

High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A study with Parallel 3DFFT

Krishna Kandalla⁽¹⁾, Hari Subramoni⁽¹⁾,
Karen Tomko⁽²⁾, Dmitry Pekurovsky⁽³⁾,
Sayantan Sur⁽¹⁾ and Dhabaleswar. K. Panda⁽¹⁾

⁽¹⁾Computer Science & Engineering Department,
The Ohio State University

⁽²⁾ The Ohio Supercomputer Center

⁽³⁾ San Diego Supercomputer Center

Outline

- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

Introduction

- Parallel applications can scale beyond 100,000 cores
- InfiniBand is commonly used across commodity clusters
- Message Passing Interface (MPI) is the de-facto programming model



**Tsubame
Supercomputer
73,278 cores**

Collective Communication in MPI

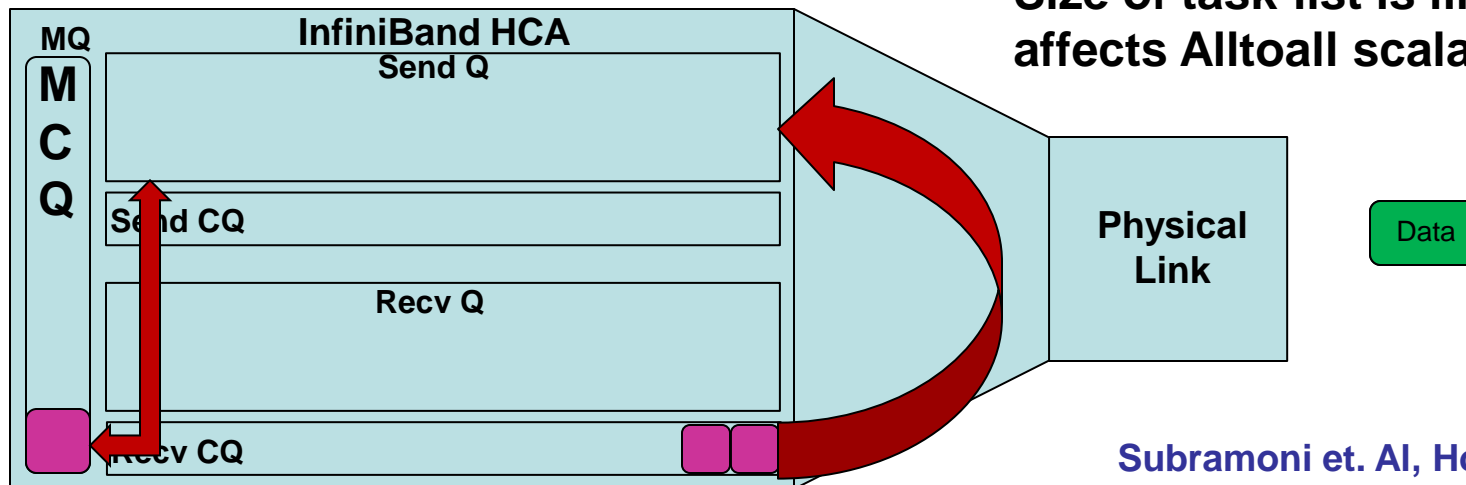
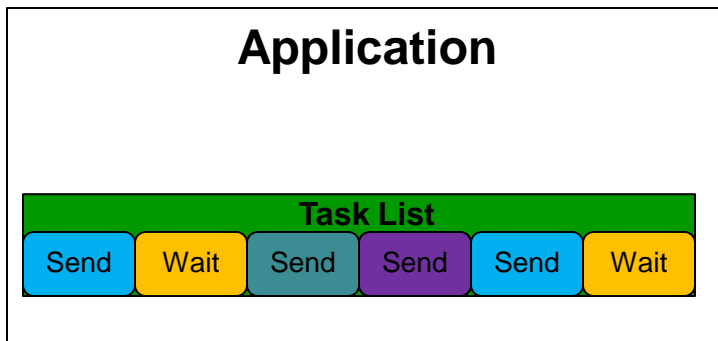
- MPI-2.2 defines blocking collective operations – limits performance and scalability of dense operations (Alltoall)
- MPI-3 may support non-blocking collectives
- Hoefler et. al, proposed host-based approaches
- Latest ConnectX-2 adapters from Mellanox supports network offload features
- Study benefits with real scientific libraries – P3DFFT

Overview of InfiniBand Collective Offload

- Applications can offload task-lists to the NIC
- A CQE gets created on the MCQ after execution

Problems:

- Alltoall is extremely communication intensive
- Size of task-list is limited – Directly affects Alltoall scalability

