ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

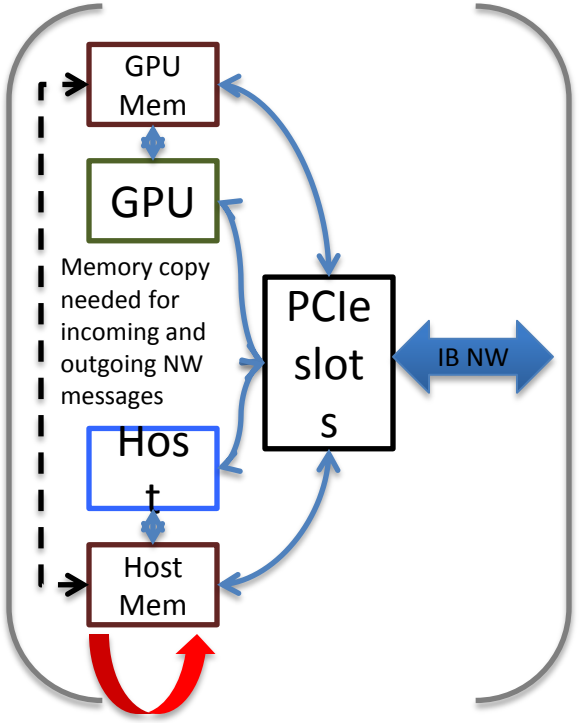
CSCS
Swiss National Supercomputing Centre

Towards Transparent and Efficient GPU Communication on InfiniBand Clusters

Sadaf Alam
Jeffrey Poznanovic
Kristopher Howard
Hussein Nasser El-Harake

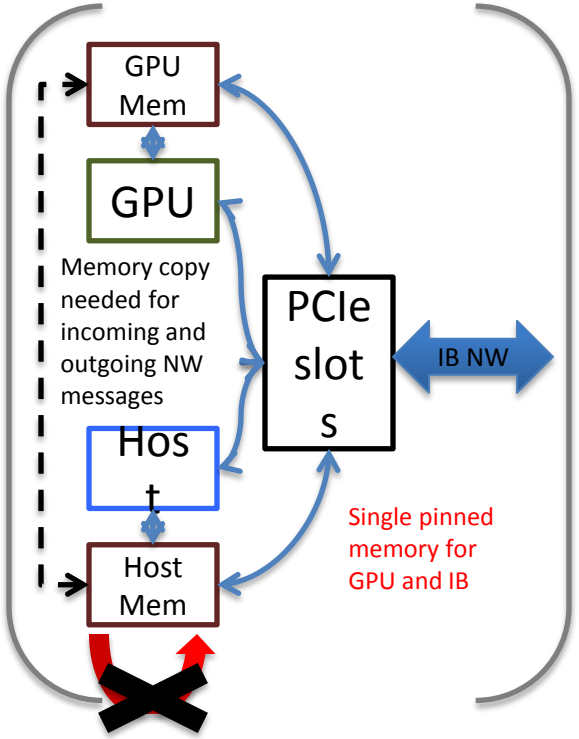
MPI and I/O from GPU vs. CPU

- Traditional CPU point-of-view
  - InfiniBand adapter is a PCIe device (typically x4 lanes)
  - GPU accelerators also PCIe devices (traditionally x16 lanes)
- Traditionally
  - CPU is workhorse for HPC FLOP workload
  - GPU not in picture
- Today
  - Top Top500 system delivers Linpack performance from GPU



GPUDirect

- To make data transfer from GPU to communication network efficient
- Eliminate internal copying and overhead by the host CPU
- Potential benefits when transferring data over IB interconnect
- Application requirement: MPI comm. of CUDA pinned memory



Outline

- Setting up GPUDirect
  - Prerequisites
  - Step-by-step guide
- Setting up benchmarks
  - CUDA example
  - OpenCL example
- Results and analysis
  - Exploiting GPUDirect with full-scale GPU accelerated apps
- Future directions

Pre-requisites

- Hardware
  - x86 platform
  - NVIDIA Tesla cards (Fermi generation)
  - QDR IB adapters
  - QDR switch

- Software
  - OS: RHEL 5 Update 4 +
  - IB software: OEFD 1.5.x with GPUDirect support
  - MPI: MVAPICH version 1.5+
  - NVIDIA Development Driver for Linux version 256.35
  - NVIDIA CUDA Toolkit 3.2+ (incl. OpenCL driver)

Reference: AMBER 11 (PMEMD) with GPUDirect white paper

Setup

- Kernel installation

  1. Install the required kernel rpm
  2. Modify the boot loader configuration
  3. Reboot the machine with new kernel

- Install IB driver (with updates, if applicable)

  1. Mount the ISO files
  2. Run the installation script (with –all option for Mellanox OFED)
  3. Restart the driver

Reference: AMBER 11 (PMEMD) with GPUDirect white paper

Setup Contd.

