Guía de mejores prácticas de SLURM para usuarios de HPC.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Contenido

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

1.- INTRODUCCIÓN[1]

Slurm (**Simple Linux Utility for Resource Management**) es uno de los sistemas de gestión de tareas más utilizado en clústeres de computación.

Tiene tres funciones principales:

1. **Asignación de recursos**: asigna a los usuarios acceso a los recursos (nodos de cómputo) a los usuarios durante un período de tiempo determinado para que puedan realizar su trabajo.
2. **Inicio y monitoreo del trabajo**: proporciona un marco para iniciar, ejecutar y monitorear el trabajo en el conjunto de nodos asignados.
3. **Arbitraje de recursos**: asigna arbitrariamente los recursos mediante la administración de una cola de trabajo pendiente.

Slurm es un código abierto, de gestión de clústeres y programación de trabajos, tolerante a fallos, que no requiere modificaciones del kernel para su funcionamiento. Se emplea ampliamente en supercomputadores y clústers, y más concretamente es empleado por algunos de los socios de **NCC Spain**.

Por lo que desde **NCC Spain**, consideramos de gran interés para el usuario que solicite el acceso a nuestros recursos de **HPC**, ya sea mediante el **EuroCC Spain Testbed** u otros medios, conozcan y sepan emplearlo, y tengan a su disposición una serie de buenas prácticas y/o tips que puedan facilitar su uso. Es por ello que se ha recabado en este documento una serie de recomendaciones y comandos que pueden resultar de utilidad

Slurm está escrito en lenguaje C y utiliza un motor de configuración GNU **autoconf**. Si bien inicialmente fueron escritos para Linux, otros sistemas operativos similares a UNIX deberían ser objetivos fáciles de trasladar. El código debe adherirse al estilo de codificación del kernel de Linux.

Arquitectura

Slurm consiste en un daemon **slurmd** que se ejecuta en cada nodo de cómputo y un daemon central de **slurmctld** que se ejecuta en un nodo de administración (con gemelo de conmutación por error opcional).

Los daemons **slurmd** proporcionan comunicaciones jerárquicas tolerantes a errores.

Las entidades administradas por estos daemons de Slurm incluyen **nodos**, el recurso informático de Slurm, **particiones**, que agrupan los nodos en conjuntos lógicos (posiblemente superpuestos), **trabajos** o asignaciones de recursos asignados a un usuario para una cantidad de tiempo especificada y **pasos de trabajo**, que son conjuntos de tareas (posiblemente paralelas) dentro de un trabajo. Las particiones se pueden considerar colas de trabajos, cada una de las cuales tiene una variedad

[1] Fuente: Slurm Workload Manager - Descripción general (schedmd.com)

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

de restricciones como el límite de tamaño del trabajo, el límite de tiempo del trabajo, los usuarios autorizados a usarlo, etc. A los trabajos ordenados por prioridad se les asignan nodos dentro de una partición hasta que los recursos (nodos, procesadores, memoria, etc.) dentro de esa partición se agotan. Una vez a un trabajo se le asigna un conjunto de nodos, el usuario puede iniciar un trabajo paralelo en la forma de los pasos de trabajo en cualquier configuración dentro de la asignación. Por ejemplo, se puede iniciar un solo paso de trabajo que utilice todos los nodos asignados al trabajo, o varios pasos de trabajo pueden utilizar de forma independiente una parte de la asignación.

2.- CONCEPTOS BÁSICOS[i]

Los conceptos más importantes dentro de slurm son:

- Nodos de cómputo.
- Particiones.
- Trabajos.
- Tareas que representan un proceso dentro de un trabajo.

a) Nodos de login

Desde los nodos de login es desde donde el usuario interactúa con Slurm y desde donde lanza y monitoriza sus trabajos. Desde aquí, el usuario accede a sus datos y a los resultados de las ejecuciones.

b) Particiones

Las particiones son un mecanismo para agrupar en forma lógica los nodos, cada una de las cuales pueden tener una variedad de restricciones como límite de tamaño del trabajo, límite de tiempo del trabajo, usuarios autorizados a usarlo, etc. Al lanzarse los trabajos en una partición concreta, éstos son ordenados por prioridad dentro de una partición hasta que los recursos (nodos, procesadores, memoria, etc.) dentro de esa partición están agotados pudiendo quedar a la espera.

c) Comandos más comunes en SLURM

Para familiarizarse con el gestor de tareas se recogen los comandos más comunes de slurm. Para obtener más información acerca del comando como opciones se puede ejecutar

man <command>
<command> --help

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

- **sbatch** <script file>: lanzar un script al gestor de colas.