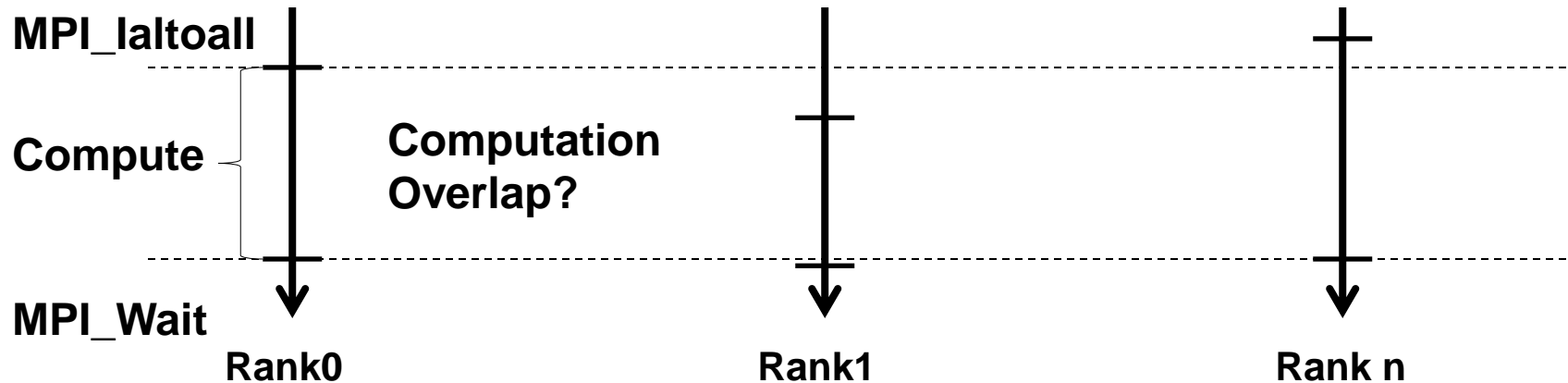


Subramoni et. al, Hot Interconnects 2010

Outline

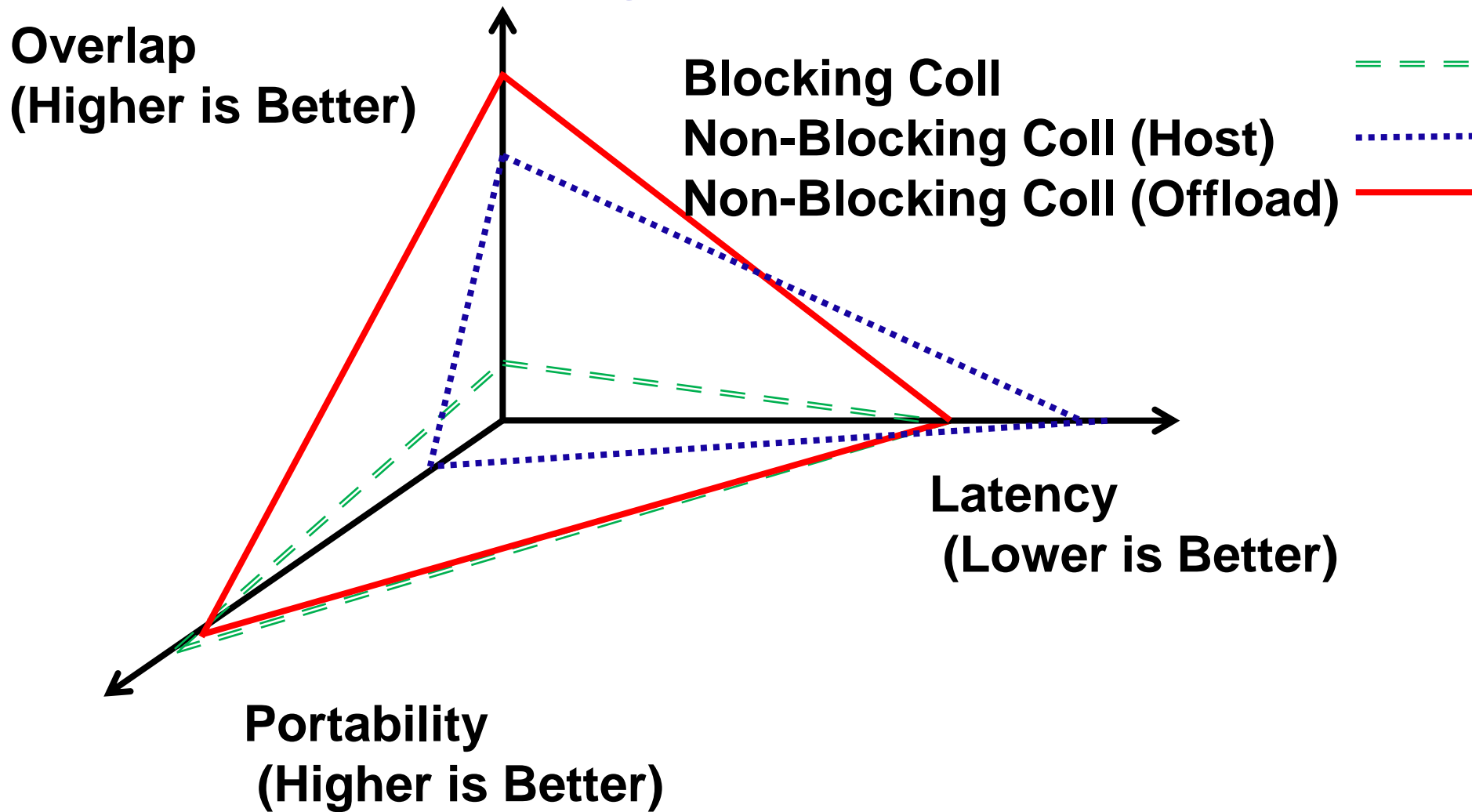
- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

