Data Transfer To Remote GPUs Over High Performance Networks

MPI CUDA METHOD DATA TRANSFER

Andrea Bocci[1] - Ali Marafi[2] - Prof.Mohammad Almulla[2]

(1) Cern, (2) Kuwait University

Experiment Goal

The main purpose of this experiment is to find the most efficient method of transferring data over a network connection from a client machine to a remote GPU in a server machine, performing a computation on the remote GPU, and transferring the results back to the client.

To evaluate the impact of using a remote GPU, we need a reference that is not affected by any network overhead. Therefore, we use a reference the performance of a machine that does the same computation on a local GPU.

Naively, the local GPU should be the faster compared to any method that uses a remote GPU.

Experiment Environment: Hardware Part

The measurements were performed on two pair of machines.

The first pair is equipped with dual Intel Xeon Gold 6130[1] "Skylake" CPUs (with PCIe 3.0), Mellanox ConnectX-5 100Gb/s network cards[2] that can operate in InfiniBand and RDMA over Converged Ethernet (RoCE) mode, and Intel 10Gb/s Ethernet network cards; one of the two machines is equipped with an NVIDIA Tesla T4 GPU[3]; the machines are connected to each other via both Intel and Mellanox cards.

The second pair is equipped with single AMD EPYC 7502P[4] "Rome" CPUs (with PCIe 4.0), Mellanox ConnectX-5 Ex 100Gb/s network cards that can operate in InfiniBand and RoCE mode, and Broadcom NetXtreme 1Gb/s Ethernet network cards; one of the two machines is equipped with an NVIDIA A10 GPU[5]; the machines are connected to each other via both Broadcom and Mellanox cards.

The Broadcom and Intel cards support only the IP protocol. The Mellanox cards have been tested using the InfiniBand, RoCE, and IP-over-InfiniBand protocols.

Experiment Environment: Software Part

To perform the measurements using both local and remote GPUs we wrote two applications: a first one for programming a local GPU, cudaTimeMeasurment, and a second for programming a remote GPU, mpiCudaGeneric.

Both programs generate two arrays of single precision floating point numbers with a size variable between 10 and 400 million elements, then transfer them to the GPU to perform a computation, and copy the resulting array back to the host memory. The GPU simply computes the pairwise sum of the two arrays; to emulate the impact of a more complex computation, the sum can be repeated an arbitrary number of times.

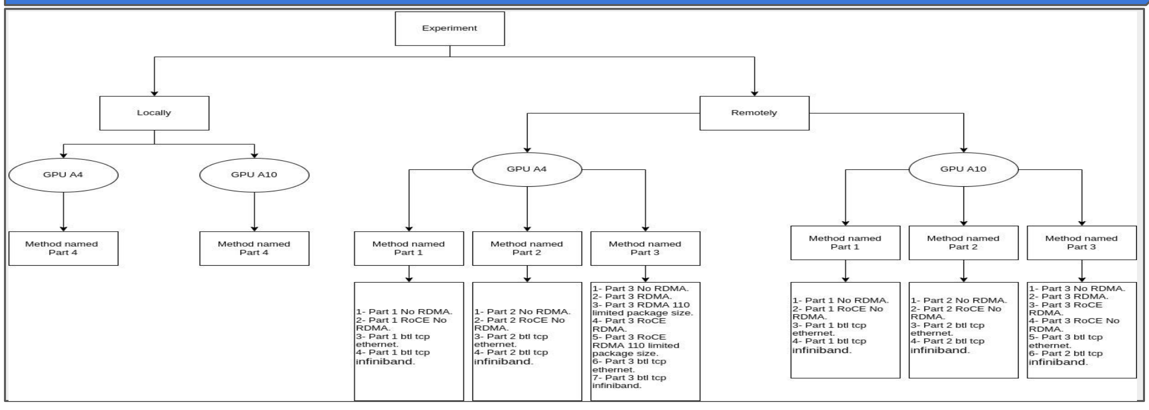
Finally, the result is compared with the result of the same operation performed on the CPU; if the results are correct, the program prints the amount of time spent in the various parts of the task: data transfers and GPU computations.

Both programs use CUDA 11.5[6] to program the GPUs.

mpiCudaGeneric uses OpenMPI 4.1[7] and optionally the Unified Communication X (UCX)[8] library for the inter-process communications.

When using the Mellanox cards in InfiniBand or RoCE mode with the UCX library, the communication with the remote GPU can take advantage of the NVIDIA GPUDirect Remote Direct Memory Access (RDMA) to transfer the data directly to or from the remote GPU memory, bypassing the machine's host memory.

