Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2523 – Sistemas Distribuidos (II/2017)

Actividad 2

Lunes 25 de Septiembre

1. Objetivo

- Comprender el funcionamiento de *Slurm*, un *Scheduler* que maneja los recursos de un *cluster* Linux, a través del uso y edición de *scripts bash*.

2. Slurm

Slurm (*Simple Linux Utility for Resource Management*) es un *scheduler* que maneja una cola de trabajos y los asigna a los recursos que tiene disponible. Para poder funcionar, *Slurm* utiliza un nodo de administración en el que corre un proceso de control (`slurmctld`) y cada nodo que puede procesar trabajo utiliza un un *slurm daemon* `slurmd`. Gracias a esto, se pueden correr los comandos de *Slurm* en cualquier nodo. A continuación hay una lista de comandos que pueden resultar útiles. La descripción completa, y otros comandos disponibles pueden encontrarse en `https://slurm.schedmd.com/man_index.html`:

- `sbatch [options] script [args...]`, se usa para entregar un *script* bash a *Slurm*. Esta ejecución termina cuando se ha logrado transferir el *script* al controlador y se le ha asignado un *ID* de trabajo o *Job ID*.
- `squeue [OPTIONS...]`, se usa para obtener una lista de los trabajos que están actualmente en la cola de *Slurm*.
- `sinfo [OPTIONS...]` permite obtener información del estado de los nodos y particiones (conjuntos de nodos) disponibles.

3. Ejemplo

A continuación se muestra un *script* de *Slurm*. Cada línea que comienza con `#SBATCH` define un parámetro de la ejecución para el *script* bash. La descripción de cada opción se encuentra en la documentación[1].

```
1 #!/bin/bash
2 #SBATCH --job-name=trabajito
3 #SBATCH -t 0-2:00                          # time (D-HH:MM)
4 #SBATCH -o slurm-%a.out                    # STDOUT
5 #SBATCH -e slurm-%a.err                    # STDERR
6 #SBATCH --mail-type=END,FAIL               # notifications for job do    ne & fail
7 #SBATCH --mail-user=user@gmail.com         # send-to address
```

[1] `https://slurm.schedmd.com/sbatch.html`

```
8  #SBATCH --workdir=/user/iic2523/slurm/results   #cambiar esto /user/<algo>    /slurm
9  #SBATCH --array 1-100%10                        # 100 procesos, 10 simult'aneos
10
11 srun python3 ../average.py $SLURM_ARRAY_TASK_ID
```

Los parámetros relevantes son:

- `-e file_name` y `-o file_name`: define el nombre del archivo de output y el archivo de log de errores respectivamente.
- `--workdir=path_to_folder`: define la carpeta de trabajo del *script*. Se puede definir con un path absoluto o relativo. Pero, si se desconoce la carpeta de inicio, es mejor definir un camino absoluto.
- `--array=indices`: envía un arreglo de trabajos al *scheduler* con parámetros idénticos. En el ejemplo, correrá 100 trabajos, con índices de 1 a 100. Añadir *%10* limita a que se corran 10 trabajos simultáneos de los 100 totales.

4. Descripción Actividad

Para esta actividad debe conectarse con su cuenta al *cluster* GRIMA. Tienen a su disposición *scripts* realizados por los ayudantes. Estos archivos se encuentran en el cluster, en el directorio `/user/iic2523/slurm`:

- `array.sh`: código base en bash que se utilizará para añadir nuevos trabajos a la cola de *Slurm*.
- `average.py`: programa en python3 que se ejecutará en los nodos del *cluster*. Inicialmente, genera N números aleatorieos y calcula su promedio. N es el parámetro que recibe el programa, y en este ejemplo será el *Task ID* del trabajo que lo ejecuta (ver línea 12 del ejemplo).
- `combine.py`: código en python3 que reúne los diferentes resultados para combinarlos.

Primero, deben copiar los scripts en sus *home*s:

```
1 cp -r /user/iic2523/slurm ~/
```

Para editar los archivos, pueden utilizar *vim*, *nano*, o bien copiarlos a sus computadores y luego volverlos a subir:

```
1 mkdir slurm
2 scp hercules.ing.puc.cl:/user/<algo>/slurm/* ./slurm
3 scp ./slurm/* hercules.ing.puc.cl:/user/<algo>/slurm/
```

Para ejecutar el ejemplo (en el cluster):

```
1 cd slurm
2 sbatch array.sh
```

Para ver el estado de sus trabajos (en el cluster):

```
1 squeue
```

Una vez que sus trabajos hayan terminado, en el directorio `results` encontrarán varios archivos con los resultados de cada trabajo. Para combinar los resultados (promediar todos los números en los archivos), deben ejecutar:

```
python3 combine.py
```

Su trabajo es modificar estos *scripts* para encontrar números primos. Deben crear el programa *primes.py* que, al igual que *average.py*, recibirá como parámetro el *TASK ID* asignado por *Slurm*. Cada instancia de su programa deberá trabajar en un conjunto disjunto de 100 números (0-99, 100-199, ...), e imprimir todos los números primos que encuentre en dicho rango. También deberán modificar *combine.py* para combinar todos los números encontrados en un único archivo.

5. Análisis

A continuación, deberán responder las siguientes preguntas. Pueden investigar en internet para dar sus respuestas, pero deben indicar sus fuentes.

1. Comparar funcionamiento de su aplicación de manera local usando uno de los nodos disponibles (`titan`, `tripio`, `caleuche`, ó `trauco`), versus la versión distribuida usando *slurm*. ¿Qué diferencias y semejanzas puede apreciar en cuanto a tiempo y uso de recursos? ¿Por qué podría existir una diferencia en tiempo de ejecución al usar Slurm?

2. Explicar cómo se podría modificar el script para utilizar una cantidad fija de nodos.

3. Describir cómo Slurm distribuye los recursos del cluster.

6. Entrega

Debe entregar, a través de dos cuestionarios disponibles en SIDING, un archivo *.zip* que incluya los *scripts* utilizados y un archivo **PDF** con su análisis.

Los *scripts* deben entregarlos **hoy** (13:00) y el análisis lo pueden entregar hasta el **miércoles 27 de septiembre, 13:00**.