THE GOD, THE BAD AND THE UGLY

D.Rossetti, 8/16/2016

NVIDIA

DUELING FOR GOLD

In the Southwest during the Civil War, a mysterious stranger, Joe (Clint Eastwood), and a Mexican outlaw, Tuco (Eli Wallach), form an uneasy partnership -- Joe turns in the bandit for the reward money, then rescues him just as he is being hanged. When Joe's shot at the noose goes awry during one esc... More

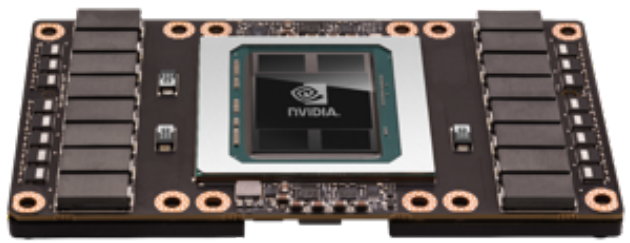
Release date: December 29, 1967 (USA)

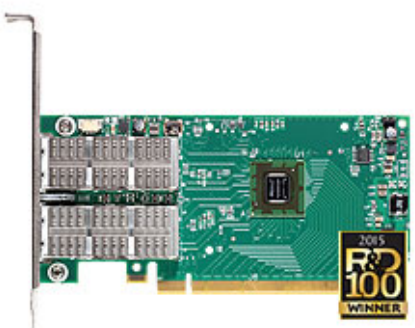
Director: Sergio Leone

Featured song: The Good, the Bad and the Ugly

Music composed by: Ennio Morricone

DUELING FOR POWER





The three chaps still around

OAK RIDGE National Laboratory

