

Exploiting Full Potential of GPU Clusters with InfiniBand using MVAPICH2-GDR

Presentation at Mellanox Theater (SC'15)

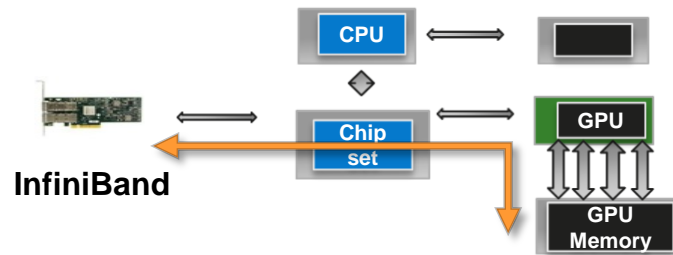
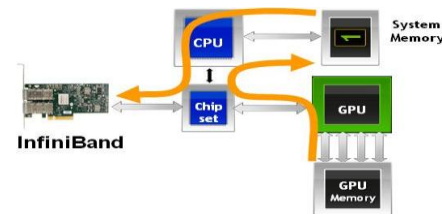
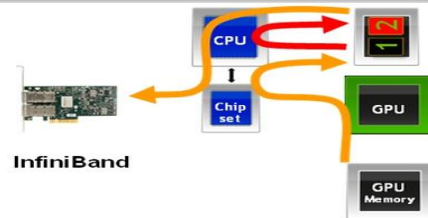
Dhabaleswar K. (DK) Panda - The Ohio State University

panda@cse.ohio-state.edu



- **Communication on InfiniBand Clusters with GPUs**
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- New Enhancements
 - Non-Blocking Collectives (MPI3 NBC) support
 - Multi-Stream for efficient MPI Datatype Processing
- On-Going Work (GDR-Async and Managed Memory)
- OpenACC-Aware support
- Conclusions

- Before CUDA 4: Additional copies
 - Low performance and low productivity
- After CUDA 4: Host-based pipeline
 - Unified Virtual Address
 - Pipeline CUDA copies with IB transfers
 - **High performance and high productivity**
- After CUDA 5.5: GPUDirect-RDMA support
 - GPU to GPU direct transfer
 - Bypass the host memory
 - Hybrid design to avoid PCI bottlenecks



- High Performance MPI and PGAS Library for InfiniBand, 10-40Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE) and Accelerators

MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 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

Used by more than 2,475 organizations in 76 countries

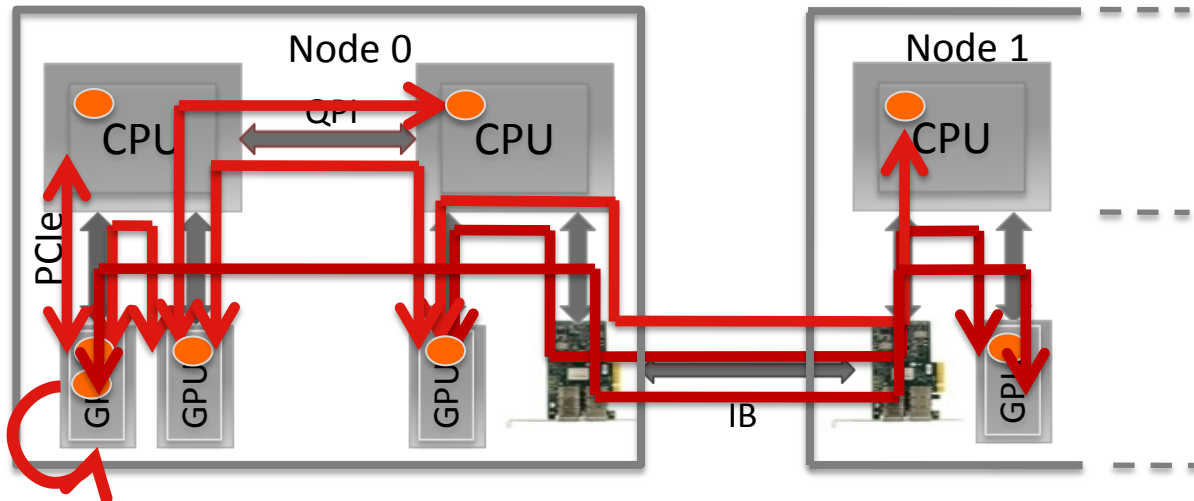
More than 308,000 downloads from the OSU site directly

Empowering many TOP500 clusters (Nov '15 ranking)

- 10th ranked 519,640-core cluster (Stampede) at TACC
- 13th ranked 185,344-core cluster (Pleiades) at NASA
- 25th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
- Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)

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

- Connected as PCIe devices – Flexibility but Complexity



● Memory buffers

1. Intra-GPU
2. Intra-Socket GPU-GPU
3. Inter-Socket GPU-GPU
4. Inter-Node GPU-GPU
5. Intra-Socket GPU-Host
6. Inter-Socket GPU-Host
7. Inter-Node GPU-Host

