Exploiting Computation and Communication Overlap in MVAPICH2 and MVAPICH2-GDR MPI Libraries

Talk at

Overlapping Communication with Computation Symposium (April '18)

by

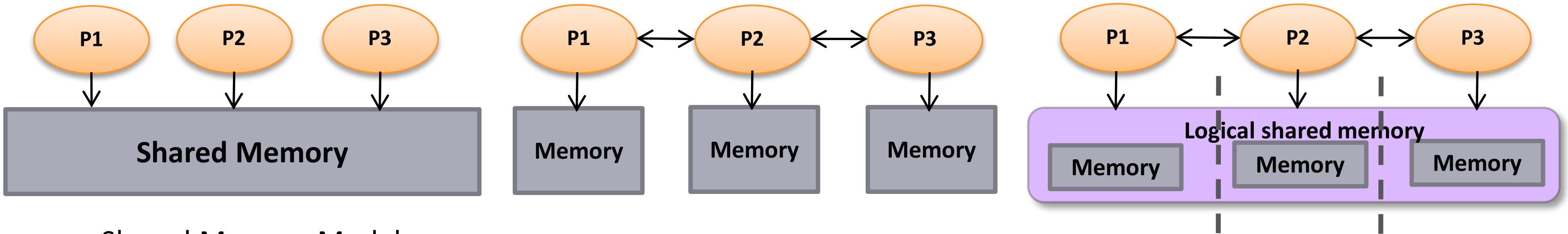
Dhabaleswar K. (DK) Panda

The Ohio State University

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

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

Parallel Programming Models Overview



Shared Memory Model	Distributed Memory Model	Partitioned Global Address Space (PGAS)
SHMEM, DSM	MPI (Message Passing Interface)	Global Arrays, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications

Middleware

Programming Models
MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenMP, OpenACC, Cilk, Hadoop (MapReduce), Spark (RDD, DAG), etc.

Communication Library or Runtime for Programming Models

- Point-to-point Communication
- Collective Communication
- Energy-Awareness
- Synchronization and Locks
- I/O and File Systems
- Fault Tolerance

Networking Technologies
(InfiniBand, 40/100GigE, Aries, and Omni-Path)

Multi-/Many-core Architectures

Accelerators (GPU and FPGA)

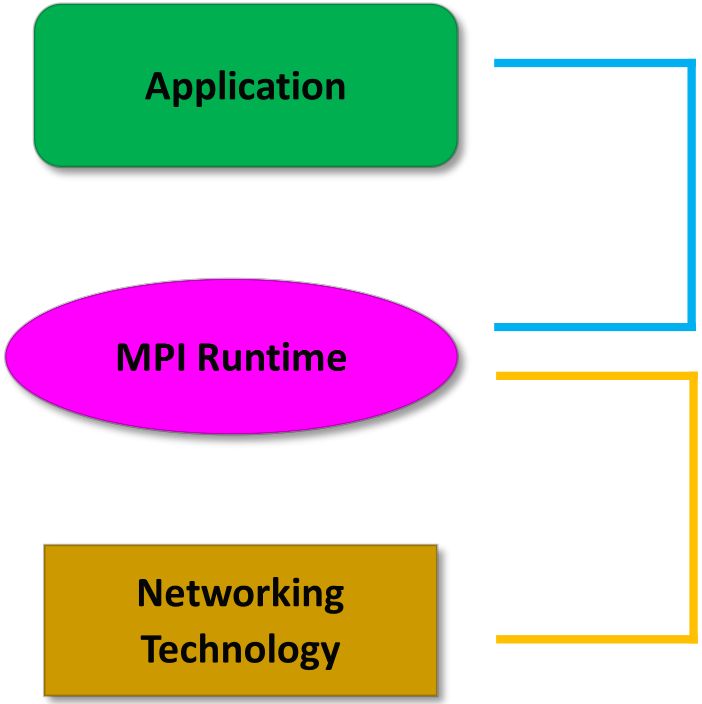
Co-Design Opportunities and Challenges across Various Layers

Performance

Scalability

Resilience

Basic Concept of Overlapping Communication with Computation



Take Advantage of Overlap

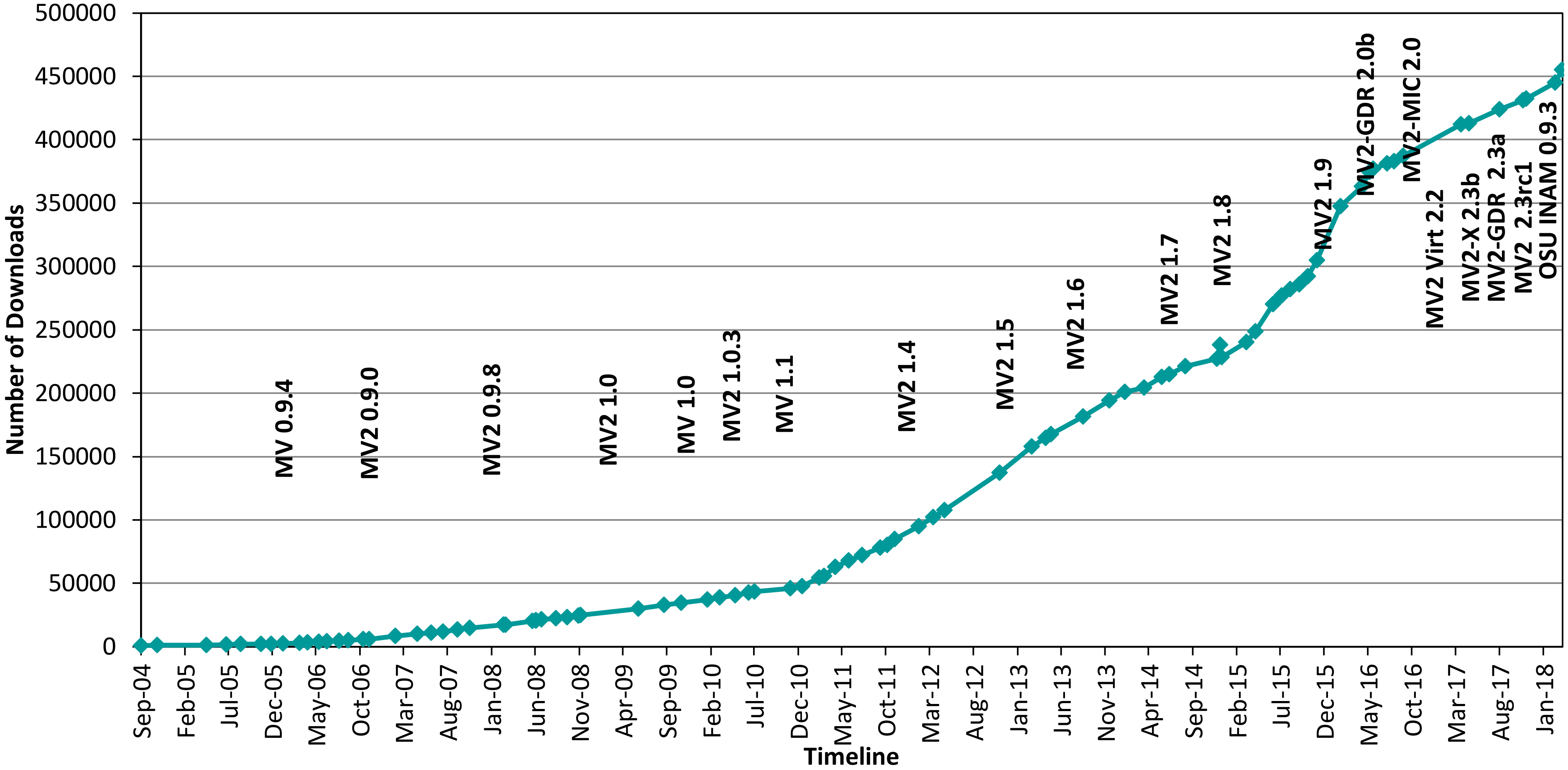
- Transparently
- Co-design

Design MPI Primitives Exploiting Overlap Capabilities of Network Mechanisms

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,875 organizations in 86 countries**
  - **More than 460,000 (> 0.46 million) downloads from the OSU site directl**
  - Empowering many TOP500 clusters (Nov ‘17 ranking)
    - **1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China**
    - 9th, 556,104 cores (Oakforest-PACS) in Japan
    - 12th, 368,928-core (Stampede2) at TACC
    - 17th, 241,108-core (Pleiades) at NASA
    - 48th, 76,032-core (Tsubame 2.5) at Tokyo Institute of Technology
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - **http://mvapich.cse.ohio-state.edu**
- Empowering Top500 systems for over a decade

MVAPICH2 Release Timeline and Downloads



Architecture of MVAPICH2 Software Family

High Performance Parallel Programming Models

- Message Passing Interface (MPI)
- PGAS (UPC, OpenSHMEM, CAF, UPC++)
- Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)

High Performance and Scalable Communication Runtime

Diverse APIs and Mechanisms

- Point-to-point Primitives
- Collectives Algorithms
- Job Startup
- Energy-Awareness
- Remote Memory Access
- I/O and File Systems
- Fault Tolerance
- Virtualization
- Active Messages
- Introspection & Analysis

Support for Modern Networking Technology (InfiniBand, iWARP, RoCE, Omni-Path)

Transport Protocols: RC, XRC, UD, DC

Modern Features: UMR, ODP, SR-IOV, Multi Rail

Support for Modern Multi-/Many-core Architectures (Intel-Xeon, OpenPower, Xeon-Phi, ARM, NVIDIA GPGPU)

Transport Mechanisms: Shared Memory, CMA, IVSHMEM, XPMEM*

Modern Features: MCDRAM*, NVLink*, CAPI*

* Upcoming

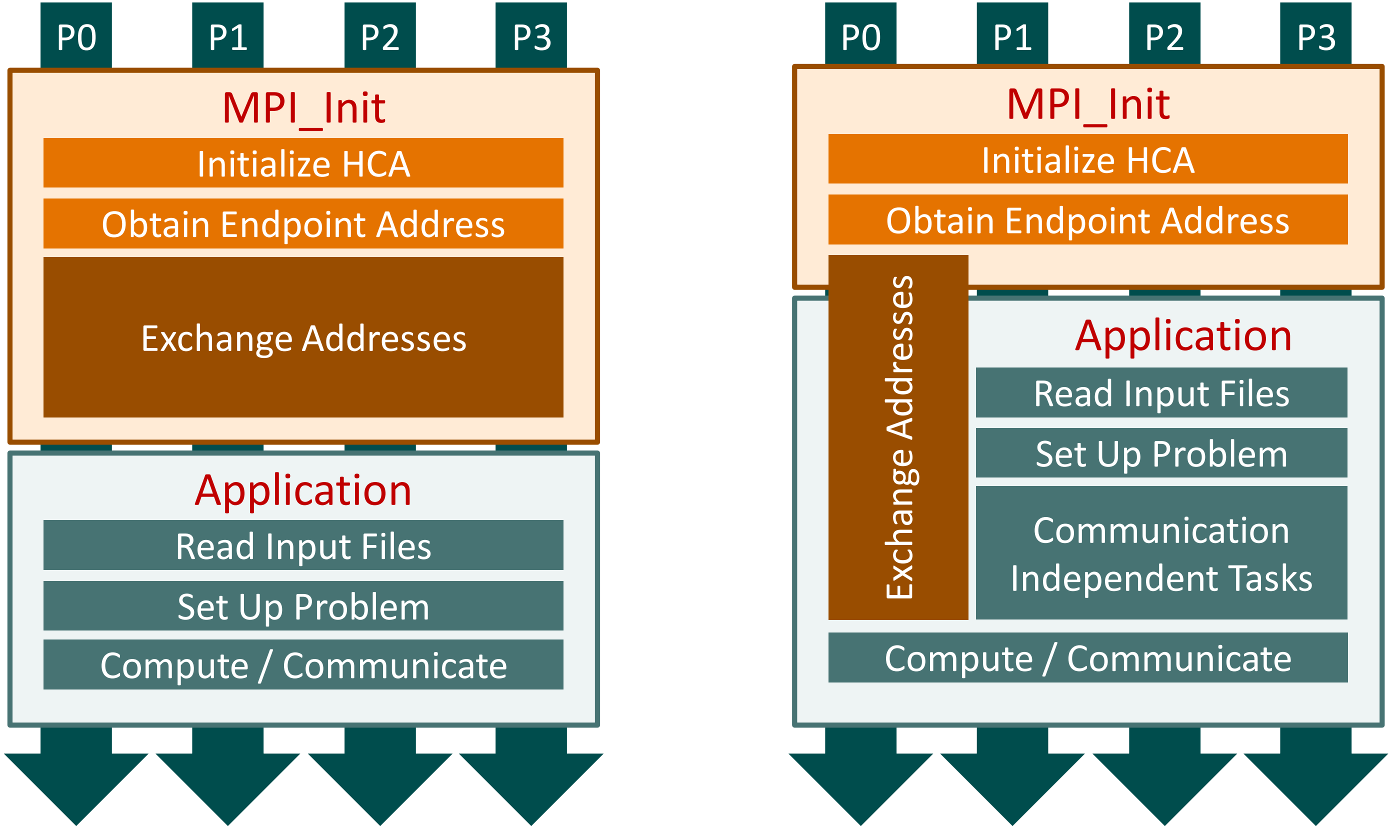
MVAPICH2 Software Family

High-Performance Parallel Programming Libraries	
MVAPICH2	Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE
MVAPICH2-X	Advanced MPI features, OSU INAM, PGAS (OpenSHMEM, UPC, UPC++, and CAF), and MPI+PGAS programming models with unified communication runtime
MVAPICH2-GDR	Optimized MPI for clusters with NVIDIA GPUs
MVAPICH2-Virt	High-performance and scalable MPI for hypervisor and container based HPC cloud
MVAPICH2-EA	Energy aware and High-performance MPI
MVAPICH2-MIC	Optimized MPI for clusters with Intel KNC
Microbenchmarks	
OMB	Microbenchmarks suite to evaluate MPI and PGAS (OpenSHMEM, UPC, and UPC++) libraries for CPUs and GPUs
Tools	
OSU INAM	Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration
OEMT	Utility to measure the energy consumption of MPI applications

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand Core-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

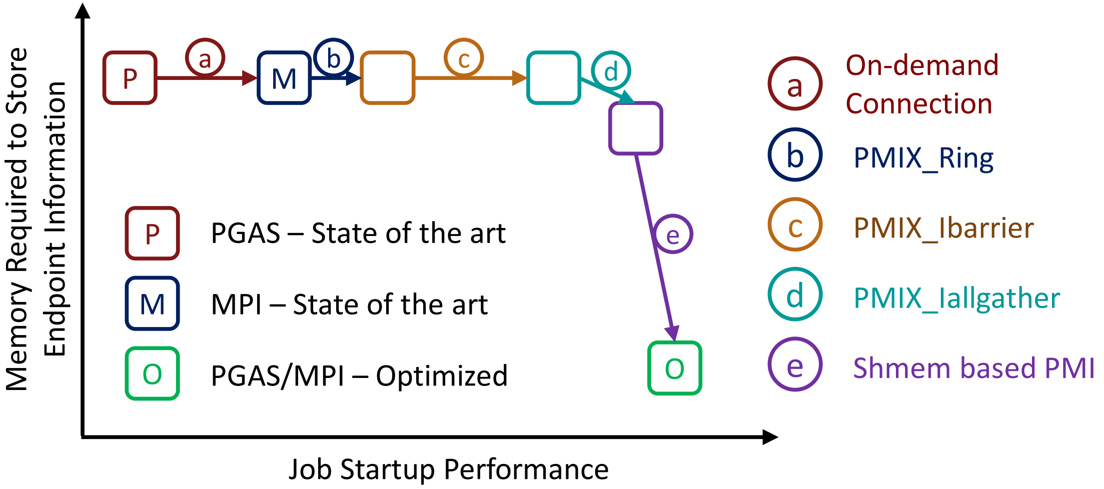
Overlapping Application Compute with MPI Startup



No Overlap between MPI_Init and Application Computation

MPI can continue to initialize in the background while Application starts

Towards High Performance and Scalable Startup at Exascale



- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by O(processes per node)
- 1GB Memory saved per node with 1M processes and 16 processes per node

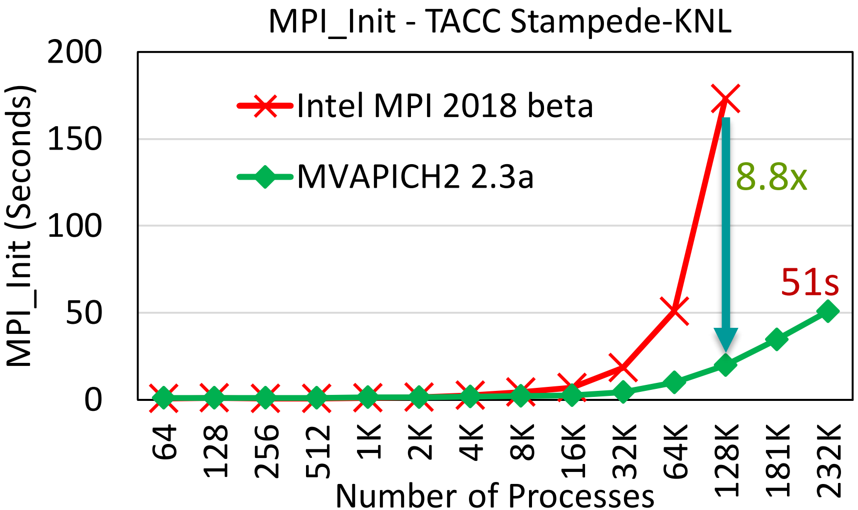
a **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS ’15)

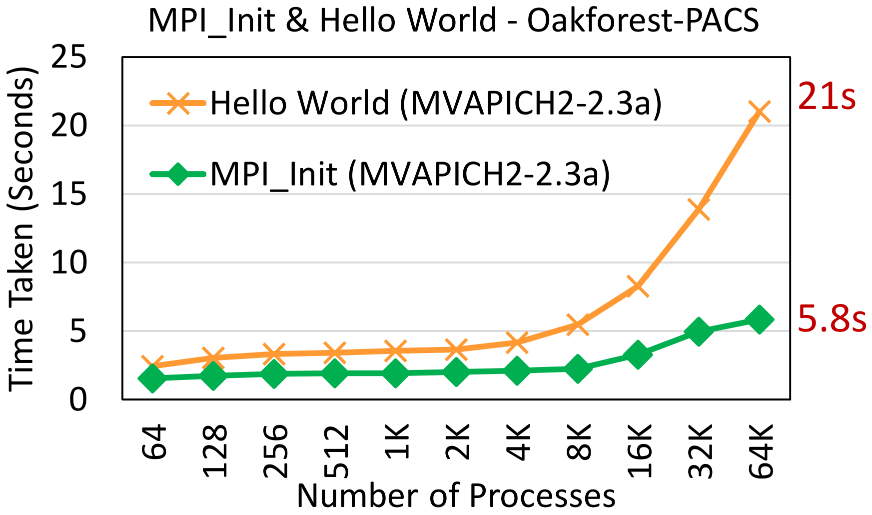
b **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia ’14)

c d **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’15)

e **SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability.** S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’16)

