Curso Avanzado - Ejercicios

v.04.23

Introducción

A lo largo de esta sesión se realizarán ejercicios prácticos que permitirán adquirir conocimientos enfocados a usos más específicos en el Cluster Leftraru-Guacolda.

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
**************** MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
* BIENVENID@ A * MMMMMMMMMMMMMMMMMMMMNmhhhyhmMMMMMMMMMMMMMMMM NLHPC MMMMMMM
*   GUACOLDA   * MMMMMMMMMMMMMMMMMy/-..``````.oNMMMMMMMMMMMMMMMMMMMMMMMMMMM
*   LEFTRARU   * MMMMMMMMMMMMMMMMMy:---:--.--::+MMMMMMMMMMMMMMMMMMMMMMMMMMMM
**************** MMMMMMMMMMMMMMMMdo://:+o:///+::yMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMNs+-/://:o/+/:+`::oNMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMd+`  ` .-/+ssso:-`   `:yNMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMh-.````.ososss+o+s+-     `.NMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMd.``..` :-sssososss--      `oMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMy+`      -/o-ss+ooos-/o/      `:odNMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMN:        ..` .oso+/` ```   `       .omMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMd.       ` `-/+so/+s/:.`/+/o:`        `sNMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMNo.+`:` ``-//+:/:--....+ssos:`      . `dMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMy++/::/:-::-:-:::::::---/oooss+.    .o hMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMNo//////:::::-:::::----.-+sssssssss:    -+MMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMo////////:-------/yho.  `+sssosssss:  :NMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMm::::::--..--:o/-+yhhh++ys+-+s/sssssss.oMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMN:-----......:d:.-/yss+o-:/o+ .::++//:::+MMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMs-------`    `+   /:os:/-    `   .::::::::dMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMN:/ossss-```      `+oss+so+o     `hd---....:NMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMssosssss/`.....``  // .+` +.     yMMy.----:/oMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMM+sosss/.--::       /: .s  o.     mMMMh:osssodMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMysssss-`-:///`      +:..s  o-     hMMMMNoossss+mMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMdoss+/-.--:///-   `.+. `o  :-    `/sNMMMMy/osssoNMMMMMMMMMM
MMMMMMMMMMMMMMMMMMosso+s-.----//:  ```+` `o  /:    .:`+yNMMN+sssssyMMMMMMMMMM
MMMMMMMMMMMMMMMMMhssssss:.-----::    `:  `:  ::    ./+:.+ohM+ssssooNMMMMMMMMM
MMMMMMMMMMMMMMMMMossssssoMy------`    .-  `:  ::     :+++/--o/sso+:-oMMMMMMMMM
M*KID*MMMMMMMMMm+ssss/NMMNo---.    `--:osos+::-    ./++++/-.-.....:MMMMMMMMM
MMMMMMMMMMMMMMMd+//://+MMMMMNh.`    `/ Guacolda/     -:/++++:``....:MMMMMMMMM
MMMMMMMMMMMMMMy/+oo/-sMMMMMMo``    -:/:/+//:/+s`      :::++++/`....:MMMMMMMMM
MMMMMMMMMMMMMMy+dmmmMMMMMhhhhhhhhhyhhdhhh:`yhhdmmdhyyssyhdddmmmmddmMmmNMMMMM
MMMd:yMMMMMMMN  mMMMd``mMMMMMMMMMN. 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 -NMMMMMMMMNoy
Md -MMMMm: o`.MMMMy -MMMMMMMMMo :dddmmm  NMMMM+ .ohNMMMMMMN` hMMMMMMMd+.+N
M+ oMMMMMMo  :MMMM: oNdys++/sM- sMMMMMN  mMMMM+ /MMMMMMMMMN. yMMMmy+.-sNMM
M/ yMMMMMMMdohMMMM- `.:/osssmM/.dMMMMMM/:mMMMMy -MMMMMMMMMMd:``..-+ymMMMMM
MMNMMMMMMMMMMMMMMMMMMNMMMMMMMMMMMMMMMMMMMMMMMMMMMMmNMMMMMMMMMMMMMMMMMMMMMMMMMM

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.