Experiment FlowChart



Abbreviations

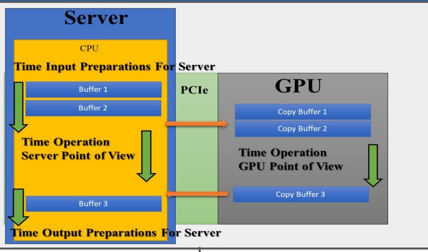
RDMA: Remote Direct Memory Access is a direct memory access from one machine to the system or GPU memory of another machine without involving either one's operating system, achieving high-throughput, low-latency data transfers. Of the cards tested in this work, only the Mellanox cards support RDMA.

RoCE: RDMA Over Converged Ethernet allows remote direct memory access over an Ethernet network, by encapsulating the InfiniBand transport packets over Ethernet. In this work we used RoCE v2, an IP-based protocol that supports routing. Of the cards tested in this work, only the Mellanox cards support RoCE.

TCP Over Infiniband: TCP/IP protocol over an InfiniBand connection, encapsulating the IP packets inside InfiniBand packets.

TCP Over Ethernet: TCP/IP protocol over an Ethernet connection, encapsulating the IP packets inside Ethernet packets

Local Experiment



The time measurements are taken in four points:

Time Input Preparations: copy data from **Pageable** Buffer to **Pinned** Buffer, then copy from Pinned Buffer to GPU Buffer.

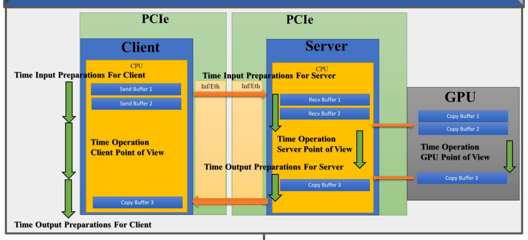
Time Operations on CPU PointView: Computations done on the GPU but time taken by CPU.

Time Operations on Device PointView: Computations done on the GPU and taken by GPU.

Time Output Preparations: Copy the results from GPU Buffer to **Pinned** Buffer, then Copy from **Pinned** to **Pageable** Buffer.

The calculation of Latency time taken for the Local transfer data is: [(Time input perpartions + Time Operations on Device + Time output preparations) - Time Operations on Server].

Remote Experiment For Part 1 & 2



For **Part 1**, the time measurements are taken in seven points:
Time Input Preparations on Client: Copy data from the Client **Pageable** Buffers to the Server **Pageable** Buffers through MPI with different Methods: RoCE with No RDMA, TCP over Ethernet, TCP over Infiniband.

Time Input Preparations on Server: Copy data from the Server **Pageable** Buffers to the GPU Buffers.

Time Operations on Client PointView: Computations done on the GPU but time is taken by the Client CPU.

Time Operations on Server PointView: Computations done on the GPU but time is taken by the Server CPU.

Time Operations on Device PointView: Computations done on the GPU and the time is taken by the GPU.

Time Output Preparations on Server: Copy data from the GPU buffer to the Server **Pageable** Buffer.

Time Output Preparations on Client: Copy data from the Server **Pageable** buffer to the Client **Pageable** Buffer through MPI with different Methods: RoCE with No RDMA, TCP over Ethernet, TCP over Infiniband.

For **Part 2**, the time measurements are taken in seven points:
Time Input Preparations on Client: Copy data from the Client **Pageable** Buffers to the Server **Pinned** Buffers through MPI with different Methods: RoCE with No RDMA, TCP over Ethernet, TCP over Infiniband.

Time Input Preparations on Server: Copy data from the Server **Pinned** Buffers to the GPU Buffers.

Time Operations on Client PointView: Computations done on the GPU but time is taken by the Client CPU.

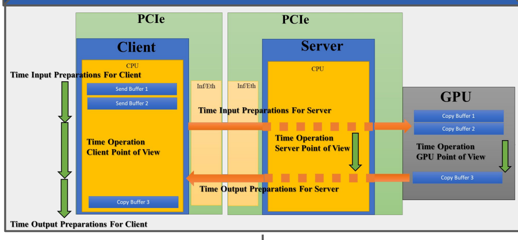
Time Operations on Server PointView: Computations done on the GPU but time is taken by Server the CPU.

Time Operations on Device PointView: Computations done on the GPU and the time is taken by the GPU.

Time Output Preparations on Server: Copy data from the GPU buffer to the Server **Pinned** Buffer.

Time Output Preparations on Client: Copy data from the Server **Pinned** buffer to the Client **Pageable** Buffer through MPI with different Methods: RoCE with No RDMA, TCP over Ethernet, TCP over Infiniband.

Remote Experiment For Part 3



For **Part 3**, the time measurements are taken in seven points:
Time Input Preparations on Client: Send data from a Client **Pageable** Buffers to a remote GPU through MPI with different Methods: RDMA, RoCE with RDMA, TCP over Ethernet, TCP over Infiniband.

Time Input Preparations on Server: Receive data from Client **Pageable** Buffers to the GPU Buffers.