Startup Performance on KNL + Omni-Path



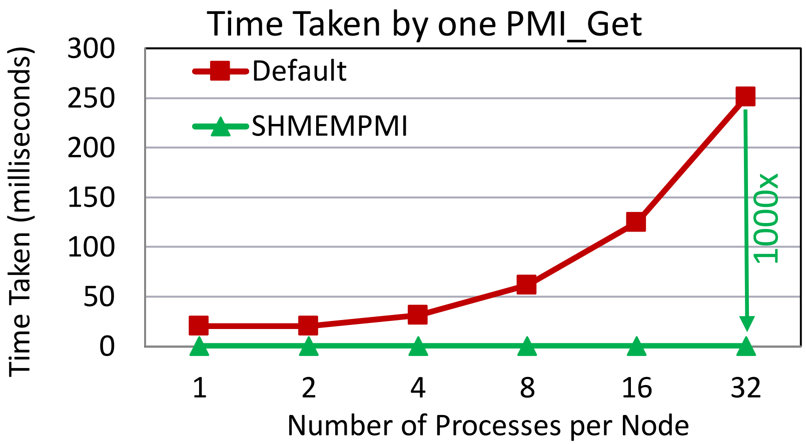


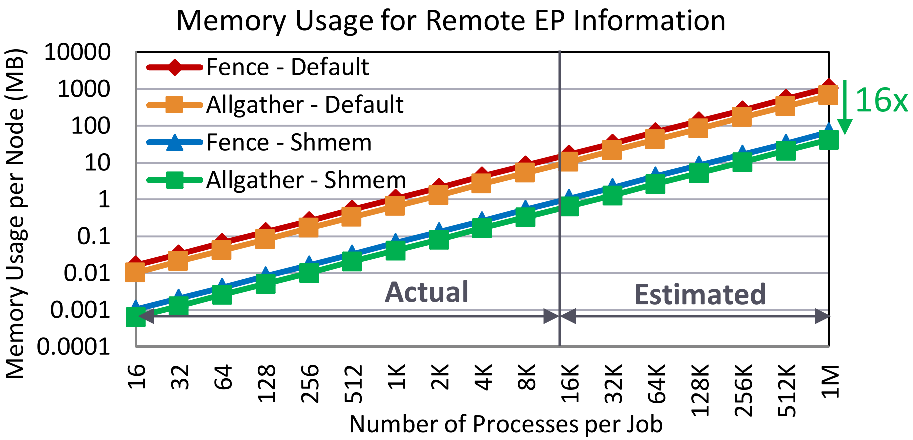
- MPI_Init takes 51 seconds on 231,956 processes on 3,624 KNL nodes (Stampede – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

New designs available in MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1

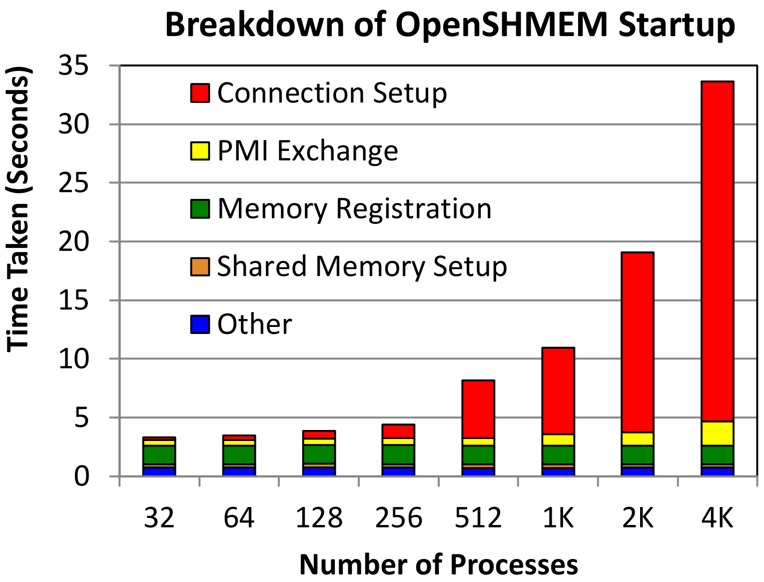
Process Management Interface (PMI) over Shared Memory (SHMEMPMI)

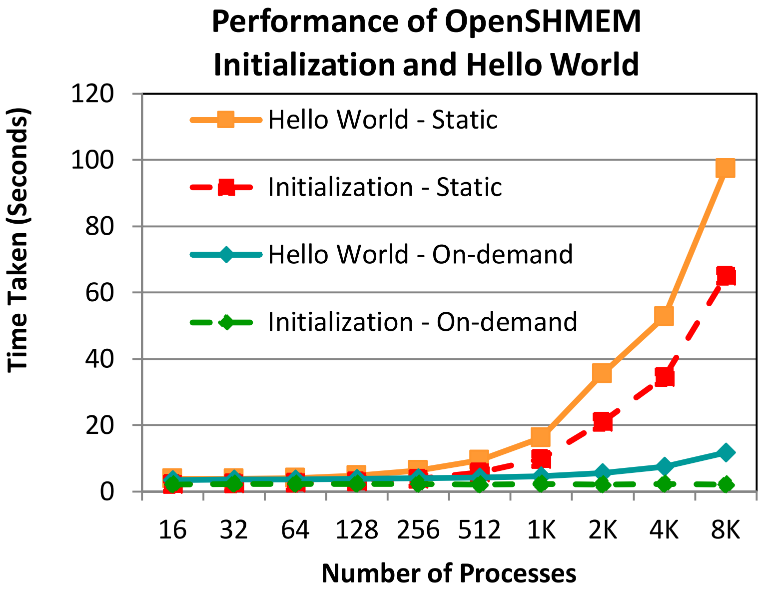
- SHMEMPMI allows MPI processes to directly read remote endpoint (EP) information from the process manager through shared memory segments
- Only a single copy per node - O(processes per node) reduction in memory usage
- Estimated savings of 1GB per node with 1 million processes and 16 processes per node
- Up to 1,000 times faster PMI Gets compared to default design
- **Available since MVAPICH2 2.2rc1 and SLURM-15.08.8**





On-demand Connection Management for OpenSHMEM+MPI





- Static connection establishment wastes memory and takes a lot of time
- On-demand connection management improves OpenSHMEM initialization time by 29.6 times
- Time taken for Hello World reduced by 8.31 times at 8,192 processes
- **Available since MVAPICH2-X 2.1rc1**

How to Get the Best Startup Performance with MVAPICH2?

- **MV2_HOMOGENEOUS_CLUSTER=1** //Set for homogenous clusters
- **MV2_ON_DEMAND_UD_INFO_EXCHANGE=1** //Enable UD based address exchange

Using SLURM as launcher

- **Use PMI2**
  - ./configure --with-pm=slurm --with-pmi=pmi2
  - srun --mpi=pmi2 ./a.out
- **Use PMI Extensions**
  - Patch for SLURM available at http://mvapich.cse.ohio-state.edu/download/
  - Patches available for SLURM 15, 16, and 17
  - PMI Extensions are automatically detected by MVAPICH2

Using mpirun_rsh as launcher

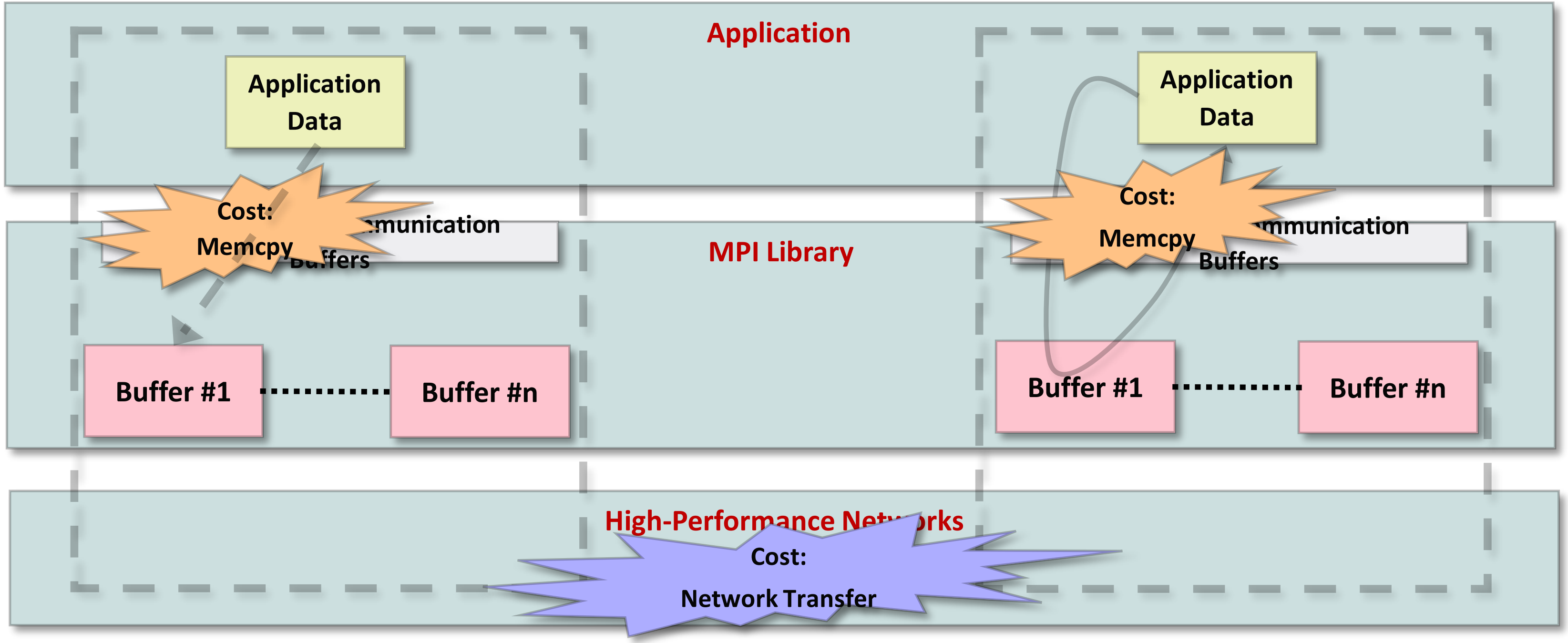
- **MV2_MT_DEGREE**
  - degree of the hierarchical tree used by mpirun_rsh
- **MV2_FASTSSH_THRESHOLD**
  - #nodes beyond which hierarchical-ssh scheme is used
- **MV2_NPROCS_THRESHOLD**
  - #nodes beyond which file-based communication is used for hierarchical-ssh

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand Core-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

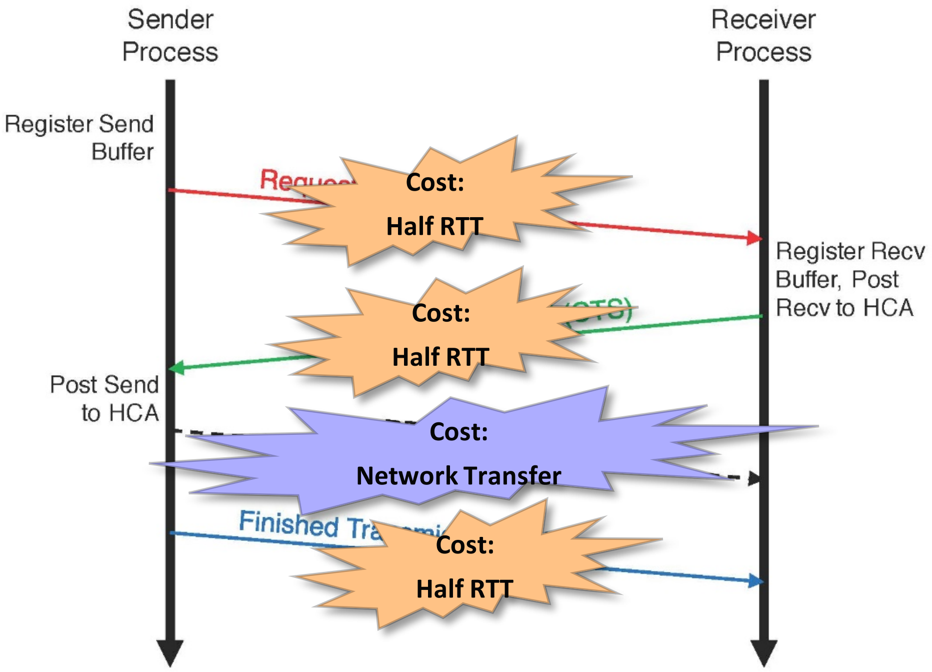
Communication Costs of Point-to-point Protocols - Eager

- Good communication performance for smaller messages
- No synchronization required between sender and receiver
- Cost of extra copies is high for large messages

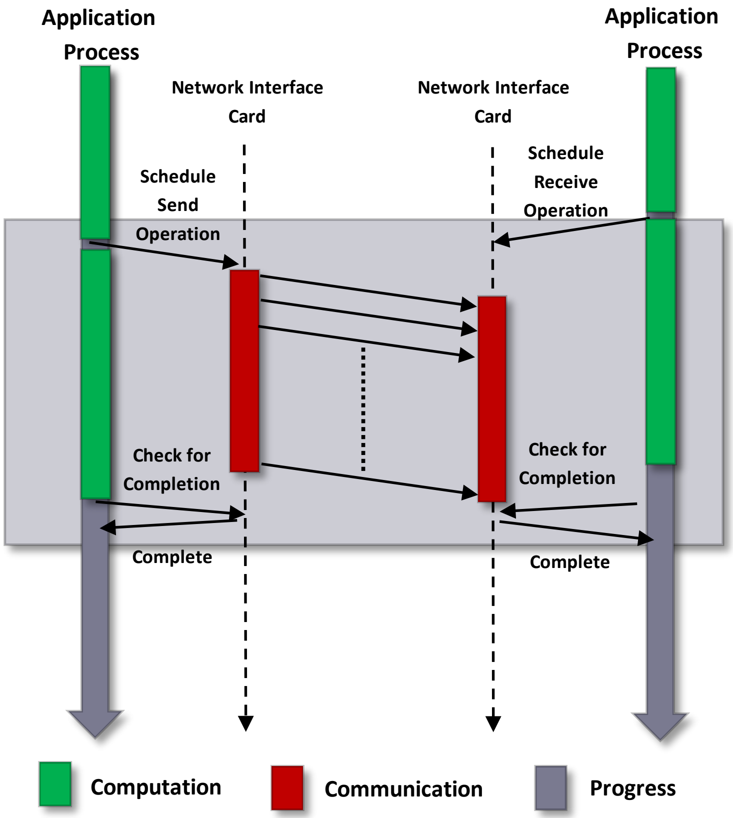


Communication Costs of Point-to-point Protocols - Rendezvous

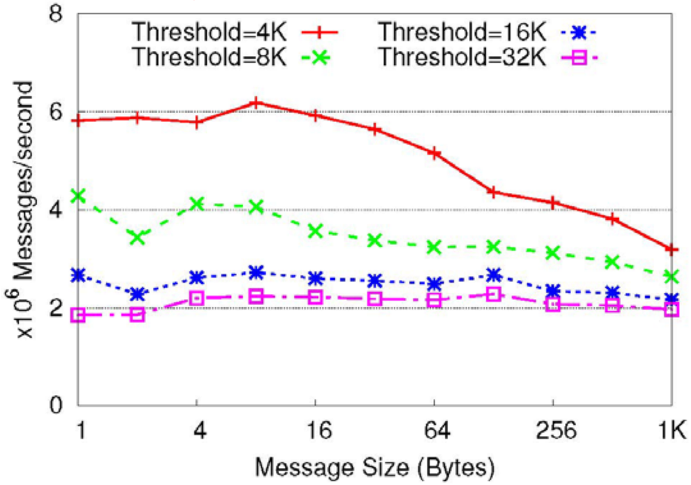
- Avoid extra copies for larger messages
- Synchronization required between sender and receiver
- Can be based on RDMA Read or RDMA Write (shown here)



Analyzing Overlap Potential of Eager Protocol

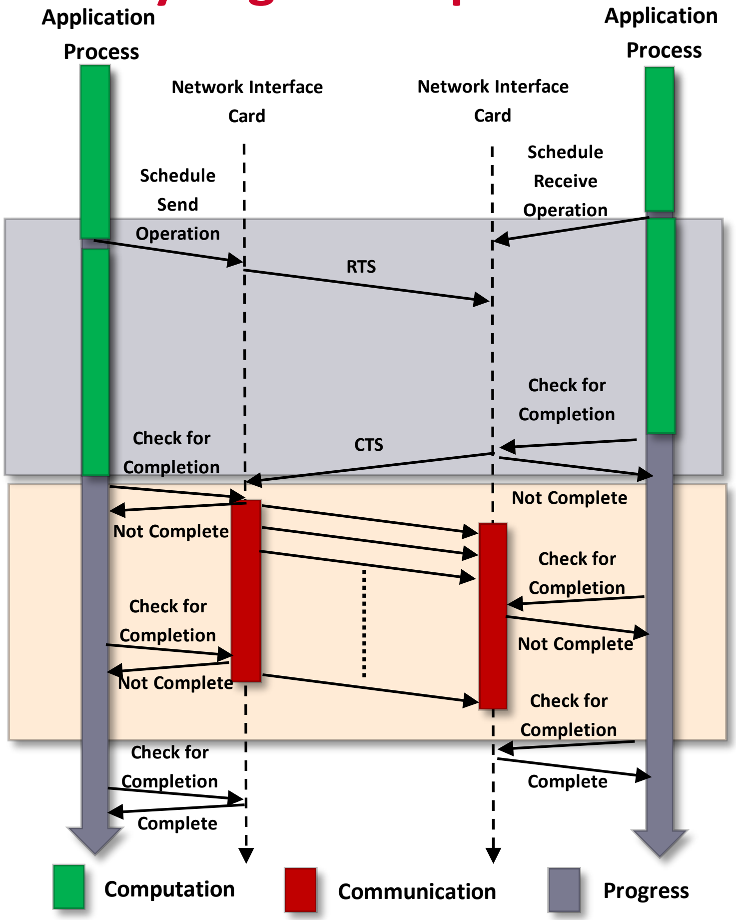


- Application processes schedule communication operation
- Network adapter progresses communication in the background
- Application process free to perform useful compute in the foreground
- **Overlap of computation and communication => Better Overall Application Performance**
- **Increased buffer requirement**
- **Poor communication performance if used for all types of communication operations**



Impact of changing Eager Threshold on performance of multi-pair message-rate benchmark with 32 processes on Stampede

Analyzing Overlap Potential of Rendezvous Protocol

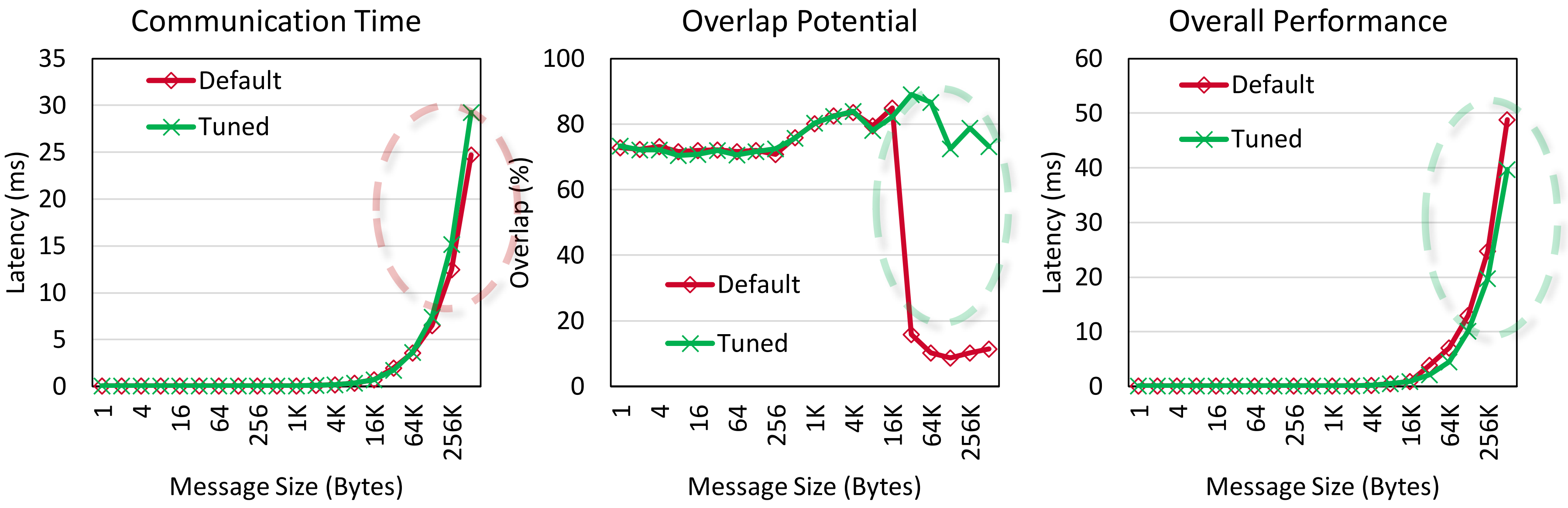


- Application processes schedule communication operation
- Application process free to perform useful compute in the foreground
- Little communication progress in the background
- All communication takes place at final synchronization