Ejemplo de Script básico

Los siguientes ejercicios requerirán que los usuarios puedan lanzar tareas al Gestor de Tareas, y que los usuarios puedan crear sus propios *scripts*.

El presente ejemplo, es un script básico para tomar a modo de referencia, el cual podrán copiar de ser necesario para ejecutar de manera exitosa los ejercicios que se verán durante la presentación.

Script

Cuando se quiera editar un script, se puede utilizar un editor como **VIM**, **Nano** o su preferido.

El contenido del script debe ser similar como el siguiente ejemplo:

```
#!/bin/bash
#SBATCH -J ejemplo
#SBATCH -p slims
#SBATCH -n 1
#SBATCH -c 1
#SBATCH -o archivo_%j.out
#SBATCH -e archivo_%j.err
#SBATCH --mail-user=foo@example.com
#SBATCH --mail-type=ALL

sleep 10
```

Ejecución de Script

Una vez que haya editado su script, deberá lanzarlo, reemplazando *test.sh* por el nombre de su propio script:

```
# sbatch test.sh
```

Una vez enviado a la cola de ejecución, se obtendrá un mensaje similar a:

```
Submitted batch job 24232120
```

Ejercicios

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

Ejercicio 1

1. Descarga el siguiente código OpenMP: pi_omp.c en tu directorio de trabajo.
2. Compile el código con el comando `icc pi_omp.c -o pi_omp -qopenmp`
3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones:
    - La partición a lanzar es slims.
    - Ejecute 1 proceso.
    - Esta tarea debe tener 20 cpu asignadas.
    - Cada CPU requiere 2300MB de memoria RAM.
    - El binario a ejecutar es `pi_openmp`.
4. Ejecuta el script en el gestor de tareas y observa el archivo de salida.
5. Repite el ejercicio variando el número de cpu asignadas ¿qué diferencia es posible ver?

Objetivo

- Compilar el código OpenMP.
- Ejecutarlo mediante Slurm.
- Variar parámetros en la tarea Slurm para verificar y comparar el comportamiento.

Solución

Punto 1

Podemos descargar el código fuente vía web, y desde consola podemos ejecutar:

```
curl -o pi_omp.c
https://raw.githubusercontent.com/nlhpc-training/Curso-Avanzado/master/sou
rce/pi_omp.c
```

Punto 2

Para compilar el código descargado, ejecutaremos:

```
icc pi_omp.c -o pi_omp -qopenmp
```

El archivo binario para ejecutar es **pi_omp**.

Punto 3

Este es el script de Slurm que utilizaremos:

```
#!/bin/bash
##---------------SLURM Parameters - NLHPC ----------------
#SBATCH -J ejercicio1
#SBATCH -p slims
#SBATCH -n 1
#SBATCH -c 20
#SBATCH --mem-per-cpu=2300
#SBATCH --mail-user=foo@bar.com
#SBATCH --mail-type=ALL
#SBATCH -o ejercicio1_%j.out
#SBATCH -e ejercicio1_%j.err
./pi_omp
```

Punto 4

Una vez editado nuestro script, lo guardaremos con el nombre **job_script.sh** y lo ejecutaremos con:

```
sbatch job_script.sh
```

Si verificamos la salida en el archivo **ejercicio1_23903272.out** o equivalente, obtendremos un mensaje similar a:

```
Hay 20 hilos en ejecución

Pi es aproximadamente 3.1415926535897922, el error cometido es 0.0000000000000009

Tiempo de ejecución: 1.395446 segundos
```

Punto 5

Si editamos nuestro script y modificamos el parámetro **-c 10** obtendremos un resultado similar a:

```
Hay 10 hilos en ejecución

Pi es aproximadamente 3.1415926535898007, el error cometido es 0.0000000000000075

Tiempo de ejecución: 2.785664 segundos
```

Se puede observar que el tiempo se duplica al disminuir a la mitad el número de recursos.

