

# Requesting and using GPUs

## GPU Nodes

Both Curnagl and Urblauna have nodes with GPUs - on Curnagl these are in a separate partition.

### Curnagl

Currently there are 7 nodes each with 2 NVIDIA A100 GPUs. One additional node is in the interactive partition

### Urblauna

Currently there are 2 nodes each with 2 NVIDIA A100 GPUs. The GPUs are partitioned into 2 GPUs with 20GB of memory so it appears that each node had 4 distinct GPUs. These GPUs are also available interactively.

## Requesting GPUs

In order to access the GPUs they need to be requested via SLURM as one does for other resources such as CPUs and memory.

The flag required is `--gres=gpu:1` for 1 GPU per node and `--gres=gpu:2` for 2 GPUs per node.

An example job script is as follows:

```
#!/bin/bash -l

#SBATCH --cpus-per-task 12
#SBATCH --mem 64G
#SBATCH --time 12:00:00

# GPU partition request only for Curnagl
#SBATCH --partition gpu

#SBATCH --gres gpu:1
#SBATCH --gres-flags enforce-binding

# Set up my modules
```

```
module purge
module load my list of modules
module load cuda

# Check that the GPU is visible

nvidia-smi

# Run my GPU enable python code

python mygpucode.py
```

If the `#SBATCH --gres gpu:1` is omitted then no GPUs will be visible even if they are present on the compute node.

If you request one GPU it will always be seen as device 0.

The `#SBATCH --gres-flags enforce-binding` option ensures that the CPUs allocated will be on the same PCI bus as the GPU(s) which greatly improves the memory bandwidth. This may mean that you have to wait longer for resources to be allocated but it is strongly recommended.

## Partitions

The `#SBATCH --partition` can take different options depending on whether you are on Curnagl or on Urblauna.

Curnagl:

- `cpu` (default)
- `gpu`
- `interactive`

Urblauna:

- `urblauna` (default)
- `interactive`

## Using CUDA

In order to use the CUDA toolkit there is a module available

```
module load cuda
```

This loads the nvcc compiler and CUDA libraries. There is also a cudnn module for the DNN tools/libraries

# Containers and GPUs

Singularity containers can make use of GPUs but in order to make them visible to the container environment an extra flag "--nv" must be passed to Singularity

```
module load singularity
```

```
singularity run --nv mycontainer.sif
```

The full documentation is at <https://sylabs.io/guides/3.5/user-guide/gpu.html>

---

Révision #12

Créé 3 février 2020 12:11:48 par Ewan Roche

Mis à jour 7 juin 2024 08:19:33 par Cristian Ruiz