Motivation



- Challenge: **progress the collective schedules in an asynchronous manner *with*:**
 - **performance portability**
 - **minimal host processor intervention**
 - **acceptable communication latency**

Design Space of Collective Algorithms



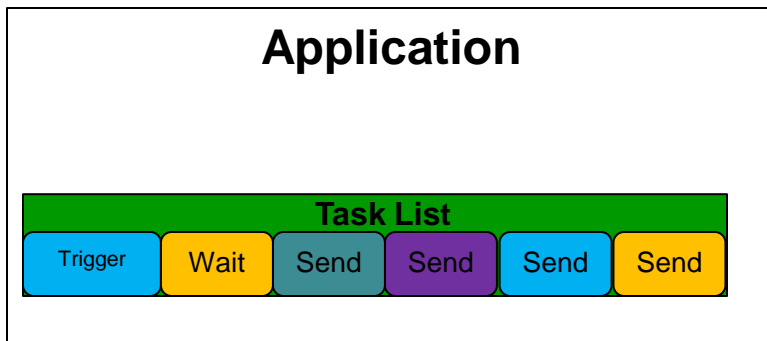
Problem Statement

- Can we leverage network offload to design MPI_Ialltoall?
- Will network offload help overlap with collectives?
- Can network offload improve application throughput?
- Can we re-design scientific libraries (such as 3DFFT) to leverage our proposed MPI_Ialltoall?