  Se utiliza para enviar un script de trabajo para su posterior ejecución. Normalmente, el script contendrá uno o más comandos de ejecución para iniciar tareas paralelas.

- **squeue:** para verificar el estado de los trabajos en las colas.

  Informa del estado de los trabajos. Tiene una amplia variedad de opciones de filtrado, clasificación y formato. De forma predeterminada, informa de los trabajos en ejecución en orden de prioridad y, a continuación, de los trabajos pendientes.

- **scancel** <job_id list>: para cancelar un trabajo.

  Se utiliza para cancelar un o ejecutar el trabajo o el paso del trabajo. También se puede utilizar para enviar una señal arbitraria a todos los procesos asociados a un trabajo en ejecución o a un paso de trabajo

- **scontrol show job** <job_id>: para obtener información del estado de un trabajo.

  Es la herramienta administrativa que se utiliza para ver y/o modificar el estado de Slurm.

- **sinfo**: para ver estado de las colas del sistema.

  Informa del estado de las particiones y nodos gestionados por Slurm. Tiene una amplia variedad de filtrado, clasificación y formato opciones.

- **salloc** <opciones>: para iniciar sesión interactiva (obtener un nodo para ejecutar).

  Utilizada para asignar recursos a un trabajo en tiempo real. Normalmente, esto se utiliza para asignar recursos y generar un Shell, es decir, ejecutar una sesión interactiva.

- **srun** <aplicacion>: para enviar un trabajo a ejecutar o inicia los pasos del trabajo en tiempo real.

  SRUN tiene una amplia variedad de opciones para especificar los requisitos de recursos, entre los que se incluyen: mínimo y el número máximo de nodos, el número de procesadores, los nodos específicos que se van a utilizar o no, y características específicas del nodo (memoria, espacio en disco, características, etc.). Un trabajo puede contener varios pasos de trabajo que se ejecutan secuencialmente o en paralelo en recursos independientes o compartidos dentro de la asignación de nodos del trabajo

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

- **sacct**: para consultar el accounting de la propia cuenta.

  Se emplea para reportar información sobre los trabajo activos o completados.

- **sstat:** obtener información sobre los recursos utilizados por un trabajo en ejecución.

  Se utiliza para obtener información acerca de los recursos utilizados por un trabajo en ejecución o un paso de trabajo.

- **sview** es una interfaz gráfica de usuario para obtener y actualizar la información de estado de los trabajos, particiones y nodos administrados por Slurm.

3.- EMPLEO DE COMANDOS ÚTILES EN SLURM[2]

Slurm proporciona una variedad de herramientas que permiten al usuario administrar y comprender sus trabajos. En esta apartado de presentaráns estas herramientas y e proporcionarán detalles sobre cómo usarlas.

a) Encontrar información en la cola de trabajo con squeue

El comando squeue es una herramienta que se usa para obtener información sobre los trabajos que están en la cola. De manera predeterminada, el comando squeue imprimirá el ID del trabajo, la partición, el nombre del trabajo, el usuario del trabajo, el estado del trabajo, el tiempo que lleva en ejecución, la cantidad de nodos y la lista de nodos asignados:

```
squeue
JOBID  PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
111111     batch   my_job   myuser  R    1:21:59      1 node0101-1
```

Podemos generar información no abreviada con la marca --long. Esta bandera imprimirá la información predeterminada no abreviada con la adición de un campo de límite de tiempo:

```
squeue -l
squeue --long
```

El comando squeue también brinda a los usuarios un medio para calcular la hora de inicio estimada de un trabajo agregando el indicador --start a nuestro comando.