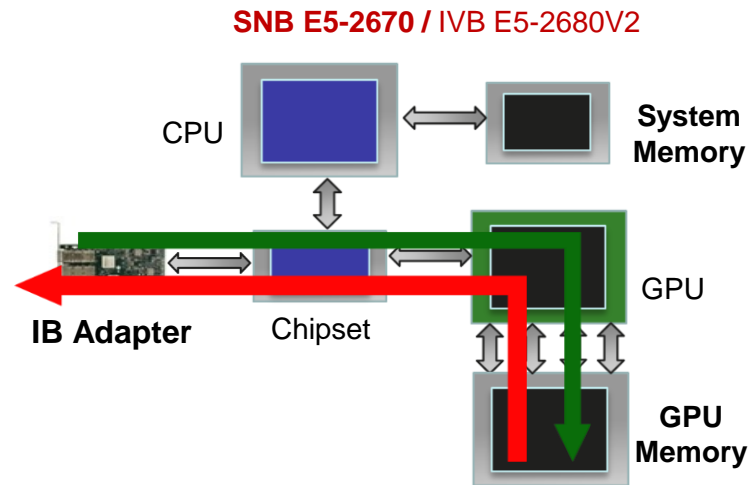
8. Inter-Node GPU-GPU with IB adapter on remote socket
and more . . .

- For each path different schemes: Shared_mem, IPC, GPUDirect RDMA, pipeline ...
- Critical for runtimes to optimize data movement while hiding the complexity

- Communication on InfiniBand Clusters with GPUs
- **MVASPICH2-GPU with GPUDirect-RDMA (GDR)**
- New Enhancements
 - Non-Blocking Collectives (MPI3 NBC) support
 - Multi-Stream for efficient MPI Datatype Processing
- On-Going Work (GDR-Async and Managed Memory)
- OpenACC-Aware support
- Conclusions

GPUDirect RDMA (GDR) with CUDA

- Hybrid design using GPUDirect RDMA
 - GPUDirect RDMA and Host-based pipelining
 - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
- 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

P2P write: 5.2 GB/s

P2P read: < 1.0 GB/s

IVB E5-2680V2

P2P write: 6.4 GB/s

P2P read: 3.5 GB/s

S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy and D. K. Panda, Efficient Inter-node MPI Communication using GPUDirect RDMA for InfiniBand Clusters with NVIDIA GPUs, Int'l Conference on Parallel Processing (ICPP '13)

- MVAPICH2-2.2a with GDR support can be downloaded from

<https://mvapich.cse.ohio-state.edu/download/mvapich2gdr/>

- System software requirements

- Mellanox OFED 2.1 or later
- NVIDIA Driver 331.20 or later
- NVIDIA CUDA Toolkit 6.5 or later
- Plugin for GPUDirect RDMA

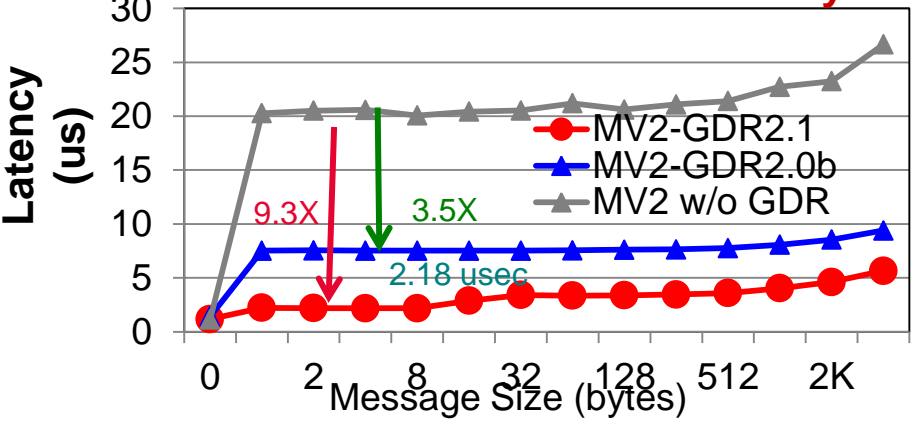
http://www.mellanox.com/page/products_dyn?product_family=116

Strongly Recommended: use the new GDRCOPY module from NVIDIA

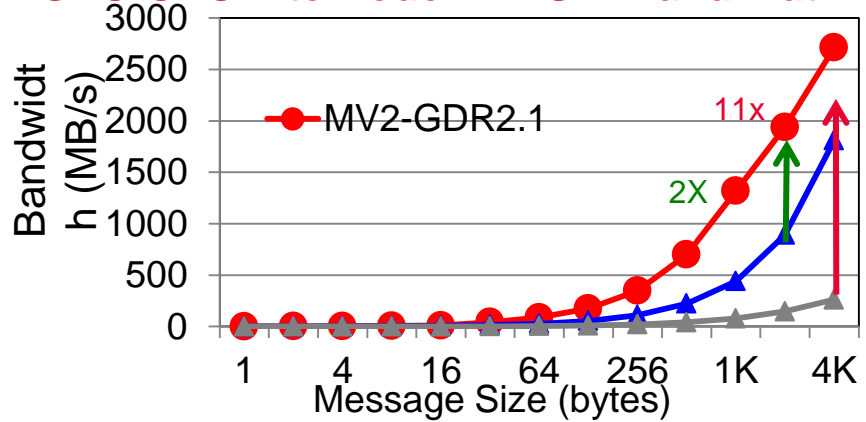
- Has optimized designs for point-to-point and collective communication using GDR
- Contact MVAPICH help list with any questions related to the package

