Investigar un trabajo fallido

No siempre los trabajos se ejecutan correctamente. Hay una lista de motivos por los que los trabajos o las aplicaciones se detienen o fallan. Las causas más comunes son:

- Exceder los límites de recursos solicitados, como la memoria o el tiempo límite, por ejemplo.
- Errores específicos del software.
- Errores en el procesamiento de los datos.

> **Nota**
>
> Es importante recopilar mensajes de error/salida ya sea escribiendo dicha información en la ubicación predeterminada o especificando ubicaciones específicas mediante las opciones `--error/--output`, por ejemplo, aunque esto dependerá de cada software.
>
> Los mensajes de error y de salida son el punto de partida para investigar un error en el trabajo.

Exceder los límites de recursos

Cada partición define límites máximos y predeterminados para el tiempo de ejecución y el uso de la memoria. Dentro de la especificación del trabajo, los límites actuales se pueden definir dentro de los rangos. Para una mejor programación, se deben estimar los requisitos del trabajo y adaptar los límites a las necesidades. Cuanto más bajos sean los límites mejor, así SLURM podrá encontrar un lugar. Además, cuanto menos gastos generales de recursos se especifican, menos recursos se desperdician, como por ejemplo, con la memoria.

> **Info**
>
> Si un trabajo excede el tiempo de ejecución o el límite de memoria, SLURM matará el trabajo.

Información de error

En ambos casos, el archivo de errores proporciona información adecuada:

- Tiempo límite:

```
(...)
slurmstepd: error: *** JOB 41239 ON fnode01 CANCELLED AT 2016-11-30T11:22:57 DUE
TO TIME LIMIT ***
(...)
```

- Límite de memoria:

```
(...)
slurmstepd: error: Job 41176 exceeded memory limit (3940736 > 2068480), being killed
slurmstepd: error: Exceeded job memory limit
slurmstepd: error: *** JOB 41176 ON fnode01 CANCELLED AT 2016-11-30T10:21:37 ***
(...)
```

Errores de Software

Slurm captura el código de salida de un trabajo y lo guarda como parte del registro del trabajo. **Para trabajos lanzados con sbatch, se captura el código de salida del script. Para trabajos lanzados con srun, el código de salida será el valor de retorno del comando ejecutado. Cualquier código de salida distinto de cero se considera un error de trabajo y da como resultado un estado de trabajo de FAILED**.

Si una señal es responsable de la terminación de un trabajo/paso, el número de la señal también se capturará y se mostrará después del código de salida (separado por dos puntos).

Según el orden de ejecución de los comandos en el script sbatch, es posible que un comando específico falle, pero el script por lotes devolverá cero, lo que indica que se realizó correctamente. Por ejemplo:

fail.R

```
var<-sq(1,1000000000)
```

job_sbatch.sh

```
#!/bin/bash

#SBATCH --job=my_r_test            #  Nombre del trabajo
#SBATCH --nodes=1                 # Nº de nodos
#SBATCH --constrains=sandy         # sandy, ilk (icelake)... arquitecture
```

```
#SBATCH --time=02:00                # Límite de tiempo

#SBATCH --output=file_%j.log        # Log de salida estandar
#SBATCH --error=file_%j.err         # Log de salida errores
#SBATCH --chdir=.                   # Directorio de trabajo

#SBATCH --mail-user=email           # Donde será enviado el mail
#SBATCH --mail-type=END,FAIL        # Eventos email
##############################################################

Rscript fail.R
echo "Script finished"
```

El código de salida y el estado indican incorrectamente que el trabajo finalizó correctamente:

```
sbatch job_sbatch.sh
Submitted batch job 41585

sacct -j 41585
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
41585        Simple E +        all         id          1  COMPLETED      0:0
41585.batch       batch                    id          1  COMPLETED      0:0
```

Solo el archivo de salida específico de R muestra el error:

fail.Rout

```
(...)
> var<-sq(1,1000000000)
Error: could not find function "sq"
Execution halted
```

Puede evitar este problema "saliendo" con un código de salida adecuado tan pronto como falle el comando:

job_sbatch.sh

```
#!/bin/bash

#SBATCH --job=my_r_test             #  Nombre del trabajo
#SBATCH --nodes=1                   # Nº de nodos
#SBATCH --constrains=sandy          # sandy, ilk (icelake)... arquitecture
#SBATCH --time=02:00                # Límite de tiempo

#SBATCH --output=file_%j.log        # Log de salida estandar
#SBATCH --error=file_%j.err         # Log de salida errores
#SBATCH --chdir=.                   # Directorio de trabajo

#SBATCH --mail-user=email           # Donde será enviado el mail
#SBATCH --mail-type=END,FAIL        # Eventos email
##############################################################
```

```
Rscript fail.R || exit 91
echo "Script finished"
```

Ahora, el código de salida y el estado coinciden con el verdadero resultado:

```
sbatch job_sbatch.sh
Submitted batch job 41925

sacct -j 41925
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
41925        Simple E +        all         id          1     FAILED     91:0
41925.batch       batch                   id          1     FAILED     91:0
```

Compruebe siempre los archivos de salida específicos de la aplicación en busca de mensajes de error.