- NVIDIA driver and SDK installation

1. Install NVIDIA Development Driver for x86_64 platforms
2. Load NVIDIA driver to confirm installation
3. Ensure the correct version for the driver
4. Install CUDA Toolkit and SDK (version 3.2 and above)
5. Run sample codes from the SDK to ensure GPU devices are setup and accessible

Monitoring the GPUDirect Activities

- How to ensure if the activity is taking place or not
  - Small message sizes may not use RDMA => no GPUDirect
- IB core parameters to observe
  - /sys/module/ib_core/parameters/gpu_direct_enable
    - As root, set “0” for disable and “1” for enable
  - /sys/module/ib_core/parameters/gpu_direct_pages
    - Values should change after the activity
  - /sys/module/ib_core/parameters/gpu_direct_shares
    - Values should change after the activity
- Observe the parameters while running an application
  - watch -d -n5 cat /sys/module/ib_core/parameters/*

Enabling and Disabling GPUDirect

- Method 1: environment variables
  - Enable GPUDirect
    - export MV2_RNDV_PROTOCOL=RPUT
    - export MV2_USE_RDMA_ONE_SIDED=1
  - Disable GPUDirect
    - export MV2_RNDV_PROTOCOL=R3
    - export MV2_USE_RDMA_ONE_SIDED=0

- Method 2: ib_core parameters
  - Enable/Disable GPUDirect (on each node as root; 1=enable, 0=disable)
    - echo 1 > /sys/module/ib_core/parameters/gpu_direct_enable

MPI and **CUDA** Micro-benchmark

- Modified version of the OSU Bandwidth micro-benchmark with GPUDirect

```
#ifdef PINNED
    if (myid == 0) printf("Using PINNED host memory!\n");
    cudaMallocHost( (void**) &s_buf1, MYBUFSIZE);
    cudaMallocHost( (void**) &r_buf1, MYBUFSIZE);
#else
    if (myid == 0) printf("Using PAGEABLE host memory!\n");
    s_buf1 = (char*) malloc(MYBUFSIZE);
    r_buf1 = (char*) malloc(MYBUFSIZE);
#endif
```