- **Reduced buffer requirement**
- **Good communication performance if used for large message sizes and operations where communication library is progressed frequently**

- **Poor overlap of computation and communication => Poor Overall Application Performance**

Impact of Tuning Rendezvous Threshold on 3D-Stencil



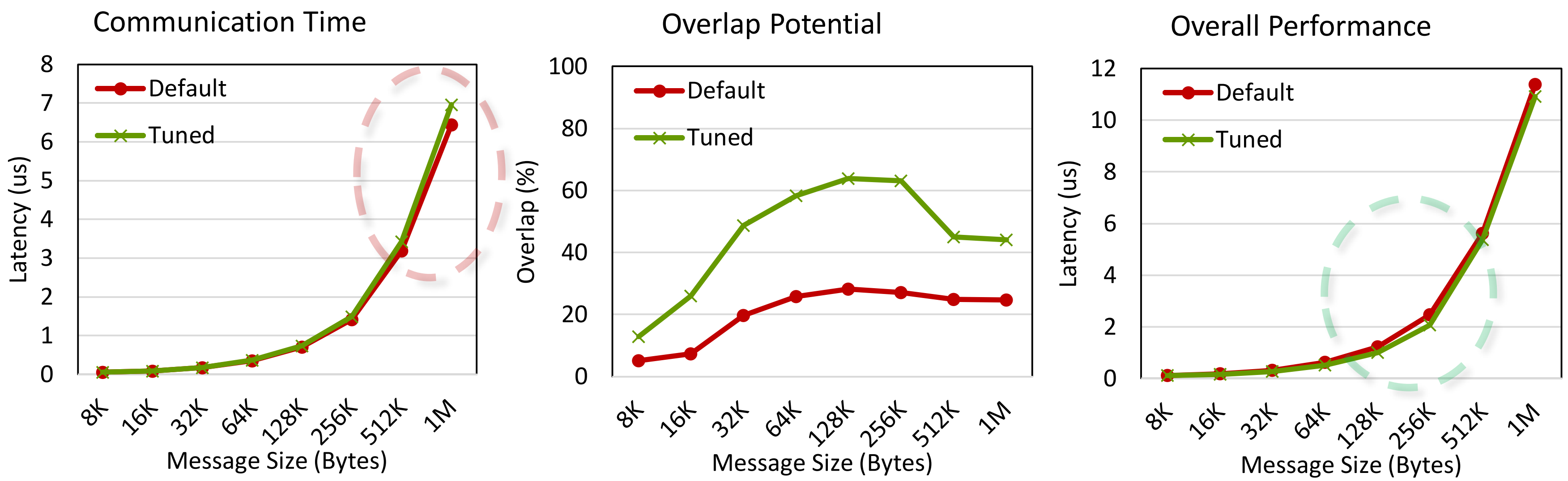
- Increased eager threshold from 16KB to 512KB
- Very small degradation in raw communication performance
- Significant improvement in overlap of computation and communication
- ~18% Improvement in overall performance

MV2_IBA_EAGER_THRESHOLD=512K
MV2_SMP_EAGERSIZE=512K

(Applicable to both InfiniBand and Omni-Path)

8192 Processes, SandyBridge + FDR

Impact of Tuning Rendezvous Protocol on 3D-Stencil



- RDMA Read based protocol (RGET) used instead of RDMA Write
- Very minor penalty in raw performance
- Offers more overlap due to less synchronization overhead
- Up to 15% improvement in overall execution time

MV2_RNDV_PROTOCOL=RGET

(Applicable to InfiniBand)

64 Processes, Broadwell + EDR

Dynamic and Adaptive MPI Point-to-point Communication Protocols

- Different communication protocols have different trade-offs
  - Need to consider performance, overlap, memory requirement
  - Manual tuning is difficult and time-consuming

- Can the MPI library select the best protocol at runtime?
  - Use different protocols and thresholds between different pair of processes
  - Deliver good performance and minimize resource consumption
  - Dynamically adapt to the application's communication requirements at runtime

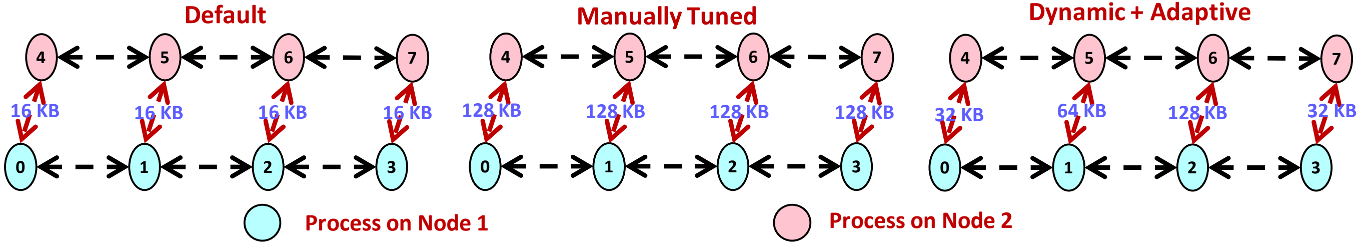
Default	**Poor overlap**; **Low memory requirement**	**Low Performance**; **High Productivity**
Manually Tuned	**Good overlap**; **High memory requirement**	**High Performance**; **Low Productivity**
Dynamic + Adaptive	**Good overlap**; **Optimal memory requirement**	**High Performance**; **High Productivity**

Dynamic and Adaptive MPI Point-to-point Communication Protocols (cont.)

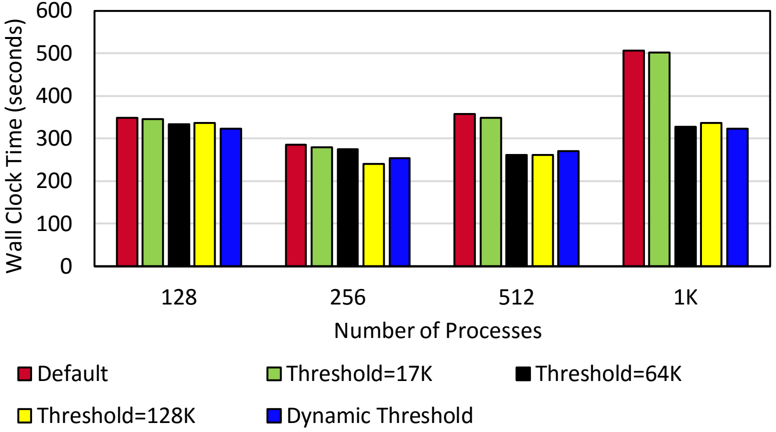
Desired Eager Threshold

Process Pair	Eager Threshold (KB)
0 – 4	32
1 – 5	64
2 – 6	128
3 – 7	32

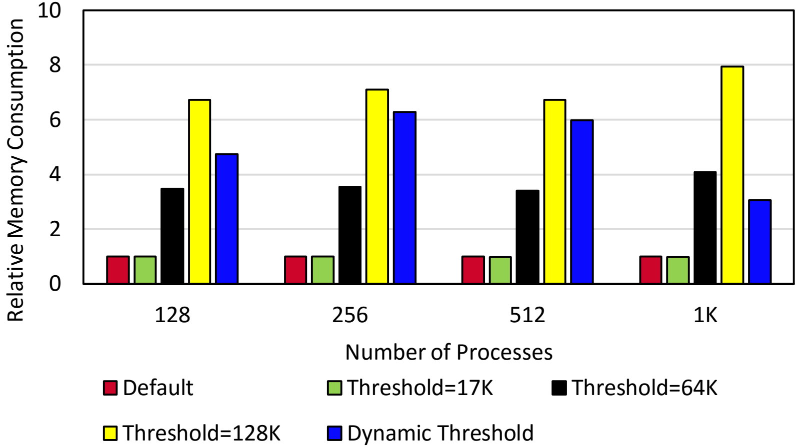
Eager Threshold for Example Communication Pattern with Different Designs



Execution Time of Amber



Relative Memory Consumption of Amber



H. Subramoni, S. Chakraborty, D. K. Panda, Designing Dynamic & Adaptive MPI Point-to-Point Communication Protocols for Efficient Overlap of Computation & Communication, ISC'17 - **Best Paper**

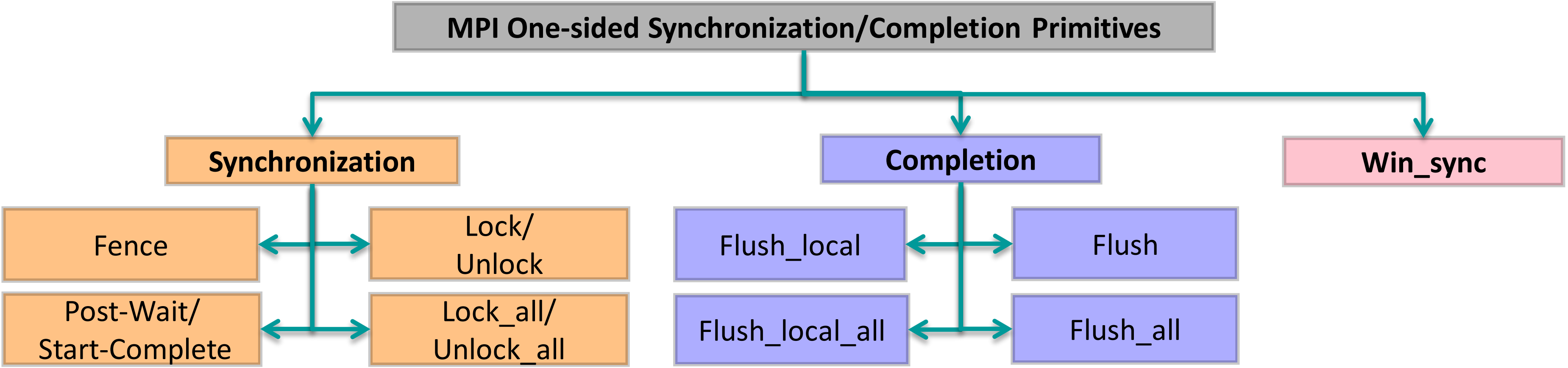
Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand Core-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

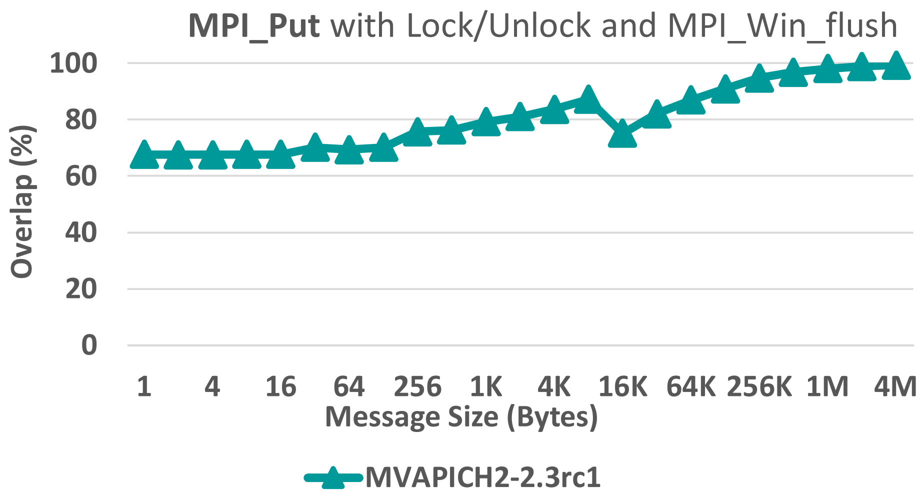
MPI-3 RMA: Communication and synchronization Primitives

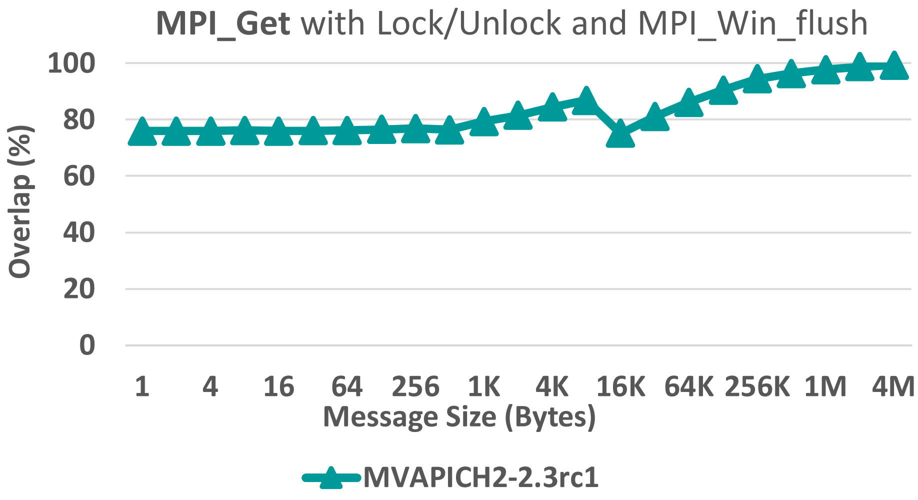
- Non-blocking one-sided communication routines
  - Put, Get (Rput, Rget)
  - Accumulate, Get_accumulate
  - Atomics
- Flexible synchronization operations to control initiation and completion

MVAPICH2 supports all RMA communication with Best performance and overlap



Overlap between Computation and RMA Operations

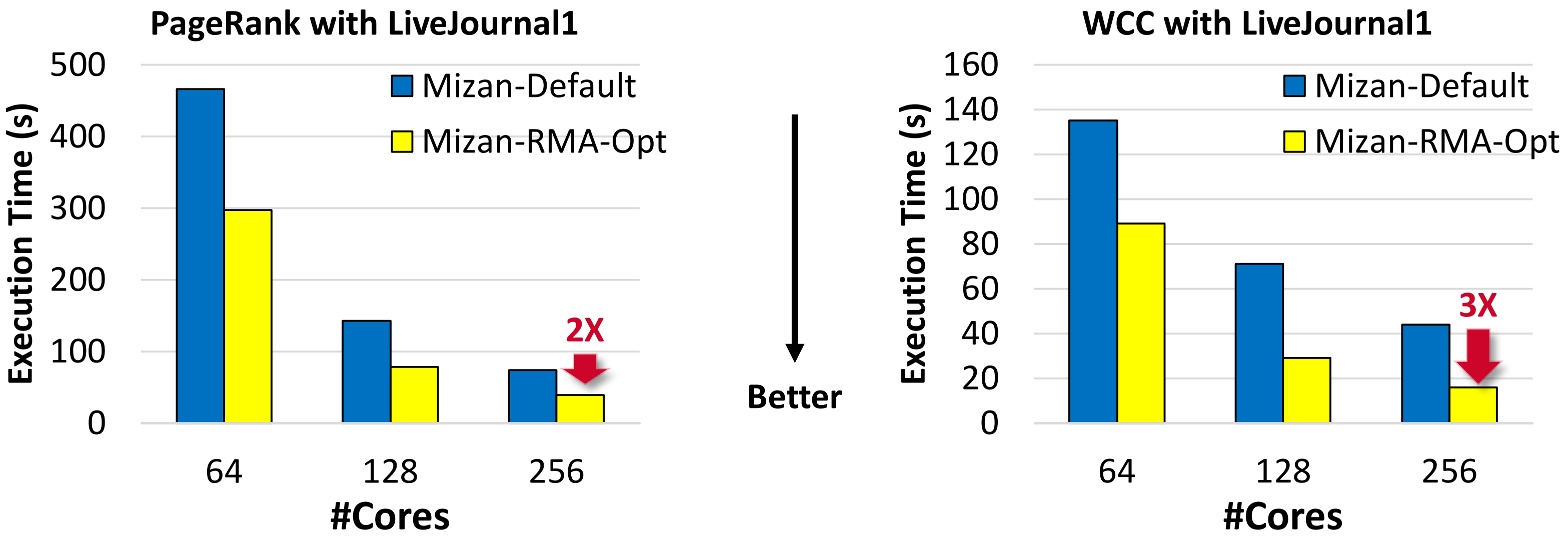




- **67-99%** overlap between **MPI_Put** and computation
- **75-99%** overlap between **MPI_Get** and computation

MVAPICH2-2.3rc1
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
Mellanox Connect-X4 EDR HCA
Mellanox OFED 4.3

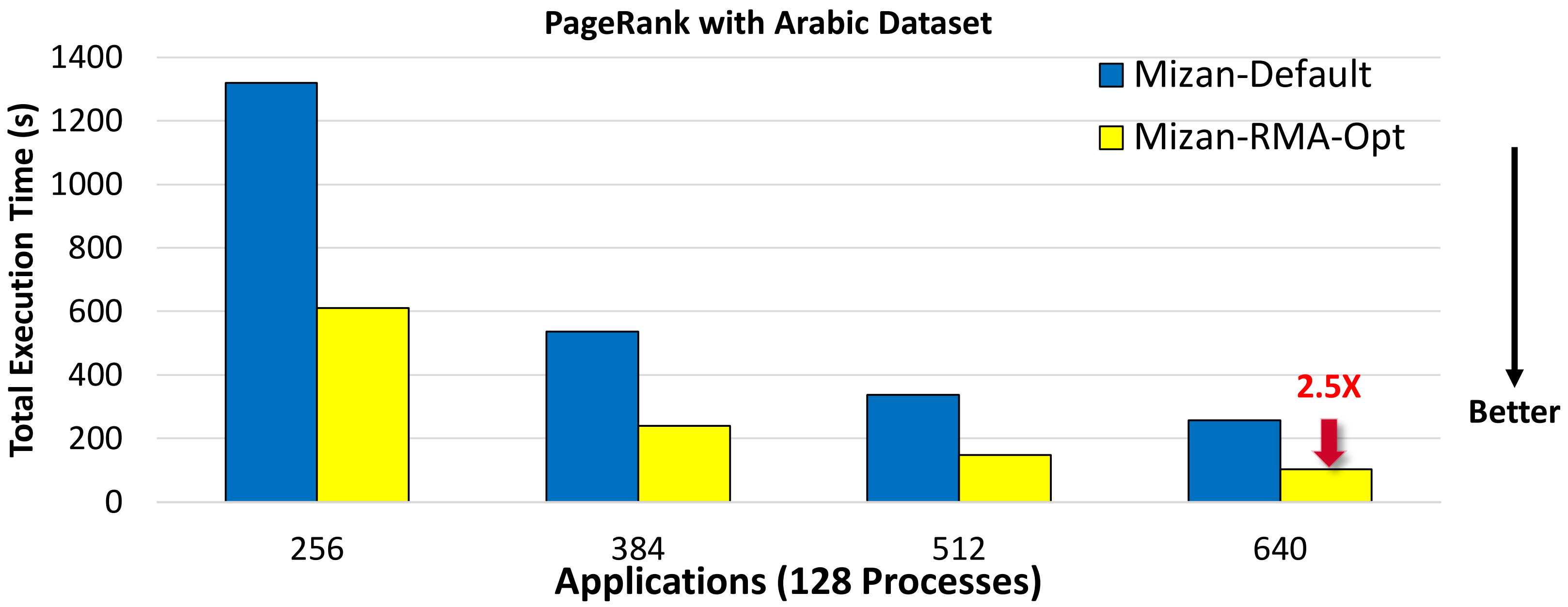
Graph Processing Framework with Optimized MPI RMA