[2] Fuente: https://doc.hpc.iter.es/2023.03/slurm/how_to_slurm_useful_commands/

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Esto agregará la hora de inicio estimada de Slurm para cada trabajo en nuestra información de salida.

```
squeue --user=username --start
```

Nota

La hora de inicio proporcionada por este comando puede ser inexacta. Esto se debe a que la hora calculada se basa en los trabajos en cola o en ejecución en el sistema. Si un trabajo con una prioridad más alta se pone en cola después de ejecutar el comando, su trabajo puede ser demorado.

Al verificar el estado de un trabajo, es posible que desee llamar repetidamente al comando squeue para buscar actualizaciones. Podemos lograr esto agregando el indicador --iterate a nuestro comando squeue. Esto ejecutará squeue cada n segundos, lo que permite una actualización frecuente y continua de la información de la cola sin necesidad de llamar repetidamente a squeue:

```
squeue --start --iterate=n_seconds
```

Presione ctrl-c para detener el bucle del comando y regresar a la terminal.

b) Estado de los trabajos

Una vez que se ha enviado un trabajo a una cola de trabajos, la ejecución seguirá estos estados:

- PENDING o PD: el trabajo ha entrado en la cola pero aún no están disponibles los recursos solicitados para que empiece a trabajar, es decir, que no haya nodos libres.
- RUNNING o R: el trabajo se está ejecutando en la cola con los recursos que se han solicitado.
- COMPLETED o CD: el trabajo se ha ejecutado correctamente, o al menos, lo que se ha especificado en el script de lanzamiento.
- COMPLETING o CG: el trabajo está en proceso de terminar en un estado correcto.
- SUSPEND o S: la ejecución del trabajo se ha suspendido y los recursos utilizados se han liberado para otros trabajos.
- CANCELLED o CA: el trabajo ha sido cancelado o por el usuario o por los administradores de sistemas.
- FAILED o F: ha fallado la ejecución del trabajo.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

- NODE_FAIL o NF: ocurrió un error con el nodo y el trabajo no pudo lanzarse. Por defecto, Slurm relanza el trabajo de nuevo.

c) Razones de que un trabajo esté en PENDING

Cuando un trabajo se encuentra en el estado de PENDING, se añade la razón de porqué está pendiente para ejecutar y ésta puede ser:

- **(Resources)**: el trabajo está esperando hasta que estén disponibles los recursos solicitados.
- **(Dependency)**: el trabajo es dependiente de otro y, por tanto, no entrará a ejecutar hasta que se cumpla la condición establecida para la dependencia.
- **(DependencyNeverSatisfied)**: el trabajo está esperando por una dependencia que no ha sido cumplida. El trabajo se quedara en este estado para siempre, por tanto, hay que cancelar el trabajo.
- **(AssocGrpCpuLimit)**: el trabajo no se puede ejecutar porque se ha consumido la cuota asignada de CPU.
- **(AssocGrpJobsLimit)**: el trabajo no se puede ejecutar porque se ha alcanzado el límite de trabajos simultáneos que tiene permitido ejecutar el usuario o la cuenta.
- **(ReqNodeNotAvail)**: el nodo especificado no está disponible. Puede ser que esté en uso, que esté reservado o puede que esté marcado como "fuera de servicio".

Info

Para obtener más información sobre squeue, visite la página de Slurm en squeue

d) Detener trabajos con scancel

En ocasiones, es posible que necesite detener un trabajo por completo mientras se está ejecutando. La mejor manera de lograr esto es con el comando **scancel**. Este comando permite cancelar los trabajos que está ejecutando en los recursos de Research Computing utilizando la ID del trabajo. El comando se ve así:

```
scancel your_job-id
```

Para cancelar varios trabajos, puede usar una lista de ID de trabajos separados por comas:

```
scancel your_job-id1, your_job-id2, your_jobiid3
```

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Info

Para obtener más información, visite el manual de Slurm en scancel

e) Información de estado con sstat

El comando sstat permite a los usuarios obtener fácilmente información sobre el estado de sus trabajos actualmente en ejecución. Esto incluye información sobre el uso de la CPU, información de la tarea, información del nodo, tamaño del conjunto residente (RSS) y memoria virtual (VM). Podemos invocar el comando **sstat** como tal:

```
sstat --jobs=your_job-id
```

f) Formatear la salida del sstat

De forma predeterminada, **sstat** extraerá mucha más información de la que se necesitaría en la salida predeterminada de los comandos. Para remediar esto, podemos usar el indicador --format para elegir lo que queremos en nuestra salida. El indicador de formato toma una lista de variables separadas por comas que especifican los datos de salida:

```
sstat --jobs=your_job-id --format=var_1,var_2, ... , var_N
```