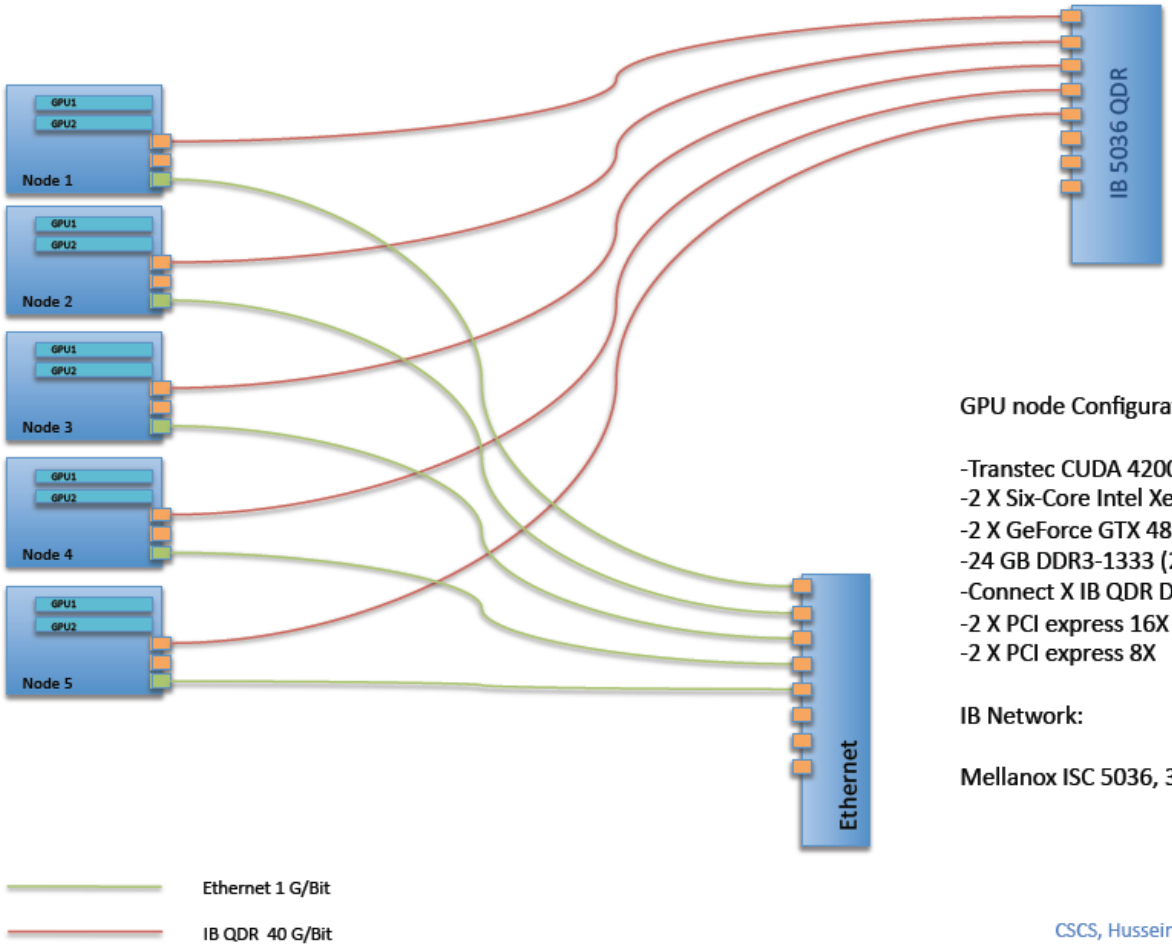
MPI and **OpenCL** Micro-benchmark

- Modified version of the OSU Bandwidth micro-benchmark with GPUDirect

```
#ifdef PINNED
   // Get platform and device information
    cl_platform_id platform_id = NULL;
    cl_device_id device_id = NULL;
    cl_uint ret_num_devices;
    cl_uint ret_num_platforms;
    cl_int ret = clGetPlatformIDs(1, &platform_id, &ret_num_platforms);
    ret = clGetDeviceIDs( platform_id, CL_DEVICE_TYPE_GPU, 1, &device_id, &ret_num_devices);

    // Create an OpenCL context
    cl_context context = clCreateContext( NULL, 1, &device_id, NULL, NULL, &ret);
    // Create a command queue
    cl_command_queue command_queue = clCreateCommandQueue(context, device_id, 0, &ret);

    // Create memory buffers on the device for each vector
    cl_mem s_mem = clCreateBuffer(context, CL_MEM_WRITE_ONLY | CL_MEM_ALLOC_HOST_PTR,,
            CL_MEM_COPY_HOST_PTR, MYBUFSIZE, NULL, &ret);
    cl_mem r_mem = clCreateBuffer(context, CL_MEM_WRITE_ONLY | CL_MEM_ALLOC_HOST_PTR,
            CL_MEM_COPY_HOST_PTR, MYBUFSIZE, NULL, &ret);

   // pinned memory (blocked call)
   s_buf1 = (char *) clEnqueueMapBuffer(command_queue, s_mem, CL_TRUE, CL_MAP_WRITE, 0,
           MYBUFSIZE, 0, NULL, NULL, &ret);
   r_buf1 = (char *) clEnqueueMapBuffer(command_queue, r_mem, CL_TRUE, CL_MAP_WRITE, 0,
           MYBUFSIZE, 0, NULL, NULL, &ret);
```

Target Platform Setup

CSCS GPU-Infiniband Test-bed Prototype Cluster



GPU node Configuration:

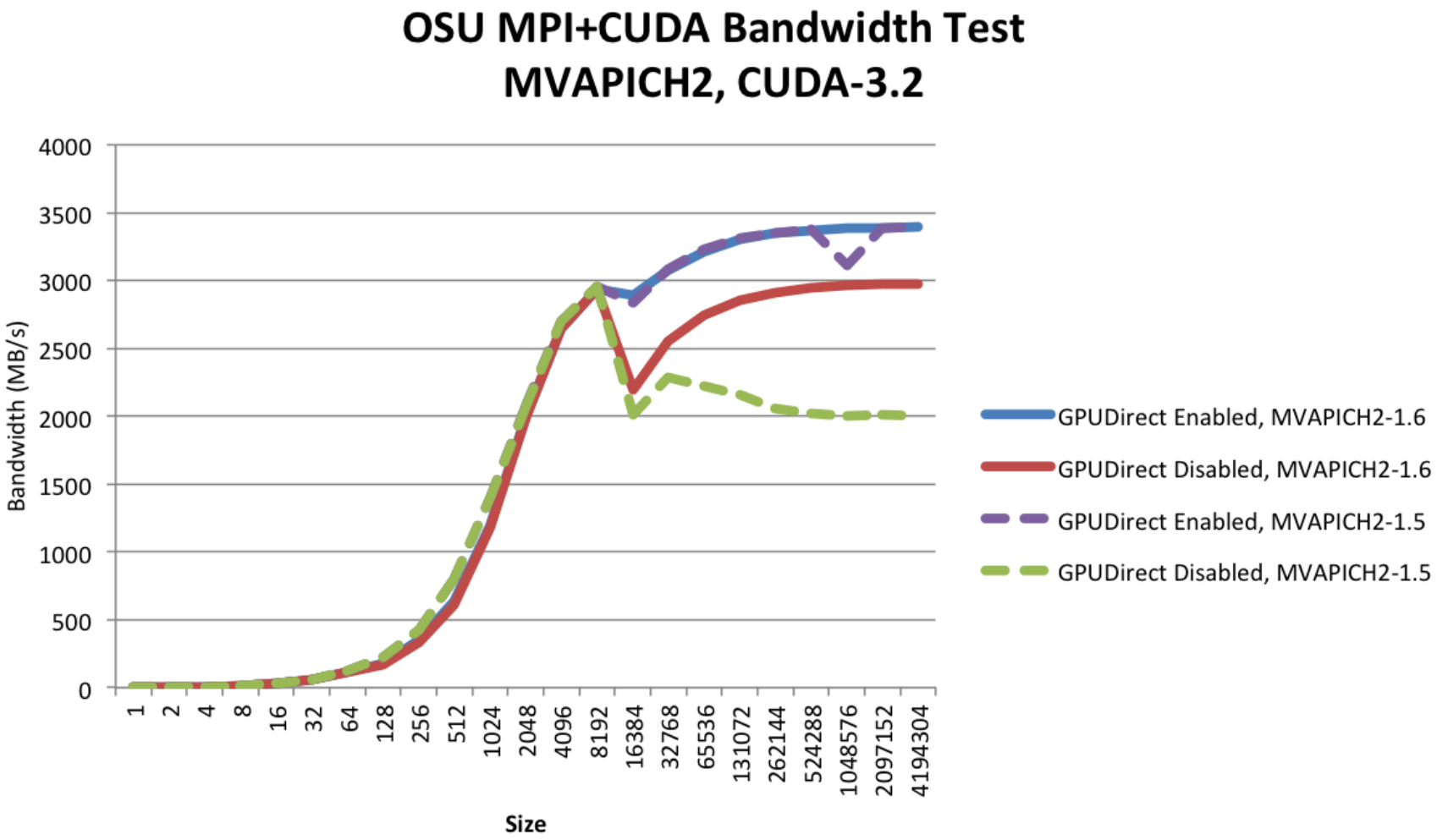
-Transtec CUDA 4200, Tower/4 U
-2 X Six-Core Intel Xeon X5670@ 2.93 GHz
-2 X GeForce GTX 480 (GF100 FERMI)
-24 GB DDR3-1333 (2GB per core)
-Connect X IB QDR Dual port 40G/bit
-2 X PCI express 16X
-2 X PCI express 8X

IB Network:

Mellanox ISC 5036, 36 Ports QDR

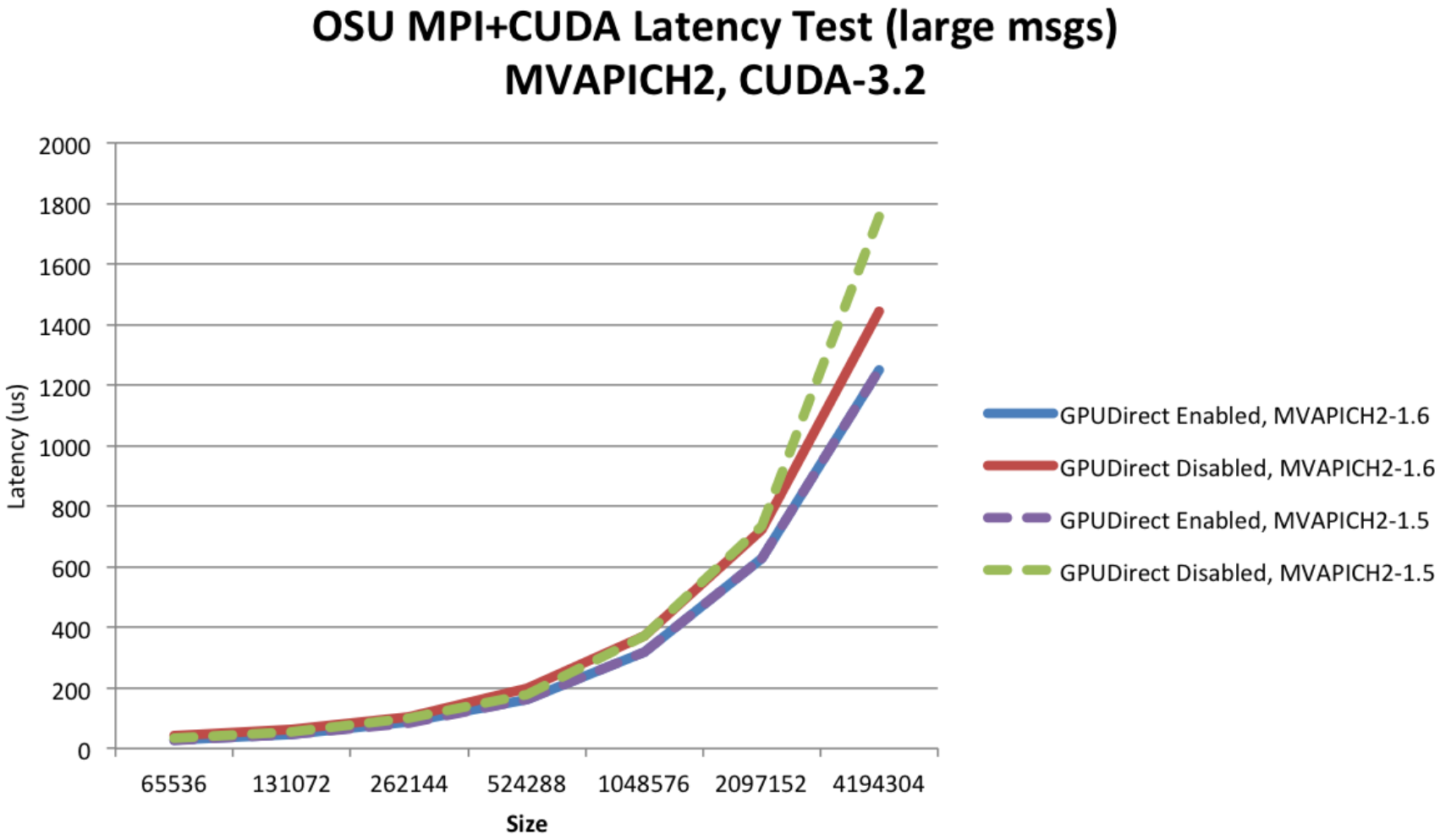
CSCS, Hussein N. Harake

Bandwidth Results



Note: PCIe transfers to/from the GPU are not included in these tests

Latency Results



Note: PCIe transfers to/from the GPU are not included in these tests

OpenCL with GPUDirect

- Identical results as CUDA pinned version
  - For performance and for IB counters

- Experiments with different memory referencing schemes did not impact performance