Lawrence Livermore National Laboratory

SUMMIT	SIERRA
150-300 PFLOPS Peak Performance	> 100 PFLOPS Peak Performance

IBM POWER9 CPU + NVIDIA Volta GPU + Mellanox Connect HCA

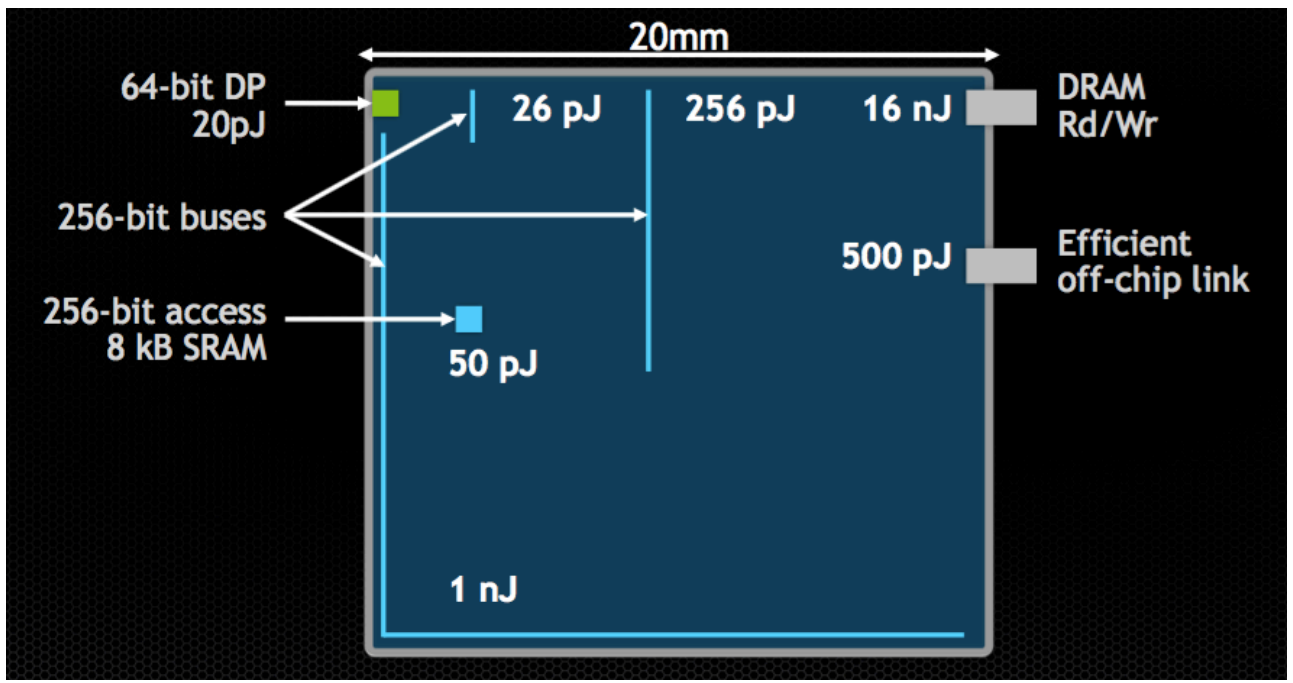
NVLink High Speed Interconnect

>40 TFLOPS per Node, >3,400 Nodes

2017

ENERGY SPENT MOVING DATA

B.Dally, 2015



power budget is fixed by thermal ~ 300W

keep data close to FP/INT ops, share data, caching, hierarchical design

integration, multi chip packages, HBM

optimize data movement, NIC on package

KEEP DATA CLOSE …

SW programming model can contribute

UVA, single address space, still needs cudaMemcpy

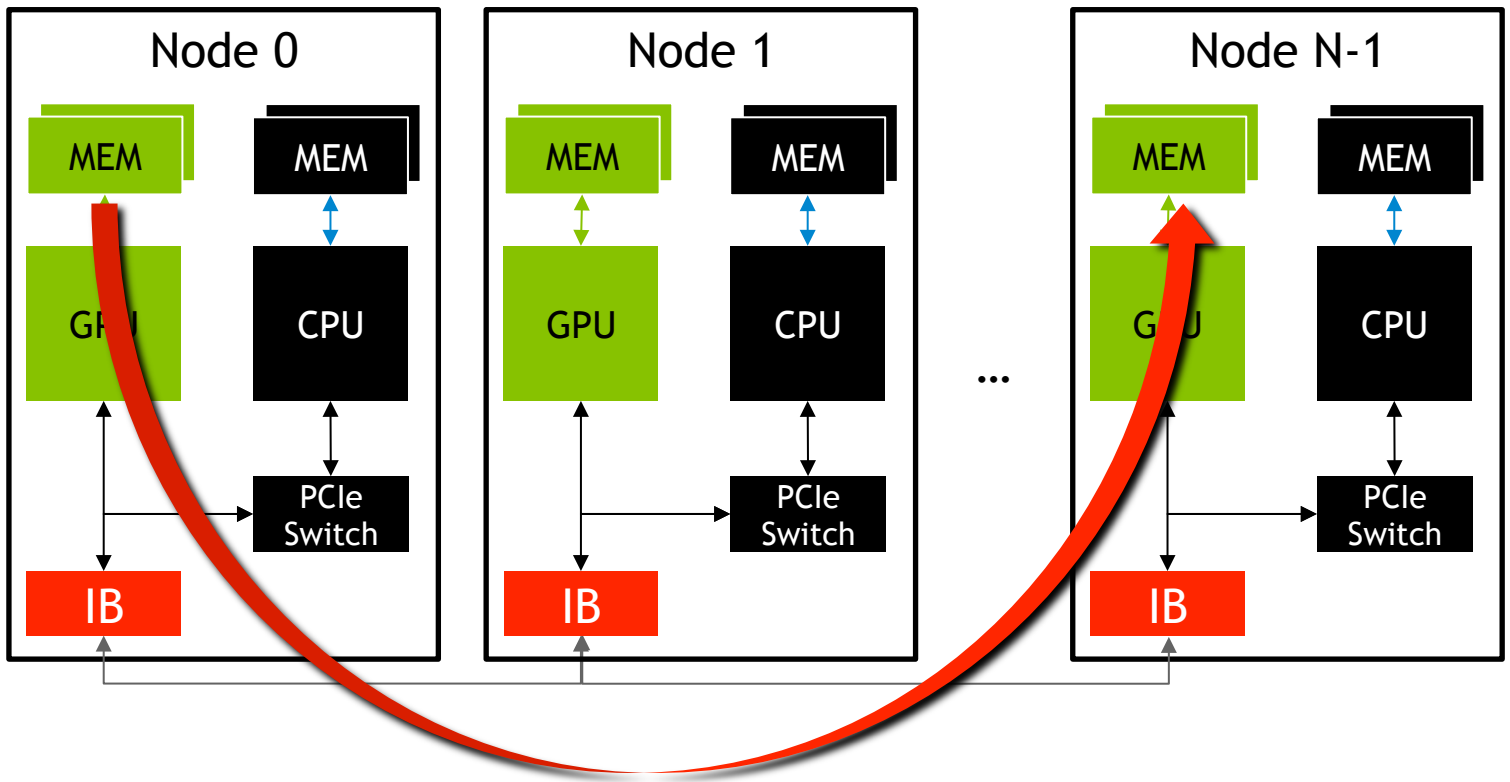
UVM *lite*, move data closer to accessor, with limitations