Time Operations on Client PointView: Computations done on the GPU but time is taken by Client CPU.

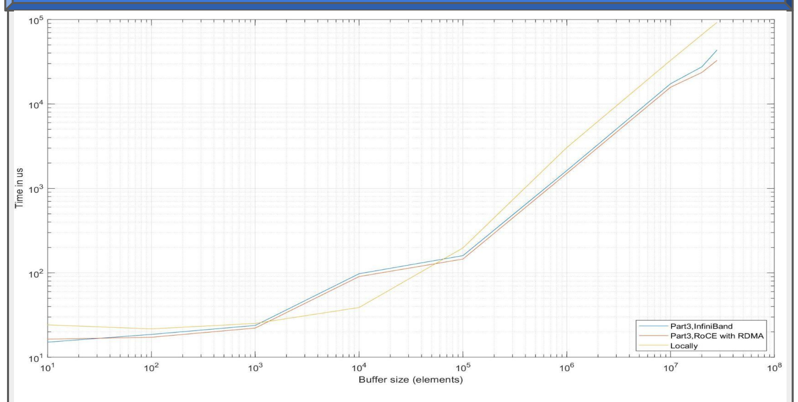
Time Operations on Server PointView: Computations done on the GPU but time is taken by the Server CPU.

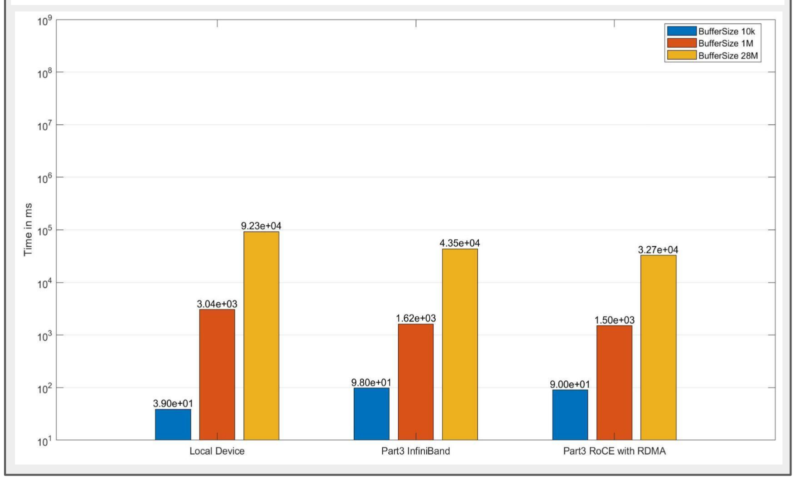
Time Operations on Device PointView: Computations done on the GPU and the time is taken by the GPU.

Time Output Preparations on Server: send data from the GPU buffer to the Client **Pageable** Buffers.

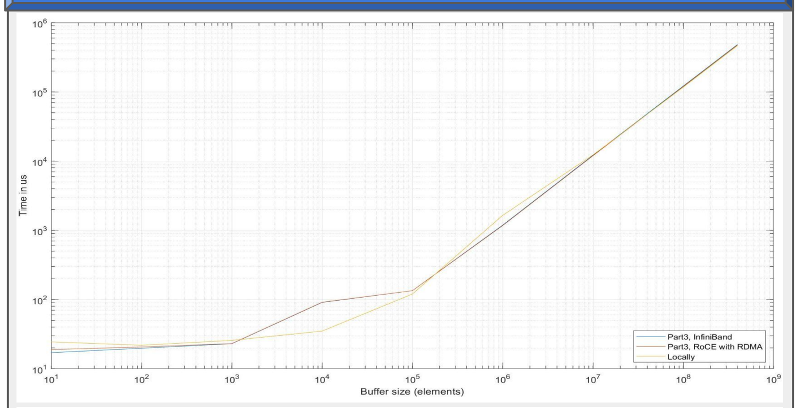
Time Output Preparations on Client: Receive data from the GPU buffer to the Client **Pageable** Buffer through MPI with different Methods: RDMA, RoCE with RDMA, TCP over Ethernet, TCP over Infiniband.

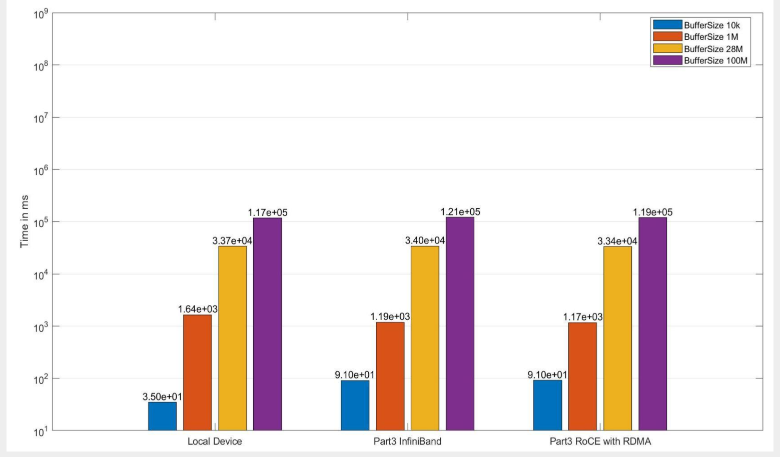
Tesla A4 GPU Results: Local, Remote over InfiniBand (with RDMA), and RoCE (with RDMA)