Y si editamos nuestro script y modificamos el parámetro **-c 5** obtendremos un resultado similar a:

```
Hay 5 hilos en ejecución

Pi es aproximadamente 3.1415926535898193, el error cometido es 0.0000000000000262

Tiempo de ejecución: 5.535774 segundos
```

Se puede observar que el tiempo se vuelve a duplicar al disminuir a la mitad el número de recursos.

Ejercicio 2

1. Descarga el siguiente código MPI: pi_mpi.c en tu directorio de trabajo.
2. Compile mediante el comando `mpiicc pi_mpi.c -o pi_mpi`
3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones:
    - La partición a lanzar es slims.
    - Ejecuta 20 procesos.
    - Deben entrar 10 procesos por cada nodo.
    - Supón que cada CPU requiere 2300MB de memoria RAM.
    - El binario a ejecutar es `pi_mpi`.
4. Ejecuta el script en el gestor de tareas y observa el archivo de salida.
5. Cambia el número de procesos ¿qué diferencias puedes notar?

Objetivo

- Compilar el código MPI.
- Ejecutarlo mediante Slurm.
- Variar parámetros en la tarea Slurm para verificar y comparar el comportamiento.

Solución

Punto 1

Podemos descargar el código desde la Web o desde la línea de comando con:

```
curl -o pi_mpi.c
https://raw.githubusercontent.com/nlhpc-training/Curso-Avanzado/master/source/pi_mpi.c
```

Punto 2

Para la compilación utilizaremos el comando:

```
mpiicc pi_mpi.c -o pi_mpi
```

Esto dejará un archivo ejecutable llamado **pi_mpi**.

Punto 3

El script para ejecutar el binario mediante el gestor de tareas queda de la siguiente manera:

```
#!/bin/bash
##---------------SLURM Parameters - NLHPC ----------------
#SBATCH -J ejercicio2
#SBATCH -p slims
```

```
#SBATCH -n 20
#SBATCH --ntasks-per-node=10
#SBATCH --mem-per-cpu=2300
#SBATCH --mail-user=foo@bar.com
#SBATCH --mail-type=ALL
#SBATCH -o ejercicio2_%j.out
#SBATCH -e ejercicio2_%j.err
srun pi_mpi
```

Punto 4

Una vez editado, guardaremos nuestro script con el nombre **ejercicio2.sh**, y lo ejecutaremos con:

```
sbatch ejercicio2.sh
```

Y en el archivo de salida obtendremos un mensaje similar a:

```
Hay 20 procesos en ejecución

Pi es aproximadamente 3.1415926535897980, el error cometido es 0.0000000000000049

Tiempo de ejecución: 2.734245 segundos
```

Punto 5

Si editamos nuestro archivo **ejercicio2.sh**, y cambiamos el número de procesos (parámetro **-n**), veremos variaciones en el tiempo de ejecución, el valor de Pi y el valor del error cometido.

Por ejemplo, con 10 procesos (**-n 10**) obtenemos:

```
Hay 10 procesos en ejecución

Pi es aproximadamente 3.1415926535898278, el error cometido es 0.0000000000000346

Tiempo de ejecución: 5.433985 segundos
```

Ejercicio 3

1. Descarga el siguiente código MPI_OpenMP híbrido: hello_hybrid.c en tu directorio de trabajo.
2. Compila utilizando el comando `mpiicc hello_hybrid.c -o hello_hybrid -qopenmp`
3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones:
    - La partición a lanzar es slims.
    - Ejecuta 2 procesos.
    - Cada proceso debe de tener 20 CPU asignadas.
    - Cada CPU requerirá 2300MB de memoria RAM.
    - El binario a ejecutar es `hello_hybrid`
4. Ejecuta el script en el gestor de tareas y observa el archivo de salida.