Algunas de estas variables pueden ser:

Variable	Descripción
avecpu	Promedio de tiempo de CPU de todas las tareas en el trabajo.
averss	Tamaño medio del conjunto residente de todas las tareas.
avevmsize	Memoria virtual promedio de todas las tareas en un trabajo.
jobid	ID del trabajo.
maxrss	Número máximo de bytes leídos por todas las tareas del trabajo.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Variable	Descripción
maxvsize	Número máximo de bytes escritos por todas las tareas en el trabajo.
ntasks	Número de tareas en un trabajo.

Por ejemplo, imprima la identificación de trabajo promedio de un trabajo, el tiempo de CPU, el máximo de rss y la cantidad de tareas. Podemos hacer esto escribiendo el comando:

```
sstat --jobs=your_job-id --format=jobid,cputime,maxrss,ntasks
```

Info

Se puede encontrar una lista completa de variables que especifican datos manejados por sstat con el indicador --helpformat o visitando la página de slurm en sstat.

g) Analizar trabajos terminados con sacct

El comando **sacct** permite a los usuarios obtener información sobre el estado de los trabajos terminados. Este comando es muy similar a sstat, pero se usa en trabajos que se ejecutaron previamente en el sistema en lugar de los trabajos que se están ejecutando actualmente. Podemos usar la identificación de un trabajo.

- **Para todos los trabajos ejecutados:**

```
sacct
```

- **Para un único trabajo, identificado por su ID:**

```
sacct --jobs=your_job_id
```

De forma predeterminada, sacct solo extraerá los trabajos que han ejecutado en **el día en curso**. Podemos usar el indicador --starttime para decirle al comando que mire más allá de su caché de trabajos a corto plazo.

```
sacct –-jobs=your_job-id –-starttime=YYYY-MM-DD
```

Para ver una versión no abreviada de la salida de sacct, use el indicador --long:

```
sacct –-jobs=your_job-id –-starttime=YYYY-MM-DD --long
```

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

h) Formatear la salida del sacct

Al igual que sstat, es posible que la salida estándar no proporcione la información que queremos. Para remediar esto, podemos usar el indicador --format para elegir lo que queremos en nuestra salida. De manera similar, el indicador de formato es manejado por una lista de variables separadas por comas que especifican los datos de salida:

```
sacct --user=your_rc-username --format=var_1,var_2, ... ,var_N
```

A continuación, se proporciona una lista de algunas variables:

Variable	Description
account	Cuenta con la que se ejecutó el trabajo
avecpu	Tiempo promedio de CPU de todas las tareas en el trabajo.
averss	Average resident set size of all tasks in the job.
cputime	Tiempo transcurrido de CPU usado por un job o paso
elapsed	Tiempo transcurrido de los trabajos con formato DD-HH:MM:SS
exitcode	El código de salida devuelto por el script de trabajo o salloc.
jobid	ID del trabajo.
jobname	Nombre del trabajo.
maxdiskread	Número máximo de bytes leidos por todas las tareas.
maxdiskwrite	Número máximo de bytes leidos por todas las tareas.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Variable	Description
maxrss	El código de salida devuelto por el script de trabajo o salloc.
ncpus	Cantidad de CPU asignadas.
nnodes	Número de nodos usados.
ntasks	Número de tareas en un job.
priority	Prioridad Slurm.
qos	Calidad de servicio.
reqcpu	Número de CPUs solicitados
reqmem	Cantidad de memoria necesaria para un trabajo
user	Nombre de usuario de la persona que ejecutó el trabajo.

