Sesión práctica

Uso de Guacolda-Leftraru

v.04.23

Introducción

A lo largo de esta sesión se realizarán ejercicios prácticos que permitirán adquirir conocimientos para usar un sistema gestor de recursos computacionales, tal como es el caso de SLURM, el sistema que tenemos instalado en Guacolda-Leftraru.

Metodología

La clase se dividirá en varios grupos de trabajo y en cada uno de los grupos habrá un coordinador.

Para cada uno de los ejercicios:

1. El profesor plantea el ejercicio y resuelve posibles dudas de los alumnos.
2. Se divide la clase en grupos por un tiempo prefijado. Este tiempo inicialmente será de 5 minutos. El profesor avisará si el tiempo para resolver un ejercicio concreto es distinto al señalado
3. Dentro de cada uno de los grupos, el coordinador compartirá pantalla y entre todos resolverán el ejercicio de manera colaborativa. El profesor o algún asistente ingresará a los grupos de trabajo para apoyar y aclarar dudas.
4. Al finalizar el tiempo para la ejecución del ejercicio se regresará a la sala principal.
5. Uno de los grupos será seleccionado para que su coordinador comparta su pantalla y presente la resolución del ejercicio.
6. Los participantes pueden intervenir para ayudar en la solución del problema, como también para aclarar dudas que surjan.

   NOTA: no hace falta anotar los comandos y respuestas de los ejercicios, tan solo tener clara la respuesta para poder presentarla.

¿Cómo conectarse al Cluster Leftraru-Guacolda?

Los ejercicios se deben ejecutar en el Custer Leftraru-Guacolda.

Para esto debe conectarse mediante protocolo *SSH* al Cluster con el nombre de usuario y contraseña entregado para este curso, o con su usuario y contraseña personal si ya tiene cuenta.

Desde una *Terminal* utilice el comando:

```
ssh $USERNAME@leftraru.nlhpc.cl
```

Se le solicitará su contraseña para autenticar su identidad y posteriormente podrá ver un mensaje similar a:

```
Last login: Mon Feb 28 10:21:59 2022 from pc-14-88-239-201.cm.byt.net
**************** MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
* BIENVENID@ A * MMMMMMMMMMMMMMMMMMNmhhhyhmMMMMMMMMMMMMMMMMMM NLHPC MMMMMMM
*   GUACOLDA   * MMMMMMMMMMMMMMMMMy/-..`````.oNMMMMMMMMMMMMMMMMMMMMMMMMMMM
*   LEFTRARU   * MMMMMMMMMMMMMMMMy:---:--.--::+MMMMMMMMMMMMMMMMMMMMMMMMMMM
**************** MMMMMMMMMMMMMMMdo://:+o:///+::yMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMNs+-/:///:o/+/:+`::oNMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMd+`   ` .-/+ssso:-`    `:yNMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMh-.````.ososss+o+s+-     `.NMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMd.``..` :-sssososss--      `oMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMy+`       -/o-ss+ooos-/o/       `:odNMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMN:        ..` .oso+/` ```   `        .omMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMd.       ` `-/+so/+s/:.`/+/o:`        `sNMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMNo.+`:` ``-//+:/:--....+ssos:`      . `dMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMy++/::/:-:::-:::::::---/ooooss+.    .o hMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMNo/////:::::-:::::----.-+sssssssss:    -+MMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMo/////////:-------/yho.  `+sssossssss:  :NMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMm::::::--..--:o/-+yhhh++ys+-+s/sssssss.oMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMN:-----.....:d:.-/yss+o-:/o+ .::++//:::+MMMMMMMMMMM
MMMMMMMMMMMMMMMMMMs-------`    `+   /:os:/-    `   .::::::::dMMMMMMMMMM
MMMMMMMMMMMMMMMMMN:/ossss-```      `+oss+so+o     `hd---....:NMMMMMMMMM
MMMMMMMMMMMMMMMMMssossss/`.....``  // .+` +.     yMMy.----:/oMMMMMMMMMM
MMMMMMMMMMMMMMMMM+sosss/.--::      /: .s  o.     mMMMh:ossssodMMMMMMMMM
MMMMMMMMMMMMMMMMyysssss-`-:///`    +:..s  o-     hMMMMNoossss+mMMMMMMMM
MMMMMMMMMMMMMMMdoss+/-.--:///-   `.+. `o  :-    `/sNMMMy/osssoNMMMMMMMM
MMMMMMMMMMMMMMMosso+s-.----//:   ```+` `o  /:    .:`+yNMN+ssssyMMMMMMMM
MMMMMMMMMMMMMMhssssss:.------::    `:  `:  ::    ./+:.+ohM+ssssooNMMMMMMM
MMMMMMMMMMMMMMossssoMy------`    .-  `:  ::     :+++/--o/sso+:-oMMMMMMM
M*KID*MMMMMMMm+ssss/NMMNo---.    `--:osos+::-    ./++++/-.-.....:MMMMMMM
MMMMMMMMMMMd+//:://+MMMMNh.`     `/ Guacolda/     -:/++++:``....:MMMMMMM
MMMMMMMMMMMy/+oo/-sMMMMMo``    -:/:/+//:/+s`     :::++++/`....:MMMMMMM
MMMMMMMMMMMy+dmmmMMMMMhhhhhhhhhyhhdhhh:`yhhdmmdhyyssyhdddmmmmddmMmmNMMMM
MMMd:yMMMMMMN  mMMMd``mMMMMMMMMN. mMMMMM` mMMh:.``.-:--..sMMMMMmo-`.` sMMM
MMM-  /mMMMMh -MMMMs `NMMMMMMMMs -MMMMMd .MMMdh/ /MMMMMo /MMMd: :yNy``hMMM
MMd /y.`oNMMo sMMMM: +MMMMMMMMM: yMMMMM+ +MMMMN``mMMdo..sNMMo -dMMMNdNMMMM
MM: dMNs`.hM: mMMMN` dMMMMMMMMm  .---.`  hMMMMy .+-`-odMMMMs -NMMMMMMMNoy
Md -MMMMm: o`.MMMMy -MMMMMMMMo :dddmmm  NMMMM+ .ohNMMMMMN` hMMMMMMd+.+N
M+ oMMMMMMo  :MMMM: oNdys++/sM- sMMMMMN  mMMMM+ /MMMMMMMMN. yMMMmy+.-sNMM
M/ yMMMMMMdohMMMM- `.:/osssmM/.dMMMMMM/:mMMMMy -MMMMMMMMMd:``..-+ymMMMMM
MMNMMMMMMMMMMMMMMMMNMMMMMMMMMMMMMMMMMMMMMMMMMMmNMMMMMMMMMMMMMMMMMMMMMMM

Laboratorio Nacional de Computacion de Alto Rendimiento (NLHPC)
Centro de Modelamiento Matematico (CMM)

Universidad de Chile
```

