GPUs en TeideHPC

GPU es el acrónimo de *Graphics Processing Unit* y representa precisamente el corazón de una tarjeta gráfica al igual que la CPU lo hace en un PC. Aparte del corazón, también es su cerebro, ya que es la encargada de realizar todos los cálculos complejos que permiten que algunos programas pueden ejecutare mucho más rápido que en una CPU.

Entre las principales usos de las GPUs están los siguientes:

- Edición de vídeo
- Renderización de gráficos 3D
- Aprendizaje automático
- Aplicaciones científicas
- etc...

El clúster TeideHPC ofrece 2 modelos diferentes de GPU para usar con sus trabajos. Recomendamos echar un vistazo a la descripción del clúster para tener una idea de cómo se ve.

Modelos de GPU disponibles

Estas son las GPU disponibles actualmente en TeideHPC:

modelos	**# nodos**	**# GPU/ nodo**	**tipo slurm**	**Cores CPU/ nodo**	**Memoria CPU/ nodo**	**Capacidad de cómputo (*)**	**Memoria GPU (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

(*) *Capacidad de cómputo o Compute Capability* es un término técnico creado por NVIDIA como una forma compacta de describir qué funciones de hardware están disponibles en algunos modelos de GPU y no en otros. No es una medida de rendimiento y solo es relevante si está compilando sus propios programas de GPU. Consulte la página sobre programación CUDA para obtener más información.

Computación GPU en TeideHPC

El clúster TeideHPC tiene una serie de nodos que tienen conectadas unidades de procesamiento de gráficos de propósito general (GPGPU) de NVIDIA. Es posible usar herramientas CUDA para ejecutar trabajo computacional en ellas y, en algunos casos de uso, ver aceleraciones muy significativas.

Como explicamos en la sección cómo ejecutar trabajos podemos utilizar 3 formas diferentes de enviar un trabajo a la cola de trabajos: utilizando una sesión interactiva, iniciando la aplicación en tiempo real o medio de un script de ejecución.

Cómo solicitar recursos de GPU

- **Para solicitar nodos de GPU o GPU primero debe comprender cómo está configurado su clúster así como las particiones**, por este motivo, le recomendamos que consulte la página inicial donde tenemos una descripción del clúster que le puede ayudar.
- Básicamente hay **2 arquitecturas: icelake (nodos de GPU) y sandy (nodos de CPU)**.
- En base a estas arquitecturas se sólicitan los recursos dentro del cluster.

La forma de solicitar un nodo de GPU es utilizando la partición dedicada de `gpu`. Para poder utilizar esta partición hay que solicitar acceso. Además de especificar la partición, hay que utilizar las ***gres*** de SLURM

Recursos genéricos *GRES* (Generic Resource)

En Slurm, **GRES significa Recurso Genérico**. GRES es una función que le permite especificar y administrar varios tipos de recursos genéricos, como GPU.

La funcionalidad GRES de Slurm permite la asignación, la programación y el seguimiento eficientes de estos recursos para los trabajos enviados. Ayuda a garantizar que los recursos solicitados estén disponibles y sean utilizados correctamente por los trabajos que los requieren.

Para usar GRES de manera efectiva, debe comprender cómo está configurado su clúster, cómo se definen los tipos y cantidades de GRES disponibles y especificar los requisitos de GRES al enviar trabajos.

Para obtener los tipos de GRES definidos en el clúster, puede usar:

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

Para conocer la lista de *GRES* en TeideHPC mira la columna *GRES* después de ejecutar este comando.

```
$ 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)
node17101-1,node17102-1,node17103-1,node17104-1,  64         257214     ilk,gpu,a100                        gpu:a100:4(S:0-1)
```

Alternativamente, también se puede ver enumerando y filtrando el listado de nodos:

```
$ 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=4,gres/gpu:a100=4
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
...
```

Como se puede observar, actualmente cada nodo de GPU dispone de 4 tarjetas A100 de Nvidia.

Ejemplos de uso de *GRES*

- Para un nodo de GPU (icelake) con una GPU (Nvidia A100).

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

Tip

Recuerde que para utilizar GPUs tiene que utilizar la partición gpu, para la cual tiene que haber solicitado acceso.

Para solicitar una sola GPU en slurm simplemente agregue la directiva `#SBATCH --gres=gpu:<model>` a su script de envío de trabajo y le dará acceso a una GPU. Para solicitar varias GPU, agregue `#SBATCH --gres=gpu:<modelo>:n`, donde '*n*' es la cantidad de GPU.

Entonces, si desea 1 CPU y 2 GPU de nuestros nodos GPU de uso general en la partición 'gpu', debe especificar:

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

Si prefiere usar una sesión interactiva, puede usar:

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

Mientras está en el nodo GPU, puede ejecutar `nvidia-smi` para obtener información sobre las GPU asignadas.

Scripts de ejemplo para un trabajo de GPU

- Ejemplo usando una GPU Nvidia A100 compleata

```
#!/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
```

Más ejemplos

Visita nuestro github https://github.com/hpciter/user_codes