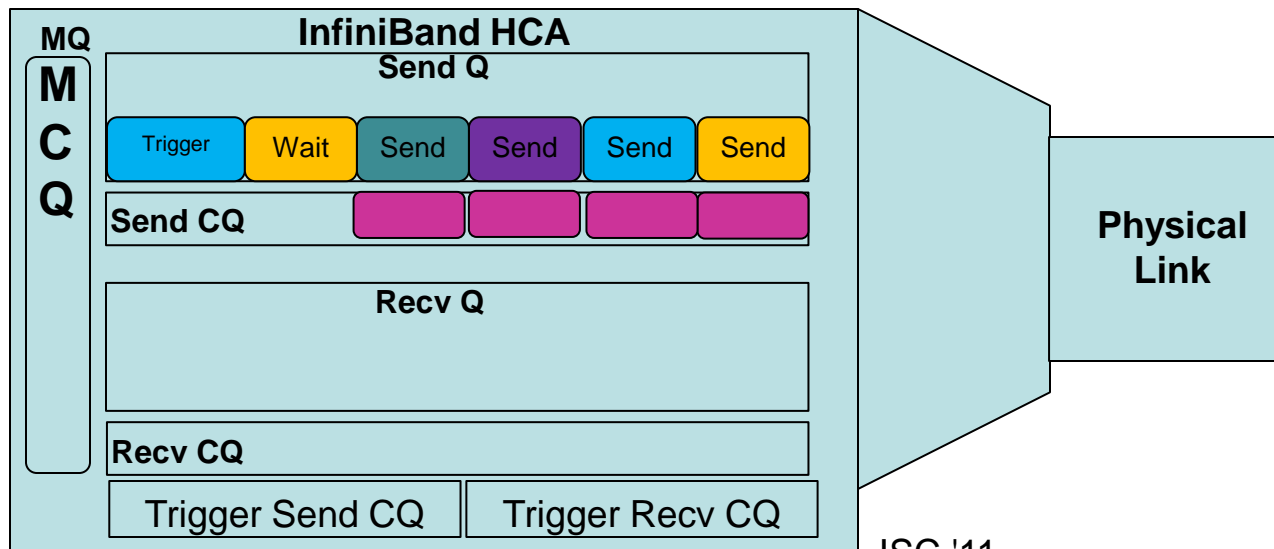
Outline

- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

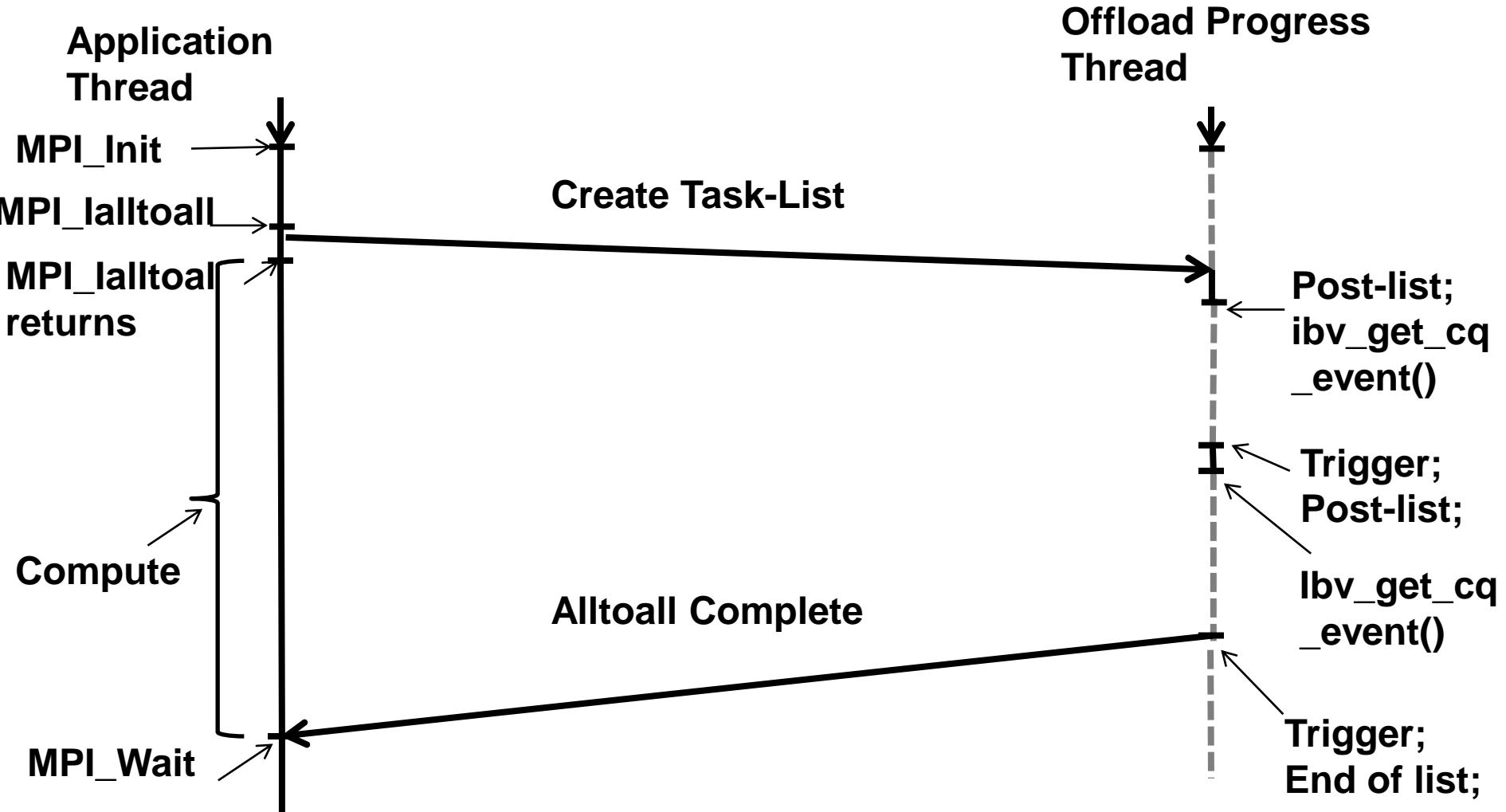
Creating Task-lists with the “trigger” operation



- Task list with multiple phases
- A phase has send, wait & trigger tasks
- Progress thread calls “ibv_get_cq_event()” and blocks
- Trigger task generates an interrupt which signals the progress thread