Si necesita más información para conectarse exitosamente, puede consultar al presentador del curso o visitar nuestro Tutorial de acceso a Leftraru vía SSH.

Ejercicios

A continuación se presentan los ejercicios que se realizarán durante la presentación.

Ejercicio 1

1. Ejecuta el comando `hostname` en la partición *slims* de Leftraru con el comando `srun`.
    - Con un único proceso
    - Con dos procesos iguales
    - Con dos procesos en distintos nodos
    - Con un proceso con dos *cores* reservados
2. ¿Qué resultados se han obtenido?
3. En la partición *slims*:
    - ¿Cuántos *cores* puedo reservar por proceso?
    - ¿Por qué ese número?
    - ¿Qué diferencia hay en la partición *General*?
    - ¿Qué ocurre si reservo más *cores* de los disponibles?
4. ¿Qué ocurre si no especifico la partición en la que quiero ejecutar mi comando?

Objetivo

- Ejecutar un comando común de Linux mediante *Slurm* en modo interactivo, reservando distinta cantidad de recursos y ver el resultado en diferentes escenarios.
- Identificar la cantidad de recursos en la partición *slims* y comprobar la diferencia con la partición *General*.
- Forzar un error al requerir más *cores* de los disponibles.
- Identificar la partición de recursos *por defecto* existente en el Cluster.

Solución

Punto 1

Los comandos a ejecutar son los siguientes:

```
srun -n 1 hostname
```

```
srun -n 2 hotname
```

```
srun -n 2 --ntasks-per-node=1 hostname

srun -n 1 -c 2 hostname
```

Punto 2

Los resultados pueden variar, ya que se mostrará el nombre de los distintos *hosts* en donde se ejecuten, obteniendo nombres con la nomenclatura *cnXXX*.

Punto 3

Se pueden reservar hasta 20 *cores* en la partición *slims*, ya que es el número de procesadores disponibles por nodo.

En la partición *General* es posible reservar hasta 44 *cores*.