mvapich-help@cse.ohio-state.edu

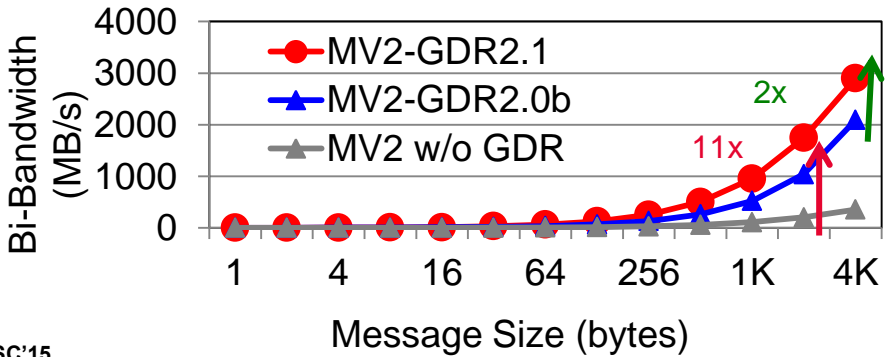
GPU-GPU Internode MPI Latency



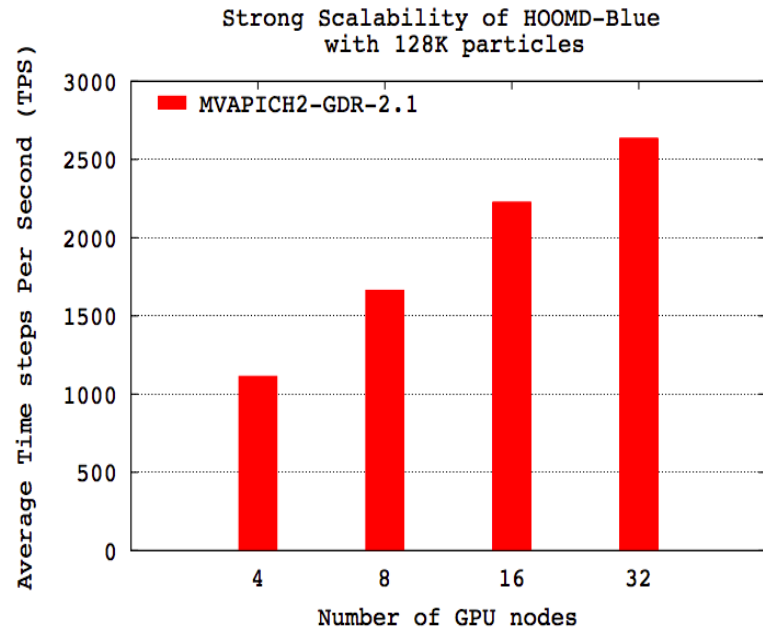
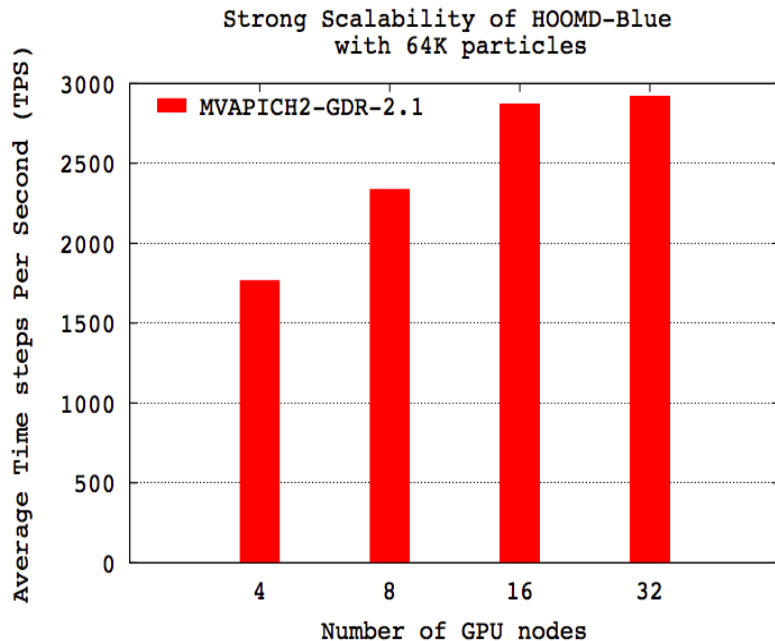
GPU-GPU Internode MPI Uni-Bandwidth



