S9653 - HOW TO MAKE YOUR LIFE EASIER IN THE AGE OF EXASCALE COMPUTING USING NVIDIA GPUDIRECT TECHNOLOGIES

Davide Rossetti, Elena Agostini
Tue 3/19, 2PM, Room 211A
AGENDA

• GPUDirect & Topology
  • How system topology may affect GPUDirect technologies and communication API
  • A case study

• GPUDirect RDMA:
  • Memory consistency problems when dealing with you NIC
    • Problem statement and possible solutions
  • L4T (Tegra)
    • Xavier topology insights
    • Application guideline
GPUDIRECT & SYSTEM TOPOLOGY: A CASE STUDY
THE ISING BETHE LATTICE

Overview

- A system of binary variables (i.e., variables that can assume only one out of two possible values) that interact each other.
- The variables are the vertices of a random graph. The graph is bipartite meaning that the red variables interact only with the blue ones
  - Same-type variables can run in parallel
  - Each red vertex has only 4 blue neighbors and vice versa
- The simulation performs a sort of relaxation dynamics that emulates the training of artificial neural networks (corresponding to the minimization of the loss function in a high-dimensional space).

Paper "Benchmarking multi-GPU applications on modern multi-GPU integrated systems", M. Bernaschi, E. Agostini, D.Rossetti
Submitted to "Special Issue of Concurrency and Computation, Practice and Experience 2018"
THE ISING BETHE LATTICE

Multi-GPU system

- Variables are distributed among all the GPUs in the system
- Interaction pattern, each variable may interact with any number of other GPUs
- Exchanging during each step of the simulation the single chunks of memory needed by each variable would result in a huge amount of small size messages among GPUs
- Most convenient to exchange all the red results (i.e. the entire device memory buffer) at the end of their interaction with the blue and vice versa
THE ISING BETHE LATTICE

Device buffers communication

<table>
<thead>
<tr>
<th>Technology</th>
<th>Communication API</th>
<th>Single Process</th>
<th>Multi-Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPUDirect P2P (CE)</td>
<td>cudaMemcpyPeer</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GPUDirect P2P (SM)</td>
<td>NcclAllGather</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GPUDirect RDMA</td>
<td>MVAPICH2 GDR</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- MVAPICH2 + GPUDirect RDMA support: directly exchange device memory
- NCCL 2.2: single and multi-process modes
- AllGather
Not all the GPU pairs have the same type of connection:

- GPUs 0 and 1, directly connected, 1 NVLink, BW 50 GB/sec
  - P2P with CE or NCCL (SM)
- GPUs 0 and 3, directly connected, 2 NVLinks, BW 100 GB/sec
  - P2P with CE or NCCL (SM)
- GPUs 0 and 5, not directly connected. Best connection path could be through NVLink to GPU 1 or alternatively, CPU or HCA
  - P2P with NCCL (SM)
  - IB cards with MVAPICH2-GDR or NCCL
THE ISING BETHE LATTICE

DGX-1V - speed up

Speed-up single process configurations with respect to mono-GPU configuration, grid size $2^{25}$

Speed-up multi-process configurations with respect to mono-GPU configuration, grid size $2^{25}$
THE ISING BETHE LATTICE
IBM AC922 - Power9 CPU

- Only 4 GPUs in the system
- GPU and CPU P9 connected through 3 NVLinks -> 150 GB/s
- GPU 0 is connected to:
  - GPU 1 with NVLink
  - GPU 2 and 3 through SMP bus -> effective P2P BW is 20 GB/s (experimentally)
- NVLink transactions can be tunneled over SMP bus -> GPUDirect P2P (CE) is supported across sockets
- NCCL and P2P are always applicable
  - No need to use IB cards
Due to the limited bandwidth when crossing the two POWER9 NUMA nodes, the performance does not improve when using 4 GPUs.