Designing Scalable Offload Alltoall



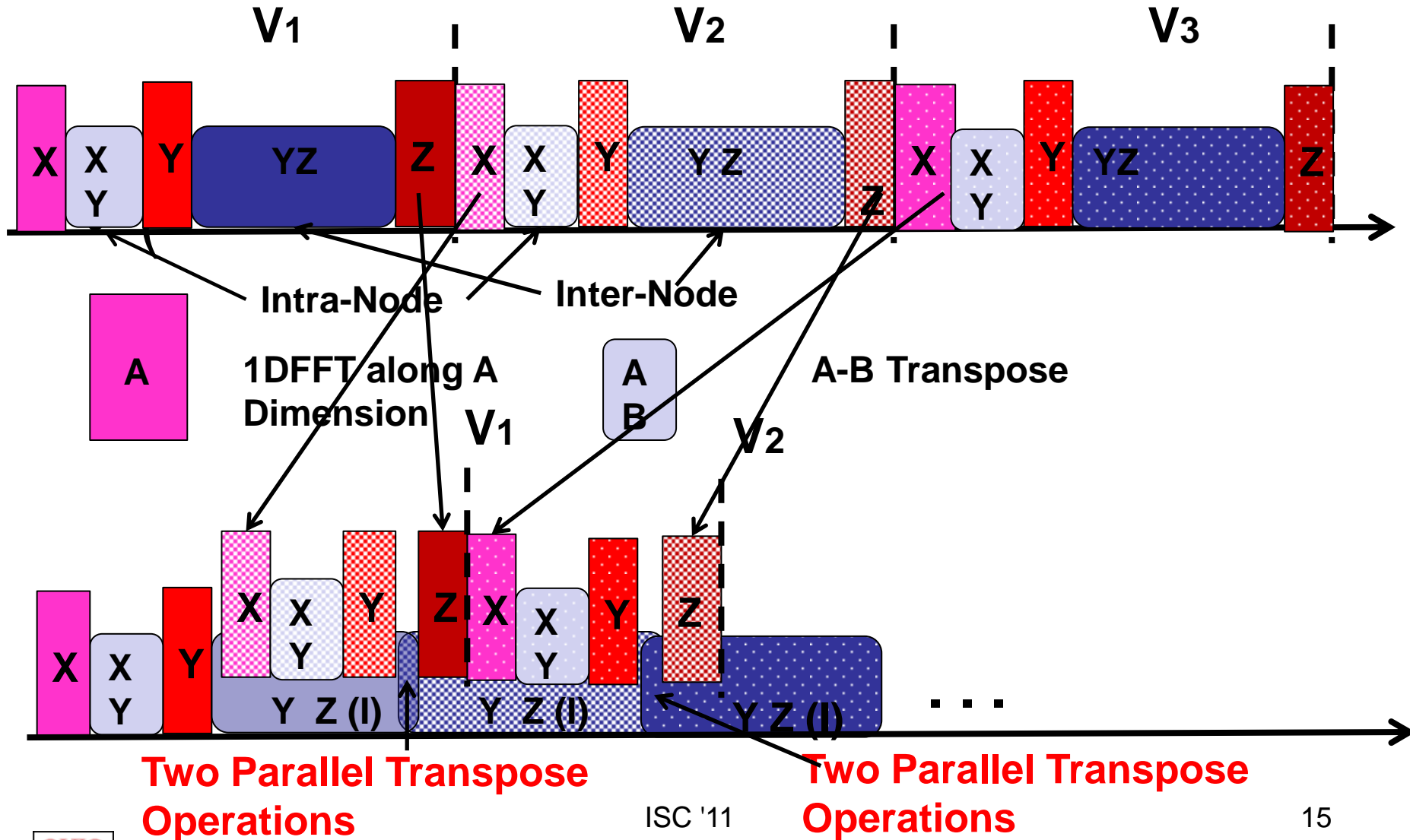
Outline

- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

Parallel 3DFFT Library

- Applications in areas related to simulations of turbulence, Astrophysics etc. rely heavily on 3DFFT
- P3DFFT from San Diego Supercomputer Center (SDSC) is a portable, high performance implementation of 3DFFT (<http://code.google.com/p/p3dffft/>)
- P3DFFT uses a 2D *pencil* decomposition to maximize parallelism
- P3DFFT relies on expensive large message Alltoall operations to implement the transpose operations

Re-designing P3DFFT for Overlap



Outline

- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

Experimental Setup

- Intel Xeon E5640 (2.53 GHz), 12 GB memory per node
- MT26428 QDR ConnectX-2 with PCI-Ex interfaces, 171-port Mellanox QDR switch, OFED 1.5.1
- RHEL 5.4, 2.6.18-164.e15 kernel version
- MVAPICH2 – A High Performance MPI implementation over InfiniBand and other RDMA networks (v1.6)
 - <http://mvapich.cse.ohio-state.edu/>
 - Used by more than 1580 organizations world-wide