GPU-GPU Internode Bi-directional Bandwidth



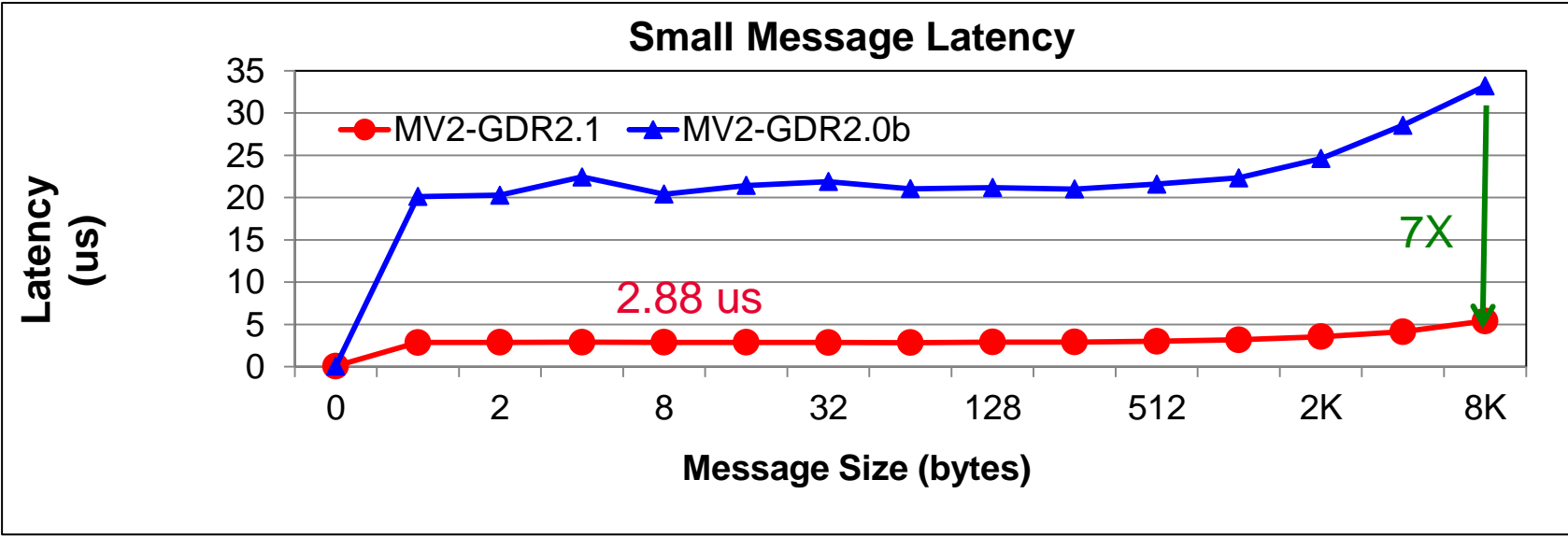
MVAPICH2-GDR-2.1
 Intel Ivy Bridge (E5-2680 v2) node - 20 cores
 NVIDIA Tesla K40c GPU
 Mellanox Connect-IB Dual-FDR HCA
 CUDA 7
 Mellanox OFED 2.4 with GPU-Direct-RDMA



- 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

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

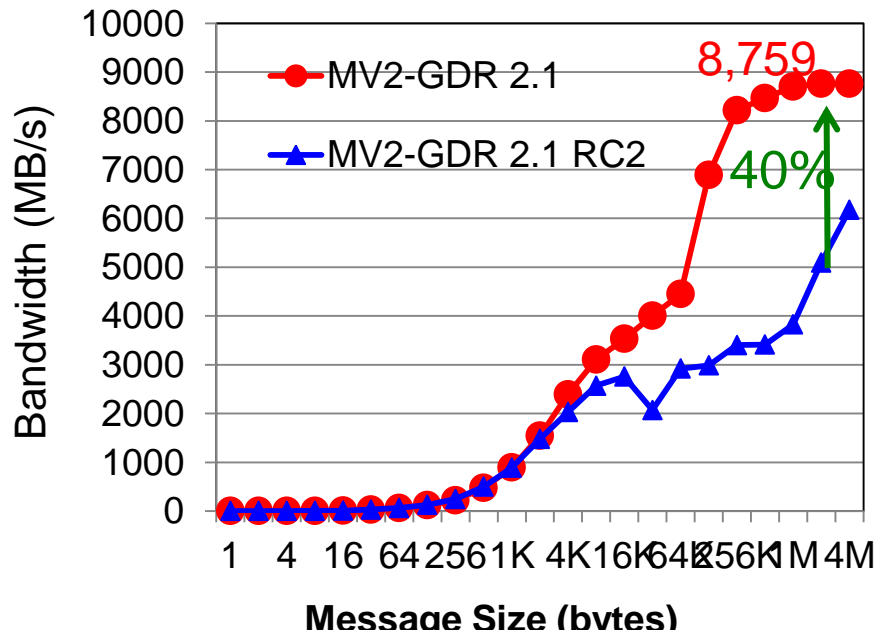
MPI-3 RMA provides flexible synchronization and completion primitives



