



**EDUCACIÓN CON
RESPONSABILIDAD
SOCIAL**

Universidad de Colima

Facultad de Telemática - Ingeniería de Software

Práctica 5: Ejecución de un programa con

MPI en el clúster KNL

Programación paralela

Profesor: PhD Juan Manuel Ramírez Alcaraz

Estudiante:

Gilberto Felipe Ramírez García | 5K

Colima, Colima 18 de octubre de 2019

ÍNDICE

ÍNDICE

INTRODUCCIÓN

DESARROLLO

1. Elaborar el programa a ejecutar en c localmente.
2. Copiar el programa por scp al clúster.
3. Compilar el programa y crear el ejecutable.
4. Copiar el script para enviar el trabajo a la cola (queue) correspondiente.
5. Ejecutar el script desde la carpeta personal.
6. Capturar los resultados con los diferentes nodos.
 - 1 nodo y 2 procesadores
 - 3 nodos y 5 procesadores
 - 10 nodos y 10 procesadores

OBSERVACIONES Y CONCLUSIONES

REFERENCIAS

INTRODUCCIÓN

- Elaborar un reporte del proceso para ejecutar el programa Hola Mundo con MPI en el clúster KNL del LNS de la BUAP.
- Probar la ejecución con diferente número de nodos y procesadores.
- Agregar capturas de pantalla del proceso.
- Recuerda registrar los problemas que te encuentres en el proceso.
- El reporte debe estar en el formato de práctica acordado para la clase.

El objetivo de esta práctica es repasar y familiarizarnos con el procedimiento, comandos, etc. para ejecutar programas sencillos utilizando la librería MPI (Message Passing Interface). En esta ocasión vamos a utilizar el clúster KNL de la BUAP.

DESARROLLO

Seguiré los siguientes pasos para ejecutar el programa en el clúster. Pueden existir otros. En este caso, sigo el tutorial propuesto en la clase para hacerlo en *c*.

1. Elaborar el programa a ejecutar en *c* localmente.
2. Copiar el programa por scp al clúster.
3. Compilar el programa y crear el ejecutable.
4. Copiar el script para enviar el trabajo a la cola (*queue*) correspondiente.
 - a. Ejecutar el script con 2 nodos y 2 procesadores.
 - b. Ejecutar el script con 3 nodos y 5 procesadores.
 - c. Ejecutar el script con 10 nodos y 10 procesadores.
5. Ejecutar el script desde la carpeta personal.
6. Capturar los resultados con los diferentes nodos.

1. Elaborar el programa a ejecutar en *c* localmente.

```
#include <stdio.h> // librería estándar de c
#include <mpi.h>    // librería de MPI

int main(int argc, char *argv[]) // función principal y declaración variables
```

```

/*
    argc Puntero al número de argumentos
    argv Puntero al vector de argumentos
*/

{
    int this_proc, total_procs;

    /* Aquí comienza el paralelismo */

    MPI_Init(&argc, &argv);
    /* Función que inicia una sesión MPI.
       Es la primera función que se debe ejecutar
    */

    MPI_Comm_size(MPI_COMM_WORLD, &total_procs);
    /* Función que determina el número total de procesos. */

    MPI_Comm_rank(MPI_COMM_WORLD, &this_proc);
    /* Función que determina el identificador (rank) del proceso actual. */

    printf("Hola mundo! Soy el proceso nro. %d de un total de %d procesos.\n",
this_proc, total_procs);
    /* Esta función de C para imprimir en pantalla. */

    MPI_Finalize();
    /* Esta función termina la sesión MPI.*/
}

```

2. Copiar el programa por scp al clúster.

```

gilberto@gilberto-Satellite-U940:~$ scp /home/gilberto/Documents/ParallelProgram
ming/miprograma.c curso11@148.228.4.17:/mnt/zfs-pool/home/curso11

```

Utilicé el comando scp para copiar el programa.c local a la carpeta personal del clúster.

El comando scp (*Secure Copy*) tiene como argumentos /directorio/local usuario@servidor:/directorio/remoto.