Objetivo

- Compilar el código híbrido (mpi/openmp)
- Lanzarlo mediante Slurm
- Identificar cómo se ejecuta entre distintos hilos y distintos nodos.

Solución

Punto 1

Descargamos el código desde la Web o desde la consola con el comando:

```
wget
https://raw.githubusercontent.com/nlhpc-training/Curso-Avanzado/master/sou
rce/hello_hybrid.c
```

Punto 2

Para compilar el código ejecutaremos el siguiente código:

```
mpiicc hello_hybrid.c -o hello_hybrid -qopenmp
```

El binario que podremos ejecutar se llama **hello_hybrid**.

Punto 3

Utilizaremos el siguiente script con el gestor de tareas:

```
#!/bin/bash
##---------------SLURM Parameters - NLHPC ----------------
#SBATCH -J ejercicio3
#SBATCH -p slims
#SBATCH -n 2
#SBATCH -c 20
```

```
#SBATCH --mem-per-cpu=2300
#SBATCH --mail-user=foo@bar.com
#SBATCH --mail-type=ALL
#SBATCH -o ejercicio3_%j.out
#SBATCH -e ejercicio3_%j.err
srun hello_hybrid
```

Punto 4

Grabamos nuestro archivo con el nombre **ejercicio4.sh** y ejecutaremos nuestra tarea con el comando:

```
sbatch ejercicio4.sh
```

Si verificamos el archivo de salida, veremos un resultado similar a:

```
Hello from thread 6 out of 20 from process 0 out of 2 on cn007
Hello from thread 11 out of 20 from process 0 out of 2 on cn007
Hello from thread 5 out of 20 from process 0 out of 2 on cn007
Hello from thread 18 out of 20 from process 0 out of 2 on cn007
…
Hello from thread 13 out of 20 from process 1 out of 2 on cn016
Hello from thread 9 out of 20 from process 1 out of 2 on cn016
Hello from thread 5 out of 20 from process 1 out of 2 on cn016
Hello from thread 12 out of 20 from process 1 out of 2 on cn016
```

Ejercicio 4

1. Descarga el siguiente código: average.py en tu directorio de trabajo.
2. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones:
    - La partición a lanzar es slims.
    - Ejecuta un proceso.
    - Este proceso debe tener asignados 2 CPU.
    - Cada CPU requiere 2300MB de memoria RAM.
    - Utiliza una versión reciente de Python.
    - La dimensión del arreglo es de 100 y deben entrar de a 10 simultáneos
3. ¿Qué has obtenido al ejecutar la tarea?

Objetivo

- Descargar script de Python
- Ejecutarlo mediante el gestor Slurm
    - Utilizar *Arreglos* de tareas
- Identificación de archivos de salidas en otras carpetas

Solución

Punto 1

Descargamos el código desde la Web o con el siguiente comando:

```
wget
https://raw.githubusercontent.com/ialab-puc/cluster/master/doc/samples/array/avera
ge.py
```

Punto 2

Hemos descargado un código Python por lo que utilizaremos el siguiente script para ejecutarlo mediante el gestor de tareas:

```
#!/bin/bash
##---------------SLURM Parameters - NLHPC ----------------
#SBATCH -J ejercicio4
#SBATCH -p slims
#SBATCH -n 1
#SBATCH -c 2
#SBATCH --mem-per-cpu=2300
#SBATCH --mail-user=foo@bar.com
#SBATCH --mail-type=ALL
#SBATCH --array=1-100%10
#SBATCH -o out/ejercicio4_%A_%a.out
#SBATCH -e out/ejercicio4_%A_%a.err
```

```
ml  Python/3.8.2
python average.py $SLURM_ARRAY_TASK_ID
```