- Proposed design performs better than default implementation
- For Weakly Connected Components (WCC) on 256 cores, proposed design could reduce the total execution time by 2X compared with the default scheme

M. Li, X. Lu, K. Hamidouche, J. Zhang and D. K. Panda, "Mizan-RMA: Accelerating Mizan Graph Processing Framework with MPI RMA," IEEE HiPC, 2016

Graph Processing Framework with Optimized MPI RMA



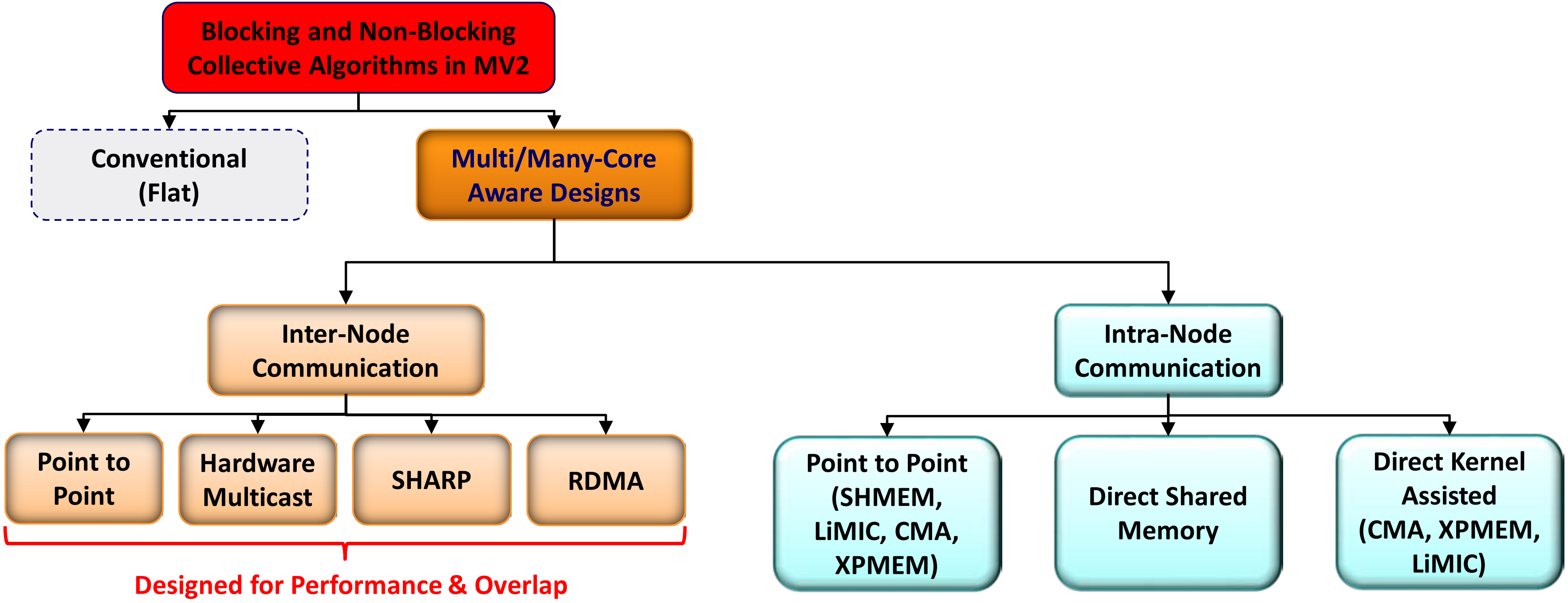
- Proposed design shows good strong scaling
- Proposed design scales better than default implementation

M. Li, X. Lu, K. Hamidouche, J. Zhang and D. K. Panda, "Mizan-RMA: Accelerating Mizan Graph Processing Framework with MPI RMA," IEEE HiPC, 2016

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand Core-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

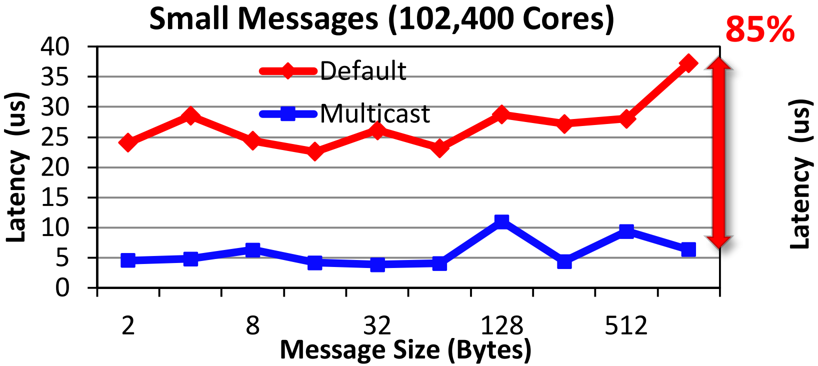
Collective Communication in MVAPICH2

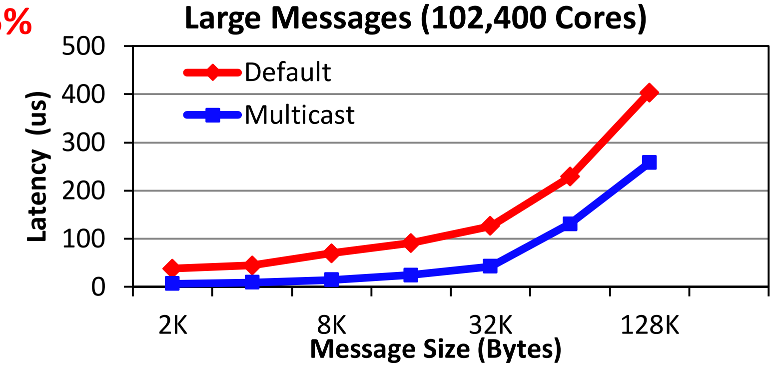


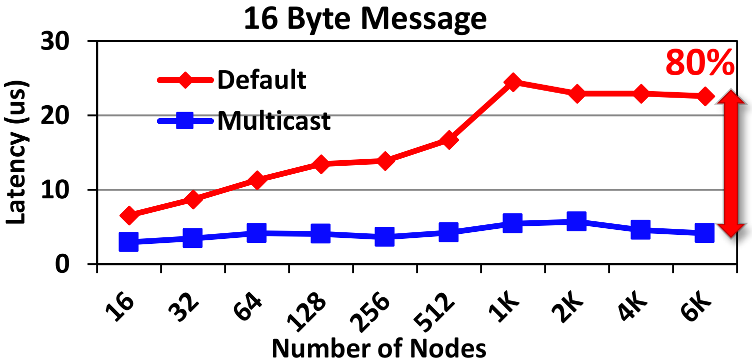
Run-time flags:

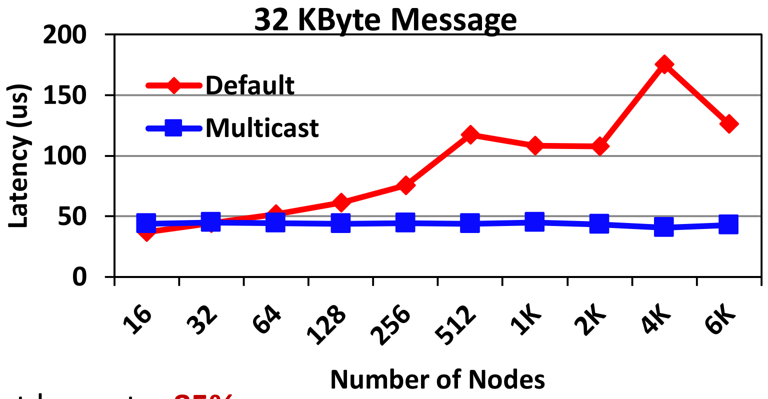
All shared-memory based collectives	:	MV2_USE_SHMEM_COLL (Default: ON)
Hardware Mcast-based collectives	:	MV2_USE_MCAST (Default : OFF)
CMA-based collectives	:	MV2_USE_CMA_COLL (Default : ON)

Hardware Multicast-aware MPI_Bcast on TACC Stampede



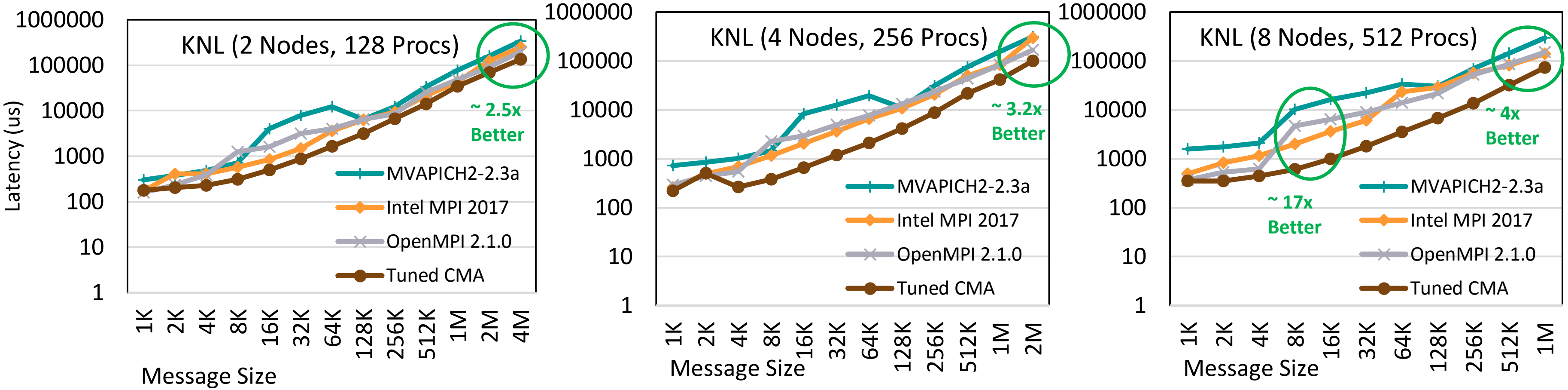






- MCAST-based designs improve latency of MPI_Bcast by up to **85%**
- Use MV2_USE_MCAST=1 to enable MCAST-based designs

Optimized CMA-based Collectives for Large Messages



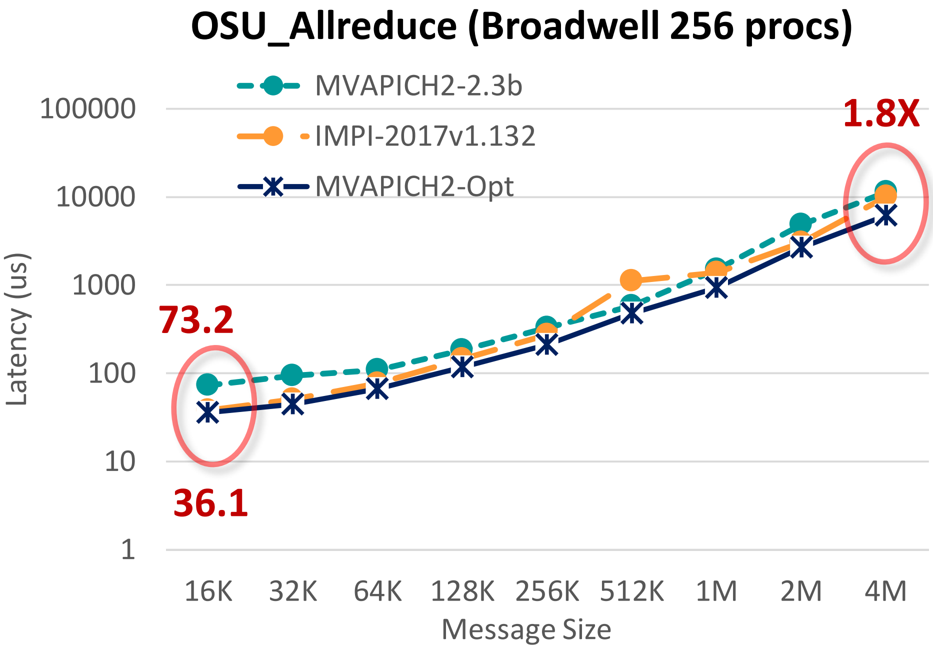
Performance of MPI_Gather on KNL nodes (64PPN)

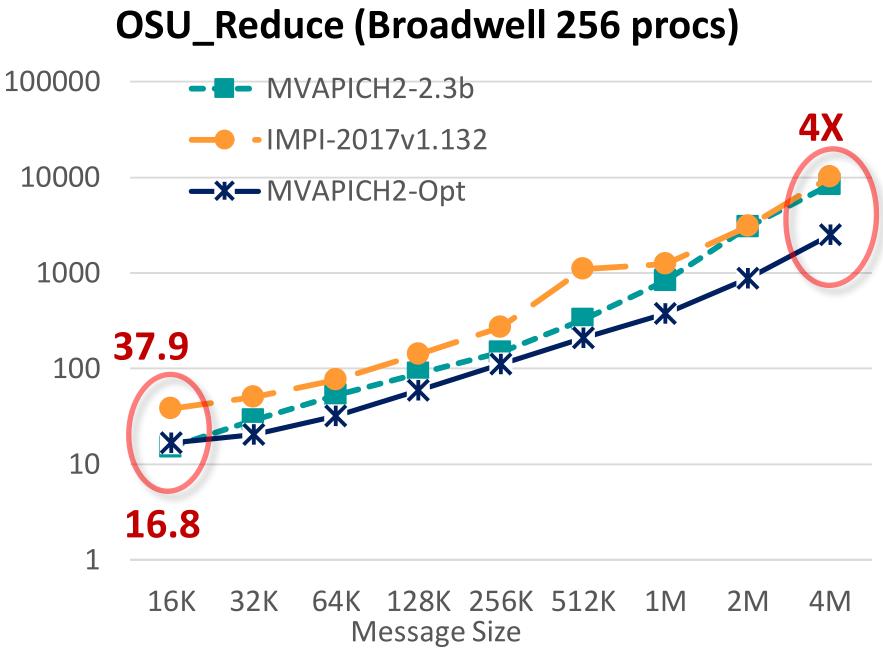
- Significant improvement over existing implementation for Scatter/Gather with 1MB messages (up to 4x on KNL, 2x on Broadwell, 14x on OpenPower)
- New two-level algorithms for better scalability
- Improved performance for other collectives (Bcast, Allgather, and Alltoall)

S. Chakraborty, H. Subramoni, and D. K. Panda, **Contention Aware Kernel-Assisted MPI Collectives for Multi/Many-core Systems,** *IEEE Cluster '17, BEST Paper Finalist*

Available in MVAPICH2-X 2.3b

Shared Address Space (XPMEM)-based Collectives Design





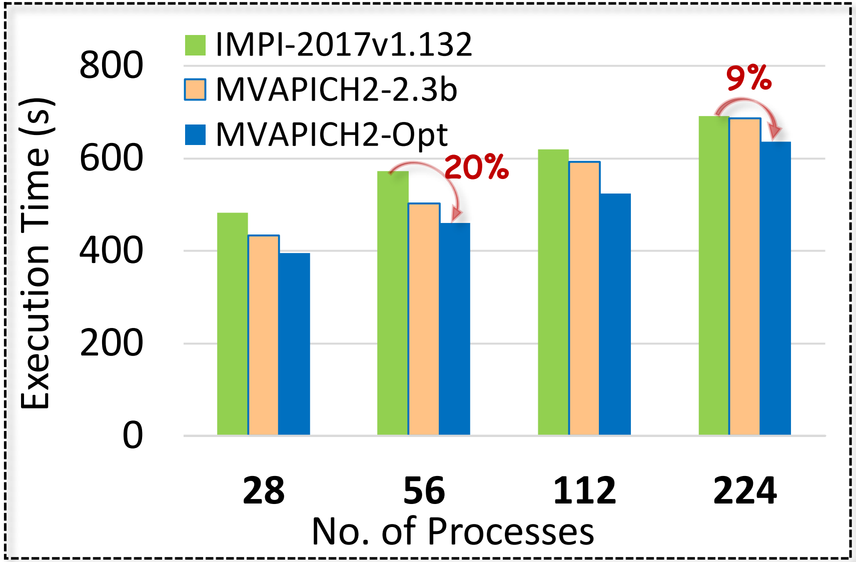
- "*Shared Address Space*"-based true *zero-copy* Reduction collective designs in MVAPICH2
- Offloaded computation/communication to peers ranks in reduction collective operation
- Up to **4X** improvement for 4MB Reduce and up to **1.8X** improvement for 4M AllReduce

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. Panda, Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018.

