What Amdahl says about GPU communication

- If you make your GPU computation infinitely fast, performance will be bound by your communication.

- GPU-2-GPU communication has:
  - Higher latency (additional hop over PCIe)
  - Lower bandwidth (limited by lowest bandwidth link)

- G2G communication cannot be an afterthought when running at scale.
How do GPUs communicate?

MPI_Send()  
GPU0  
CPU0  

MPI  
MPI_Recv()  
GPU1  
CPU1  

But what really happens here?
Until recently…

GPU0

`cudaMemcpy()`

CPU0

`MPI_Send()`

GPU1

`cudaMemcpy()`

CPU1

`MPI_Recv()`
Unified Virtual Addressing

No UVA: Multiple Memory Spaces

System Memory

0x0000
0xFFFF

GPU Memory

0x0000
0xFFFF

CPU

GPU

PCI-e

UVA: Single Address Space

System Memory

0x0000

GPU Memory

0xFFFF

CPU

GPU

PCI-e
One address space for all CPU and GPU memory
- Determine physical memory location from a pointer value
- Enable libraries to simplify their interfaces (e.g. MPI and cudaMemcpy)

Supported on Tesla starting with Fermi 64-bit applications on Linux and Windows TCC
With UVA and CUDA-aware MPI

//MPI rank 0
MPI_Send(s_buf_d,size,...);

//MPI rank n-1
MPI_Recv(r_buf_d,size,...);

No UVA and regular MPI

//MPI rank 0
cudaMemcpy(s_buf_h,s_buf_d,size,...);
MPI_Send(s_buf_h,size,...);

//MPI rank n-1
MPI_Recv(r_buf_h,size,...);
cudaMemcpy(r_buf_d,r_buf_h,size,...);

CUDA-aware MPI makes MPI+CUDA easier.
CUDA-Aware MPI Libraries May

- Use RDMA to completely remove CPU from the picture.
- Stage GPU buffers through CPU memory automatically.
- Pipeline messages
- All the programmer needs to know is they’ve passed a GPU pointer to MPI, the library developer can optimize the rest.
Cray began supporting GPU-awareness in 5.6.3
- Functions on XK7, but not optimally performing
- Expected to work very well on XC30

Must be explicitly enabled via run-time environment variable
- MPICH_RDMA_ENABLED_CUDA
- Works with both CUDA and OpenACC

Version 5.6.4 adds a pipelining feature that should help large messages
- Enabled with MPICH_G2G_PIPELINE
OMB Latency

- Host-to-Host will always have the lowest latency (fewest hops)
- Staging through host memory explicitly adds significant latency
- GPU-aware library is able to fall in the middle.

Note: 2 nodes on separate blades.
Once again, H2H wins out (probably by a difference of latency)

Direct RDMA suffers badly with this benchmark.

Setting `MPICH_G2G_PIPELINE=64` pipelines messages and opens up more concurrency.
OMB Bandwidth, Varying Pipelines

- OMB sends messages in a window of 64, so that is naturally optimal.
- Counter-intuitive that no intermediate values seemed to help.
- Additionally tried varying chunk sizes with no benefit.
Optimizing Performance of a Message

- MPI vendors know how to optimize performance for an interconnect
  - Different approaches for different message sizes
  - Multiple algorithms

- Unfortunately, on the XK7, this may not be the optimal approach.
MPI Lacks the Ability to Express Dependencies

- One way to negate the cost of G2G communication is to overlap with something computation.
- Restructuring order of computation may allow such overlapping.
- Some communication patterns have a natural concurrency that isn’t easily exploited.

Exploiting Communication Concurrency

This cannot be expressed in GPU-aware MPI today.
do $i = N, W$

  MPI_Irecv($i$)

enddo

do $i = N, W$

  packBufferAsync($i$)
  copyBufferD2HAsync($i$)

 enddo

while (more to send/recv)

  if Irecv completed
    copyBufferH2DAsync
    unpackBufferAsync()
  endif
  if D2H completed
    MPI_Isend()
  endif

done
Talks Related to this Optimization

Porting Strategy: Pack/Exchange/Unpack

- For each cycle
  - Launch edge_pack kernel for the cycle in a unique stream
  - Call a cudaEventRecord for the stream’s packing event

15. Start looking at timelines showing communication, host execution and accelerator

HOMME - Matt Norman – CUG2012

S3D - John Levesque – CUG2013
Summary

- Optimizing kernels will only take you so far as you scale, communication cannot be an afterthought.
- GPU-aware MPI libraries are becoming available
  - Easier to program
  - Can optimize performance of individual message transfers
- Some communication patterns have a natural concurrency that can be exploited to make communication “free”, but this takes additional effort.