

## SMG

Se recomienda la lectura la [guía de usuario](#) del Cluster SMG antes de la realización de las prácticas.

### Práctica 1

El objetivo de esta práctica es tomar contacto con el uso del cluster, así como del sistema de PBS/TORQUE instalado en él:

Conéctese al frontend de SMG (ui.macc.unican.es) mediante el comando `ssh` (Linux/Mac OS) o el programa [?PyTTY](#) (Windows). Para ello, previamente cada alumno ha recibido una cuenta y una clave de acceso que debe cambiar:

```
[user@ui ~]$ ssh user@ui.macc.unican.es
```

- Desde el frontend, utilizando el comando `qstat`. Compruebe el numero de colas disponibles en cluster y si la cola 'course' se encuentra entre ellas. Si es así, por medio del comando `qstat` indique las características básicas de la cola.
- Utilizando la plantilla que se adjunta enviar un job de prueba. Para lo cual, será necesario utilizar el comando `qsub`, así como los editores `vi` o `nano`.

Plantilla:

```
#!/bin/bash

echo "Nodo: $HOSTNAME"

echo "Hora de inicio `date`"
sleep 30
echo "Hora de fin `date`"
```

- Envío del job:

```
[user@ui ~]$ qsub -q course sleep_template
```

- Comprobar el resultado obtenido en los ficheros de **output** y **error**. Como modificaría la plantilla paque que el fichero **output** se llamase 'sleep\_template\_output' y el **error** 'sleep\_template\_error'?
- Que comando he de utilizar para monitorizar solamente mis jobs?

### Práctica 2

Una vez enviado nuestro primer job, ejecutaremos un job de tipo openMP usando el código del programa [?HelloWorldOpenMP](#).

Haciendo uso del programa **module** cargar el compilador `gcc`:

```
[user@login1 ~]$ module load gcc
load gcc/4.6.3 (PATH, MANPATH, LD_LIBRARY_PATH)
```

Compilar el programa `HelloWorldOpenMP.c` y generar un ejecutable con el nombre `HelloWorldOpenMP`.

```
[user@login1 ~]$ gcc HelloWorldOpenMP.c -fopenmp -o HelloWorldOpenMP
```

- Plantilla a ejecutar en el supercomputador:

```
#!/bin/bash
#@ job_name = openmp_%j
#@ initialdir = .
#@ output = openmp_%j.out
#@ error = openmp_%j.err
#@ total_tasks = 32
#@ wall_clock_limit = 00:02:00

echo "Numero de procesos: ${SLURM_NPROCS}"
```

```

echo "Numero de nodos: ${SLURM_JOB_NUM_NODES}"
echo "Numero de procesos por nodo: ${SLURM_JOB_CPUS_PER_NODE}"
echo "Nodos: ${SLURM_JOB_NODELIST}"

./HelloWorldOpenMP

```

Más información sobre las variables de entorno ( $\${SLURM\_*$ ) en el [?link](#).

- Envío del job openMP:

```

$ module load gcc
load gcc/4.6.3 (PATH, MANPATH, LD_LIBRARY_PATH)
$ mnsuubmit HelloWorldOpenMP_template
Submitted batch job 621342

```

- Una vez finalizado el job, compruebe el resultado obtenido en los ficheros **output** (openmp\_%j.out) y **error** (openmp\_%j.err). ¿ El resultado obtenido es el esperado? ¿Por qué?.

### Práctica 3

Repita el envío del job de la práctica 3 modificando el número de *threads* a ejecutar, pero sin modificar la variable *total\_tasks*. Para ello, utilice la variable de entorno OMP\_NUM\_THREADS.

- La plantilla a utilizar en ese caso será:

```

#!/bin/bash
#@ job_name = openmp_%j
#@ initialdir = .
#@ output = openmp_%j.out
#@ error = openmp_%j.err
#@ total_tasks = 32
#@ wall_clock_limit = 00:02:00

export OMP_NUM_THREADS=XX

echo "Numero de procesos: ${SLURM_NPROCS}"
echo "Numero de nodos: ${SLURM_JOB_NUM_NODES}"
echo "Numero de procesos por nodo: ${SLURM_JOB_CPUS_PER_NODE}"
echo "Nodos: ${SLURM_JOB_NODELIST}"

./HelloWorldOpenMP

```

Donde **XX** es el número de *threads* a ejecutar.

- Envíe 3 jobs con **XX** igual a 8, 16 y 32, y compruebe si los resultados son el esperados.