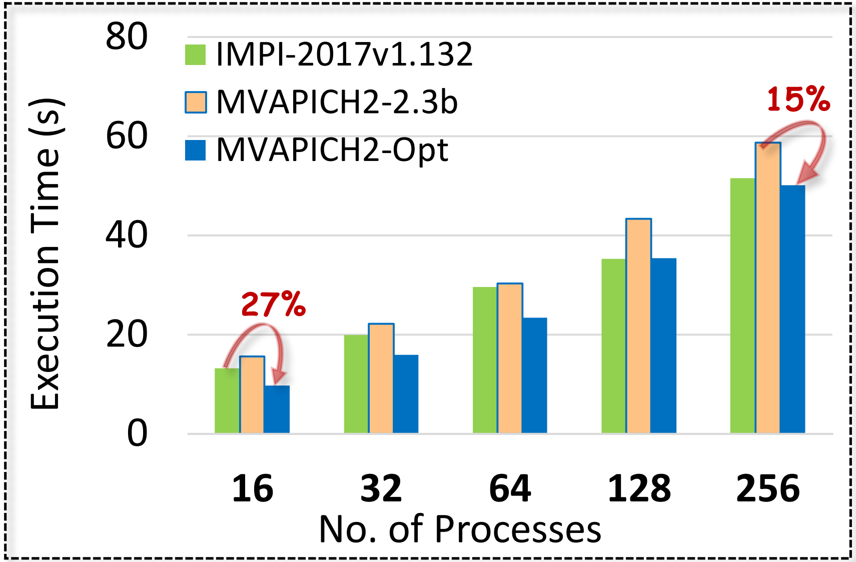
Will be available in future

Application-Level Benefits of XPMEM-Based Collectives

CNTK AlexNet Training
(Broadwell, B.S=default, iteration=50, ppn=28)

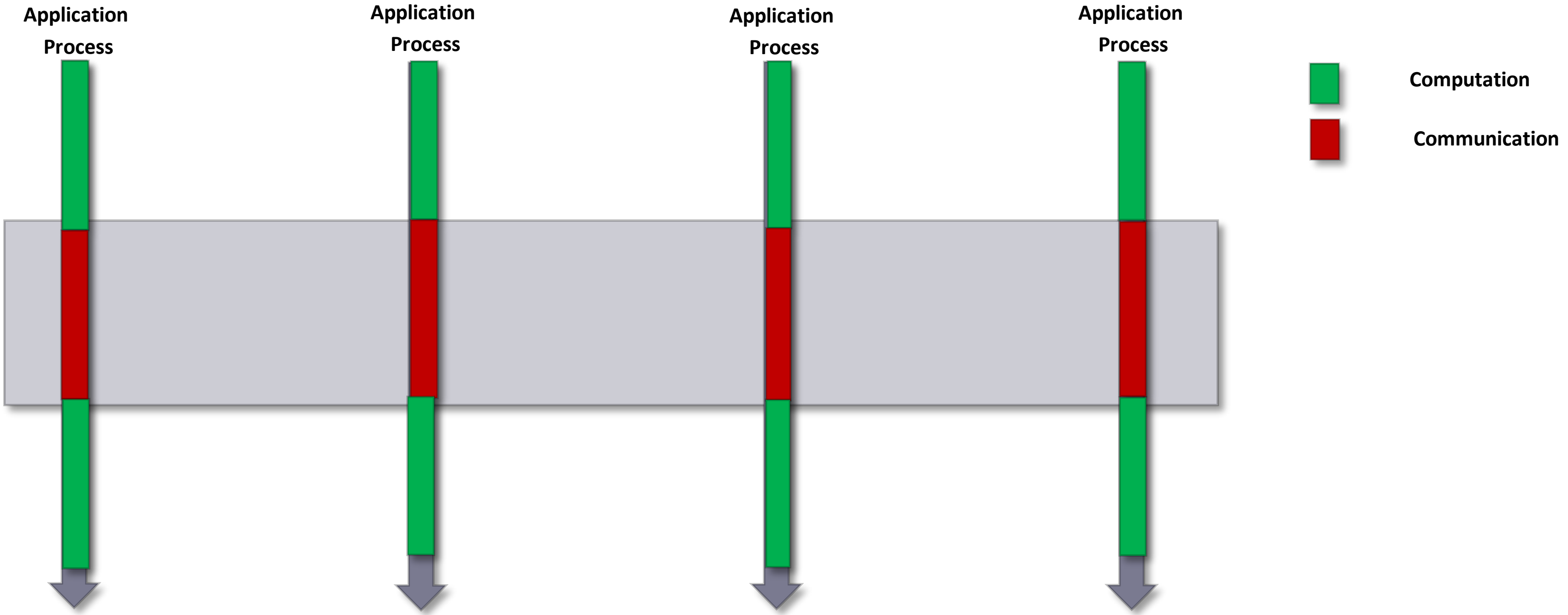


MiniAMR (Broadwell, ppn=16)



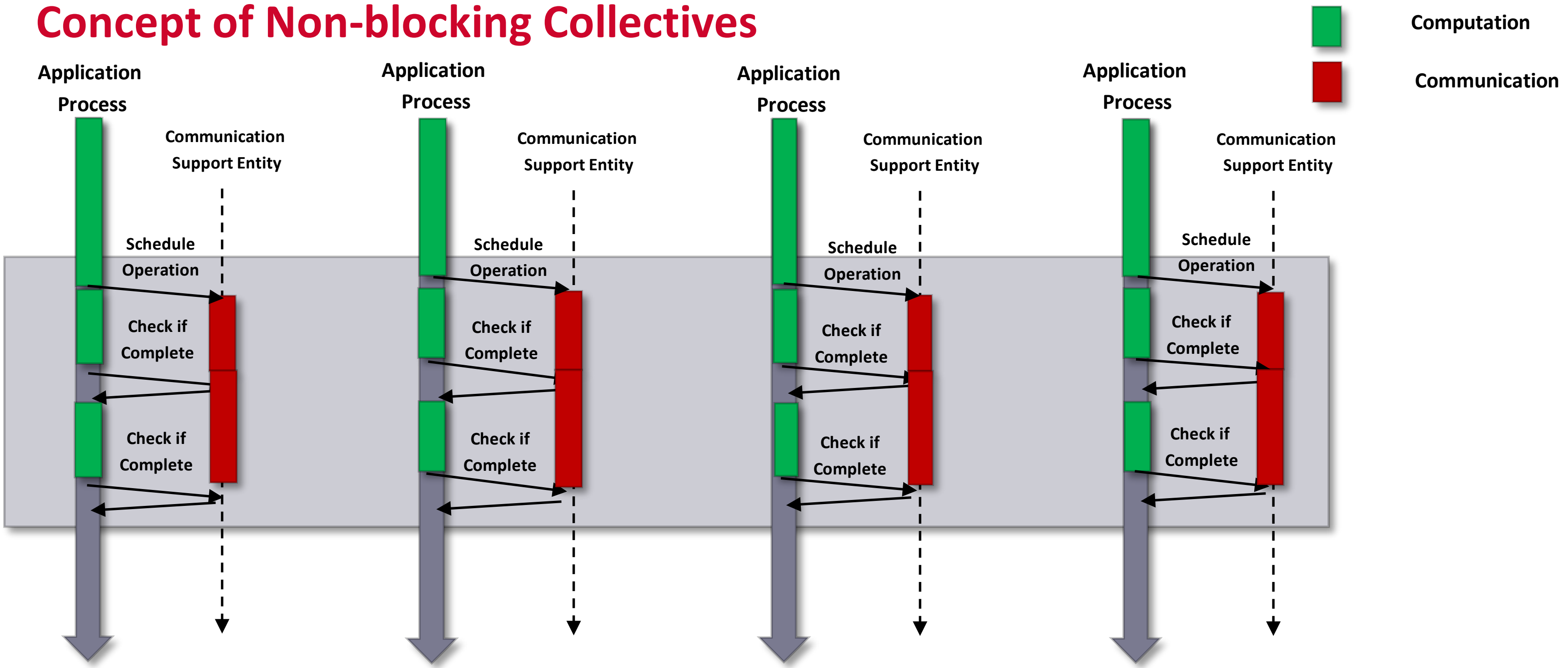
- Up to **20%** benefits over IMPI for CNTK DNN training using AllReduce
- Up to **27%** benefits over IMPI and up to **15%** improvement over MVAPICH2 for MiniAMR application kernel

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*

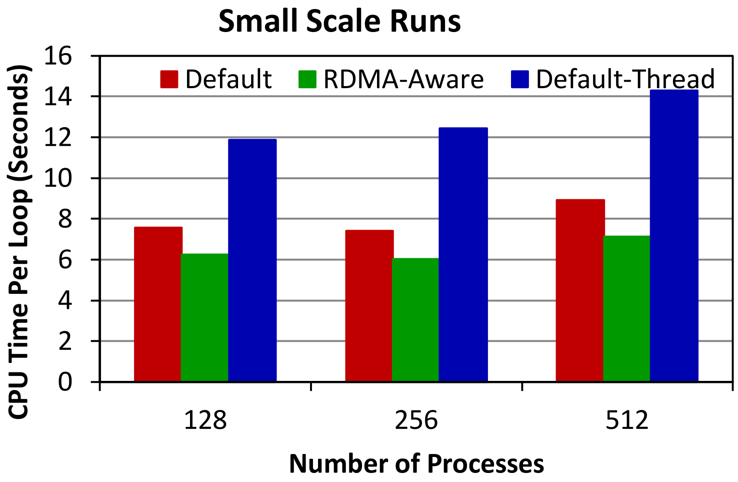
Non-blocking Collective (NBC) Operations

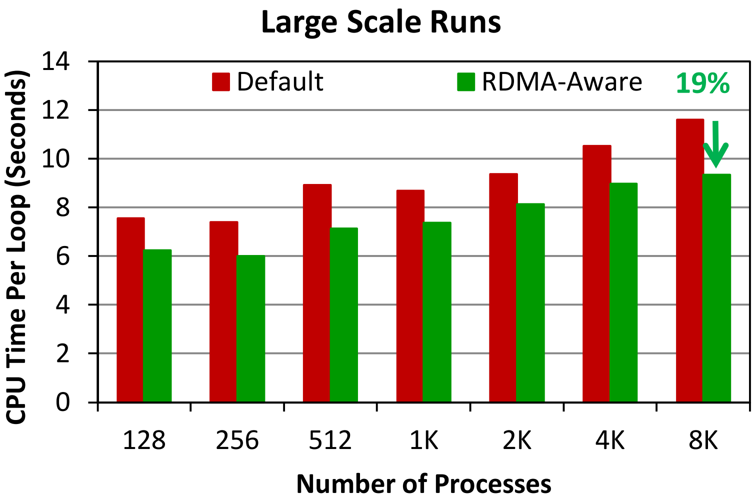
- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap
- A process 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





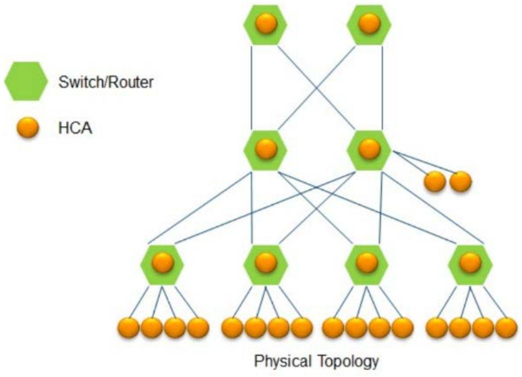
- Weak scaling experiments; problem size increases with job size
- RDMA-Aware delivers 19% improvement over Default @ 8,192 procs
- Default-Thread exhibits worst performance
  - Possibly because threads steal CPU cycles from P3DFFT
  - Do not consider for large scale experiments

Will be available in future

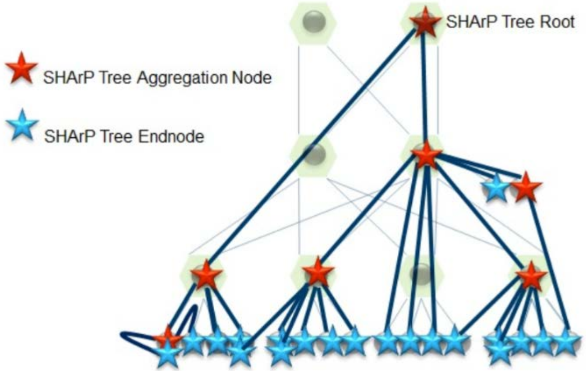
Designing Non-Blocking Personalized Collectives with Near Perfect Overlap for RDMA-Enabled Clusters, H. Subramoni , A. Awan , K. Hamidouche , D. Pekurovsky , A. Venkatesh , S. Chakraborty , K. Tomko , and D. K. Panda, ISC '15, Jul 2015

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*

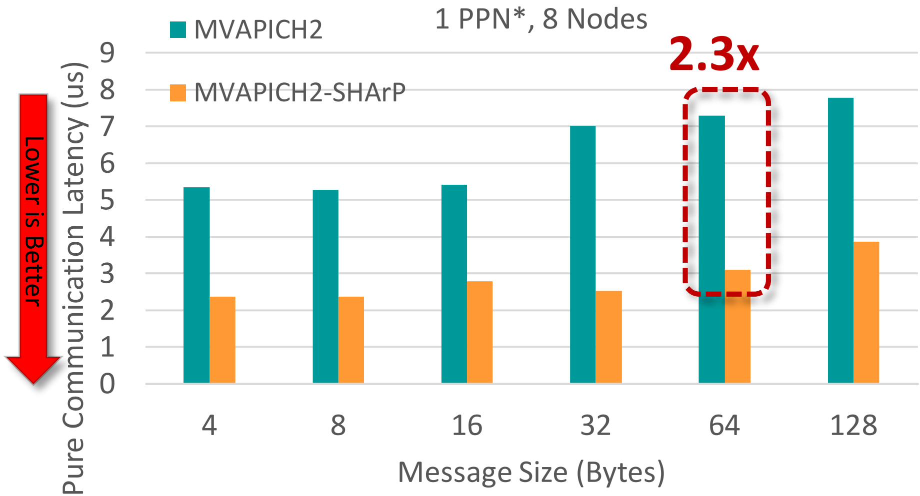


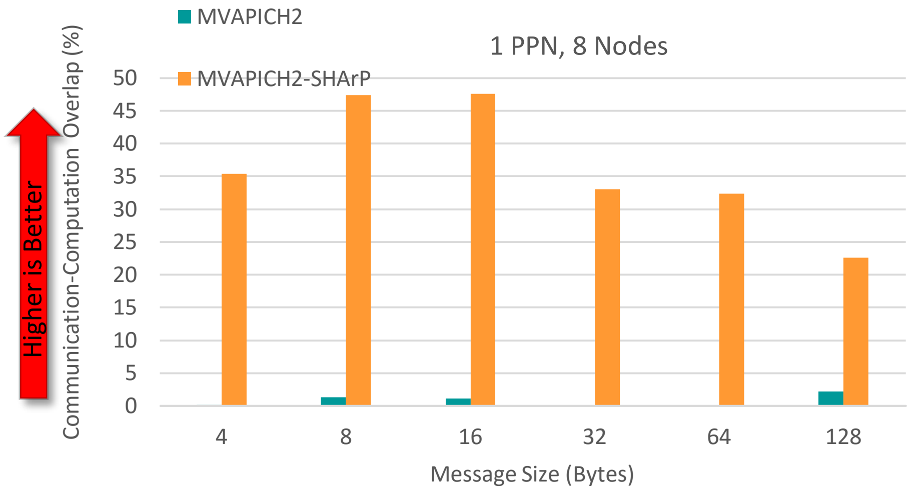
Logical SHArP Tree*

* 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

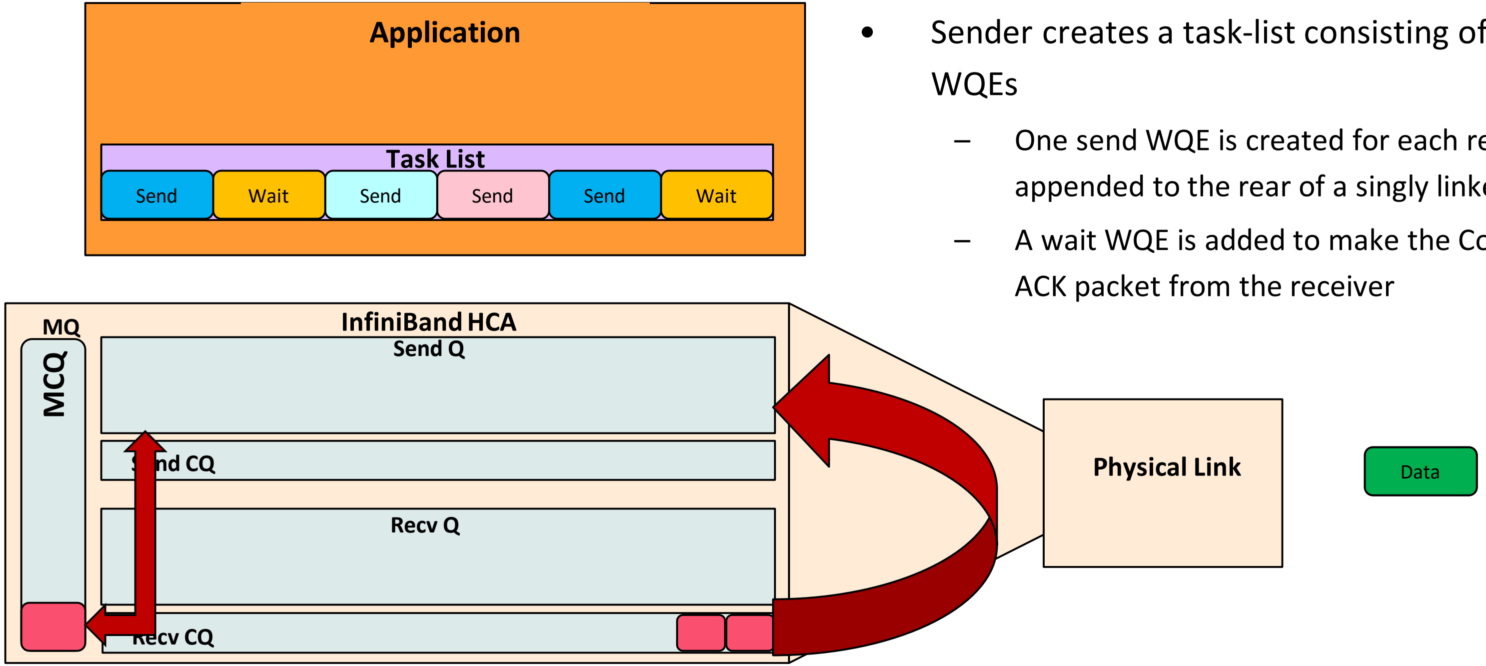
Available since MVAPICH2 2.3a

*PPN: Processes Per Node

Collective Offload in ConnectX-2, ConnectX-3, Connect-IB and ConnectX-4, ConnectX-5

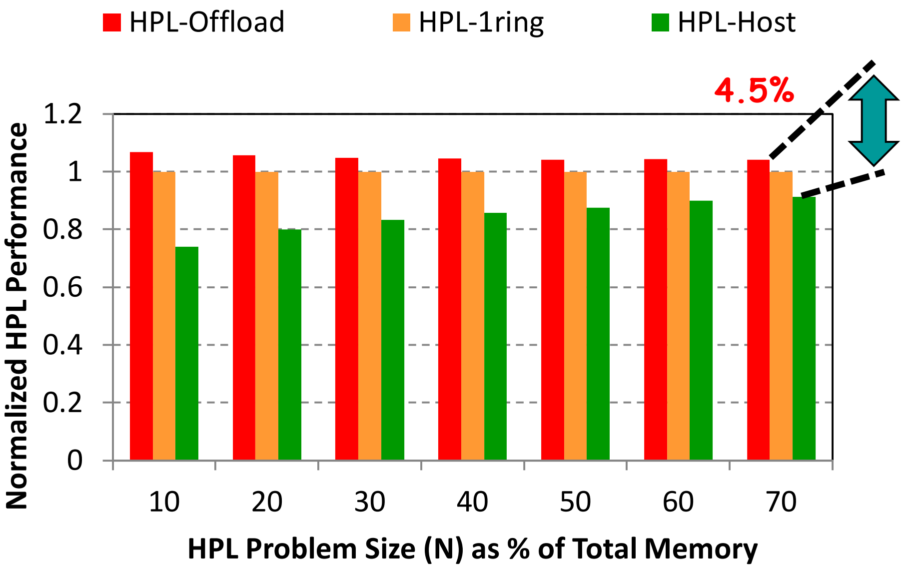
- Mellanox’s ConnectX-2, ConnectX-3, ConnectIB, ConnectX-4, and ConnectX-5 adapters feature “task-list” offload interface
  - Extension to existing InfiniBand APIs
- Collective communication with `blocking’ feature is usually a scaling bottleneck
  - Matches with the need for non-blocking collective in MPI
- Accordingly MPI software stacks need to be re-designed to leverage offload in a comprehensive manner
- Can applications be modified to take advantage of non-blocking collectives and what will be the benefits?

