GPUs at TeideHPC

GPU is an acronym for *Graphics Processing Unit* and represents precisely the heart of a graphics card just like the CPU does in a PC. Apart from the heart, it is also your brain, since it is in charge of carrying out all the complex calculations that allow some programs to run much faster than on a CPU.

Among the main uses of GPUs are the following:

- Video edition
- 3D graphics rendering
- Automatic learning
- Scientific applications
- etc...

The TeideHPC cluster offers 2 different GPU models to use with your jobs. We recommend taking a look at the cluster description to get an idea of what it looks like.

GPUs models available

These are the GPUs currently available at TeideHPC:

GPU model	**# of nodes**	**# of GPUs/ node**	**Slurm type specifier**	**CPU cores/ node**	**CPU memory/ node**	**Compute Capability (*)**	**GPU mem (GiB)**
Nvidia A100	16	4	a100	64	256GB	80	40 GB
Nvidia A100	1	8	a100	64	512GB	80	40 GB
Nvidia Tesla T4	4	1	t4	32	256GB	75	16 GB

(*) *Compute Capability* is a technical term created by NVIDIA as a compact way to describe what hardware functions are available on some models of GPU and not

on others. It is not a measure of performance and is relevant only if you are compiling your own GPU programs. See the page on CUDA programming for more.

GPU computing at TeideHPC

The TeideHPC cluster has a number of nodes that have NVIDIA general purpose graphics processing units (GPGPU) attached to them. It is possible to use CUDA tools to run computational work on them and in some use cases see very significant speedups.

As we explained in the how to run jobs section we can use 3 differents ways for sending a job to the job queue: using an interactive session, launching the application in real time or by means of an execution script.

How to request GPU resources

- **To request GPU or GPU nodes you must first understand how your cluster is configured**, for this reason we recommend that you refer to the home page where we have a cluster description that may help you.
- Basically there are **2 architectures: *icelake* (GPU nodes) and *sandy* (CPU nodes)**.
- Based on these architectures, the resources within the cluster are requested.

The way to request a GPU node is to use the dedicated `gpu` partition. In order to use this partition, access must be requested.

In addition to specifying the partition, the SLURM ***gres*** must be used.

Generic Resource *GRES*

In Slurm, **GRES stands for Generic Resource**. GRES is a feature that allows you to specify and manage various types of generic resources such as GPUs (Graphics Processing Units) within a computing cluster.

Slurm's GRES functionality enables efficient allocation, scheduling, and tracking of these resources for jobs submitted to the cluster. It helps ensure that the requested resources are available and properly utilized by the jobs that require them.

To use GRES effectively, you need to understand how your cluster is configured, how available GRES types and quantities are defined and specify GRES requirements when submitting jobs.

To get type of GRES defined in the cluster you can use:

```
$ scontrol show config | grep Gres
GresTypes = gpu
```

To find out list of *GRES* at TeideHPC look at the column *GRES* after execute this command.

```
$ sinfo -o "%40N  %10c  %10m  %35f  %30G"

NODELIST                                 CPUS       MEMORY     AVAIL_FEATURES
GRES
node18109-1                              64         257214     ilk,gpu,a100                   gpu:a100:8
node2204-[3-4]                           20         31906      ivy                            (null)
node17109-1,node17110-1,node18110-1,node 64         257214     ilk,viz,t4
gpu:t4:1
node0303-2,node0304-[1-4],node1301-[1-4] 16         30000+     sandy
(null)
node17102-1                              64         257214     ilk,gpu,a100,3g.20gb,2g.10gb,1g.5gb
gpu:3g.20gb:1(S:0),gpu:2g.10gb
node17101-1,node17103-1,node17104-1,node 64         257214
ilk,gpu,a100                     gpu:a100:4(S:0-1)
```

Alternatively, it can also be viewed by listing and filtering the nodes:

```
$ scontrol show nodes | egrep "NodeName|gres"
....
NodeName=node1315-4 Arch=x86_64 CoresPerSocket=8
NodeName=node2204-3 Arch=x86_64 CoresPerSocket=10
...
NodeName=node17101-1 Arch=x86_64 CoresPerSocket=32
  CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
NodeName=node17102-1 Arch=x86_64 CoresPerSocket=32
  CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=7,gres/gpu:1g.5gb=2,gres/gpu:
2g.10gb=1,gres/gpu:3g.20gb=1,gres/gpu:a100=3
NodeName=node17103-1 Arch=x86_64 CoresPerSocket=32
  CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
NodeName=node17104-1 Arch=x86_64 CoresPerSocket=32
  CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
...
```

As you can see, each node in cluster may have a different definition. For example this node has 4 GPUs NVidia A100.

```
NodeName=node17104-1 Arch=x86_64 CoresPerSocket=32
  CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
```

GRES usage example.

- For a GPU node (icelake) with one GPU (Nvidia A100).

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --gres=gpu:a100=1
```

Tip

Remember that to use GPUs you have to use the gpu partition, for which you have to have requested access.

To request a single GPU on slurm just add `#SBATCH --gres=gpu` to your submission script and it will give you access to a GPU. To request multiple GPUs add `#SBATCH --gres=gpu:n` where '*n*' is the number of GPUs.

So if you want 1 CPU and 2 GPUs from our general use GPU nodes in the 'gpu' partition, you would specify:

```
#SBATCH -p gpu
#SBATCH -n 1
#SBATCH --gres=gpu:a100:2
```

If you prefer to use *interactive session* you can use:

```
salloc -p gpu --mem 8000 --gres=gpu:a100:1
```

While on GPU node, you can run `nvidia-smi` to get information about the assigned GPU's.

Job script example for GPU job

- Full GPU Nvidia A100

```
#!/bin/bash

#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:a100:1
#SBATCH --mem=8G
#SBATCH --time=1:00:00

module purge
module load CUDA/12.0.0

nvidia-smi
sleep 20
```

```
sbatch 01_gpu_basic_a100.sbatch
```

More examples

Visit our repository in github https://github.com/hpciter/user_codes