  - CL_MEM_WRITE_ONLY
  - CL_MEM_READ_ONLY
  - CL_MEM_READ_WRITE
  - CL_MEM_ALLOC_HOST_PTR

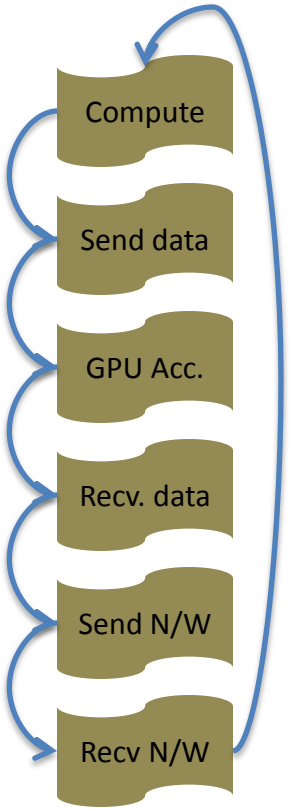
GPUDirect for HPC Applications (I)

- Benefits reported for HPC applications by vendor studies
  - Nvidia-Mellanox, GPUDirect Technology Brief, 2010
  - QLogic White Paper, Maximizing GPU Cluster Performance, 2011

- We experimented with various multi-GPU applications, for example:
  - Himeno benchmark
  - NAMD
  - AMBER11

GPUDirect for HPC Applications (II)

- Our experiments are far less successful due to possible issues
  - GPGPU accelerated applications currently do not share communication and computation data
    - Historical reasons
  - Message size constraints
    - Recall impact of GPUDirect for certain message sizes
  - System setup
    - Recall impact of environment variables
  - Experimental setup
    - Input configurations for applications



Solving Possible Pitfalls (I)

- Does the application communicate using CUDA pinned memory?
  - This is a requirement

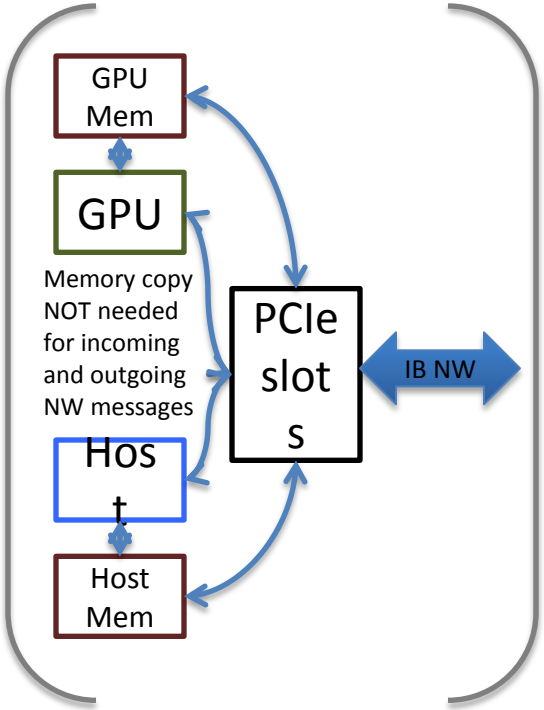
- MPI library build configuration
  - Must have RDMA communication enabled
  - Some *custom* MPI configurations may not work
  - For example, our Nemesis config of MVAPICH2 does not show GPUDirect traffic for larger applications (e.g. AMBER11)

Solving Possible Pitfalls (II)

- Message sizes need to be above the RDMA threshold
  - Find the message sizes in the application (e.g. using a performance tool)
  - Use environment variables to control the RDMA threshold; for example:
    - export MV2_VBUF_TOTAL_SIZE=10000
    - export MV2_IBA_EAGER_THRESHOLD=10000
  - Test to ensure performance does not drop

Future Directions

- Reduce further copying overhead and host involvement
  - Can CUDA 4.0 GPUDirect 2.0 be exploited?
- Possible changes in benchmarking
  - Showcase how GPUDirect can be exploited
- Possible benefits to applications
  - Currently being investigated



Summary

- GPUDirect 1.0—very first step for reducing traditional CPU<->GPU overheads for MPI communication
  - Application development process oblivious to improved features
- Should GPU always rely on CPU for access to system resources?
  - Yes—not much needs to be done… incremental changes should suffice
  - No—a lot needs to be done
- R&D in programming and runtime systems needed to guide application developers

Acknowledgements

- Gilad Shainer, Mellanox
- Pak Lui, Mellanox
- DK Panda, Ohio State University
- Sayantun Sur, Ohio State University
- Timothy Lanfear, NVIDIA