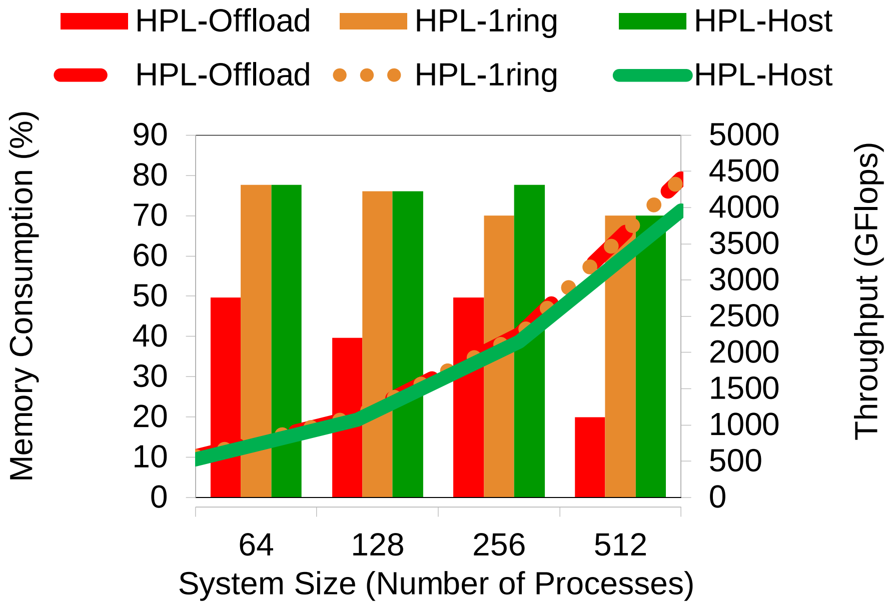
Collective Offload Support in ConnectX InfiniBand Adapter (Recv followed by Multi-Send)



- 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

Co-designing HPL with Core-Direct and Performance Benefits





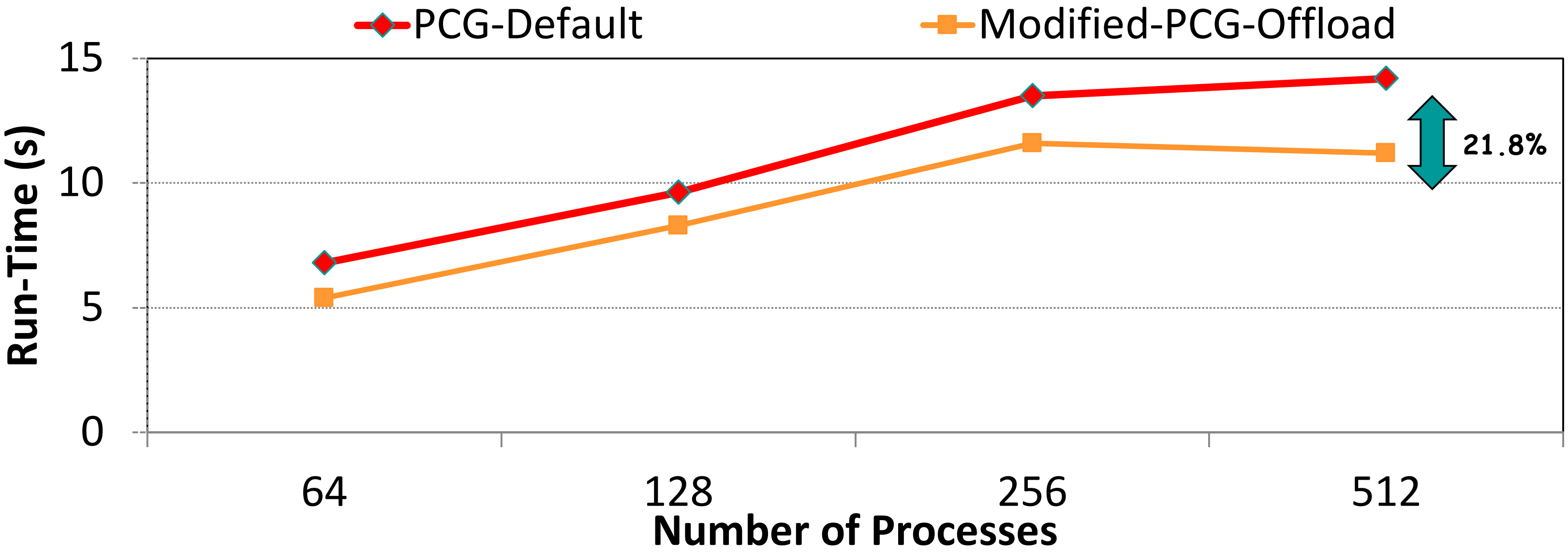
HPL Performance Comparison with 512 Processes

HPL-Offload consistently offers higher throughput than HPL-1ring and HPL-Host. Improves peak throughput by up to 4.5 % for large problem sizes

HPL-Offload surpasses the peak throughput of HPL-1ring with significantly smaller problem sizes and run-times!

K. Kandalla, H. Subramoni, J. Vienne, S. Pai Raikar, K. Tomko, S. Sur, and D K Panda,
Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, (HOTI 2011)

Pre-conditioned Conjugate Gradient (PCG) Solver Performance with Non-Blocking Allreduce based on CX-2 Collective Offload



64,000 unknowns per process.
Modified PCG with Offload-Allreduce performs 21% better than default PCG

K. Kandalla, U. Yang, J. Keasler, T. Kolev, A. Moody, H. Subramoni, K. Tomko, J. Vienne and D. K. Panda, Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS '12, May 2012.

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand Core-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GPU

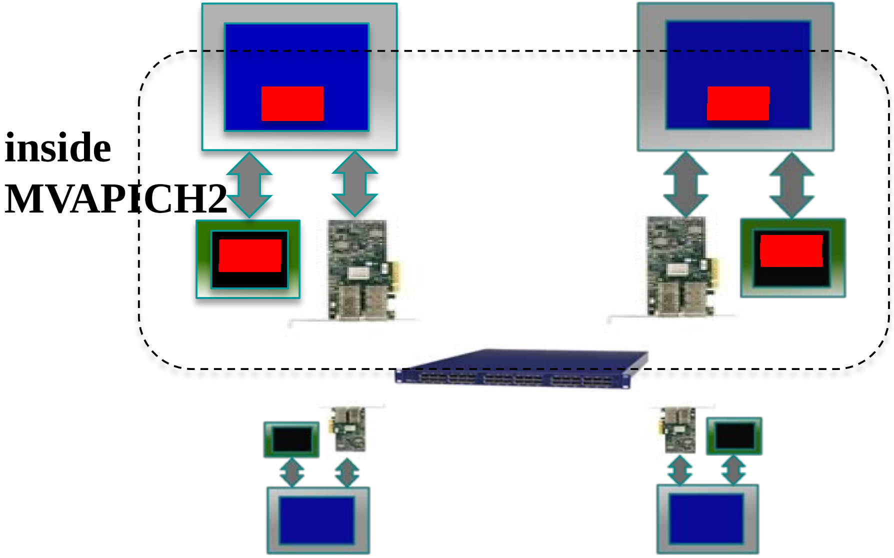
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```



High Performance and High Productivity

CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.3 Releases

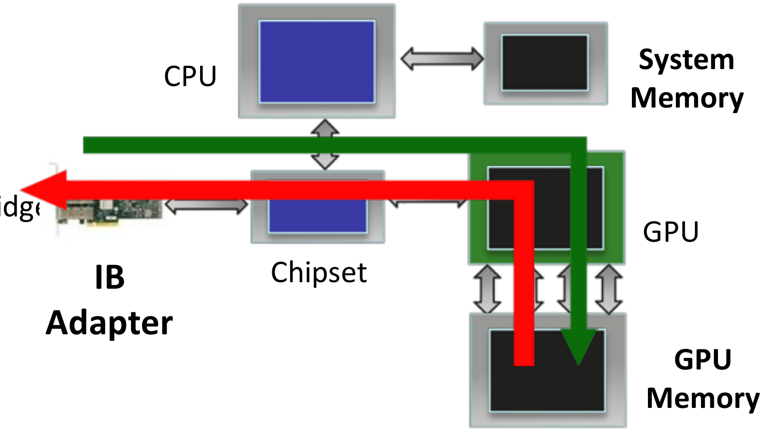
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers
- Unified memory

GPU-Direct RDMA (GDR) with CUDA

- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
    - Similar bottlenecks on Haswell
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters

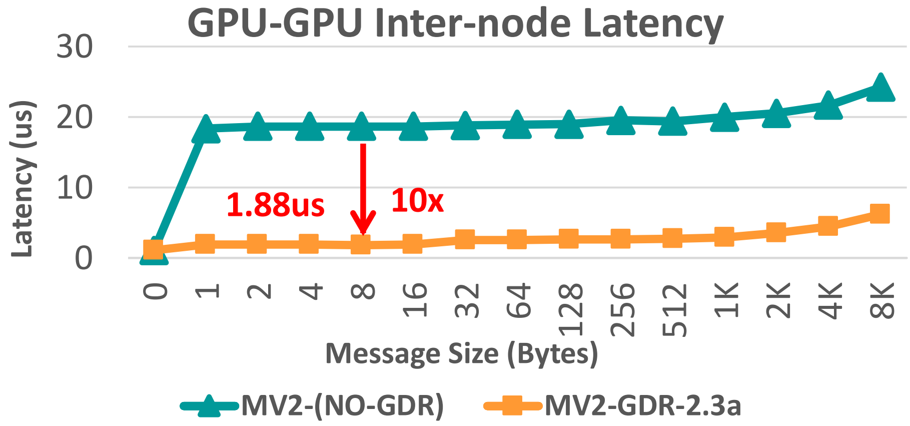
SNB E5-2670 /

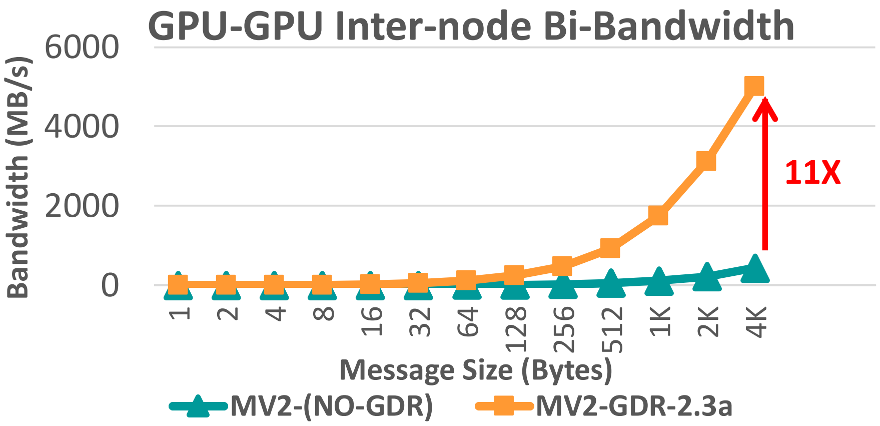
IVB E5-2680V2

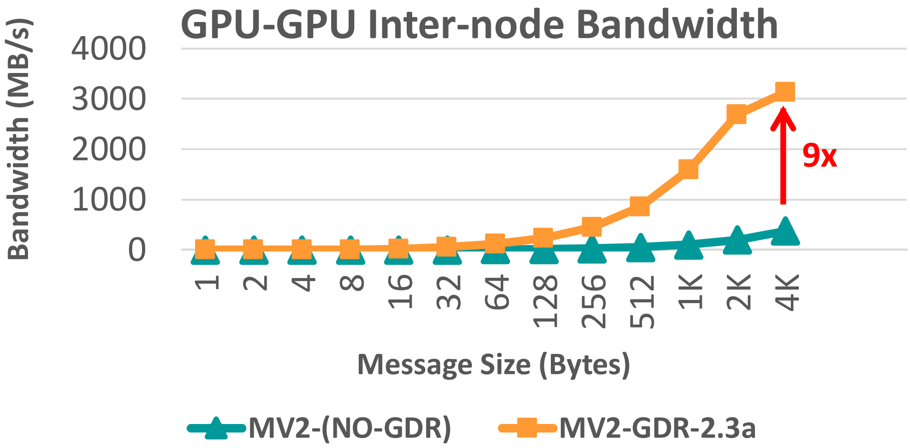


	SNB E5-2670 Intra-socket	SNB E5-2670 Inter-sockets	IVB E5-2680V2 Intra-socket	IVB E5-2680V2 Inter-sockets
P2P read	<1.0 GBs	<300 MBs	3.5 GBs	<300 MBs
P2P write	5.2 GBs	<300 MBs	6.4 GBs	<300 MBs

Optimized MVAPICH2-GDR Design

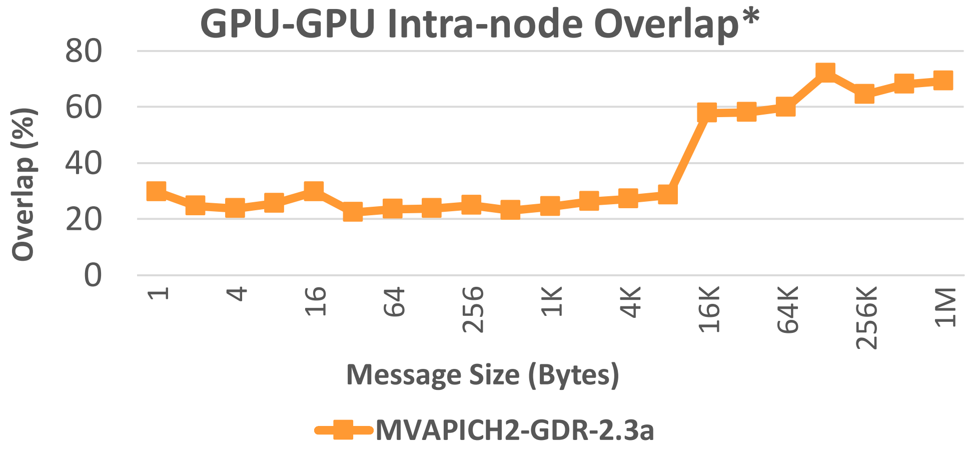


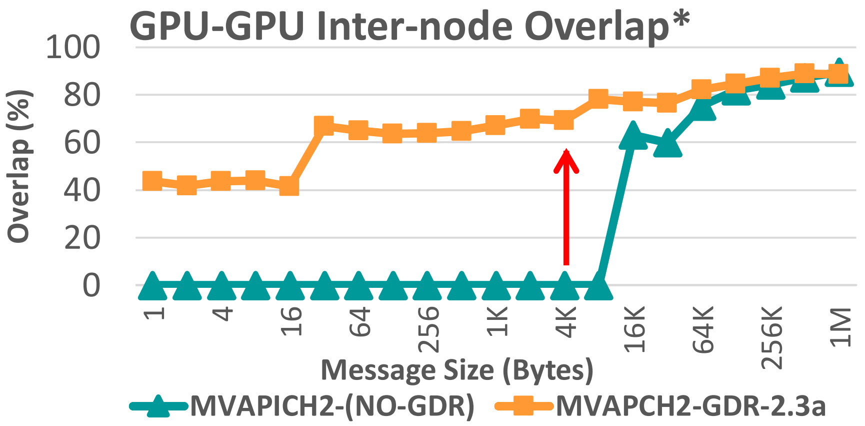




MVAPICH2-GDR-2.3a
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA

Overlap with Optimized MVAPICH2-GDR Design



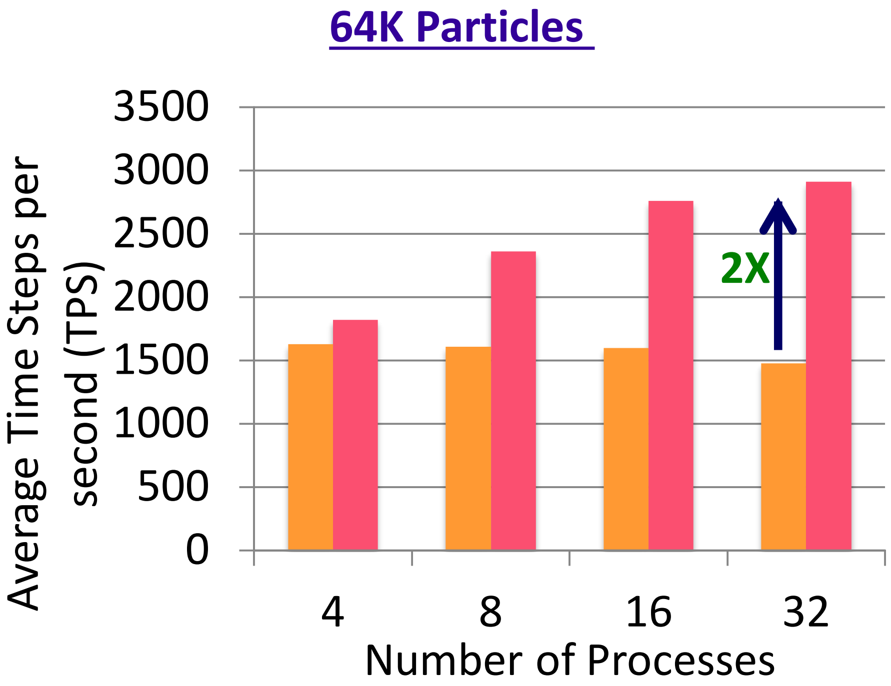


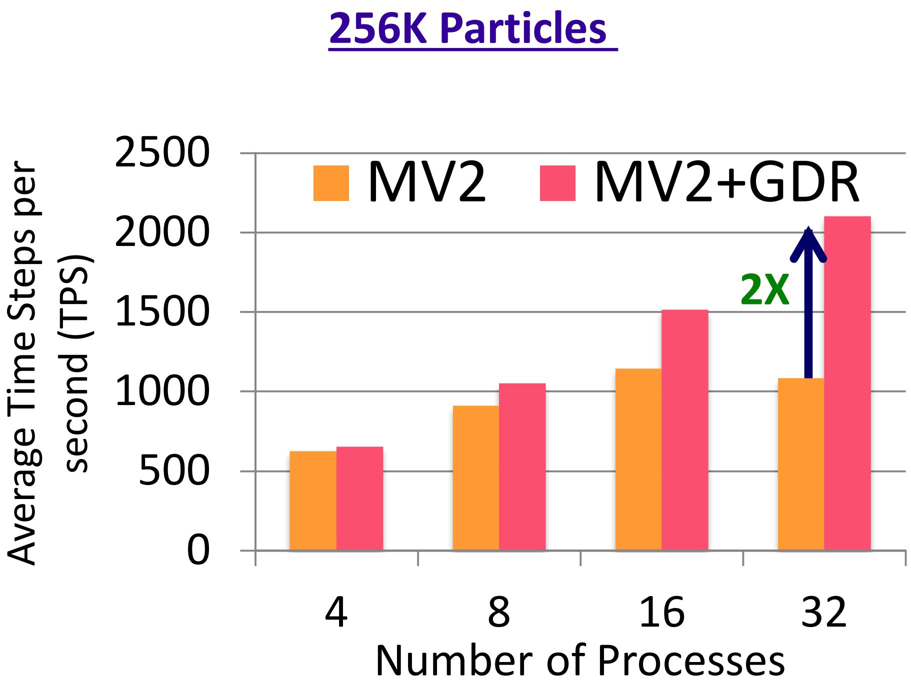
- Up to **69%** overlap* for intra-node GPU-GPU communication
- With GDR, up to **78%** overlap* for inter-node small and medium message transfers
- With intelligent pipeline, up to **88%** overlap* for inter-node large message transfers

MVAPICH2-GDR-2.3a
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA

*Overlap between GPU-to-GPU communication and CPU computation

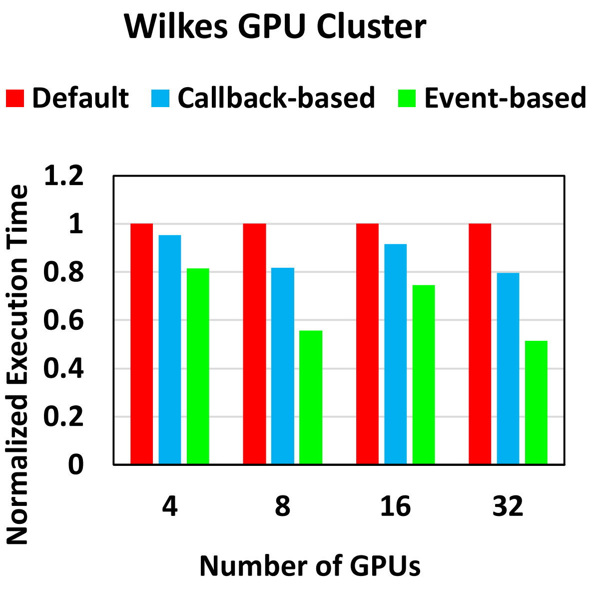
Application-Level Evaluation (HOOMD-blue)