UVM *full*, simultaneous access, atomics

goals:

- usability
- performance, e.g. transparent data movement
- broaden design space, e.g. platform dependent optimizations, cache coherency

FOR CLUSTERS ?



GETTING RID OF THE CPU

past and current efforts

APEnet+, NaNet, NaNet-10 D.Rossetti et al.

PEACH2, T.Hanawa, T.Boku, et al.

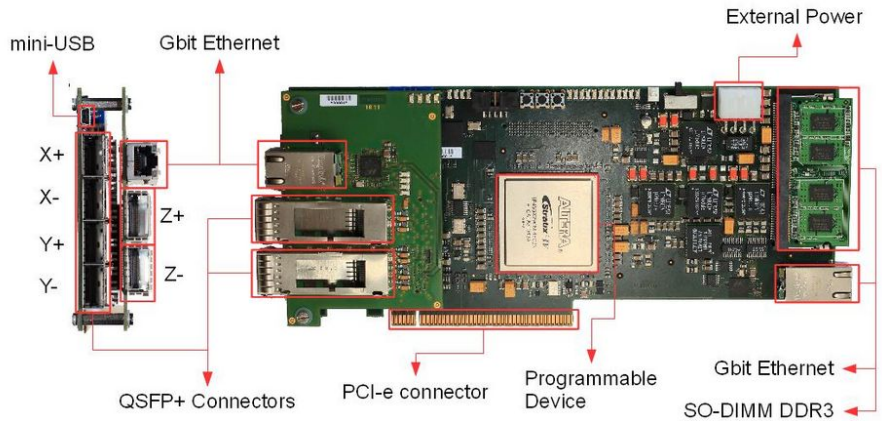
GGAS, H.Fröning, L.Oden

project Donard, S.Bates

GPUnet, GPUfs, M. Silberstein et al.

…

challenges: optimize data movement, be friendly to the SIMT model



GETTING RID OF THE CPU

a tentative road map …

A. plain comm

B. GPU-aware comm: MVAPICH2, Open MPI

C. CPU-prepared, CUDA stream triggered: Offloaded MPI + Async

D. CPU-prepared, CUDA SM triggered: in-kernel comm APIs

E. CUDA SM: PGAS/SHMEM/RMA

(C) OFFLOADED MP(I)

Async as optimization

```
while (!done) {
  mp_irecv (…, rreq)
  pack_boundary <<<…,stream1>>> (buf)
  compute_interior <<<…,stream2>>> (buf)
  mp_isend_on_stream(…, sreq, stream1)
  mp_wait_on_stream (rreq, stream1)
  unpack_boundary <<<…,stream1>>> (buf)
  compute_boundary <<<…,stream1>>> (buf)
  //synchronize between CUDA streams
}
mp_wait_all(sreq)
cudaDeviceSynchronize()
```

two implementations

- baseline, CUDA Async APIs
- HW optimized, need NIC features

helps strong scaling

(D) CUDA SM TRIGGERED COMMS

CPU Code

```
mp::mlx5::send_desc_t tx_descs[n_tx];
mp::mlx5::send_desc_t rx_descs[n_rx];
for(...) {
  mp_irecv(recv_buf, nbytes, rreq);
  mp::mlx5::get_descriptors(&rx_descs[i])
  mp_send_prepare(send_buf, nbytes, peer,
reg, sreq);
  mp::mlx5::get_descriptors(&tx_descs[i],
sreq);
}
fused_pack_unpack<<<>>>(…);
```