Micro-Benchmark Evaluations

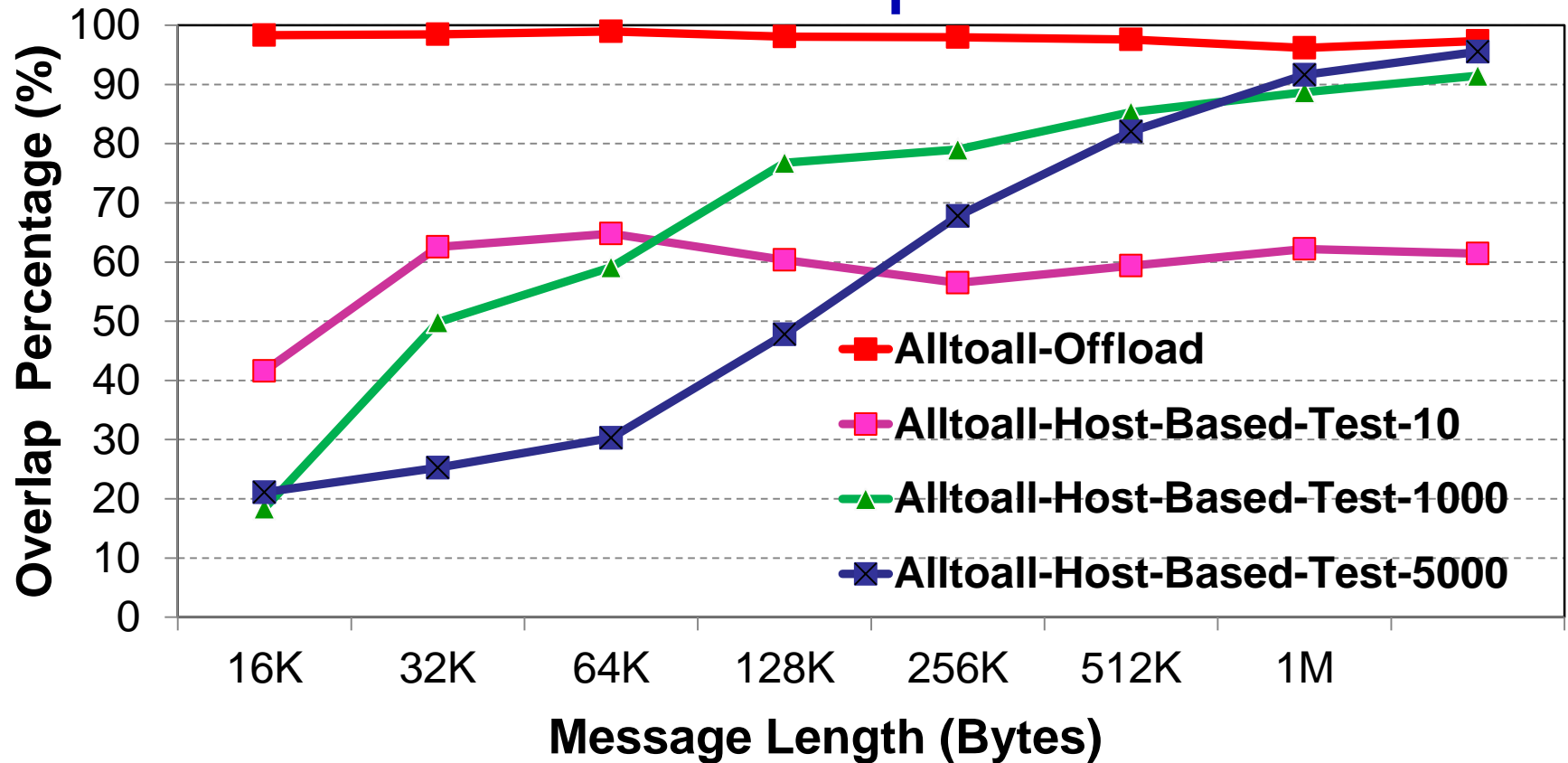
Overlap Benchmark

- Measure average MPI_Ialltoall latency
- Overlap Percentage:
start_overlap_timer()
MPI_Ialltoall(..)
while(time < alltoall_latency)
 update timer
MPI_Wait(..)
end_overlap_timer()

Throughput Benchmark

```
start_throughput_timer()  
MPI_Ialltoall(..)  
CBLAS_DGEMM()  
MPI_Wait(..)  
end_throughput_timer()
```