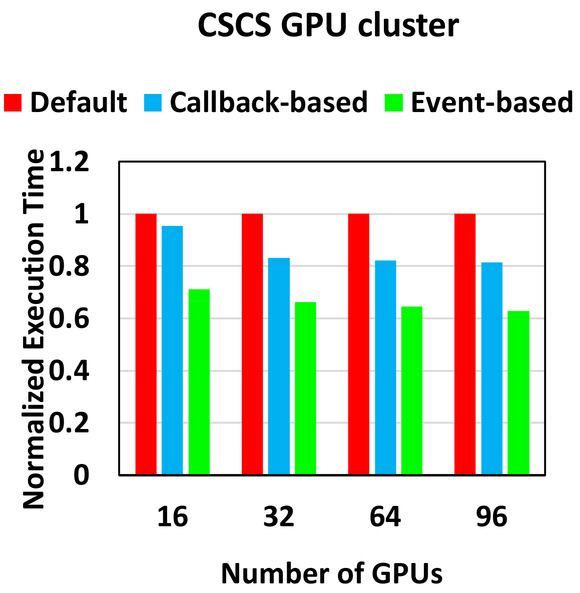


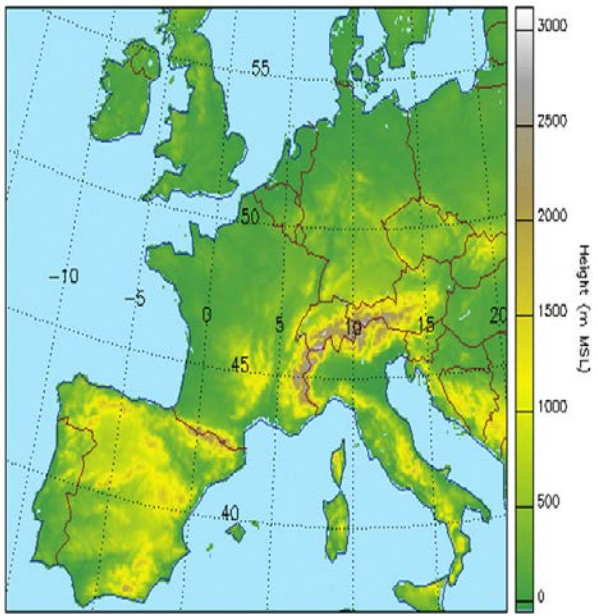


- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomdBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland







Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/

- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

On-going collaboration with CSCS and MeteoSwiss (Switzerland) in co-designing MV2-GDR and Cosmo Application

C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee , H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand CORE-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- OSU Caffe

Motivation: Exploiting CORE-Direct and GPUDirect RDMA

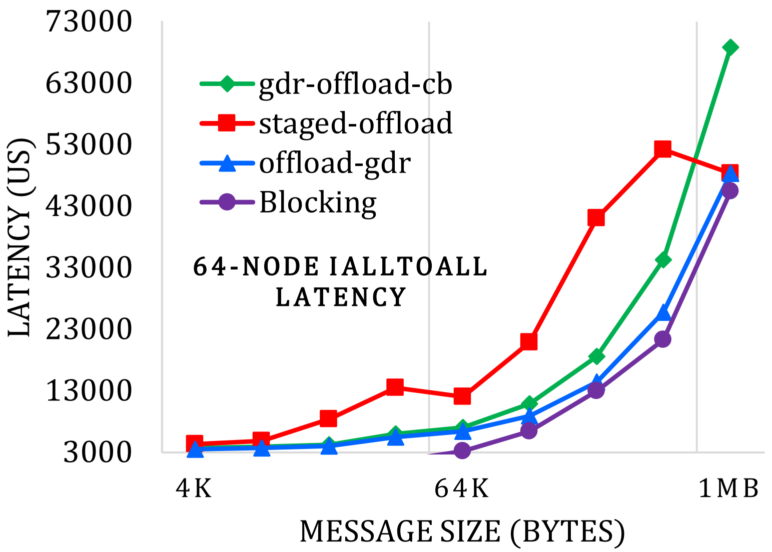
- Applications use GPU/CPU resources for computation and MPI for communication directly from GPU buffers
- MPI collectives common in GPU applications. E.g.: Alltoall for FFTs
- Collectives are time consuming with scale so MPI-3.0 introduced NBCs
- Non-blocking communication operations from GPU buffers can
  - Allow CPU to overlap GPU-based communication with CPU compute
  - Ease GPU kernels redundancy in waiting for non-dependent communication
  - Allow power efficient execution from CPU perspective
- Rich set of GPU and network primitives available for NBC designs but architectural limitations must be addressed

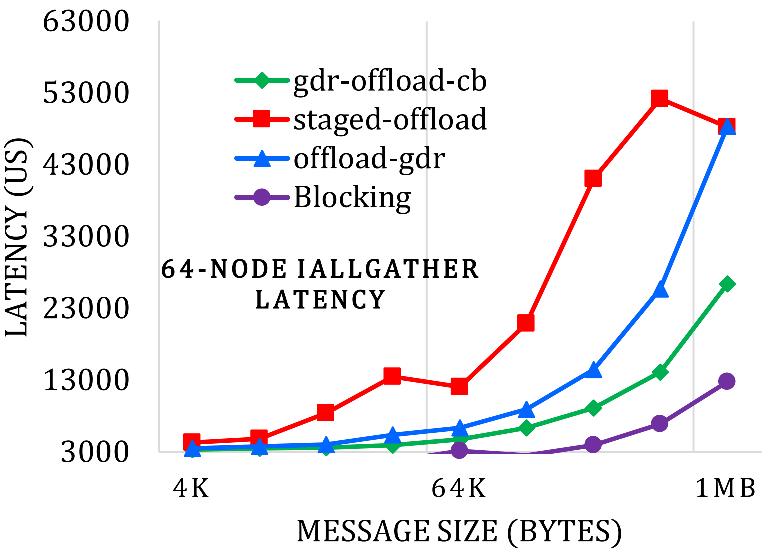
A. Venkatesh, K. Hamidouche, H. Subramoni, and D. K. Panda, Offloaded GPU Collectives using CORE-Direct and CUDA Capabilities on IB Clusters, HiPC ’15

Overview of Core-Direct + GPUDirect Designs

- Realized through mapping of MPICH schedule abstraction
  - Schedule composed of sched_send, sched_barrier, sched_recv, sched_start etc
  - Mapped to Core-Direct primitives with collective-specific GPU↔Host done additionally
- Multiple designs explored
  - Naïve Design: Host-assisted GPU NBC (Scatter)
  - Offload-Staged: Host-assisted GPU NBC + Core-Direct
  - Offload-GDR: (GDR + Core-Direct)-based NBC
  - Offload-Callback: (Core-Direct, GDR, CUDA)-based NBC

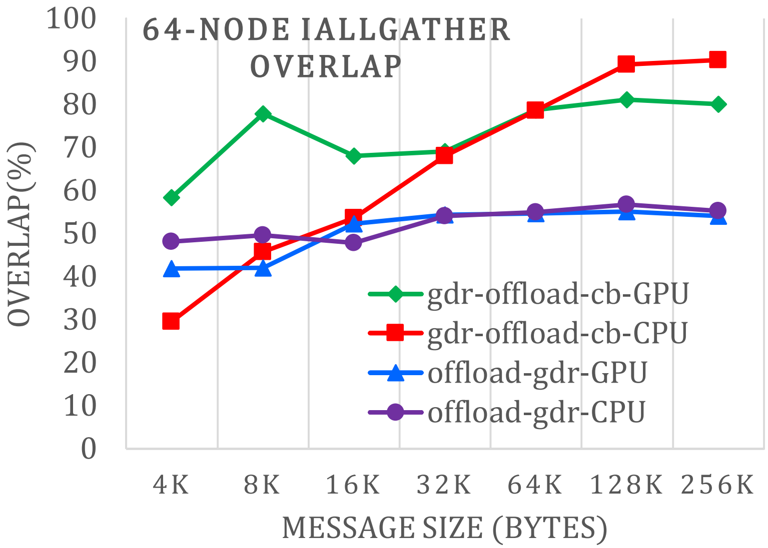
Latency Comparison with Blocking Collectives

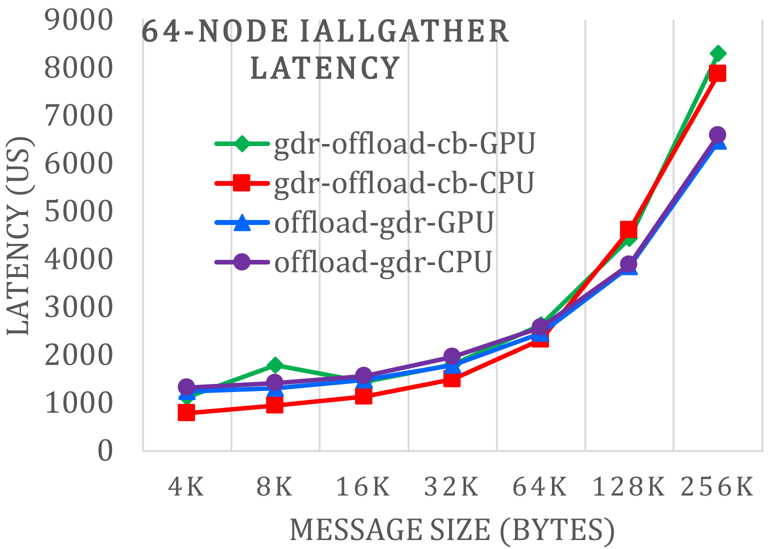




- Use of GDR and CUDA callback mechanisms improve latency (comparable for alltoall)
- Latency high for the case of alltoall even though callback designed to avoid staging latency

Effect of Compute Location on Overlap/Latency





- New schemes are able to exploit overlap well

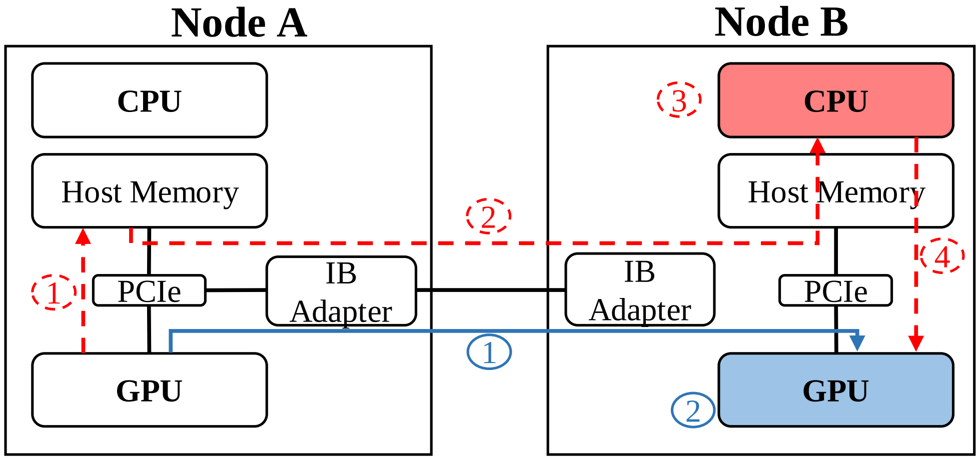
Available in MVAPICH2-GDR 2.3a

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand CORE-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

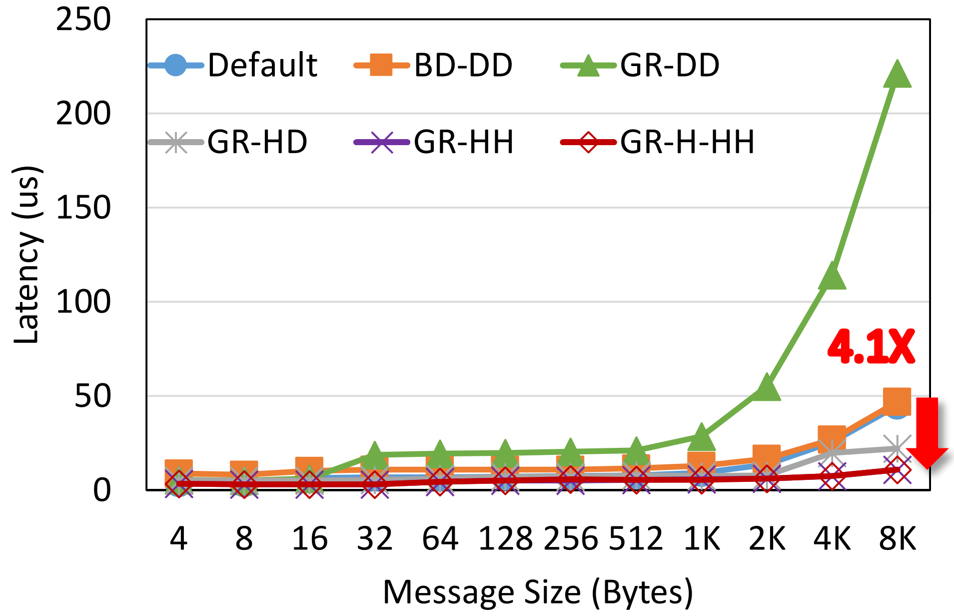
GPU-kernel based Reduction

- Scientific parallel applications spend a considerable amount of time in GPU-based collective communication operations
  - E.g. Deep learning frameworks such as TensorFlow and Caffe
- Optimized computation-intensive collectives in MVAPICH2-GDR
  - MPI_Reduce and MPI_Allreduce
  - Exploring the best combinations
    - Computation on
      - CPU or GPU
    - Communication through
      - Host or GPU memory

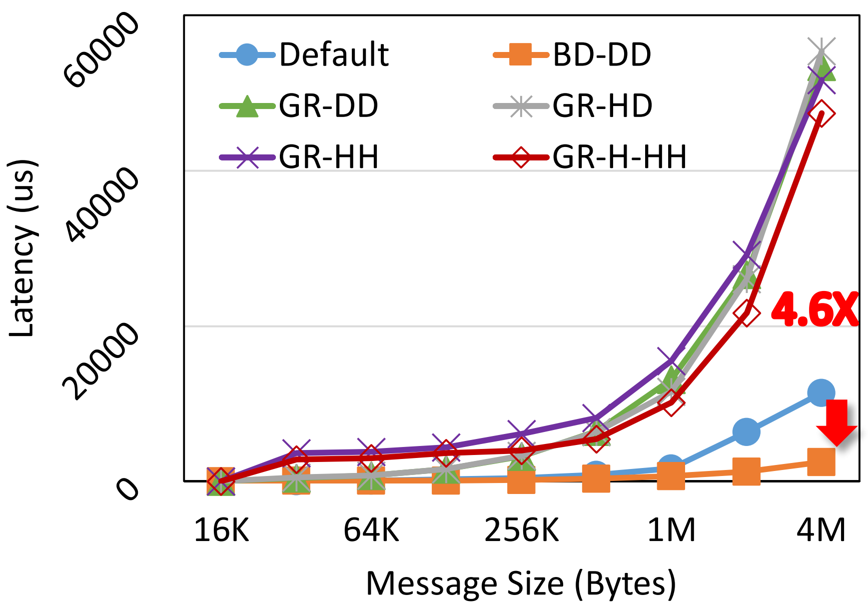


Evaluation - MPI_Reduce @ CSCS (96 GPUs)

Gather-first approaches* win for small messages



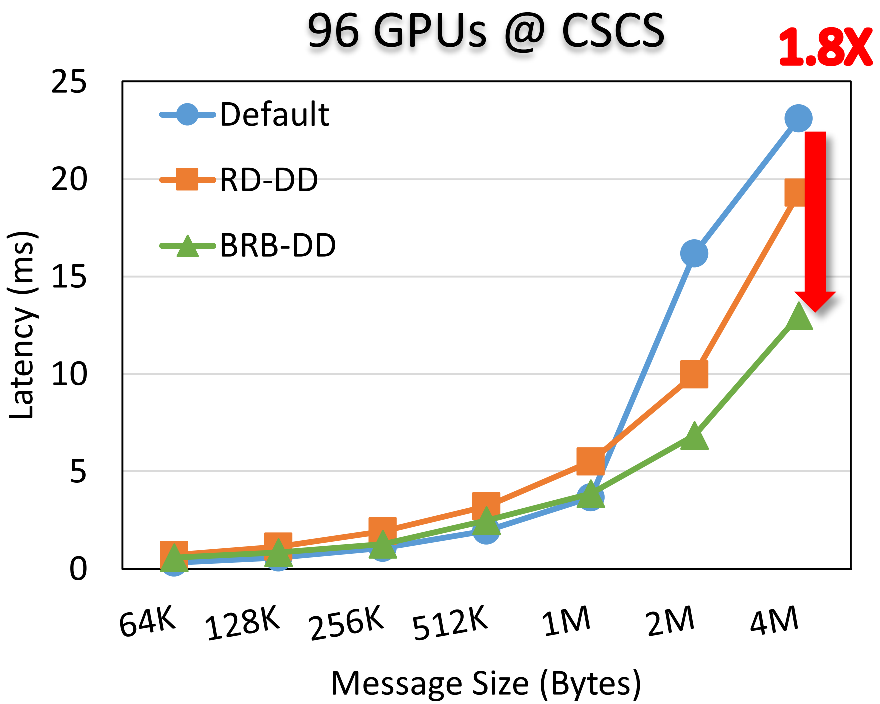
K-nomial GPU-based approach* win for large messages



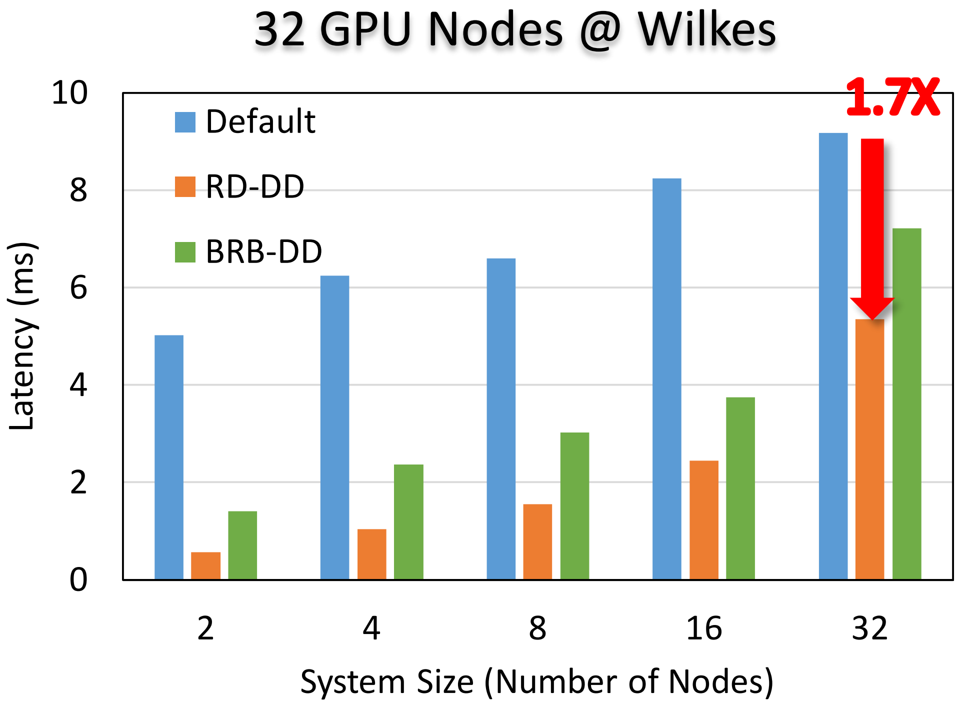
*Ching-Hsiang Chu, Khaled Hamidouche, Akshay Venkatesh, Ammar Ahmad Awan, and Dhabaleswar K. Panda, "CUDA Kernel based Collective Reduction Operations on Large-scale GPU Clusters, " IEEE/ACM CCGrid'16.

