



**EDUCACIÓN CON
RESPONSABILIDAD
SOCIAL**

Universidad de Colima

Facultad de Telemática - Ingeniería de Software

Práctica 6: Suma de bloques de vector empleando paralelismo

Programación paralela

Profesor: PhD Juan Manuel Ramírez Alcaraz

Estudiante:

Gilberto Felipe Ramírez García | 5K

Colima, Colima 28 de octubre de 2019

ÍNDICE

ÍNDICE

INTRODUCCIÓN

DESARROLLO

1. Codificar el programa a ejecutar en C dentro del clúster.
2. Compilar el programa y crear el ejecutable
3. Codificar el script correspondiente
4. Ejecutar el script enviándolo a la cola con el comando qsub
5. Capturar los resultados con diferente número de procesadores y parámetros

Ejecutar el script con 2 procesadores

Ejecutar el script con 5 procesadores.

Ejecutar el script con 5 procesadores cambiando parámetros.

Ejecutar el script con 10 procesadores

Ejecutar el script con 100 procesadores

Ejecutar el script con 50 procesadores.

Ejecutar el script con 20 procesadores

OBSERVACIONES Y CONCLUSIONES

REFERENCIAS

INTRODUCCIÓN

- Elaborar y ejecutar un programa que genere un vector de n números aleatorios y que cada procesador solicitado realice la suma de un bloque de los n números en el clúster de la BUAP.
- n debe ser múltiplo del número de procesadores solicitados (Para que el tamaño del bloque sea un número entero).
- Probar la ejecución con diferente número de procesadores y tamaños del vector.
- 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.

DESARROLLO

Seguiré los siguientes pasos para ejecutar el programa de suma de vector en el clúster.

1. Codificar el programa a ejecutar en *C* dentro del clúster.
2. Compilar el programa y crear el ejecutable.
3. Codificar el script correspondiente.
4. Ejecutar el script enviándolo a la cola con el comando `qsub`.
5. Capturar los resultados cambiando el número de procesadores y de parámetros.

1. Codificar el programa a ejecutar en C dentro del clúster.

Programa suma.c

```
/*=====
LIBRERÍAS
=====*/

#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/*===== FIN LIBRERÍAS =====*/
```

```

// PROGRAMA PRINCIPAL
int main(int argc, char *argv[])
{
    // ENTRADA
    int this_proc, total_procs;
    int size_A = 10;
    int vector[size_A];

    // GENERAR ARRAY CON NÚMEROS ALEATORIOS
    for (int c = 0; c < size_A; c++)
    {
        vector[c] = rand()%11;
        printf("%d ", vector[c]);
    }

    /* Aquí comienza el paralelismo */

    // INICIO MPI
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &total_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &this_proc);

    // INICIALIZAR ARRAY SUMA
    int suma[total_procs];
    for (int x = 0; x < total_procs; x++)
    {
        suma[x] = 0;
    }

    // * FUNCIÓN BROADCAST DE MPI
    MPI_Bcast(&suma, total_procs, MPI_INT, 0, MPI_COMM_WORLD);

    // VARIABLES PARA FUNCIÓN RELOJ
    int block_size = size_A / total_procs;
    clock_t tini, tfin, total_t;

    // GENERAR SUMA DE BLOQUES (X PROCESADORES) Y TOMAR TIEMPO DE EJECUCIÓN
    tini = clock();
    for (int i = this_proc * block_size; i < (this_proc + 1) * block_size; i++)
    {
        suma[this_proc] += vector[i];
    }
    tfin = clock();

```

```

total_t = (double)(tfin - tini) / CLOCKS_PER_SEC;

// FINALIZAR MPI
MPI_Finalize();

// IMPRIMIR SALIDAS
printf("\nTiempo total de ejecución: %f segundos \n", total_t);

for (int i = 0; i < total_procs; i++)
{
    printf("La suma del procesador %d de %d procesadores = %d \n", i,
total_procs, suma[i]);
}

return 0;
}

```

Con esto queda explicado el código.

2. Compilar el programa y crear el ejecutable

Cargar el compilador de mpi para C

```
[curso11@rogueone sumaVector]$ module load Compilers/Parallel-Studio-XE-2018
```

Compilar el programa en C

```
[curso11@rogueone sumaVector]$ mpicc sumaVector.c -o sumaVector
```

3. Codificar el script correspondiente

En el reporte de la práctica anterior, explicamos línea por línea del script. Ahora, solo mostramos el código. Guardamos el script con el nombre `mpi_run.sh`.

```
[curso11@rogueone sumaVector]$ cat mpi_run.sh
#!/bin/bash

#PBS -l nodes=2:ppn=1,walltime=00:00:10
#PBS -N mpi_sumaVector
#PBS -q staff
#PBS -d /mnt/zfs-pool/home/curso11/sumaVector
#PBS -o mpi_sumaVector.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`
cat ${PBS_NODEFILE} | sort -u > $PBS_O_WORKDIR/machines.LINUX
mpirun -np $NPROCS -machinefile machines.LINUX ./sumaVector > mpi_sumaVector.out
[curso11@rogueone sumaVector]$
```