3. Compilar el programa y crear el ejecutable.

Para compilar primero activamos el módulo de MPI para C.

```
gilberto@gilberto-Satellite-U940:~$ ssh curso11@148.228.4.17
curso11@148.228.4.17's password:
Last login: Thu Oct 17 19:02:32 2019 from 187.204.100.213
[curso11@rogueone ~]$ module load Compilers/Parallel-Studio-XE-2018
```

Luego, ejecutamos el comando

```
[curso11@rogueone ~]$ mpiicc miprograma.c -o miprograma
```

4. Copiar el script para enviar el trabajo a la cola (*queue*) correspondiente.

Debemos seguir las políticas del clúster de la BUAP. Estas nos piden que utilicemos un script para mandar a la cola de trabajo correspondiente. Este paso podría omitirse de ser nuestro clúster. Este es el script. Explico dentro del mismo los comandos.

```
#!/bin/bash
/* #! Esta secuencia de caracteres se llama shebang y hace que el
program loader esa línea inicial como una dirección de intérprete.
/bin/bash indica al SO que vamos a usar el shell bash. El shell es un
intérprete de línea de comandos que trae su propia interfaz de línea
de comandos.
*/

/* Explicamos el bloque de comando #PBS
PBS (Portable Batch System) es un software que realiza agendar
trabajos y hacerlos más eficientes.
#PBS es el comando para agendar los trabajos
*/
#PBS -l nodes=2:ppn=1,walltime=00:00:10
/* El comando especifica el número de nodos y de procesadores/cores a
utilizar por MPI, y el marco de tiempo para realizar el trabajo. */

#PBS -N mpi_gil_programa
/* El comando identifica el nombre del programa/trabajo (job). */
```

```

#PBS -q staff
/* El comando especifica el nombre de la cola de procesamiento a
donde se formará nuestro programa.*/

#PBS -d /mnt/zfs-pool/home/curs011/hello-world
/* El comando especifica el directorio donde se encuentra el programa
de MPI. */

#PBS -o mpi_curso_programa.log
/* El comando especifica el nombre del archivo de la salida/output
(-o) estándar del ejecutable MPI. En este caso, es mi archivo de
historial mpi_curso_programa.log */

#PBS -j oe
/* Este comando instruye que la salida de errores e se una (join -j)
con el output o, es decir, con mpi_curso_programa.log */

#PBS -V
/* Este comando especifica la variable de entorno para PBS */

#PBS -S /bin/bash
/* El comando con la opción -S declara la ruta y el nombre del
lenguaje del script usado para interpretar el script, en este caso,
bash. */

source $MODULESHOME/init/bash
/* El comando indica la variable de entorno de bash. */

module purge
/* El comando descarga todos los módulos de MPI.*/

module load Compilers/Parallel-Studio-XE-2018
/* El comando carga el compilador de MPI para C.*/

NPROCS=`wc -l < $PBS_NODEFILE`
/* El comando asigna a la variable NPROCS la variable $PBS_NODEFILE
que contiene el listado de todos los nodos.
El comando wc -l (w de write, c de counter, -l es la opción de

```

líneas) imprime el número de líneas del input de \$PBS_NODEFILE.*/

```
cat ${PBS_NODEFILE} | sort -u > $PBS_O_WORKDIR/machines.LINUX
/* El comando muestra la variable de entorno $PBS_NODEFILE y la
entuba al comando sort -u. Este comando va a ordenar la lista de
nodos excluyendo nodos repetido. El output va a ser redireccionado al
archivo $PBS_O_WORKDIR/machines.LINUX */
```

```
mpirun -np $NPROCS -machinefile machines.LINUX ./miprograma >
mpi_miprograma.out
/* El comando ejecuta programas seriales o paralelos en MPI.
-np indica el número de procesos.
$NPROCS la variable que contiene el número de nodos/procesos que se
van a ejecutar.
-machinefile especifica la ruta al archivo host que lista los nodos
del clúster, en este caso, machines.LINUX.
./miprograma ejecuta mi programa y redirecciona el output al archivo
mpi_miprograma.out */
```

Guardamos el script con el nombre `mpi_run.sh`.

5. Ejecutar el script desde la carpeta personal.

Se ejecuta el script `bash mpi_run.sh` usando el comando `qsub` que sirve para enviar nuestro job a una cola.

```
[curso11@rogueone hello-world]$ qsub mpi_run.sh
5881.rogueone
```

Nos regresa el id o número de proceso asignado a nuestra trabajo/programa.

Para ver el estado del job en la cola de procesos, utilizamos el comando `qstat`.

```
5881.rogueone      mpi_gil programa curso11      00:00:00 C staff
[curso11@rogueone hello-world]$
```

`qstat` nos regresa el estado de todos los procesos. En la lista encontramos el nuestro. La C significa completed. Como nuestro trabajo era tan pequeño, el cluster tardó menos de un segundo en realizarlo. Por eso, el tiempo aparece 00:00:00.