Evaluation - MPI_Allreduce

Available in MVAPICH2-GDR 2.3a



Good Scalability



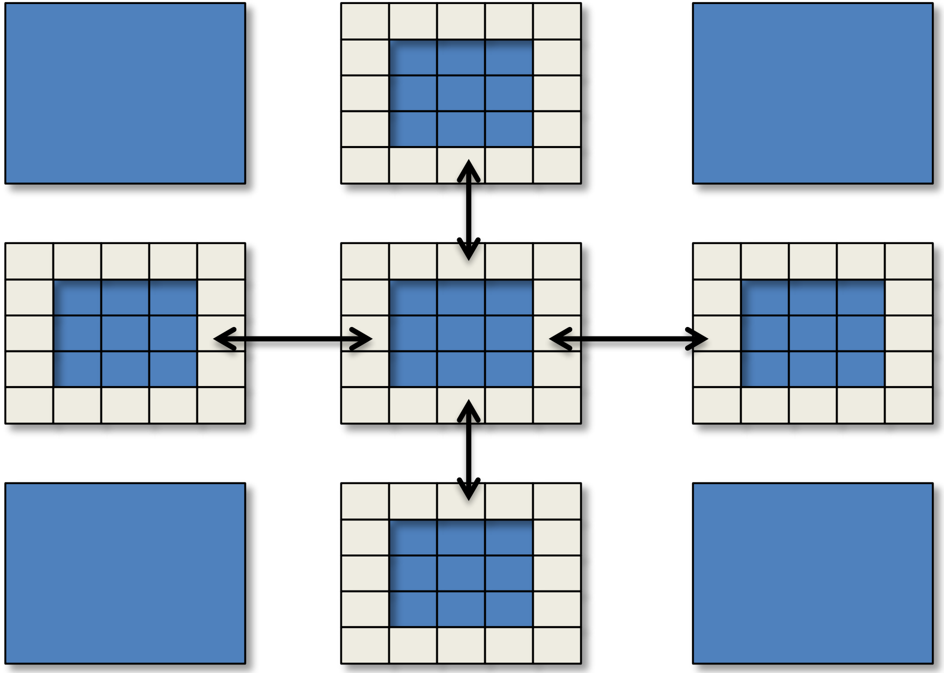
Ching-Hsiang Chu, Khaled Hamidouche, Akshay Venkatesh, Ammar Ahmad Awan, and Dhabaleswar K. Panda, "CUDA Kernel based Collective Reduction Operations on Large-scale GPU Clusters, " IEEE/ACM CCGrid'16.

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand CORE-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

Non-contiguous Data Exchange

Halo data exchange



- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions
- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration

MPI Datatype support in MVAPICH2

- Datatypes support in MPI
  - Operate on customized datatypes to improve productivity
  - Enable MPI library to optimize non-contiguous data

At Sender:

MPI_Type_vector (n_blocks, n_elements, stride, old_type, &new_type);

MPI_Type_commit(&new_type);

…

MPI_Send(s_buf, size, new_type, dest, tag, MPI_COMM_WORLD);

- Inside MVAPICH2
  - Use datatype specific CUDA Kernels to pack data in chunks
  - Efficiently move data between nodes using RDMA
  - In progress - currently optimizes *vector* and *hindexed* datatypes
  - Transparent to the user

H. Wang, S. Potluri, D. Bureddy, C. Rosales and D. K. Panda, GPU-aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation, IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 10, pp. 2595-2605 , Oct 2014.

MPI Datatype Processing (Computation Optimization)

- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels
- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - Subarray
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - Indexed_block
    - MV2_CUDA_KERNEL_IDXBLK_XDIM

MPI Datatype Processing (Communication Optimization)

Common Scenario

MPI_Isend(Buf1, ...,req1);

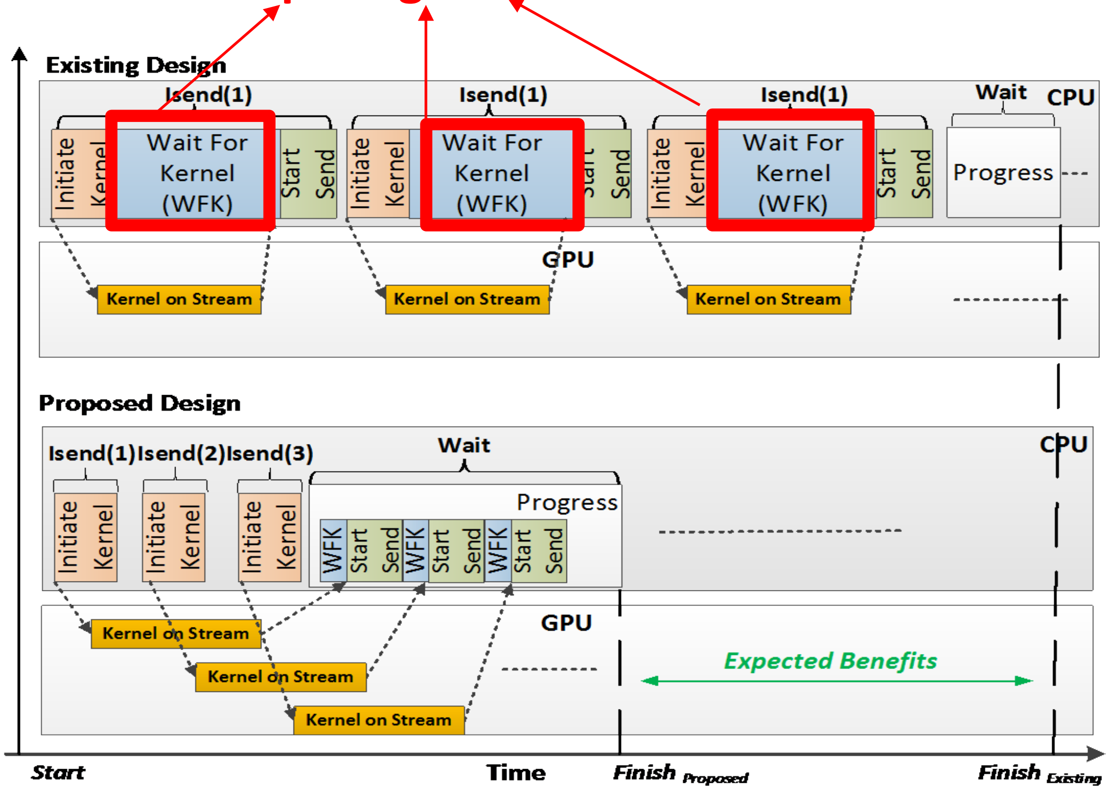
MPI_Isend(Buf2, ...,req2);

Application work on the CPU/GPU

MPI_Waitall(requests, ...)

*Buf1, Buf2...contain non-contiguous MPI Datatype

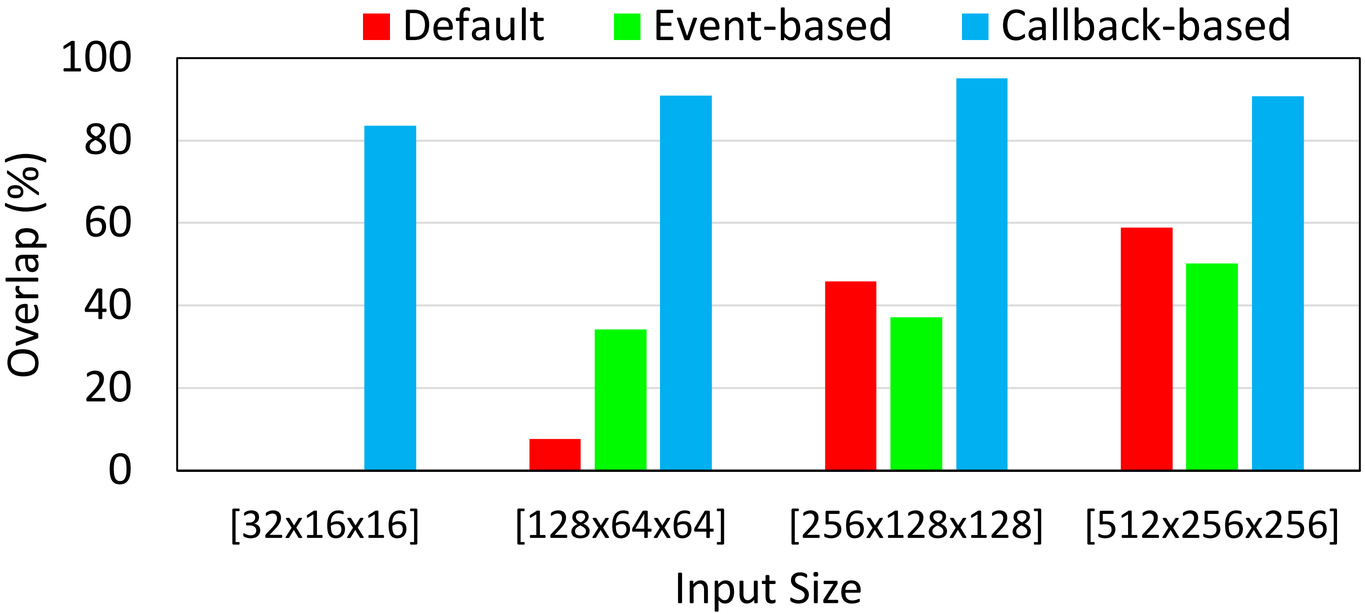
Waste of computing resources on CPU and GPU



MPI Datatype Processing (Communication Optimization)

Available in MVAPICH2-GDR 2.3a

- **Modified 'CUDA-Aware' DDTBench for *NAS_MG_y***
  - Up to 90% overlap between datatype processing and other computation



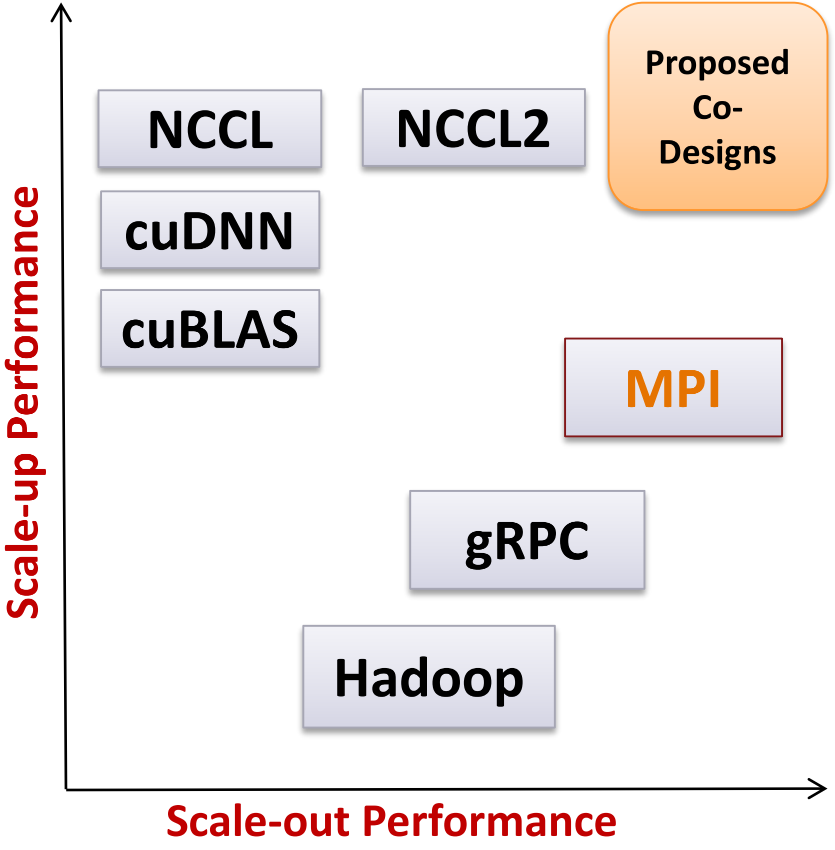
C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee , H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

Presentation Outline

- MVAPICH2/MVAPICH2-X
  - Job Startup
  - Point-to-point Communication
  - Remote Memory Access (RMA)
  - Collective Communication
- MVAPICH2-GDR
  - Support for InfiniBand CORE-Direct
  - GPU-kernel based Reduction
  - Datatype Processing
- Deep Learning Application: OSU Caffe

Deep Learning: New Challenges for MPI Runtimes

- Deep Learning frameworks are a different game altogether
  - Unusually large message sizes (order of megabytes)
  - Most communication based on GPU buffers
- Existing State-of-the-art
  - cuDNN, cuBLAS, NCCL --> **scale-up** performance
  - NCCL2, CUDA-Aware MPI --> **scale-out** performance
    - For small and medium message sizes only!
- Proposed: Can we **co-design** the MPI runtime (**MVAPICH2-GDR**) and the DL framework (**Caffe**) to achieve both?
  - Efficient **Overlap** of Computation and Communication
  - Efficient **Large-Message** Communication (Reductions)
  - What **application co-designs** are needed to exploit **communication-runtime co-designs**?



A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (PPoPP '17)

OSU-Caffe: Proposed Co-Design Overview

- To address the limitations of Caffe and existing MPI runtimes, we propose the **OSU-Caffe (S-Caffe)** framework
- At the application (DL framework) level
  - Develop a fine-grain workflow – i.e. layer-wise communication instead of communicating the entire model
- At the runtime (MPI) level
  - Develop support to perform reduction of very-large GPU buffers
  - Perform reduction using GPU kernels

OSU-Caffe is available from the HiDL project page (http://hidl.cse.ohio-state.edu)

Optimized Data Propagation and Gradient Aggregation using NBC Designs

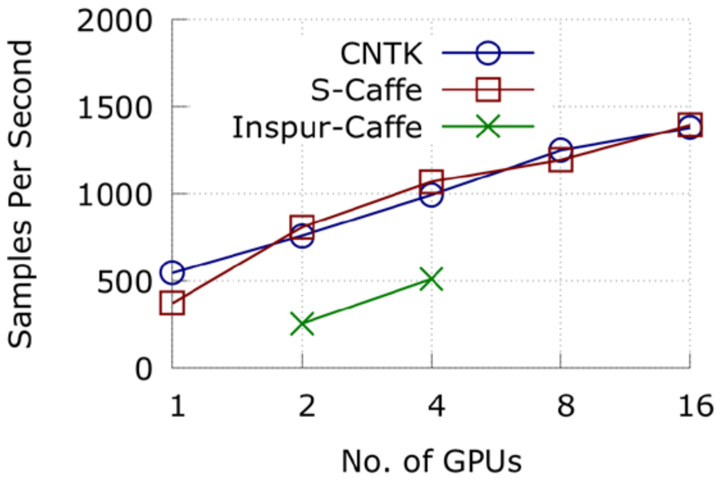
- Exploit Non-Blocking Collective (NBC) operations in MPI-3
  - Divide communication into fine-grain steps
  - Overlap computation of layer **"i"** with communication of layer **"i+1"**
  - **MPI_Ibcast** to post all communication in advance
    - Wait in an on-demand fashion
  - Allow for runtime selection of data propagation design
    - Based on message (DL model) size, number of GPUs, and number of nodes
- Co-design gradient aggregation at application level
  - **Helper thread** based approach to realize a **non-blocking MPI_Reduce**

A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters, PPoPP '17

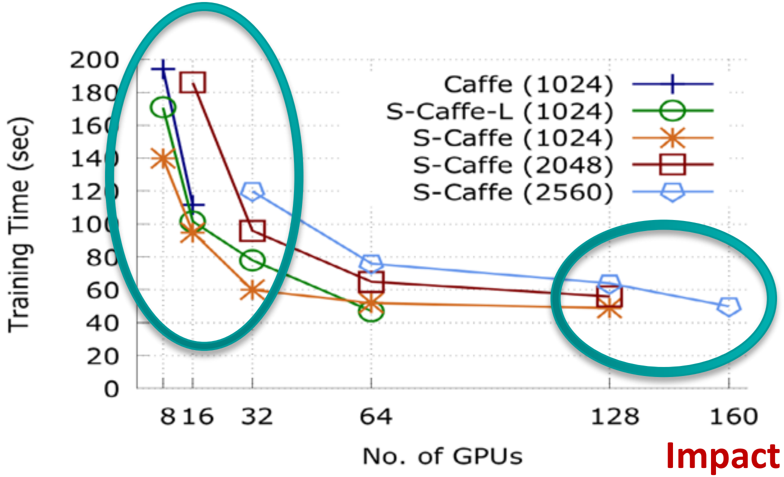
S-Caffe vs. Inspur-Caffe and Microsoft CNTK

- AlexNet: Notoriously hard to scale-out on multiple nodes due to comm. overhead!
- Large number of parameters ~ 64 Million (comm. buffer size = 256 MB)

- GoogLeNet is a popular DNN
- 13 million parameters (comm. buffer size = ~50 MB)



Up to 14% improvement (Scale-up)



Impact of HR

S-Caffe delivers better or comparable performance with other multi-node capable DL frameworks

Concluding Remarks

- Exploiting overlap between computation and communication is significant in HPC
- Presented some of the approaches and results along these directions taken by the MVAPICH2 and MVAPICH2-GDR Libraries
- Allows applications to take advantage of the overlap capabilities
- As exascale systems are getting more complicated in their architectures, solutions exploiting overlap capabilities will be important

Additional Presentation and Tutorials

- Tutorial: How to Boost the Performance of Your MPI and PGAS Applications with MVAPICH2 Libraries
  - 04/05/18, 9:00 am-12:00 noon
- Tutorial: High Performance Distributed Deep Learning: A Beginner's Guide
  - 04/05/18, 1:00pm-4:00 pm

Funding Acknowledgments

Funding Support by

Office of Science U.S. Department of Energy
NSF
Mellanox Technologies
Microsoft
QLogic
Cray The Supercomputer Company
intel
Linux Networx
NVIDIA
NetApp
Sun microsystems
Cisco Systems

Equipment Support by

Mellanox Technologies
intel
AMD
Sun microsystems
APPRO
advanced clustering technologies, inc.
Microway Technology you can count on
NVIDIA
QLogic

Personnel Acknowledgments

Current Students (Graduate)

- A. Awan (Ph.D.)
- R. Biswas (M.S.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- H. Javed (Ph.D.)
- P. Kousha (Ph.D.)
- D. Shankar (Ph.D.)
- H. Shi (Ph.D.)
- J. Zhang (Ph.D.)

Current Students (Undergraduate)

- N. Sarkauskas (B.S.)

Current Research Scientists

- X. Lu
- H. Subramoni

Current Post-doc

- A. Ruhela
- K. Manian

Current Research Specialist

- J. Smith
- M. Arnold

Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Research Scientist

- K. Hamidouche
- S. Sur

Past Programmers

- D. Bureddy
- J. Perkins

Past Post-Docs

- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- H. Wang

Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory

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

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/