4. Ejecutar el script enviándolo a la cola con el comando qsub

Se ejecuta el script bash `mpi_run.sh` usando el comando `qsub` que sirve para enviar nuestro job a una cola. Nos regresa el id o número de proceso asignado a nuestra trabajo/programa.

```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
6335.rogueone
```

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

```
6335.rogueone      mpi_sumaVector  curso11      00:00:00 C staff
[curso11@rogueone sumaVector]$
```

`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.

5. Capturar los resultados con diferente número de procesadores y parámetros

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

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

a. Ejecutar el script con 2 procesadores

Este es una ejecución de control, para ver si el programa realiza bien la suma.

Parámetros: Array[10 n aleatorios entre 0 y 10].

La suma es correcta.

```
[curso11@rogueone sumaVector]$ tail -f mpi_sumaVector.out
897 802 765 992 1 521 220 380 729 969 897 802 765 992 1 521 220 380 729 969 Tardé 0
Duración en MS 0.000
Posición 0 = 3457
Posición 1 = 0
Tardé 0
Duración en MS 0.000
Posición 0 = 0
Posición 1 = 2819
^C
```

b. Ejecutar el script con 5 procesadores.

Esta es otra ejecución de control, para ver si el programa funciona con más de 2 procesadores.

Parámetros: Array[10 n aleatorios entre 0 y 10].

```
[curso11@rogueone sumaVector]$ cat mpi_run.sh
#!/bin/bash

#PBS -l nodes=5:ppn=1,walltime=00:01:00
#PBS -N mpi_sumaVector
#PBS -q staff
#PBS -d /mnt/zfs-pool/home/curso11/sumaVector
#PBS -o mpi_sumaVector.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`
cat ${PBS_NODEFILE} | sort -u > $PBS_O_WORKDIR/machines.LINUX
mpirun -np $NPROCS -machinefile machines.LINUX ./sumaVector > mpi_sumaVector.out
[curso11@rogueone sumaVector]$
```

```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
6342.rogueone
[curso11@rogueone sumaVector]$
```

```
6342.rogueone      mpi_sumaVector      curso11      00:00:00 C staff
[curso11@rogueone sumaVector]$
```

```
Activities Terminal dom oct 27, 22:38 curso11@rogueone:~/sumaVector
File Edit View Search Terminal Help
[curso11@rogueone sumaVector]$ cat suma.out
6 10 6 2 1 4 0 6 3 1 6 10 6 2 1 4 0 6 3 1 6 10 6 2 1 4 0 6 3 1 6 10 6 2 1 4 0 6 3 1
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 5
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 16
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 8
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 6
La suma del procesador 4 de 5 procesadores = 0

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 4
[curso11@rogueone sumaVector]$
```

c. *Ejecutar el script con 5 procesadores cambiando parámetros.*

Número de procesadores: 5.

