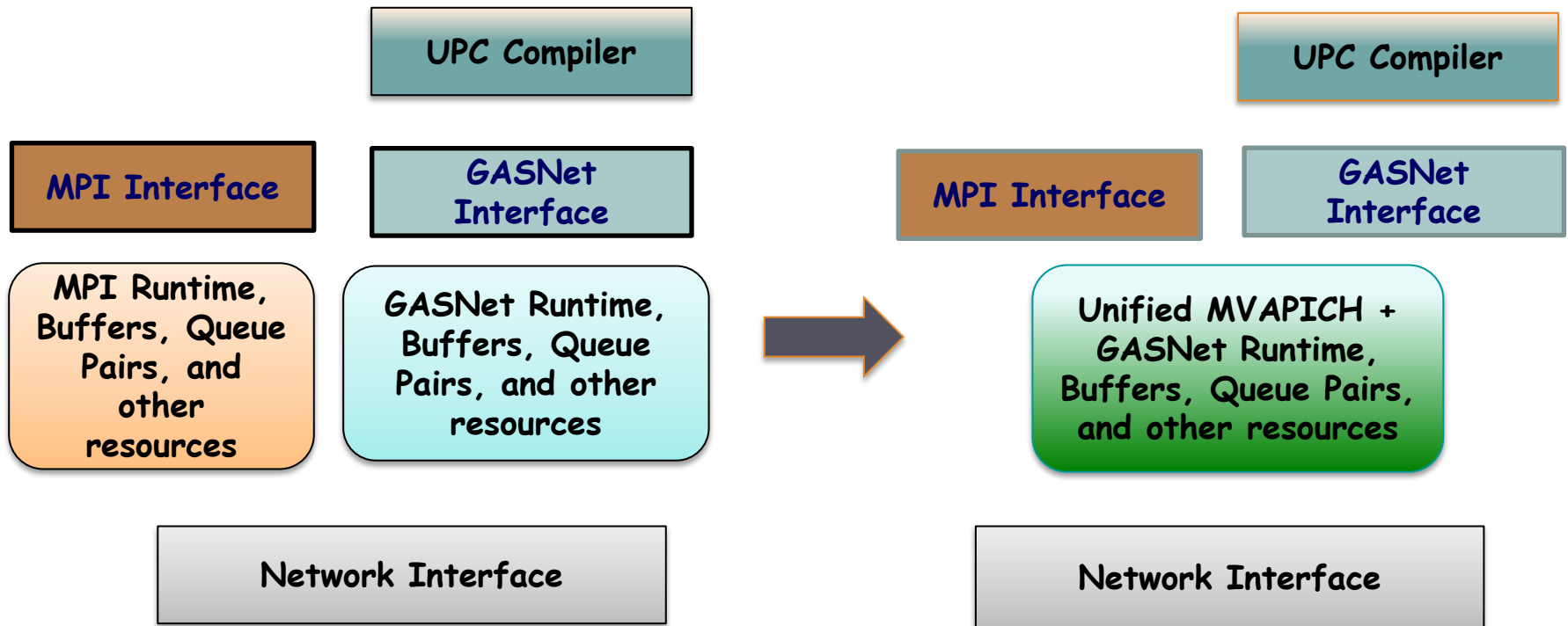
- Experiments on TACC Ranger cluster – 64x64x64 data grid per process – 25 iterations – 32KB messages
- On 4K processes – 11% with Async-2sided and 12% with Async-1sided (RMA)
- On 8K processes – 6% with Async-2sided and 10% with Async-1sided (RMA)