Si se reservan más *cores* de los disponibles por nodo, se obtiene un error similar a:

```
srun: error: CPU count per node can not be satisfied
srun: error: Unable to allocate resources: Requested node configuration is
not available
```

Punto 4

Si no se especifica una partición, se utilizará la partición *por defecto* que corresponde a los nodos de la partición *Slim*.

Ejercicio 2

1. Crear un *script* de ejecución para lanzarlo mediante *sbatch*, con las siguientes consideraciones:
   - Utilizar la partición *slims*
   - Reservar un único *core*
   - Ejecutar el comando `stress -c 1`
2. Considerando que se está utilizando el comando `stress`, responda las siguientes preguntas:
   - ¿Cuánto tiempo estará la tarea en ejecución?
   - ¿Qué comando puede utilizar para obtener información acerca de la tarea en ejecución?
   - ¿Cómo puedo cancelar la tarea?
3. Para poder lanzar tareas, debo conocer el uso del *Cluster* ¿Qué comando me permite conocer el estado de las particiones?

Objetivo

Este ejercicio permite conocer la sintaxis básica para la ejecución de tareas de manera no interactiva, con la flexibilidad que permite enviar a la cola de trabajo y así poder esperar los resultados sin la necesidad de encontrarnos frente a nuestra computadora. También saber que los tiempos de ejecución tienen un máximo de 30 días para las cuentas estándar, y las de práctica/estudiantes un tiempo de ejecución de 30 minutos.

Además, conocer el estado de las distintas particiones permite tomar la decisión de qué tipo de recursos están disponibles al momento de querer ejecutar/lanzar una tarea.

Solución

Punto 1

El script básico es:

```
#!/bin/bash

#SBATCH -J ejercicio-stress
#SBATCH -p slims
#SBATCH -c 1

stress -c 1
```

Punto 2

El comando en principio no debería finalizar. Debido a los límites de uso para los usuarios bajo *Slurm*, las tareas podrán durar un máximo de 30 días para cuentas estándar, y para las cuentas de estudiantes 30 minutos. Pasado ese tiempo la tarea será cancelada por el gestor de recursos.

Para obtener información de las tareas en ejecución o en la cola de espera se puede utilizar el comando `squeue`.

Para obtener detalles de una tarea, se puede ejecutar el comando `scontrol show job $JOBID`.

> Solo se puede obtener información de los trabajos del propio usuario.
> No es posible ver los trabajos de otros usuarios.

En el caso de necesitar cancelar alguna de las tareas que estén en espera o ejecución, se utiliza el comando `scancel $JOBID`.

Punto 3

Para conocer el estado del *Cluster* y de sus recursos se ejecuta el comando `sinfo`, el que mostrará un resumen de las particiones, sus nodos y el estado de los mismos.

> También es posible visitar la página web Dashboard.

Ejercicio 3

1. Crea un *script* para lanzarlo con `sbatch`, teniendo en consideración:
   - La partición a utilizar es *slims*.
   - Reserva un nodo completo.
   - Ejecutar el comando `stress -c 40 -t 15m`.
2. Verifica en qué nodo se está ejecutando tu tarea y acceder mediante `ssh` al nodo para ejecutar `htop`.
3. ¿Cuántos recursos de *CPU* está utilizando la tarea? Puedes compartir tu pantalla o realizar una captura de pantalla.
4. ¿Cómo sería la manera correcta de lanzar la tarea con el fin de hacer un uso óptimo de los *cores* disponibles en el nodo?
5. Compara y explica el uso de *CPU* entre los puntos 3 y 4.

Objetivo

Ejecutar un comando que haga sobreuso de los recursos de los nodos, verificar su comportamiento realizando una tarea similar que utilice de manera óptima los recursos de los nodos.

Solución

Punto 1

Script:

```
#!/bin/bash
#SBATCH -J sobreutilizacion
#SBATCH -p slims
#SBATCH -n 20
#SBATCH --ntasks-per-node=20

stress -c 40 -t 15m
```

Posterior a eso, se debe ejecutar con `sbatch $NOMBRE_SCRIPT`

Punto 2

Se puede verificar nuestra tarea, verificando la cola de trabajos con `squeue`.

También se puede ver el detalle utilizando `scontrol show job $JOBID`

Una vez que hayamos identificado el nodo en donde se encuentra nuestra tarea en ejecución, bastará ejecutar `ssh $NODO`. Deberemos ingresar la contraseña de nuestro usuario.

Finalizamos ejecutando `htop` desde el nodo en donde se encuentra nuestra tarea.

