Compiling multi-GPU MPI-CUDA code on Casper

Follow the example below to build and run a multi-GPU, MPI/CUDA application on the Casper cluster. The example uses the **nvcc** NVIDIA CUDA compiler to compile a C code. (Use the **nvfortran** compiler via the **nvhpc** module if your code is CUDA Fortran.)

Any libraries you build to support an application should be built with the same compiler, compiler version, and compatible flags that were used to compile the other parts of the application, including the main executable(s). Also, when you run the applications, be sure you have loaded the same module/version environment in which you created the applications. This will avoid job failures that can result from missing mpi launchers and library routines.

Copy sample files

Log in to either Casper or Cheyenne, then copy the sample files from this directory to your own GLADE file space:

/glade/u/home/csgteam/Examples/mpi_cuda_hello

Start interactive job

Run **execcasper** to start an interactive job on a GPU-accelerated Casper node and request 2 GPUs and 2 MPI tasks for this example. Multi-GPU jobs require v100 GPUs and are assigned by default if gpu_type is not user defined.

```
execcasper -l select=1:ngpus=2:mpiprocs=2
```

Compile code

Load a module environment for compiling MPI CUDA programs. The example below uses the NVIDIA nvhpc compilers, CUDA, and OpenMPI, along with the ncarcompilers module which simplifies linking CUDA. It is possible to use a different compiler than nvhpc (e.g. Intel) although loading the CUDA module is required.

```
module purge
module load nvhpc openmpi cuda ncarcompilers
```

Use the NVIDIA compiler (**nvcc**) to compile portions of your code that contain CUDA calls. (As an alternative to doing each of the following compiling and linking steps separately, you can run **make** to automate those steps. The necessary makefile is included with the sample files.)

```
nvcc -c gpu_driver.cu
nvcc -c hello.cu
```

Compile any portions of the code containing MPI calls.

```
mpicc -c main.c
```

Link the object files.

```
mpicxx -o hello gpu_driver.o hello.o main.o
OR
mpicc -o hello gpu_driver.o hello.o main.o -lstdc++
```

Launch executable

Launch the executable with **mpirun**.

```
mpirun -n 2 ./hello
```

Sample output:

```
[task 1] Contents of data before kernel call: HdjikhjcZ
Using 2 MPI Tasks
[task 0] Contents of data before kernel call: HdjikhjcZ
there are 2 gpus on host casper08
[task 0] is using gpu 0 on host casper08
there are 2 gpus on host casper08
[task 1] is using gpu 1 on host casper08
[task 0] Contents of data after kernel call: Hello World!
[task 1] Contents of data after kernel call: Hello World!
```