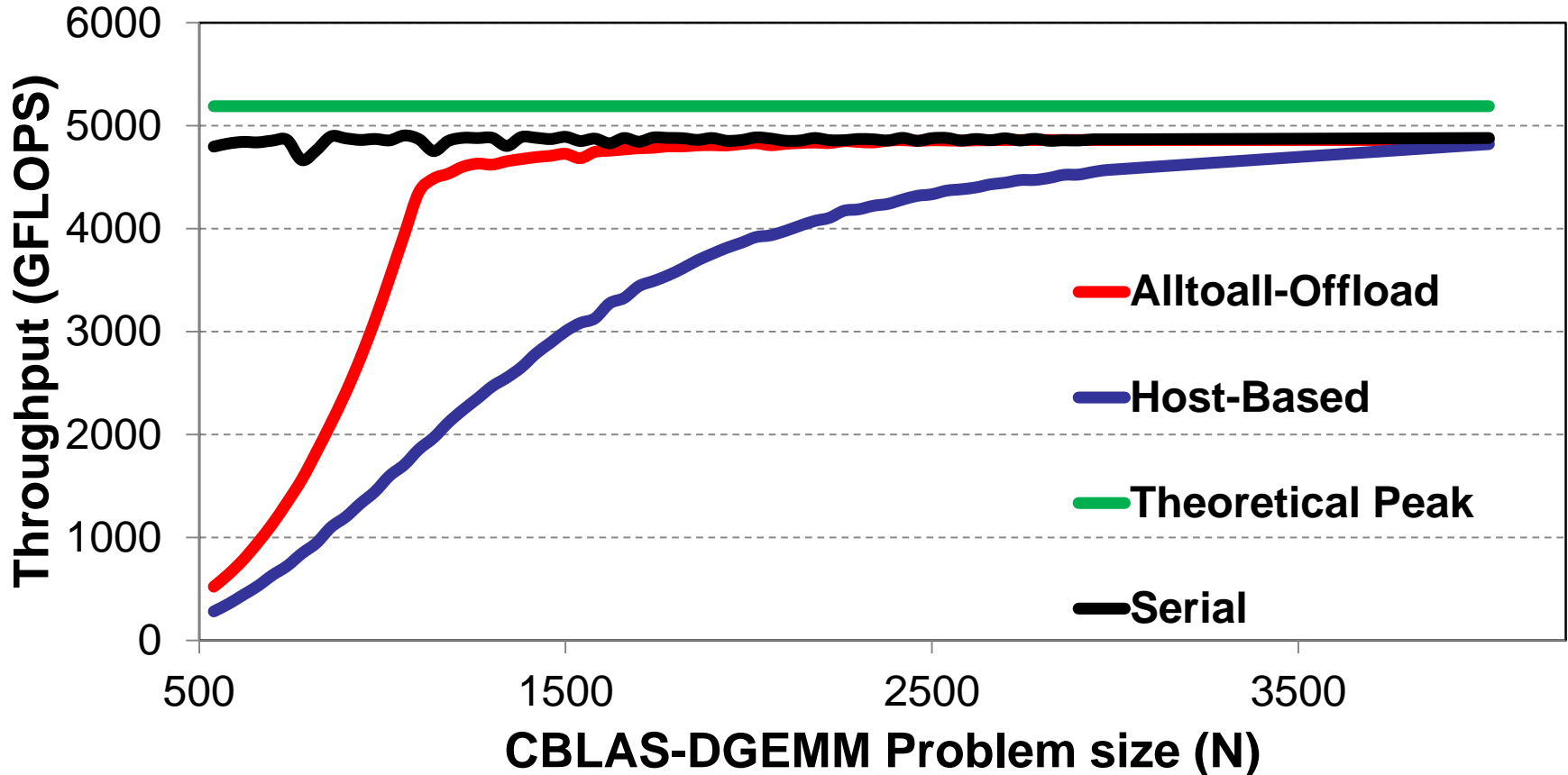
Communication/Computation Overlap



Alltoall Overlap Comparison with 256 Processes

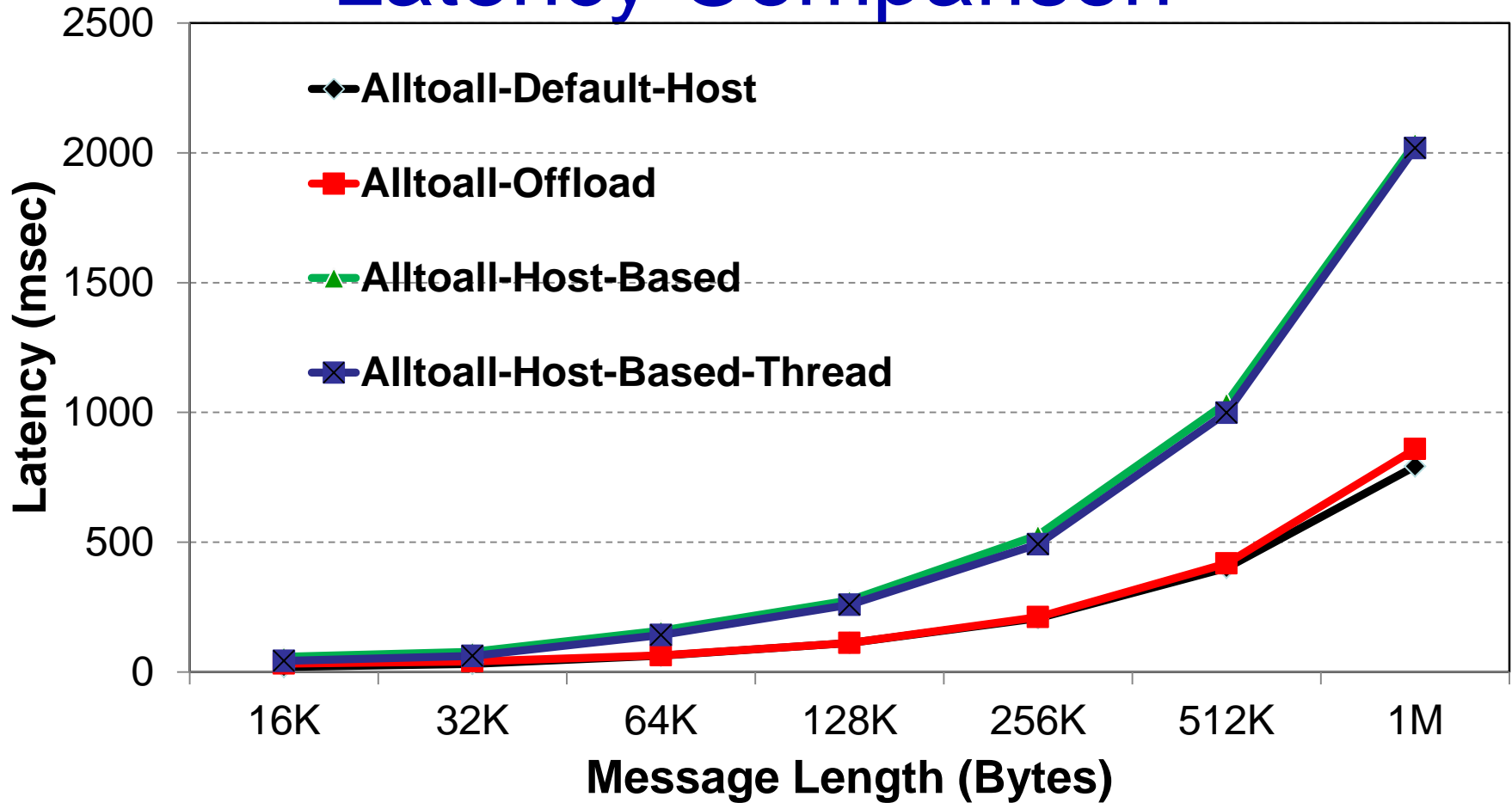
Alltoall-Offload delivers near perfect communication/computation overlap for all messages in a portable manner

DGEMM Throughput Comparison



CBLAS-DGEMM overlapped with Offload-lalltoall delivers better throughput upto **110%** when compared to Host-Based lalltoall with 512 processes