Punto 3

Para verificar nuestra tarea, y desde `htop`, podemos filtrar por nuestro nombre de usuario, presionando la tecla *U* de nuestro teclado y seleccionando nuestro usuario.

Veremos que los procesos utilizan *50%* o menos en la columna *CPU*, debido a que los nodos *slims* tienen *20 cores* cada uno y nuestro comando solicita un total de *40*. Entonces, cada proceso y debido a los cambios de contexto, no existe un uso óptimo de los recursos.

Punto 4

Para un uso óptimo, nuestro script debería ser:

```
#!/bin/bash
#SBATCH -J sobreutilizacion
#SBATCH -p slims
#SBATCH -n 20
#SBATCH --ntasks-per-node=20

stress -c 20 -t 15m
```

Haciendo coincidir el parámetro de *SBATCH -n 20* con el parámetro en el comando `stress -c 20`.

Posterior a eso, bastará ejecutar nuestro script con `sbatch $NOMBRE_SCRIPT`.

Punto 5

Si verificamos el comportamiento de nuestro nuevo script en el nodo en donde se esté ejecutando. Veremos que el comando `htop` presenta un uso óptimo cercano al *100%* de cada *CPU*.

Ejercicio 4

1. Crea un *script* para lanzarlo con `sbatch`, teniendo en consideración:
   - La partición a utilizar es *slims*.
   - Reserva un único *core*.
   - Debe enviar un correo electrónico cuando la tarea cambie de estado.
   - El comando a ejecutar es `stress -m 1 --vm-bytes 2048M -t 15m`.
2. ¿Qué ocurre con la ejecución? ¿Cuál es la razón?

3. Modifica el script para ejecutar la tarea.
4. ¿Cuántos recursos de *RAM* está utilizando la tarea?

Objetivo

Obtener un error de tipo *OOM* para posteriormente modificar nuestra tarea asignando los recursos necesarios para su correcta ejecución.

Además, podremos enviar correos electrónicos a la cuenta que se indique en el *script* a ejecutar. Recibiendo notificaciones de cuando nuestra tarea entre en ejecución, finalización y cancelación.

Solución

Punto 1

Script:

```
#!/bin/bash
#SBATCH -J uso-memoria
#SBATCH -p slims
#SBATCH -c 1
#SBATCH --mail-user=USER@HOST
#SBATCH --mail-type=ALL
#SBATCH -o uso-memoria_%j.out
#SBATCH -e uso-memoria_%j.err

stress -m 1 --vm-bytes 2048M -t 15m
```

Ejecutar con `sbatch $NOMBRE_SCRIPT`.

Punto 2

Una vez que nuestra tarea se ejecute, presentará un fallo, el que se verá reflejado en el archivo `uso-memoria_%j.err`.

El mensaje será similar a:

```
slurmstepd: error: *** JOB 23827949 ON cn094 CANCELLED AT 2022-02-23T11:53:05
***
slurmstepd: error: Detected 1 oom-kill event(s) in StepId=23827949.batch.
Some of your processes may have been killed by the cgroup out-of-memory
handler.
```

Este error se produce debido a que no hemos asignado la memoria suficiente necesaria para la ejecución del comando `stress -m 1 --vm-bytes 2048M -t 15m` que requiere al menos 2Gib.

Por defecto *Slurm* asigna 1GB de *RAM* si no se especifica otro valor.

Punto 3

Script:

```
#!/bin/bash
#SBATCH -J uso-memoria
#SBATCH -p slims
#SBATCH -c 1
#SBATCH --mail-user=USER@HOST
#SBATCH --mail-type=ALL
#SBATCH --mem-per-cpu=2300
#SBATCH -o uso-memoria_%j.out
#SBATCH -e uso-memoria_%j.err

stress -m 1 --vm-bytes 2048M -t 15m
```

En esta oportunidad hemos asignado un poco más de 2Gb de RAM.

Punto 4

Nuestra tarea se encontrará correctamente ejecutada, haciendo el uso adecuado de la memoria que hemos asignado a la tarea *Slurm*.

Ejercicio 5

1. Descarga el siguiente código Python: n-queens-problem-3.py en tu directorio de trabajo.
2. Crea un *script* para lanzarlo con `sbatch`, teniendo en consideración:
   - Utilizar la partición *slims*.
   - Cada trabajo reserva un único *core*.
   - Supondremos que cada trabajo reserva un total de *2300Mb* de *RAM*.
   - Ejecuta el código descargado con la última versión de *Python* disponible.
   - ¿Cuál es el resultado obtenido?