CUDA code

```
__global__ fused_pack_unpack(desc descs,
txbuf, rxbuf) {
  block_id = elect_block();
  if (block_id < n_pack_nthrs) {
    pack(send_buf);
    __threadfence();
    last_block = elect_block();
    if (last_block && threadIdx.x < n_tx)
      mp::device::mlx5::send(tx_descs[…]);
    __syncthreads();
  }
  else { block_id -= n_pack_threads;
    if (!block_id) {
      if (threadIdx.x < n_tx) {
        mp::device::mlx5::wait(rx_descs[…]);
      __syncthreads();
      if (threadIdx.x == 0) sched.done = 1;
    }
    while (ThreadLoad<LOAD_CG>(&sched.done));
    unpack(recv_buf);
```

(E) SHMEM like

Long running CUDA kernels

Communication within parallel region

```
__global__ void 2dstencil (u, v, sync, …)
{
    for(timestep = 0; …) {
        if (i+1 > nx) {
            v[i+1] = shmem_float_g (v[1], rightpe);
        }
        if (i-1 < 1) {
            v[i-1] = shmem_float_g (v[nx], leftpe);
        }

        u[i] = (u[i] + (v[i+1] + v[i-1] . . .

        if (i < 2) {
            shmem_int_p (sync + i, 1, peers[i]);
            shmem_quiet();
            shmem_wait_until (sync + i, EQ, 1);
        }
        //intra-kernel sync
        …
    }
}
```

PROGRAMMING MODEL

A,B,C,D: pt-to-pt, RMA for coarse-grained communications

e.g. avoid many small transfers

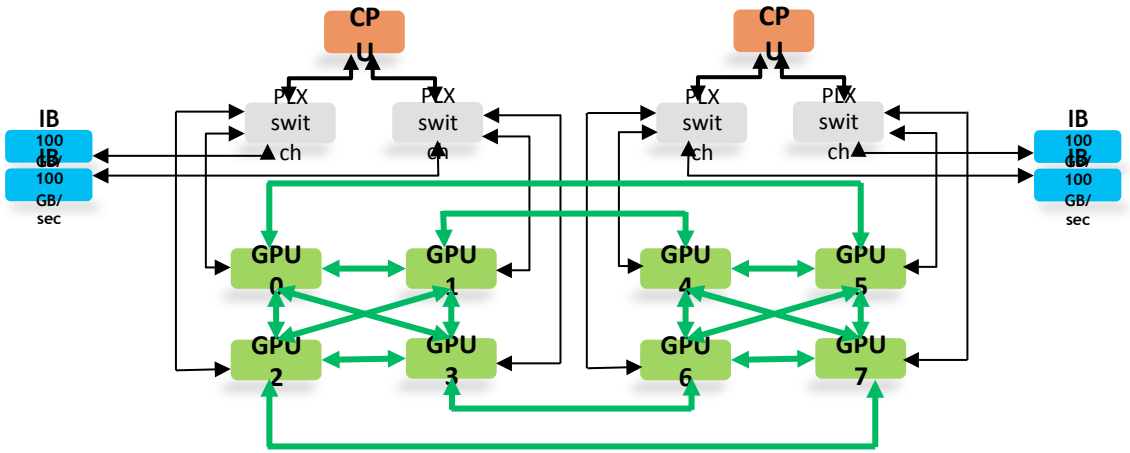
E: RMA,SHMEM for fine-grained communications

e.g. avoid many multi-word PUT/GET

Have all guarantees ? e.g. I/O vs compute

GPUs have loose consistency, SW enforced

FAT NODES



two separate networks, more power…

why not one ?

CONCLUSIONS

Three chaps compete for power

Aim: maximize Flops given technology constraints

Can we get rid of (at least) one ?

GAME OVER