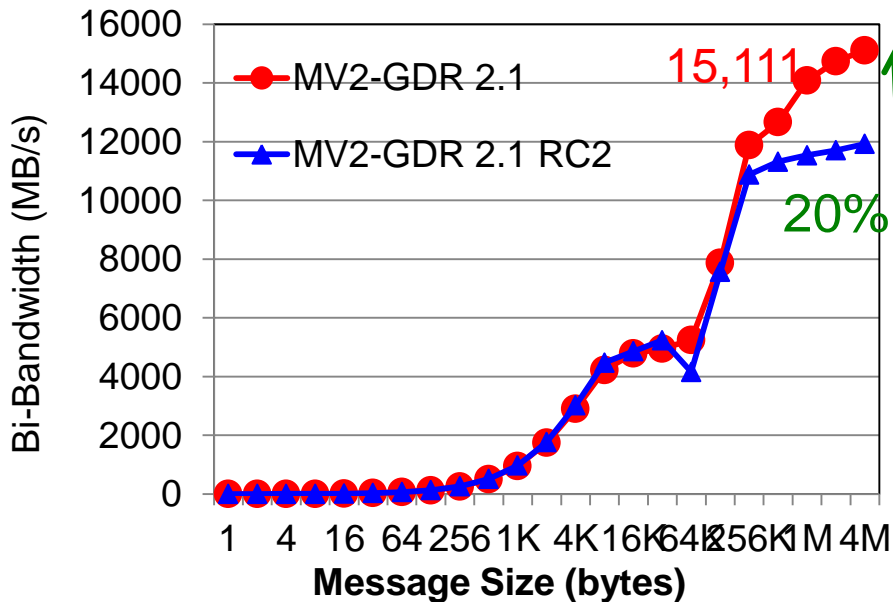
MVAPICH2-GDR-2.1
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 7, Mellanox OFED 2.4 with GPU-Direct-RDMA

Performance of MVAPICH2-GDR with GPU-Direct-RDMA and Multi-Rail Support

GPU-GPU Internode MPI Uni-Directional Bandwidth



GPU-GPU Internode Bi-directional Bandwidth



MVAPICH2-GDR-2.1

Intel Ivy Bridge (E5-2680 v2) node - 20 cores

NVIDIA Tesla K40c GPU

Mellanox Connect-IB Dual-FDR HCA CUDA 7

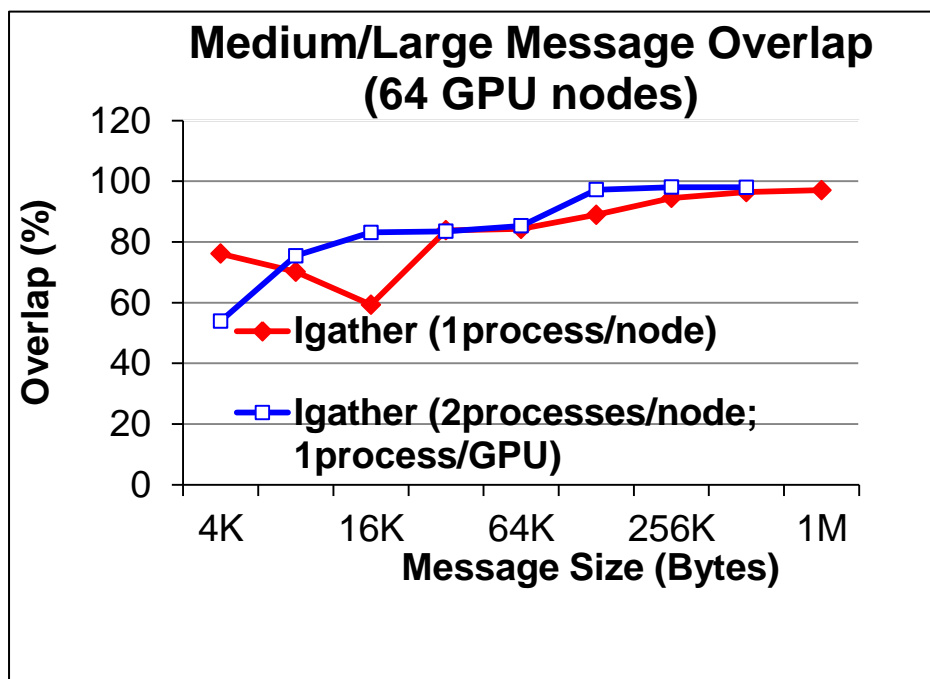
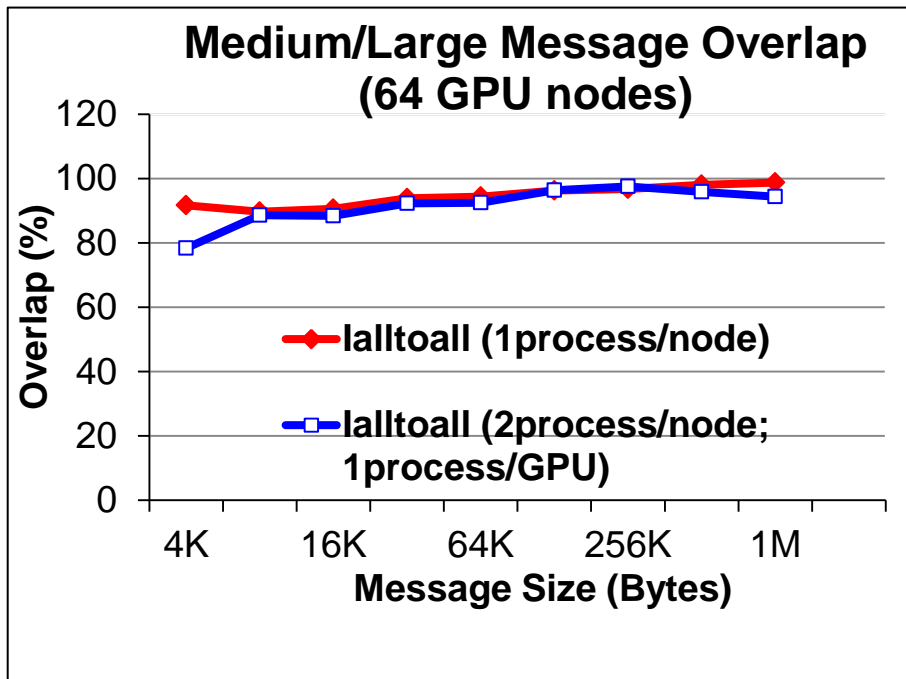
Mellanox OFED 2.4 with GPU-Direct-RDMA

