# Ejemplos de script

En esta sección, le proporcionamos un conjunto de ejemplos que le serán útiles para configurar scripts según sus necesidades de investigación. Si está interesado en echarles un vistazo, están todos disponibles en el directorio */share/user_codes/*.

Además, estos ejemplos están en nuestro repositorio. https://github.com/hpciter/user_codes.git

Para ejecutar los scripts, cópielos en su carpeta *data*.

```
$ tree /share/user_codes/
├── arrays
│   ├── 01-2node-4cpupertask
│   │   ├── 01-2nodes-4cpupertask.sbatch
│   │   └── README.md
│   ├── 02-1node-4cpupertask
│   │   ├── 02-1node-4cpupertask.sbatch
│   │   └── README.md
│   └── 03-1node-6cpupertask-ilk
│       ├── 03-1node-6cpupertask.sbatch
│       └── README.md
├── basics
│   ├── 00-sandy-nodes.sbatch
│   ├── 01-icelake-nodes.sbatch
│   └── README.md
├── gpu_basics
│   ├── 01_gpu_basic_a100.sbatch
│   ├── 02_gpu_basic_mig_partition.sbatch
│   └── README.md
├── images
│   └── teidehpc_logo.png
├── LICENSE
└── README.md
```

## Trabajos de uno y varios núcleos

Dependiendo del consumo de recursos, la simulación podría necesitar más de un procesador. En el script bash, es posible especificar cuántos núcleos se solicitan para ejecutar el trabajo.

Si la simulación usa más de un núcleo, debe agregar el parámetro *--cpu-per-task_o -c_* con la cantidad de núcleos de CPU utilizados por tarea.

Sin esta opción, el controlador asignará un núcleo. Recuerde especificar el número de tareas con el parámetro *--ntasks* o *-n*.

| Single core | Multi-core |
| --- | --- |
| #SBATCH --ntasks=X donde X => 1 | #SBATCH --ntasks=X donde X => 1<br>#SBATCH --cpus-per-task=Y donde Y > 1 |

En algunos casos, es posible que necesite ejecutar su trabajo en una arquitectura Sandy o Icelake. Esta función se puede seleccionar mediante el parámetro --*constrain* o *-C*.

## Seleccionar arquitectura de nodos

Para seleccionar la arquitectura del nodo use el parámetro *--contrain*:

| **Sandy bridge** | **Icelake** |
| --- | --- |
| #SBATCH --constrain=sandy | #SBATCH --constrain=ilk |

## OMP, MPI y Trabajos Híbridos

Es posible encontrar diferentes tipos de paralelismo: OpenMP, OpenMPI o una solución híbrida combinando ambos. Por un lado, utilizará OpenMP para el paralelismo dentro de un nodo multinúcleo.

Dado que es una implementación de subprocesos múltiples, se debe definir una variable OMP_NUM_THREADS. En cambio, si el paralelismo es entre nodos, usarás OpenMPI. Entonces, en nuestro script de muestra, será necesario especificar la cantidad de nodos, la cantidad de tareas en cada nodo y cada CPU.

| **OpenMP** | **OpenMPI** | **Hybrid** |
| --- | --- | --- |
| #SBATCH --cpus-per-tasks=X donde X > 1<br>export OMP_NUM_THREADS=X donde X > 1 | #SBATCH --ntasks=X donde X => 1<br>#SBATCH --cpus-per-task=Y donde Y>1<br>#SBATCH --nodes=Z donde Z =>2<br>#SBATCH --ntasks-per-node=W donde W>1 | Combine both options |

| OpenMP | OpenMPI | Hybrid |
| --- | --- | --- |
| | #SBATCH --ntasks-per-socket=U donde U>1<br>module load OpenMPI/2.0.2-GCC-6.3.0-2.27 | |

## Arrays de Jobs

Con el uso de arrays puede ejecutar múltiples trabajos con los mismos parámetros. En su script debe especificar la opción *--array*.

| Array |
| --- |
| #SBATCH --array=1-X donde X > 1 |

## Trabajos de GPU

Es imprescindible utilizar el parámetro *--gres* para reservar un recurso GPU y cargar el módulo CUDA.

| Nvidia A100 | Nvida Tesla T4 |
| --- | --- |
| #SBATCH --gres=gpu:a100:1<br>module load CUDA/8.0.61 | #SBATCH --gres=gpu:t4:1<br>module load CUDA/8.0.61 |