Similarly to DGX-1V, performance of NCCL single or multi-process are basically the same up to 4 GPUs, confirming that a single CPU thread is enough to manage 4 GPUs efficiently.

P2P CE is actually slightly slower that NCCL.

Speed-up all configurations with respect to mono-GPU configuration, grid size $2^{25}$.
GPUDIRECT RDMA & MEMORY CONSISTENCY
1. CUDA kernel is polling on some dev_flag
   - while(dev_flag == 0);
2. NIC receives and writes data into the GPU memory
3. NIC/CPU set dev_flag = 1
4. CUDA kernel observes dev_flag
5. CUDA kernel consumes received data

SM may observe inconsistent data!
GPUDIRECT RDMA

Memory consistency issue

- PCIe ordering guarantees are not preserved all the way inside the GPU
- Explicit fencing is required
- Fencing mechanisms:
  - GPU work launch (kernels, memory copies)
  - Read of GPU memory mapping exposed on GPU BAR1
    - Active CPU read
    - NIC proxied read
GPUDIRECT RDMA

Active CPU read

- CPU reads any GPU memory location
- CPU set dev_flag = 1
- The GPU memory location must be visible from the CPU
  - one way to create a CPU mapping of GPU memory is by using GDRCopy
  - https://github.com/NVIDIA/gdrcopy

Read dev_flag & Write dev_flag == 1
GPUDIRECT RDMA
NIC proxied read

Hack: loopback RDMA WRITE

- CPU observes nic_flag
- CPU issue NIC RDMA WRITE
  - Source is GPU BAR1, dev_src=1
  - Destination is GPU BAR1 of dev_flag
- NIC execute RDMA WRITE
  - Implicitly flushing
- GPU observe dev_flag=1
GPUDIRECT RDMA ON L4T
Tegra Jetson AGX Xavier is a 64-bit ARM high-performance SoC for autonomous machines introduced in 2018:

- iGPU 512-core Volta GPU with Tensor Cores
- CPU 8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
- Memory 16GB 256-Bit LPDDR4x | 137GB/s
- Storage 32GB eMMC 5.1
- PCIe x8 Gen2/3/4 slot
  - Any PCIe card can be connected. The PCIe slot is of x16 size to connect x16 card but operates in x8 mode.
- OS: Linux for Tegra (L4T)

- L4T v32.1 will have GPUDirect RDMA kernel API!
SYSTEM TOPOLOGY

Desktop vs Tegra

- BAR1 page size = 64KB
- PCIe access GPU memory via L2 cache
  - PCI read/write see the latest value from GPU
- GPU memory is separated from Sysmem
  - Allocator is cudaMalloc
- [https://docs.nvidia.com/cuda/gpudirect-rdma/index.html](https://docs.nvidia.com/cuda/gpudirect-rdma/index.html)

- Page size = 4 KB
- Sysmem only
- PCIe and iGPU L2 are not coherent
- cudaMalloc returns GPU cached memory
- Need to use uncached memory portion (cudaMallocHost)
GPUDIRECT RDMA

Desktop

Application
- cudaMalloc
- CU_POINTER_ATTRIBU
- TE_SYNC_MEMOPS
- Get physical pages addresses
- DMA map: Get dmable address for physical pages
- R/W PCIe - GPU memory
- Unmap
- Release
- cudaFree

Kernel module
- nvidia_p2p_get_pages
- nvidia_p2p_dma_map_pages
- nvidia_p2p_unmap_pages
- nvidia_p2p_put_pages

ioctl

GPU memory
- d_buffer

PCle device

L2 cache

dGPU
GPUDIRECT RDMA

L4T
GPUDIRECT RDMA ON L4T

next release

- Currently L4T public release is v31
- GPUDirect RDMA support starting from L4T v32.1 (JetPack 4.2)
- Note, `/usr/src/linux-headers-#KERNEL_VERSION-tegra/nvgpu/include/linux/nv-p2p.h`