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 Supercomputador Altamira, así como del sistema de colas ?SLURM instalado en él:

Conéctese al frontend de Altamira (altamira1.ifca.es) mediante el comando `ssh` (Linux/Mac OS) o el programa ?PyTTY (Windows). Para ello, previamente cada alumno ha recibido un correo con una cuenta y una clave de acceso:

```
[user@localmachine ~]$ ssh user@altamira1.ifca.es
```

- Desde el frontend, utilizando la plantilla que se adjunta enviar un job. Para lo cual, será necesario utilizar el comando **mnsubmit**, así como los editores **vi** o **nano**.

  Plantilla:

```
#!/bin/bash
#@ job_name = sleep_%j
#@ initialdir = .
#@ output = sleep_%j.out
#@ error = sleep_%j.err
#@ total_tasks = 1
#@ wall_clock_limit = 00:02:00

echo "Nodo: ${SLURM_NODELIST}"

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

- Envío del job:

```
[user@login1 ~]$ mnsubmit sleep_template
Submitted batch job 621336
```

  Para monitorizar el job use el comando **mnq**:

```
[user@login1 ~]$ mnq
 JOBID     NAME     USER    STATE       TIME TIMELIMIT CPUS  NODES NODELIST(REASON)
621336 0.sleep_     user  PENDING       0:00      2:00    1      1 (Priority)
```

- Comprobar el resultado obtenido en los ficheros de **output**(sleep_%j.out) y **error**(sleep_%j.err).

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 (${SLRUM_*}) en el ?link.

- Envió del job openMP:

```
$ module load gcc
load gcc/4.6.3 (PATH, MANPATH, LD_LIBRARY_PATH)
$ mnsubmit 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.