Antes de ejecutar la tarea crearemos la carpeta **out** indicada en los parámetros **-o** y **-e** donde serán almacenados los archivos de salida.

```
mkdir out
```

Punto 3

Ejecutamos nuestro script con:

```
sbatch ejercicio4.sh
```

Y una vez finalizado, podemos ver en la carpeta **out** varios archivos de salida asociado al arreglo de tareas ejecutadas.

```
cd out
```

```
ls -l
```

```
-rw-r--r-- 1 foo users 0 abr 22 12:51 ejercicio4_23903375_1.err
-rw-r--r-- 1 foo users 9 abr 22 12:51 ejercicio4_23903375_1.out
-rw-r--r-- 1 foo users 0 abr 22 12:51 ejercicio4_23903375_2.err
-rw-r--r-- 1 foo users 9 abr 22 12:51 ejercicio4_23903375_2.out
-rw-r--r-- 1 foo users 0 abr 22 12:51 ejercicio4_23903375_3.err
-rw-r--r-- 1 foo users 9 abr 22 12:51 ejercicio4_23903375_3.out
-rw-r--r-- 1 foo users 0 abr 22 12:51 ejercicio4_23903375_4.err
-rw-r--r-- 1 foo users 9 abr 22 12:51 ejercicio4_23903375_4.out
-rw-r--r-- 1 foo users 0 abr 22 12:51 ejercicio4_23903375_5.err
-rw-r--r-- 1 foo users 9 abr 22 12:51 ejercicio4_23903375_5.out
…
```

Y el contenido de los archivos de salida será similar a:

```
cat *.out
```

```
50003.35
50069.46
50166.41
50055.29
50059.69
50016.59
49969.48
…
```

Ejercicio 5

1. Descarga los siguientes archivos en tu directorio de trabajo: mulBy2.cu y Makefile
2. Teniendo en cuenta que es un código en Cuda y que tenemos un Makefile, mediante el comando `make` compila el código en tu directorio de trabajo
3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones:
    - La partición a lanzar es gpus.
    - Ejecuta 1 proceso.
    - Cada proceso debe de tener 1 CPU.
    - Cada CPU requiere 2300MB de memoria RAM.
    - El binario a ejecutar es `mulBy2`
4. ¿Qué obtienes al ejecutar el script?

Objetivo

- Compilar utilizando las librerías compatibles con CUDA
- Generar un script que permita ejecutarlo en el cluster en la partición **GPUS**
- Verificar el resultado obtenido

Solución

Punto 1

Descargaremos los archivos desde la Web o con los siguientes comandos:

```
wget https://raw.githubusercontent.com/nlhpc-training/Curso-Avanzado/master/source/mulBy2.cu
```

```
wget https://raw.githubusercontent.com/nlhpc-training/Curso-Avanzado/master/source/Makefile
```

Punto 2

Al ser este un archivo **CUDA** para compilarlo necesitaremos ejecutar lo siguiente en la carpeta contenedora de los archivos descargados en el punto anterior:

```
ml fosscuda

make
```

Esto cargará las librerías necesarias y generará el binario llamado **mulBy2**. Este binario es el que deberemos ejecutar mediante SLURM.

Punto 3

Ejecutaremos nuestro binario en la partición **gpus**:

```
#!/bin/bash
##---------------SLURM Parameters - NLHPC ----------------
#SBATCH -J ejercicio5
#SBATCH -p gpus
#SBATCH -n 1
#SBATCH --gres=gpu:1
#SBATCH -c 1
#SBATCH --mem-per-cpu=2300
#SBATCH --mail-user=foo@bar.com
#SBATCH --mail-type=ALL
#SBATCH -o ejercicio5_%j.out
#SBATCH -e ejercicio5_%j.err
ml purge
ml fosscuda/2019b
./mulBy2
```

Y para ejecutarlo lo lanzaremos con:

```
sbatch ejercicio5.sh
```

Punto 4

El resultado obtenido corresponde a múltiplos del número 2.

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.