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- **New Enhancements**
 - **Non-Blocking Collectives (MPI3 NBC) support**
 - Multi-Stream for efficient MPI Datatype Processing
- On-Going Work (GDR-Async and Managed Memory)
- OpenACC-Aware support
- Conclusions

Non-blocking collectives (NBC) using network tasks offload mechanism (CORE-Direct)

- MPI NBC decouple initiation(lalltoall) and completion(Wait) phases and provide overlap potential (lalltoall + compute + Wait) but CPU drives progress largely in Wait (=> 0 overlap)
- CORE-Direct feature provides true overlap capabilities by providing a priori specification of a list of network-tasks which is progressed by the NIC instead of the CPU (hence freeing it)
- We propose a design that **combines GPUDirect RDMA and Core-Direct features** to provide efficient support of CUDA-Aware NBC collectives on GPU buffers
 - Overlap communication with CPU computation
and
 - Overlap communication with GPU computation
- Extend OMB with CUDA-Aware NBC benchmarks to evaluate
 - Latency
 - Overlap on both CPU and GPU

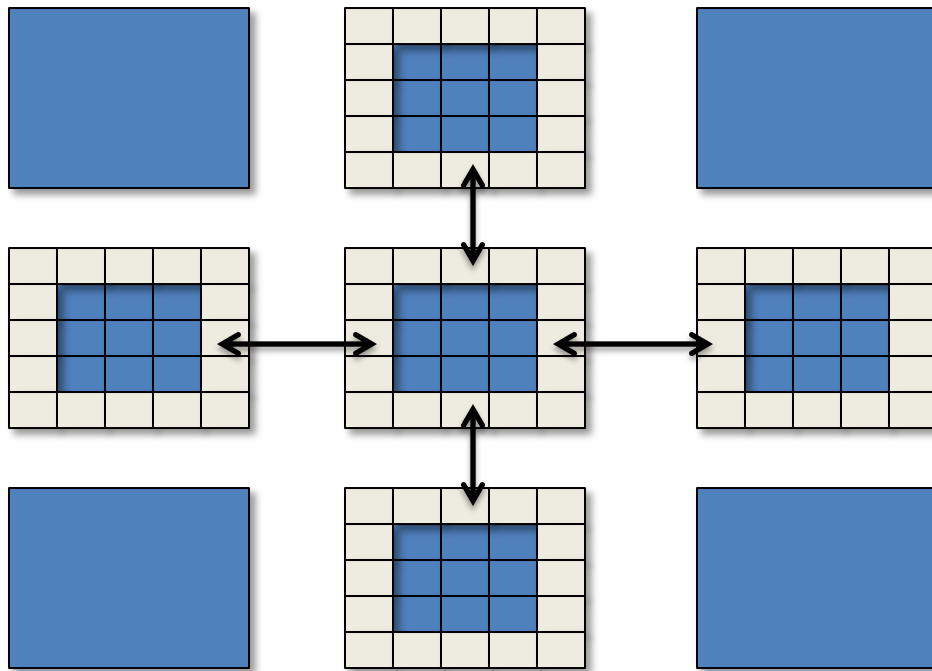
[A. Venkatesh, K. Hamidouche, H. Subramoni, and D. K. Panda, "Offloaded GPU Collectives using CORE-Direct and CUDA Capabilities on IB Clusters", HIPC, 2015](#)



