GPUDIRECT: INTEGRATING THE GPU WITH A NETWORK INTERFACE

DAVIDE ROSSETTI, SW COMPUTE TEAM
GPUDIRECT FAMILY

- GPUDirect Shared GPU-Sysmem for inter-node copy optimization
- GPUDirect P2P for intra-node, accelerated GPU-GPU memcpy
- GPUDirect P2P for intra-node, inter-GPU LD/ST access
- GPUDirect RDMA\(^2\) for inter-node copy optimization

[\(^1\)] developer info: https://developer.nvidia.com/gpudirect
[\(^2\)] http://docs.nvidia.com/cuda/gpudirect-rdma
GPUDIRECT RDMA CAPABILITIES & LIMITATIONS

- GPUDirect RDMA
  - direct HCA access to GPU memory
- CPU still driving computing + communication
  - Fast CPU needed
  - Implications: power, latency, TCO
  - Risks: limited scaling ...
MOVING DATA AROUND

CPU prepares and queues communication tasks on HCA
HCA synchronizes with GPU tasks
HCA directly accesses GPU memory
GPU synchronizes with CPU tasks
CPU prepares and queues communication tasks on HCA

Data plane
GPUDirect RDMA

Control plane
MEET NEXT THING

GPUDirect RDMA

Data plane

GPU

GPU Direct Async

CPU prepares and queues compute and communication tasks on GPU
GPU triggers communication on HCA
HCA directly accesses GPU memory

Control plane

CPU

GPU

HCA
CPU OFF THE CRITICAL PATH

- CPU prepares work plan
  - hardly parallelizable, branch intensive
- GPU orchestrates flow
  - Runs on optimized scheduling unit
  - Same one scheduling GPU work
  - Now also scheduling network communications
KERNEL+SEND
NORMAL FLOW

a_kernel<<<...,stream>>>(buf);
cudaStreamSynchronize(stream);
ibv_post_send(buf);
while (!done) ibv_poll_cq(txcq);
b_kernel<<<...,stream>>>(buf);

100% CPU utilization
Limited scaling!
a_kernel<<<...,stream>>>(buf);

b_kernel<<<...,stream>>>(buf);

gds_stream_queue_send(stream, qp, buf);

gds_stream_wait_cq(stream, txcq);

Kernel launch latency is hidden

CPU is free
while (!done) ibv_poll_cq();
a_kernel<<<...,stream>>>(buf);
cuStreamSynchronize(stream);
gds_stream_wait_cq(stream, rx_cq);

a_kernel<<<...,stream>>>(buf);

cuStreamSynchronize(stream);
USE CASE SCENARIOS

Performance mode (~ Top500)
- enable batching
- increase performance
- CPU available, additional GFlops

Economy mode (~ Green500)
- enable GPU IRQ waiting mode
- free more CPU cycles
- Optionally slimmer CPU
PERFORMANCE MODE

40% faster

[*] modified ud_pingpong test: recv+GPU kernel+send on each side.
2 nodes: Ivy Bridge Xeon + K40 + Connect-IB + MLNX switch, 10000 iterations, message size: 128B, batch size: 20
2D STENCIL BENCHMARK

- weak scaling
- 256^2 local lattice
- 2x1, 2x2 node grids
- 1 GPU per node

[*] 4 nodes: Ivy Bridge Xeon + K40 + Connect-IB + MLNX switch
ECONOMY MODE

Round-trip latency

- **25% faster**

Latency (us)

- RDMA only: 39.28 ms
- RDMA w/IRQ: 178.62 ms
- +Async: 29.46 ms

CPU utilization

- 45% less CPU load

% load of single CPU core

- RDMA only
- RDMA w/IRQ
- +Async

*size=16384

[1] modified ud_pingpong test, HW same as in previous slide
SUMMARY

- Meet Async, next generation of GPUDirect
- GPU orchestrates network operations
- CPU off the critical path
- 40% faster, 45% less CPU load

Excited about these topics?
collaborations & jobs @NVIDIA
NVIDIA REGISTERED DEVELOPER PROGRAMS

- Everything you need to develop with NVIDIA products
- Membership is your first step in establishing a working relationship with NVIDIA Engineering
  - Exclusive access to pre-releases
  - Submit bugs and features requests
  - Stay informed about latest releases and training opportunities
  - Access to exclusive downloads
  - Exclusive activities and special offers
  - Interact with other developers in the NVIDIA Developer Forums

REGISTER FOR FREE AT: developer.nvidia.com
THANK YOU

JOIN THE CONVERSATION

#GTC15  

Twitter  Facebook  LinkedIn
### PERFORMANCE VS ECONOMY

**Performance mode**

<table>
<thead>
<tr>
<th>PowerTOP 2.3</th>
<th>Overview</th>
<th>Idle stats</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td></td>
<td></td>
<td>CPU 0</td>
</tr>
<tr>
<td>C0 polling</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C1-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C3-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C6-IVB</td>
<td>89.1%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU 1</td>
<td></td>
</tr>
<tr>
<td>C0 polling</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C1-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C3-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C6-IVB</td>
<td>98.8%</td>
<td>83.5 ms</td>
<td></td>
</tr>
</tbody>
</table>

**Economy mode**

<table>
<thead>
<tr>
<th>PowerTOP 2.3</th>
<th>Overview</th>
<th>Idle stats</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td></td>
<td></td>
<td>CPU 0</td>
</tr>
<tr>
<td>C0 polling</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C1-IVB</td>
<td>0.8%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C3-IVB</td>
<td>1.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C6-IVB</td>
<td>91.3%</td>
<td>23.2%</td>
<td>1.1 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU 1</td>
<td></td>
</tr>
<tr>
<td>C0 polling</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C1-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C3-IVB</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0 ms</td>
</tr>
<tr>
<td>C6-IVB</td>
<td>99.9%</td>
<td>126.1 ms</td>
<td></td>
</tr>
</tbody>
</table>

[*] modified ud_pingpong test, HW same as in previous slide, NUMA binding to socket0/core0, SBIOS power-saving profile