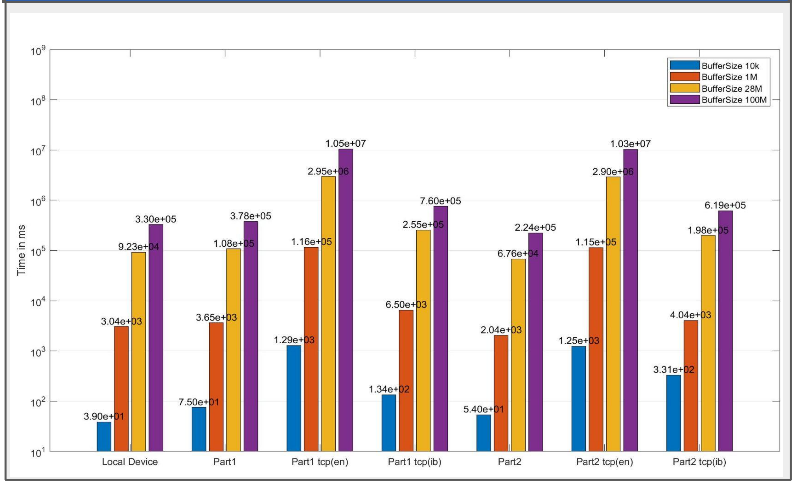


Tesla A10 GPU Results: Local, Remote over InfiniBand (with RDMA), and RoCE (with RDMA)

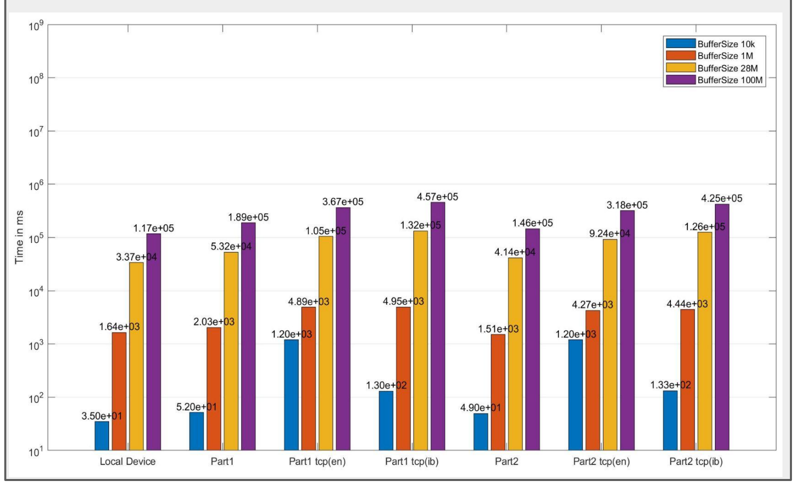




Tesla A4 GPU Results: Local, Part 1, and Part 2 (without RDMA)



Tesla A10 GPU Results: Local, Part 1, and Part 2 (without RDMA)



Conclusion

As expected, the InfiniBand cards are significantly faster than the Ethernet cards as we tested.

On the Mellanox cards:

- We did not observe a significant difference in performance between the InfiniBand and RoCE protocols;
- Even without using the RDMA capability, the native InfiniBand protocols is 2-3 times faster than the IP-over-InfiniBand protocol;
- When paired with a communication library that can leverage the RDMA capabilities, the data transfer to **remote GPUs is as fast (or even faster) than using a local GPU**.

This is a very encouraging result for extending the use of GPUs beyond those available on a local machine.

Comparing the results from the "Part 1" and "Part 2" measurements we can see that - when RDMA is not available - the use of an intermediate buffer in pinned host memory plays a vital role in speeding up the data transfers with a remote GPU.

Reference

[1]. https://ark.intel.com/content/www/us/en/ark/products/120492/intel-xeon-gold-6130-processor-22m-cache-2-10-ghz.html
[2]. https://support.mellanox.com/s/productdetails/a2v5000000052eDAAQ/connectx5-card
[3]. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-product-brief.pdf
[4]. https://www.amd.com/en/product/8796
[5]. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a10/pdf/A10-Product-Brief.pdf
[6]. https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html
[7]. https://www.open-mpi.org/
[8]. https://openucx.readthedocs.io/en/master/