Por ejemplo, suponga que desea buscar información sobre los trabajos que se ejecutaron el 12 de marzo de 2018. Desea mostrar información sobre el nombre del trabajo, la cantidad de nodos utilizados en el trabajo, la cantidad de CPU, el **maxrss** y el tiempo transcurrido. Su comando se vería así:

```
sacct --starttime=2018-03-12 --format=jobname,nnodes,ncpus,maxrss,elapsed
```

Info:

Se puede encontrar una lista completa de variables que especifican datos manejados por sacct con el indicador --helpformat o visitando la página de slurm en sacct.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

i) Control de trabajos en cola y en ejecución mediante scontrol

El comando **scontrol** proporciona a los usuarios un mayor control de sus trabajos ejecutados a través de Slurm. Esto incluye acciones como suspender un trabajo, detener la ejecución de un trabajo o extraer información detallada sobre el estado de los trabajos.

Para suspender un trabajo que se está ejecutando actualmente en el sistema, podemos usar **scontrol** con el comando suspender. Esto detendrá un trabajo en ejecución en su paso actual que se puede reanudar en un momento posterior. Podemos suspender un trabajo escribiendo el comando:

```
scontrol suspend job_id
```

Para reanudar un trabajo en pausa, usamos scontrol con el comando reanudar:

```
scontrol resume job_id
```

Slurm también proporciona una utilidad para retener trabajos que están en cola en el sistema. Retener un trabajo colocará el trabajo en la prioridad más baja, efectivamente "reteniendo" el trabajo para que no se ejecute. Un trabajo solo se puede retener si está esperando que el sistema se ejecute. Usamos el comando de espera para poner un trabajo en estado de espera:

```
scontrol hold job_id
```

Luego podemos liberar un trabajo retenido usando el comando de **release:**

```
scontrol release job_id
```

scontrol también puede proporcionar información sobre trabajos mediante el comando show job. La información proporcionada por este comando es bastante extensa y detallada, así que asegúrese de borrar la ventana de su terminal, recopilar cierta información del comando o canalizar la salida a un archivo de texto separado:

```
scontrol show job job_id
```

j) Streaming de salida a un archivo de texto

```
scontrol show job job_id > outputfile.txt
```

k) Canalizar la salida a Grep y encontrar líneas que contengan la palabra "Tiempo"

```
scontrol show job job_id | grep Time
```

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

4.- TIPS3

Tip: Usar módulos en un script

Para poder utilizar el entorno módulos, es necesario hacer un source del fichero de perfil de módulos. Para ello habrá que añadir la siguiente línea en los scripts de submit antes de utilizar cualquier comando de módulos:

```
source /etc/profile.d/profile.modules.sh
```

Tip: Reservar nodos de cómputo completos

En los nodos de cómputo se recomienda reservar nodos completos utilizando la opción -N <nodes\>, para que no interfieran ejecuciones de otros usuarios entre sí. Recuerde que la facturación de estos nodos es por uso de nodo.

Tip: Enviar un trabajo con sbatch

Es aconsejable el enviar el trabajo mediante **sbatch** así como el uso de los modificadores -D <directorio\> y -t <tiempo\>.

-t <tiempo> o su equivalente #SBATCH --t <days-HH:MM>
-D <directorio> o su equivalente #SBATCH --D <directorio>

Tip: Número de tareas por nodo

Es posible realizar ejecuciones sin utilizar todos los cores disponibles en el nodo. Para ello sólo hay que solicitar el número de nodos mediante -N X y el número de procesos a ejecutar en cada nodo con --tasks-per-node:

```
srun -N 4 --tasks-per-node=8 <software>

#SBATCH --N 4
#SBATCH --tasks-per-node=8
```

[3] Fuente: Consejos de uso - TeideHPC (iter.es)

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

★ Tip: Notificaciones del gestor de trabajos slurm

Es posible gestionar la notificación automática de ciertos eventos del trabajo con las siguientes directivas.

```
#SBATCH --mail-user=EMAIL              # Email de notificación de eventos
#SBATCH --mail-type=EVENT1, EVENT2,...     # Eventos notificables
```

Aclaración sobre los eventos por correo de Slurm:

Slurm puede enviar correos a la dirección especificada sobre una serie de eventos que le ocurran al trabajo. Dichos eventos pueden ser:

- **BEGIN**: cuando el trabajo entra en ejecución.
- **END**: cuando la ejecución del trabajo finalizada.
- **FAIL**: cuando la ejecución del trabajo falla.
- **TIME_LIMIT**: cuando el trabajo alcanza el tiempo máximo de ejecución.
- **TIME_LIMIT_50**: cuando el trabajo haya alcanzado el 50% del tiempo límite.
- **TIME_LIMIT_80**: cuando el trabajo haya alcanzado el 80% del tiempo límite.
- **TIME_LIMIT_90**: cuando el trabajo haya alcanzado el 90% del tiempo límite.
- **ARRAY_TASKS**: envía una notificación por email por cada trabajo del array. Si, al usar arrays, no se especifica esta opción, se enviará un email como si fuera un único trabajo.
- **ALL**: todos los tipos de eventos.