6. Capturar los resultados con los diferentes nodos.

Para ver la salida de nuestro proceso ejecutándose en los nodos, redireccionamos el output del script al archivo `mpi_miprograma.out`. Para ver el contenido de este archivo ejecutamos el comando `tail -f mpi_miprograma.out`.

Vamos a capturar diferentes resultados cambiando el número de nodos y procesadores que asignamos a nuestra tarea.

A. 1 nodo y 2 procesadores

```
5881.rogueone      mpi_gil programa curso11      00:00:00 C staff  
[curso11@rogueone hello-world]$
```

```
[curso11@rogueone hello-world]$ tail -f mpi_miprograma.out  
Hola mundo! Soy el proceso nro. 0 de un total de 2 procesos.  
Hola mundo! Soy el proceso nro. 1 de un total de 2 procesos.  
[
```

B. 3 nodos y 5 procesadores

```
[curso11@rogueone hello-world]$ qsub mpi_run.sh  
5882.rogueone
```

```
5882.rogueone      mpi_gil programa curso11      00:00:01 C staff
```

```
[curso11@rogueone hello-world]$ tail -f mpi_miprograma.out  
Hola mundo! Soy el proceso nro. 0 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 3 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 6 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 9 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 12 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 2 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 5 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 8 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 11 de un total de 15 procesos.  
Hola mundo! Soy el proceso nro. 14 de un total de 15 procesos.  
[
```

C. 10 nodos y 10 procesadores

```
[curso11@rogueone hello-world]$ qsub mpi_run.sh  
5884.rogueone  
[curso11@rogueone hello-world]$
```




```
5884.rogueone          mpi_gil_programa curso11          00:00:03 C staff
[curso11@rogueone hello-world]$
```

```
[curso11@rogueone hello-world]$ tail -f mpi_miprograma.out
Hola mundo! Soy el proceso nro. 38 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 48 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 58 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 68 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 78 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 88 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 98 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 99 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 56 de un total de 100 procesos.
Hola mundo! Soy el proceso nro. 76 de un total de 100 procesos.
```


OBSERVACIONES Y CONCLUSIONES

Observaciones al tutorial Manual-MPI_KNL-Corregido.


Los errores se encontraban en el archivo `mpi_corre.sh`. Para que sea más gráfico los señalamos con rojo en una captura de pantalla.

```
ARCHIVO SCRIPT TORQUE  mpi-corre.sh

#!/bin/bash

#PBS -l nodes=2:ppn=1,walltime=10:00:00
#PBS -N mpi_curso_programa
#PBS -q staff
#PBS -d /mnt/home/zfs-pool/home/cursoXX/mpi_curso 
#PBS -o mpi_curso_programa.log
#PBS -j oe
#PBS -V
#PBS -S /bin/bash

source $MODULESHOME/init/bash
module purge
module load Compilers/Parallel-Studio-XE-2018

NPROCS=wc -l < $PBS_NODEFILE 0 
cat ${PBS_NODEFILE} | sort -u > $PBS_O_WORKDIR/machines.LINUX

mpirun -np $NPROCS -machinefile machines.LINUX ./programa > mpi_curso_programa.out
```

En conclusión, en esta práctica reporté y repasé cómo se ejecuta un programa con MPI en el clúster KNL de la BUAP.

REFERENCIAS

Manual-MPI_KNL-Corregido. (16 octubre 2019). Tutorial dado por el profesor.

Tutorial Hola Mundo (C-MPI) para el KLN de la BUAP. Tutorial dado por el profesor.

PBS Grid Works. (2010). *PBS Professional User's Guide*. Altair Engineering. Recuperado el 17 de octubre de 2019 de <http://bit.ly/2BmTYTa>

Shebang. (s.f.). *Wikipedia*. Recuperado el 17 de octubre de 2019 de <http://bit.ly/2OY6AbA>

MPI Interfaz de paso de mensajes. (s.f.). *Wikipedia*. Recuperado el 17 de octubre de 2019 de <http://bit.ly/2MOjSoo>

Frequently Asked Questions on Environment Modules. (2019). *Read the docs*. Recuperado el 17 de octubre de 2019 de <http://bit.ly/32Aq6yV>