Using NVIDIA MPS in Casper GPU jobs

Some workflows benefit from processing more than one CUDA kernel on a GPU concurrently, as a single kernel is not sufficient to keep the GPU fully utilized. NVIDIA's Multi-Process Service (MPS) enables this capability on modern NVIDIA GPUs like the V100s on Casper.

Consider using MPS when you are requesting more MPI tasks than physical GPUs. Particularly for jobs with large problem sizes, using multiple MPI tasks with MPS active can sometimes offer a performance boost over using a single task per GPU.

The PBS job scheduler provides MPS support via a chunk-level resource. When you request MPS, PBS will perform the following steps on each specified chunk:

1. Launch the MPS control daemon on each job node.
2. Start the MPS server on each node.
3. Run your GPU application.
4. Terminate the MPS server and daemon.

To enable MPS on job hosts, add **mps=1** to your select statement chunks as follows:

```
#PBS -l select=1:ncpus=8:mpiprocs=8:mem=60GB:ngpus=1:mps=1
```

On each V100 GPU, you may use MPI to launch up to 48 CUDA contexts (GPU kernels launched by MPI tasks) when using MPS. MPS can be used with OpenACC and OpenMP offload codes as well, as the compiler generates CUDA code from your directives at compile time.

Jobs may not request MPS activation on nodes with GP100 GPUs.

Batch job script example

Running an MPI-enabled GPU application on Casper

In this example, we run a CUDA Fortran program that also uses MPI. The application was compiled using the NVIDIA HPC SDK compilers, the CUDA toolkit, and Open MPI. We request all GPUs on each node and use NVIDIA MPS to use multiple MPI tasks on CPU nodes for each GPU.

```
#!/bin/bash
#PBS -A project_code
#PBS -N gpu_mps_job
#PBS -q casper@casper-pbs
#PBS -l walltime=01:00:00
#PBS -l select=2:ncpus=36:mpiprocs=36:ngpus=4:mem=300GB:mps=1
#PBS -l gpu_type=v100

# Use scratch for temporary files to avoid space limits in /tmp
export TMPDIR=/glade/scratch/$USER/temp
mkdir -p $TMPDIR

# Load modules to match compile-time environment
module purge
module load ncarenv nvhpc/22.5 cuda/11.4 openmpi/4.1.4

# Run application using Open MPI
mpirun ./executable_name
```