De todos los eventos posibles, se recomienda se utilicen los relacionados con el consumo de tiempo límite permitido, *TIME_LIMIT_50, TIME_LIMIT_80 y TIME_LIMIT_90*. De esta forma, el usuario es consciente del tiempo que le queda al trabajo y, si fuese necesario, puede enviar a tiempo un correo a los administradores para que se le amplíe el tiempo de ejecución al trabajo.

★ TIPS: Encabezado de bash

Recomendamos usar `#!/bin/bash -e` en lugar de simple `#!/bin/bash`, de modo que la falla de cualquier comando dentro del script haga que su trabajo se detenga inmediatamente en lugar de intentar continuar con un entorno inesperado o datos

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

intermedios erróneos. También garantiza que sus trabajos fallidos muestren un estado de FALLADO en la salida *de sacct* .

★ TIPS: Recursos

No solicite más recursos (CPU, memoria, GPU) de los que vaya a necesitar. Además de utilizar sus horas principales más rápido, los trabajos que requieren muchos recursos tardarán más en ponerse en cola. Utilice la información proporcionada al finalizar su trabajo (por ejemplo, mediante el comando *sacct*) para definir mejor los requisitos de recursos.

★ TIPS: Tiempo de pared

Los trabajos largos pasarán más tiempo en la cola, ya que hay más oportunidades para que el programador encuentre un intervalo de tiempo para ejecutar trabajos más cortos. Por lo tanto, considere utilizar puntos de control de trabajos o, cuando sea posible, más paralelismo, para reducir la duración de los trabajos a unas pocas horas o, en el peor de los casos, días.

Deje algo de margen para la seguridad y la variabilidad entre ejecuciones en el sistema, pero trate de ser lo más preciso posible.

Si tiene muchos trabajos de menos de 5 minutos, entonces probablemente deberían combinarse en trabajos más grandes utilizando un bucle simple en el script por lotes para amortizar los gastos generales de cada trabajo (inicio, contabilidad, etc.).

★ TIPS: Memoria (RAM)

Si solicita más memoria (RAM) de la que necesita para su trabajo, esperará más tiempo en la cola y será más costoso cuando se ejecute. Por otro lado, si no solicita suficiente memoria, es posible que el trabajo se cancele por intentar exceder los límites de memoria asignados.

Le recomendamos que solicite un poco más de RAM, pero no mucha más, de la que su programa necesitará en su máximo uso.

También recomendamos usar `--mem` en lugar de `--mem-per-cpu` en la mayoría de los casos. Hay algunos tipos de trabajos para los que `--mem-per-cpu` es más adecuado.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

TIPS: Paralelismo

En general, sólo los trabajos MPI deben establecer *ntasks* mayores que 1 o usar *srun*. Si no sabe si su programa es compatible con MPI, probablemente no sea así.

Sólo los trabajos multiproceso deben configurar *cpus-per-task* . Si no sabe si su programa admite subprocesos múltiples, intente realizar una evaluación comparativa con 2 CPU y con 4 CPU y vea si hay una diferencia doble en el tiempo de trabajo transcurrido.

Las matrices de trabajos son un mecanismo eficaz para gestionar una colección de trabajos por lotes con requisitos de recursos idénticos. La mayoría de los comandos de Slurm pueden gestionar conjuntos de trabajos ya sea como elementos individuales (tareas) o como una sola entidad (por ejemplo, eliminar un conjunto de trabajos completo en un solo comando).

5.-MEJORES PRÁCTICAS PARA GRANDES CANTIDADES DE TRABAJOS

Considere la posibilidad de colocar el trabajo relacionado en un solo trabajo de Slurm con varios trabajos, tanto por razones de rendimiento como por facilidad de gestión. Cada trabajo de Slurm puede contener una multitud de pasos de trabajo y la sobrecarga en slurm para administrar los pasos del trabajo es mucho menor que el de los trabajos individuales.