Platform: Wilkes: Intel Ivy Bridge
 NVIDIA Tesla K20c + Mellanox Connect-IB
 MVA PICH2-GDR 2.2a

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- **New Enhancements**
 - Non-Blocking Collectives (MPI3 NBC) support
 - **Multi-Stream for efficient MPI Datatype Processing**
- On-Going Work (GDR-Async and Managed Memory)
- OpenACC-Aware support
- Conclusions

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

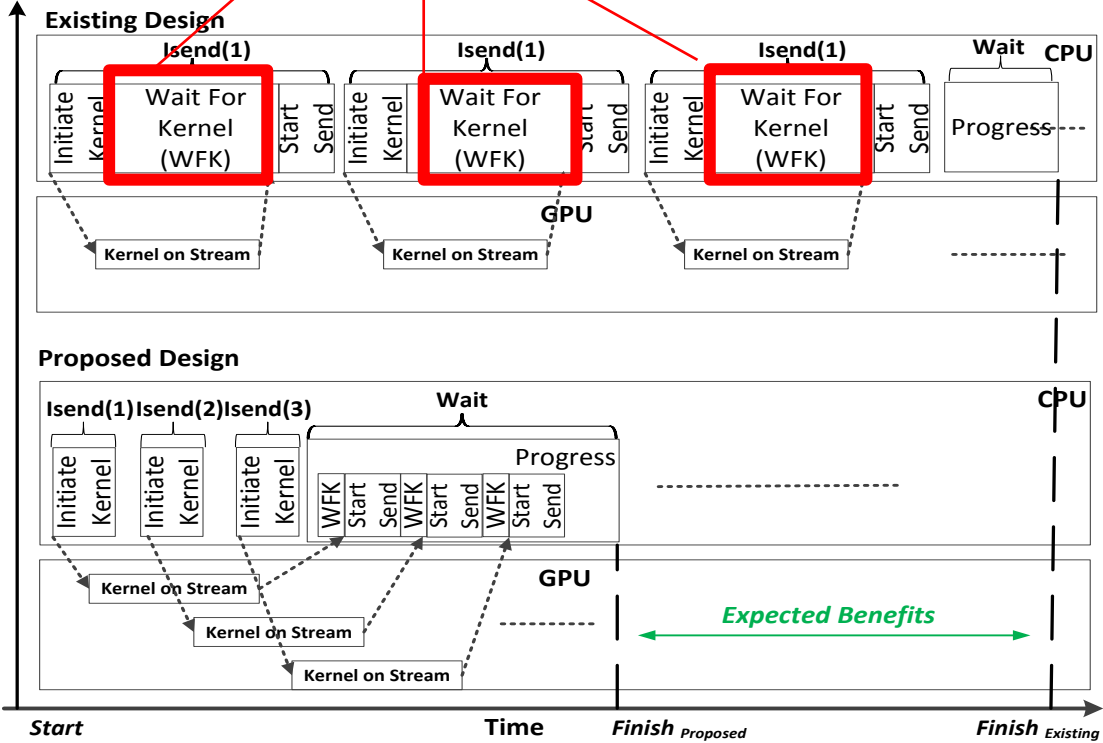
- Comprehensive support
 - Targeted kernels for regular datatypes - vector, subarray, indexed_block
 - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
 - Avoids stream conflicts with application kernels
- Flexible set of parameters 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