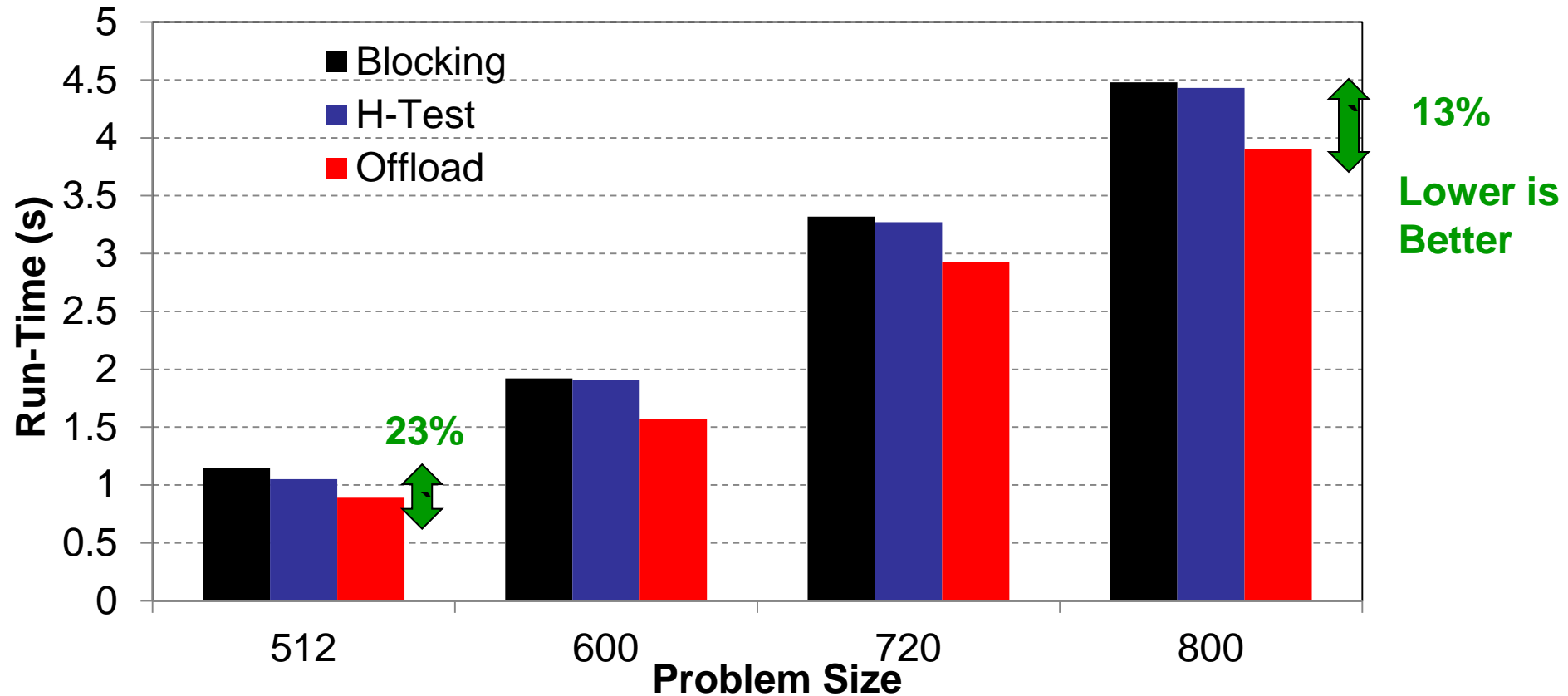
Latency Comparison



Alltoall Latency Comparison with 256 Processes

Alltoall-Offload delivers good overlap, without sacrificing on communication latency!

Parallel 3D FFT Performance



P3DFFT Application kernel run-time Comparison with 128 processes
P3DFFT with Offload-lalltoall performs about **13.5%** better than default
P3DFFT and about **12%** better than P3DFFT with Host-based-Test

Outline

- Introduction
- Problem Statement
- Designing MPI_Ialltoall with Collective Offload
- Re-designing P3DFFT for overlap
- Experimental Evaluation
- Conclusions and Future work

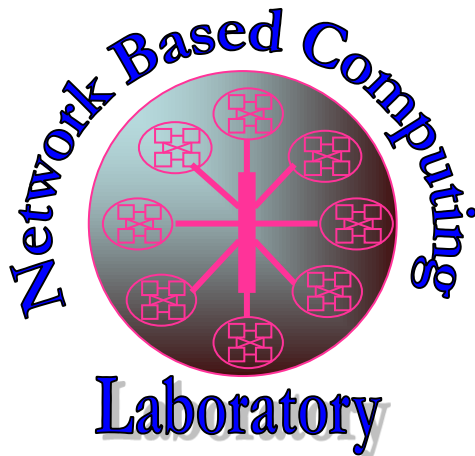
Conclusions and Future Work

- Proposed MPI_lalltoall shows near perfect (99%) overlap
- Throughput of applications improves significantly through offload-based non-blocking collectives
- P3DFFT's run-time improved by up to 23%

Future work:

- Extend Offload-based techniques for other MPI collectives and study their benefits with real applications
- Support for Offload-based collectives will be available in future MVAPICH2 releases

Thank you!



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

(1){kandalla, subramon, surs, panda}@cse.ohio-state.edu

(2)ktomko@osc.edu

(3)dmitry@sdsc.edu

(1)Network-Based Computing Laboratory, Ohio State
University

(2)The Ohio Supercomputer Center

(3)San Diego Supercomputer Center