STATE OF GPUDIRECT TECHNOLOGIES

Davide Rossetti(*)
Sreeram Potluri
David Fontaine
OUTLOOK

GPUDirect overall
GPUDirect Async
SW architecture
CUDA Async APIs
GPUDIRECT FAMILY\(^1\)

- GPUDirect Shared GPU-Sysmem for optimized inter-node copy
- GPUDirect P2P for intra-node
  - accelerated GPU-GPU memcpy
  - inter-GPU direct load/store access
- GPUDirect RDMA\(^2\) for optimized inter-node communication
- GPUDirect Async for optimized inter-node communication

\(^1\) developer info: https://developer.nvidia.com/gpudirect
\(^2\) http://docs.nvidia.com/cuda/gpudirect-rdma
GPUDIRECT IN THE CAVE

CERN’s NA62 experiment “probes decays of the charged kaon”

[*] http://apegate.roma1.infn.it mediawiki/index.php/NaNet_overview
GPUDIRECT scopes

- GPUDirect P2P → data
  - GPUs both master and slave
- GPUDirect RDMA → data
  - GPU slave, 3rd party device master
- GPUDirect Async → control
  - GPU & 3rd party device master & slave

Data plane

GPUDirect RDMA/P2P

GPU

HOST

Control plane

GPUDirect Async

GPU
GPUDIRECT scopes (2)

• GPUDirect RDMA & Async
  • over PCIe, for low latency
• GPUDirect P2P
  • over PCIe
  • over NVLink (Pascal only)
GPUDIRECT RDMA ON PASCAL

peak results, optimal PCIe fabric

- **GK110**
  - RDMA read: 6 GB/s
  - RDMA write: 9 GB/s

- **P100**
  - RDMA read: 12 GB/s
  - RDMA write: 12 GB/s

**Bandwidth (GB/s)**
GPUDIRECT P2P ON PASCAL
early results, P2P thru NVLink

Open-MPI intra-node GPU-to-GPU point-to-point BW

Bandwidth (MB/s)

0  5000  10000  15000  20000
4KB  8KB  16KB  32KB  64KB  128KB  256KB  512KB  1MB  2MB  4MB

17.9GB/s
ASYNC: MOTIVATION
VISUAL PROFILE - TRADITIONAL

(Time marked for one step, Domain size/GPU - 1024, Boundary - 16, Ghost Width - 1)
VISUAL PROFILE - TRADITIONAL

(Time marked for one step, Domain size/GPU - 128, Boundary - 16, Ghost Width - 1)

CPU bounded
SW ARCHITECTURE
GPUDIRECT SW ECOSYSTEM

MVAPICH2  Open MPI

IB verbs

CUDA RT  CUDA driver

IB core
extensions[*] for RDMA

cxgb4  mlx5

nv_peer_mem

NV display driver

HCA  GPU

RDMA

user-mode
kernel-mode

applications  benchmarks

HW

proprietary  open-source  mixed

EXTENDED STACK

- MVAPICH2
- Open MPI
- libmp
- IB verbs
- libgdsync
- CUDA RT
- CUDA driver
- NV display driver
- nv_peer_mem
- extensions[*] for RDMA/Async
- Async
- HCA
- GPU
- mlx5
- cxgb4
- IB core
- IB verbs extensions for Async
- ext. for Async
- HW
- RDMA

applications
benchmarks

user-mode

kernel-mode

proprietary
open-source
mixed

GPUDIRECT ASYNC + INFINIBAND
preview release of components

• CUDA Async extensions, preview in CUDA 8.0 EA
• Peer-direct async extension, in MLNX OFED 3.x, soon
• libgdsync, on github.com/gpudirect, soon
• libmp, on github.com/gpudirect, soon
ASYNC: APIs
GPUDIRECT ASYNC

expose GPU front-end unit

CPU prepares work plan
- hardly parallelizable, branch intensive
- GPU orchestrates flow

Runs on optimized front-end unit
- Same one scheduling GPU work
- Now also scheduling network communications
STREAM MEMORY OPERATIONS

CU_STREAM_WAIT_VALUE_GEQ = 0x0,
CU_STREAM_WAIT_VALUE_EQ = 0x1,
CU_STREAM_WAIT_VALUE_AND = 0x2,
CU_STREAM_WAIT_VALUE_FLUSH = 1<<30
CUresult cuStreamWaitValue32(CUstream stream, CUdeviceptr addr, cuuint32_t value, unsigned int flags);

CU_STREAM_WRITE_VALUE_NO_MEMORY_BARRIER = 0x1
CUresult cuStreamWriteValue32(CUstream stream, CUdeviceptr addr, cuuint32_t value, unsigned int flags);

CU_STREAM_MEM_OP_WAIT_VALUE_32 = 1,
CU_STREAM_MEM_OP_WRITE_VALUE_32 = 2,
CU_STREAM_MEM_OP_FLUSH_REMOTE_WRITES = 3
CUresult cuStreamBatchMemOp(CUstream stream, unsigned int count, CUstreamBatchMemOpParams *paramArray, unsigned int flags);

guarantee memory consistency fpr RDMA
polling on 32-bit word
32-bit word write
low-overhead batched work submission
*volatile uint32_t*) h_flag = 0;
...
cuStreamWaitValue32(stream, d_flag, 1, CU_STREAM_WAIT_VALUE_EQ);
calc_kernel<<<GSZ,BSZ,0,stream>>>();
cuStreamWriteValue32(stream, d_flag, 2, 0);
...
*(volatile uint32_t*) h_flag = 1;
...
cudaStreamSynchronize(stream);
assert(*(volatile uint32_t*)h_flag== 2);
GPUDIRECT ASYNC

APIs features

- batching multiple consecutive mem ops save ~1us each op
  - use cuStreamBatchMemOp
- APIs accept device pointers
  - memory need registration (cuMemHostRegister)
  - device pointer retrieval (cuMemHostGetDevicePointer)
- 3rd party device PCIe resources (aka BARs)
  - assumed physically contiguous & uncached
  - special flag needed
GPU PEER MAPPING
accessing 3rd party device PCIe resource from GPU

```c
struct device_bar {
    void *ptr;
    CUdeviceptr d_ptr;
    size_t len;
};

void map_device_bar(device_bar *db)
{
    device_driver_get_bar(&db->ptr,&db->len);
    CUCHECK(cuMemHostRegister(db->ptr, db->len,
                                CU_MEMHOSTREGISTER_IOMEMORY));
    CUCHECK(cuMemHostGetDevicePointer(&db->d_ptr, db->ptr, 0));
}

... cuStreamWriteValue32(stream, db->d_ptr+off, 0xfaf0, 0);
```
GPU PEER MAPPING + ASYNC

cuStreamWriteValue32(stream, db->d_ptr+offset, 0xfaf0, 0);

phys_ptr+offset

3rd party device

PCle resources

PCle iface

GPU

PCle bus
2DSTENCIL PERFORMANCE

weak scaling, RDMA vs RDMA+Async

two/four nodes, IVB Xeon CPUs, K40m GPUs, Mellanox Connect-IB FDR, Mellanox FDR switch
CAVEATS

Good platform

- GPUDirect RDMA & Async
  - need correct/reliable forwarding of PCIe transactions
- GPUDirect Async
  - GPU peer mapping limited to privileged processes (CUDA 8.0 EA)
- Platform:
  - best: PCIE switch
  - limited: CPU root-complex
THANK YOU

JOIN THE CONVERSATION
#GTC16  

JOIN THE NVIDIA DEVELOPER PROGRAM AT developer.nvidia.com/join