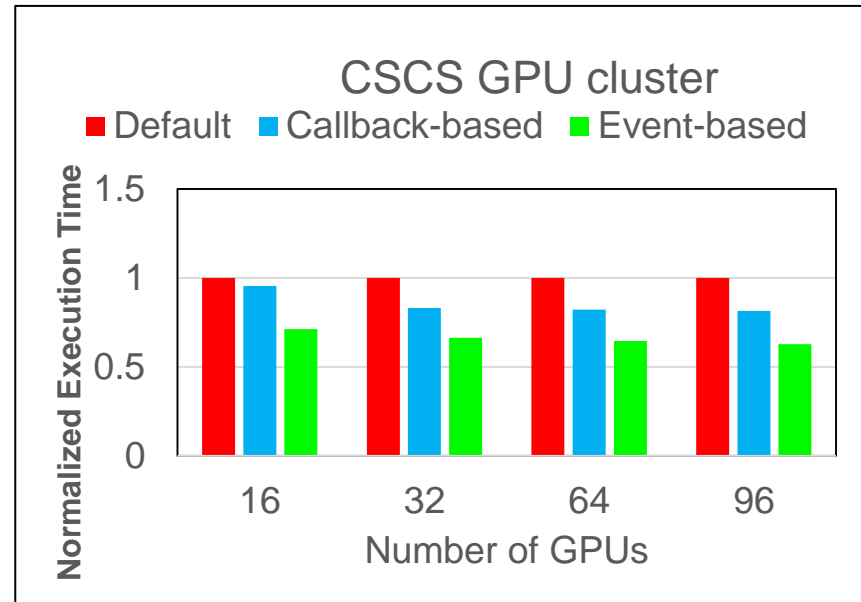
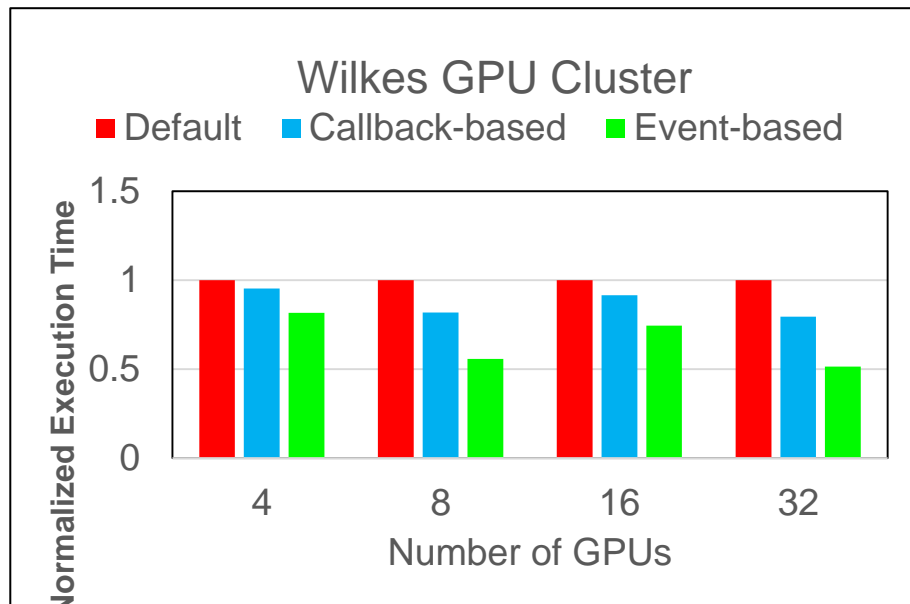
Common Scenario

```
MPI_Isend (A,.. Datatype,...)  
MPI_Isend (B,.. Datatype,...)  
MPI_Isend (C,.. Datatype,...)  
MPI_Isend (D,.. Datatype,...)  
...
```

```
MPI_Waitall (...);
```

Waste of computing resources on CPU and GPU





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

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", under review

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- New Enhancements
 - Non-Blocking Collectives (MPI3 NBC) support
 - Multi-Stream for efficient MPI Datatype Processing
- **On-Going Work (GDR-Async and Managed Memory)**
- OpenACC-Aware support
- Conclusions

- **Support for GDR_Async feature (GPUDirect RDMA Family 4)**
 - Offload control flow to the GPU
 - Issue the communication operation from/to GPU
 - Free CPU and remove from critical path
 - Hide the overhead of launching CUDA Kernels and keep the GPU busy
 - Extend OMB with GDR_Async semantics
- **Initial Support for Managed Memory**
 - MPI-Aware managed memory
 - Transparently handle the data movement of managed memory at MPI level
 - High productivity (unique pointer for both CPU and GPU work)
 - Extend OMB with managed memory semantics

- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- New Enhancements
 - Non-Blocking Collectives (MPI3 NBC) support
 - Multi-Stream for efficient MPI Datatype Processing
- On-Going Work (GDR-Async and Managed Memory)
- **OpenACC-Aware support**
- Conclusions

`acc_malloc` to allocate device memory

No changes to MPI calls

MVA PICH2 detects the device pointer and optimizes data movement

`acc_deviceptr` to get device pointer (in OpenACC 2.0)

Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations

MVA PICH2 will detect the device pointer and optimize communication

Delivers the same performance as with CUDA

```
A = acc_malloc(sizeof(int) * N);
.....
#pragma acc parallel loop deviceptr(A) ...
//compute for loop

MPI_Send (A, N, MPI_INT, 0, 1, MPI_COMM_WORLD);

.....
acc_free(A);
```

```
A = malloc(sizeof(int) * N);
.....
#pragma acc data copyin(A) ...
{
#pragma acc parallel loop ...
//compute for loop
MPI_Send(acc_deviceptr(A), N, MPI_INT, 0, 1,
MPI_COMM_WORLD);
}
.....
free(A);
```


- Communication on InfiniBand Clusters with GPUs
- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- New Enhancements
 - Non-Blocking Collectives (MPI3 NBC) support
 - Multi-Stream for efficient MPI Datatype Processing
- On-Going Work (GDR-Async and Managed Memory)
- OpenACC-Aware support
- **Conclusions**

- MVAPICH2 optimizes MPI communication on InfiniBand clusters with GPUs
- Provides optimized designs for point-to-point two-sided and one-sided communication, datatype processing and collective operations
- Efficient and maximal overlap for MPI3 NBC collectives
- Takes advantage of CUDA features like IPC and GPUDirect RDMA families
- Delivers
 - High performance
 - High productivityWith support for latest NVIDIA GPUs and InfiniBand Adapters
- **Users are strongly encouraged to use MVAPICH2-GDR 2.2a**

Network-Based Computing Laboratory

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

MVAPICH Web Page

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