Hybrid Redux: CUDA / MPI

CUDA / MPI Hybrid – Why?

- Harness more hardware
  - 16 CUDA GPUs > 1!

- You have a legacy MPI code that you'd like to accelerate.

CUDA / MPI – Hardware?

- In a cluster environment, a typical configuration is one of:
  - Tesla S1050 cards attached to (some) nodes.
    - 1 GPU for each node.
  - Tesla S1070 *server node* (4 GPUs) connected to two different host nodes via PCI-E.
    - 2 GPUs per node.
  - Sooner's CUDA nodes are like the latter
    - Those nodes are used when you submit a job to the queue "cuda".
- You can also attach multiple cards to a workstation.

CUDA / MPI – Approach

- CUDA will likely be:
  1. Doing most of the computational heavy lifting
  2. Dictating your algorithmic pattern
  3. Dictating your parallel layout
     - *i.e.* which nodes have how many cards

- Therefore:
  1. We want to design the CUDA portions first, and
  2. We want to do most of the compute in CUDA, and use MPI to move work and results around when needed.

Writing and Testing

- Use MPI for what CUDA can't do/does poorly
  - communication!
- Basic Organization:
  - Get data to node/CUDA card
  - Compute
  - Get result out of node/CUDA card
- Focus testing on these three spots to pinpoint bugs
- Be able to "turn off" either MPI or CUDA for testing
  - Keep a serial version of the CUDA code around just for this

Compiling

- Basically, we need to use both mpicc and nvcc
  - These are both just wrappers for other compilers

- Dirty trick – wrap mpicc with nvcc:

```
nvcc -arch sm_13 --compiler-bindir mpicc driver.c kernel.cu
```

- add -g and -G to add debug info for gdb

Running CUDA / MPI

- Run no more than one MPI process *per GPU*.
  - On sooner, this means each cuda node should be running no more than 2 MPI processes.

- On Sooner, these cards can be reserved by using:

#BSUB -R "select[cuda > 0]“

#BSUB -R "rusage[cuda=2]“

in your .bsub file.