Objetivo

Con este ejercicio, se aprende a cargar módulos para su uso desde trabajos enviados a la cola del gestor de recursos.

Solución

Punto 1

Es posible descargar el código visitando la URL
https://raw.githubusercontent.com/acmeism/RosettaCodeData/master/Task/N-queens-p

roblem/Python/n-queens-problem-3.py o desde la línea de comando utilizando uno de los siguientes comandos:

```
wget
https://raw.githubusercontent.com/acmeism/RosettaCodeData/master/Task/N-queens-problem/Python/n-queens-problem-3.py
```

o

```
curl -o n-queens-problem-3.py
https://raw.githubusercontent.com/acmeism/RosettaCodeData/master/Task/N-queens-problem/Python/n-queens-problem-3.py
```

Punto 2

Podemos verificar las versiones disponibles de Python, ejecutando:

```
ml spider Python
```

Posterior a esto editaremos un script similar a:

```
#!/bin/bash
#SBATCH -J test-python
#SBATCH -p slims
#SBATCH -n 1
#SBATCH -c 1
#SBATCH --mem-per-cpu=2300
#SBATCH -o test-python_%j.out
#SBATCH -e test-python_%j.err
#SBATCH --mail-user=USER@HOST
#SBATCH --mail-type=ALL


ml Python/3.9.5

python n-queens-problem-3.py
```

El archivo de salida `test-python_%j.out` tendrá una salida similar a:

```
[(1, 1), (2, 5), (3, 8), (4, 6), (5, 3), (6, 7), (7, 2), (8, 4)]
[(1, 1), (2, 6), (3, 8), (4, 3), (5, 7), (6, 4), (7, 2), (8, 5)]
[(1, 1), (2, 7), (3, 4), (4, 6), (5, 8), (6, 2), (7, 5), (8, 3)]
...
[(1, 8), (2, 2), (3, 5), (4, 3), (5, 1), (6, 7), (7, 4), (8, 6)]
[(1, 8), (2, 3), (3, 1), (4, 6), (5, 2), (6, 5), (7, 7), (8, 4)]
[(1, 8), (2, 4), (3, 1), (4, 3), (5, 6), (6, 2), (7, 7), (8, 5)]
```

Ejercicio 6

1. ¿Cómo ejecutar un programa mediante *Slurm* que necesite utilizar la partición *gpus*?
   - ¿Qué opciones existen actualmente en el *Cluster*?
   - ¿Qué módulos se deben cargar para utilizar la partición *gpus*?

Objetivo

Conocer la forma correcta de ejecutar una tarea en la partición *gpus* y los módulos necesarios que se requieren para su correcta ejecución.

Solución

El uso de particiones *gpus* debe ser invocado, especificando los parámetros `-p` y `--gres`.

Además, se requiere cargar el módulo `fosscuda` y considerar que los módulos que se vayan a utilizar hayan sido compilados para su uso.

Para verificar si un módulo, podemos verificar si incluye *fosscuda* en su descripción:

Por ejemplo:

```
ml spider NAMD/2.13-mpi

----------------------------------------------------------------------------
  NAMD: NAMD/2.13-mpi
----------------------------------------------------------------------------
    Description:
      NAMD is a parallel molecular dynamics code designed for
high-performance simulation of large biomolecular systems.


    You will need to load all module(s) on any one of the lines below before
the "NAMD/2.13-mpi" module is available to load.

      fosscuda/2019b
      icc/2018.5.274-GCC-8.2.0-2.31.1  impi/2018.4.274
...
```

Entonces los parámetros de *Slurm* y los módulos a cargar en nuestro script se verán similar a:

```
#!/bin/bash
#---------------SLURM Script - NLHPC ----------------
#SBATCH -J namd
#SBATCH -p gpus
#SBATCH -c 1
#SBATCH --gres=gpu:1
```

```
#SBATCH --mem-per-cpu=4250
#SBATCH --mail-user=USER@HOST
#SBATCH --mail-type=ALL
#SBATCH -o namd_%j.out
#SBATCH -e namd_%j.err

ml purge

ml fosscuda/2019b

ml NAMD/3.0alpha9
...
```

Enlaces de Interés

Puedes leer más información en la Wiki del NLHPC como también utilizar nuestro Generador Scripts - NLHPC para facilitarte con la edición de los *scripts de Slurm* y sus distintos parámetros.