How to request GPUs

Our cluster includes four nodes (*pirineusgpu1-4*) with Tesla P100 GPUs. Each node comprises two 24-core CPU sockets (identical to those found on standard nodes) and two GPUs.

Access to the GPU partition

Although by default new users can't submit jobs to those nodes, any regular academic group can request access to them. Please notify us that you intend to use them so we can manually grant you access to the queue.

Submitting jobs to the GPU partition

Once your group is authorised, you can submit jobs to these nodes through partition ***gpu:***

Inside the batch script:

```
#SBATCH -p gpu
```

At submit time:

```
sbatch -p gpu batch_script.slm
```

CPU time on the ***gpu*** partition is charged at the same rate as on the ***std*** partition (1 hour/CPU = 1 UC), *however* jobs sent to that partition are restricted to whole sockets of 24 CPUs, so you can only run jobs on multiples of 24 cores. If you request any other number of cores, you will actually be allotted the minimum number of 24-core sockets to satisfy your request (so, for instance, if you request 1 core you will be granted 24, if you request 32 you will be given 48, and so forth).

Note that requesting CPUs on the ***gpu*** partition is not enough to be given access to the GPUs. GPUs are a separate resource and have to be spoken for on their own.

Requesting GPUs

The standard syntax to request GPUs is:

Inside the batch script:

```
#SBATCH --gres=gpu:n
```

At submit time:

```
sbatch --gres=gpu:n batch_script.slm
```

where ***n*** is the number of GPUs that are being requested.

Inside the batch script:

```
#SBATCH --gpus-per-node=n
```

At submit time:

```
sbatch --gpus-per-node=n batch_script.slm
```

Each GPU is always accompanied by a 24-core socket, so if you specify the number of GPUs but omit the CPUs, your job is automatically assigned 24 cores per GPU. This is done to ensure that each GPU is working with the socket it is attached to for optimal data locality and performance. Notice, however, that this equivalence is unidirectional - a job requesting a GPU is automatically assigned its socket, but requesting CPUs in the ***gpu*** partition doesn't automatically assign GPUs - GPUs have to be explicitly requested.

An alternative way is to request GPUs per node:where ***n*** is the number of GPUs requested on each node. Since our nodes are equipped with two P-100 GPUs each, this value is restricted to 1-2. Note that this syntax requires that the number of nodes is specified, either directly (***-N***) or through the number of cores. As before, the entire 24-core socket attached to the GPU will be included in the job.

GPU usage is not charged separately - it is included in the rate for the accompanying CPUs. Therefore, each job run on the gpu partition will be charged in UCs as the job runtime in hours times the number of CPUs (in whole 24-core socket sets).

Examples

This will give you 1 GPU and 24 cores:

```
#SBATCH -p gpu
#SBATCH --gres=gpu:1
```

(–gres=gpu:1 requests 1 GPU; the accompanying socket is automatically assigned)

This will give you a 24-core socket, but no GPUs:

```
#SBATCH -p gpu
#SBATCH -n 1
```

(1 core is requested but they are assigned as 24-core sockets; since no GPU resource is requested, none is granted)

This will give you 48 cores (two sockets), but only 1 GPU:

```
#SBATCH -p gpu
#SBATCH -n 48
#SBATCH --gres=gpu:1
```

(48 cores are requested, which are satisfied by allocating two sockets, but only GPU is requested, so we won't have access to the second GPU; remember that while requesting GPUs automatically implies cores, cores don't imply GPUs)

This will give you 24 cores and 1 GPU:

```
#SBATCH -p gpu
#SBATCH -n 24
#SBATCH --gpus-per-node=1
```

(we request 24 cores, i.e. one socket; since that socket is in a single node, requesting also 1 GPU per node results in 1 GPU being assigned)

This will give you 2 GPUs and 48 cores in the same node:

```
#SBATCH -p gpu
#SBATCH -N 1
#SBATCH --gres=gpu:2
```

(we request 2 GPUs, so we will automatically get two sockets; since we specify a single node with *-N 1*, we will only be assigned two free sockets and GPUs if they are in the same node)

This will give you 2 GPUs and 48 cores, in two different nodes:

```
#SBATCH -p gpu
#SBATCH -N 2
#SBATCH --gpus-per-node=1
```

(we're given two nodes with 1 GPU each, and therefore also the corresponding 24-core socket)

Compiling with CUDA

A number of versions of the CUDA toolkit are available if you need to compile your own code with CUDA support.

You can see a list of the available versions with:

```
module av tools/cuda
```

To use any of them, simply load the corresponding module in addition to that of your compiler of choice. For example:

```
module load tools/gcc/8.1.0 tools/cuda/9.2
```

Related articles

- *How to run GUI applications using GPUs and VNC*
- *Benchmarks 2022*
- *How to request GPUs*
- *How to request memory*
- *How to run Gromacs*