Las matrices de trabajos son un mecanismo eficiente de administrar, una colección de trabajos por lotes con requisitos de recursos idénticos. La mayoría de los comandos de Slurm pueden administrar matrices de trabajos como elementos individuales (tareas) o como una sola entidad (por ejemplo, eliminar una matriz de trabajos completa en un solo comando).

MPI

El uso de MPI depende del tipo de MPI que se utilice. Se utilizan tres modos de funcionamiento fundamentalmente diferentes por estas diversas implementaciones de MPI.

1. Slurm inicia directamente las tareas y realiza la inicialización de comunicaciones a través de las APIs PMI2 o PMIx. (Con el apoyo de la mayoría de las implementaciones modernas de MPI).

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

2. Slurm crea una asignación de recursos para el trabajo y, a continuación, mpirun lanza tareas utilizando la infraestructura de Slurm (versiones anteriores de OpenMPI).

3. Slurm crea una asignación de recursos para el trabajo y, a continuación, mpirun lanza tareas utilizando algún mecanismo que no sea Slurm, como SSH o RSH. Estas tareas se inician fuera de la supervisión de Slurm o control. El epílogo de Slurm debe configurarse para purgar estas tareas cuando se renuncia a la asignación del trabajo. También se recomienda el uso de pam_slurm_adopt.

OPENMP[ii]

Open MP (Open Multi-Processing) es una interfaz de programación de aplicaciones (API) que permite la programación en paralelo en sistemas de memoria compartida. OpenMP se ha convertido en un estándar para programación paralela en sistemas de memoria compartida y es compatible con varios lenguajes de programación, como C, C++, y Fortran.

OpenMP se caracteriza por ser fácil de aprender, flexible y portátil. El código fuente de una aplicación puede ser adaptado a diferentes sistemas sin tener que modificar el código fuente original. Además, OpenMP ofrece una gran variedad de directivas para la creación de hilos, la sincronización y la asignación de trabajo.

Directivas de OpenMP

Las directivas OpenMP son palabras clave que se utilizan en el código fuente para indicar a la aplicación qué secciones del código se deben ejecutar en paralelo. Las directivas OpenMP también se utilizan para especificar la creación y sincronización de hilos.

Algunas de las directivas más empleadas en OpenMP:

parallel: esta directiva crea una región paralela donde el trabajo se divide entre múltiples hilos. Cada hilo ejecuta una copia de la región paralela y luego se unen al final.

for: se utiliza para paralelizar bucles. Divide el trabajo del bucle entre los hilos disponibles, donde cada hilo ejecuta una porción del bucle.

sections: se emplea para paralelizar secciones de código independientes. Cada sección se ejecuta en un hilo diferente.

single: con la directiva single, una sección de código se ejecuta solo en un hilo. Se puede utilizar para inicializaciones o para realizar operaciones que deben realizarse una sola vez.

task: se usa para crear tareas independientes que pueden ser ejecutadas por hilos disponibles. Proporciona un modelo de ejecución más flexible que las directivas for y sections.

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

critical: se utiliza para definir una sección crítica del código, donde solo un hilo puede ejecutarla a la vez. Se utiliza para proteger secciones de código que acceden a recursos compartidos.

atomic: se emplea para realizar operaciones atómicas en variables compartidas. Garantiza que la operación se realice sin interferencias de otros hilos.

barrier: se utiliza para sincronizar todos los hilos en un punto específico del programa. Los hilos esperarán hasta que todos los demás hilos alcancen el mismo punto antes de continuar.

PARA MÁS INFORMACIÓN PUEDE CONSULTAR TAMBIÉN:

GUÍA RÁPIDA DE USUARIO: https://slurm.schedmd.com/quickstart.html

VÍDEO TUTORIALES: https://www.schedmd.com/publications/

[i] Fuente: Cómo iniciar sesión en TeideHPC - TeideHPC (iter.es)

[ii] Fuente: https://medium.com/@leonardoaguirre_97179/openmp-y-la-programaci%C3%B3n-paralela-8990f14b95f3

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.