**Joint work with OSU, SDSC and TACC  
Gordon-Bell Finalist for Supercomputing 2010**

# Advanced Designs for Collectives, One-Sided and PGAS

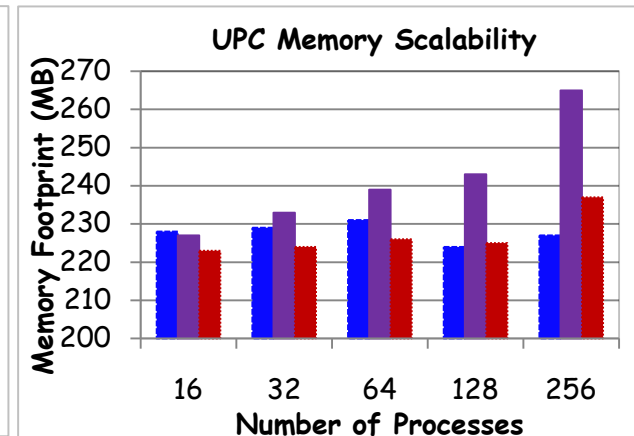
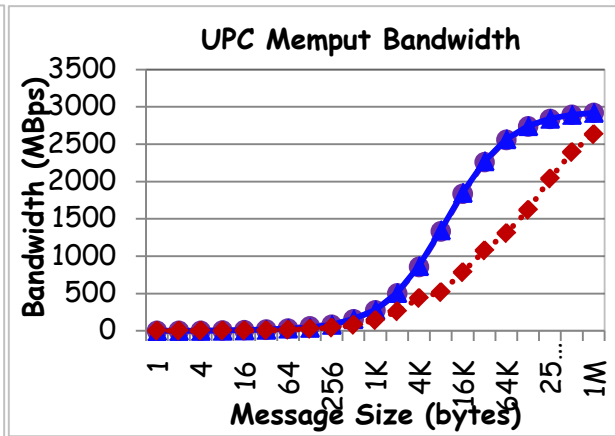
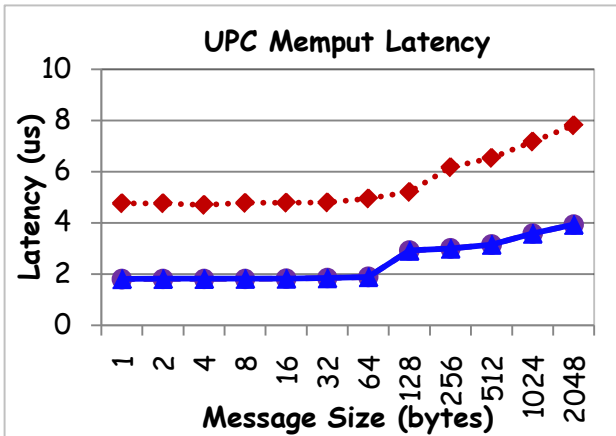
- Collective Communication
  - Multi-core Aware
  - Topology-aware
  - Power-aware
  - Exploiting Collective Offload
    - Non-blocking collectives
- One-sided communication
  - Basic inter-node performance
  - Synchronization
  - Intra-node design
  - Application Evaluation
- Partitioned Global Address (PGAS) Support

# Unifying UPC and MPI Runtimes: Experience with MVAPICH



- Currently UPC and MPI do not share runtimes
  - Duplication of lower level communication mechanisms
  - GASNet unable to leverage advanced buffering mechanisms developed for MVAPICH
- Our novel approach is to enable a truly unified communication library

# UPC Micro-benchmark Performance



MVAPICH-conduit    IBV-conduit    MPI-conduit

- BUPC micro-benchmarks from latest release 2.10.2
- UPC performance is identical with both native IBV layer and new MVAPICH conduits
- Performance of MPI conduit is not very good
  - Mismatch of MPI specification and Active messages
- MVAPICH conduit is more scalable compared native IB conduit

Will be available in future  
MVAPICH2 Release

J. Jose, M. Luo, S. Sur and D. K. Panda, "Unifying UPC and MPI Runtimes: Experience with MVAPICH", International Conference on Partitioned Global Address Space (PGAS), 2010

# Concluding Remarks

- InfiniBand is gaining momentum in HPC systems with best performance and greater usage
- As larger-scale clusters are getting deployed, new solutions are needed for collectives, one-sided and PGAS support
- Demonstrated how such solutions can be designed with MVAPICH2 and demonstrated their performance benefits
- Upcoming and future MVAPICH2 releases will have these solutions

# Funding Acknowledgments

## Funding Support by



## Equipment Support by



# Personnel Acknowledgments

## *Current Students*

- N. Dandapanthula (M.S.)
- J. Jose (Ph.D.)
- J. Huang (Ph.D.)
- K. Kandalla (M.S.)
- P. Lai (Ph.D.)
- M. Luo (Ph.D.)
- S. Pai (M.S.)
- V. Meshram (M.S.)
- X. Ouyang (Ph.D.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- H. Subramoni (Ph.D.)

## *Past Students*

- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- P. Lai (Ph. D.)
- J. Liu (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- G. Santhanaraman (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

## *Current Research*

### *Scientist*

- S. Sur

## *Current Post-Docs*

- X. Besseron
- H. Wang
- J. Vienne

## *Past Post-Docs*

- E. Mancini
- S. Marcarelli
- H.-W. Jin

## *Current*

### *Programmers*

- D. Bureddy
- M. Arnold
- J. Perkins

# Web Pointers



**MVAPICH**

MVAPICH Web Page  
<http://mvapich.cse.ohio-state.edu/>

E-mail: [panda@cse.ohio-state.edu](mailto:panda@cse.ohio-state.edu)