Parámetros: array[1,000 n aleatorios entre 0 y 1,000].

```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
6483.rogueone
```

```
6483.rogueone mpi_sumaVector curso11 00:00:00 C staff
[curso11@rogueone sumaVector]$
```

```
cat suma.out
```



```
Activities  Terminal  dom oct 27, 22:52
curso11@rogueone:~/sumaVector

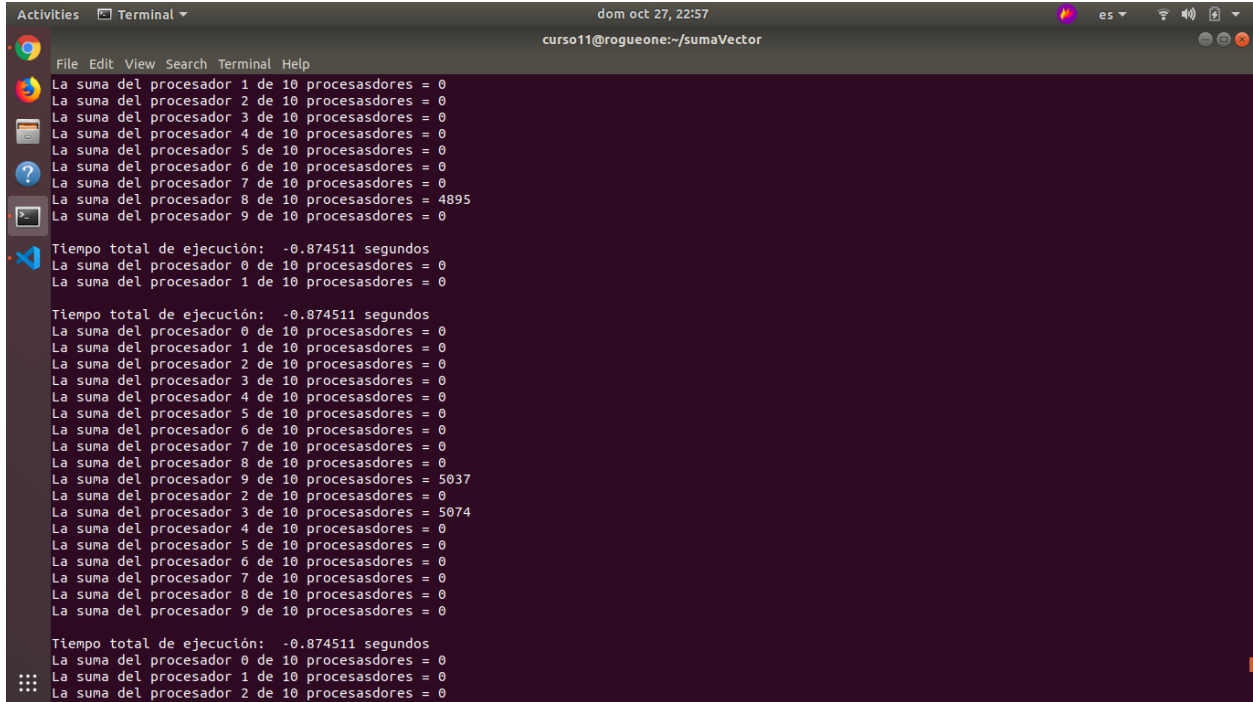
File Edit View Search Terminal Help
0 6 2 6 4 8 10 10 3 8 9 2 6 10 7 3 7 1 4 4 8 9 1 4 0 6 7 9 10 6 7 10 0 7 3 4 4 0 1 7 6 8 10 0 8 4 3 4 5 8 9 0 4 8 4 4 1
0 1 5 8 0 5 5 4 6 8 7 1 4 7 6 10 1 8 0 10 7 1 6 2 7 0 2 6 10 7 4 8 9 7 5 9 10 9 1 6 4 6 5 8 3 9 7 2 7 7 1 1 6 6 3 3 4 3
0 4 0 4 10 1 2 4 3 7 0 2 8 5 6 9 6 0 2 2 1 2 2 9 0 8 0 3 3 0 8 3 2 8 5 10 9 8 3 10 2 3 10 0 6 6 9 1 4 9 3 3 0 4 1 9
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 1123
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 978
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 1057
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 0
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 1028
La suma del procesador 4 de 5 procesadores = 0
Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 5 procesadores = 0
La suma del procesador 1 de 5 procesadores = 0
La suma del procesador 2 de 5 procesadores = 0
La suma del procesador 3 de 5 procesadores = 0
La suma del procesador 4 de 5 procesadores = 970
[curso11@rogueone sumaVector]$
```

d. Ejecutar el script con 10 procesadores

Número de procesadores: 10.

Parámetros: array[10,000 n aleatorios entre 0 y 10,000].

```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
6484.rogueone
[curso11@rogueone sumaVector]$
6484.rogueone      mpi_sumaVector      curso11      00:00:01 C staff
[curso11@rogueone sumaVector]$
```



```
Activities Terminal
dom oct 27, 22:57
curso11@rogueone:~/sumaVector

File Edit View Search Terminal Help

La suma del procesador 1 de 10 procesadores = 0
La suma del procesador 2 de 10 procesadores = 0
La suma del procesador 3 de 10 procesadores = 0
La suma del procesador 4 de 10 procesadores = 0
La suma del procesador 5 de 10 procesadores = 0
La suma del procesador 6 de 10 procesadores = 0
La suma del procesador 7 de 10 procesadores = 0
La suma del procesador 8 de 10 procesadores = 4895
La suma del procesador 9 de 10 procesadores = 0

Tiempo total de ejecución: -0.874511 segundos
La suma del procesador 0 de 10 procesadores = 0
La suma del procesador 1 de 10 procesadores = 0

Tiempo total de ejecución: -0.874511 segundos
La suma del procesador 0 de 10 procesadores = 0
La suma del procesador 1 de 10 procesadores = 0
La suma del procesador 2 de 10 procesadores = 0
La suma del procesador 3 de 10 procesadores = 0
La suma del procesador 4 de 10 procesadores = 0
La suma del procesador 5 de 10 procesadores = 0
La suma del procesador 6 de 10 procesadores = 0
La suma del procesador 7 de 10 procesadores = 0
La suma del procesador 8 de 10 procesadores = 0
La suma del procesador 9 de 10 procesadores = 5037
La suma del procesador 2 de 10 procesadores = 0
La suma del procesador 3 de 10 procesadores = 5074
La suma del procesador 4 de 10 procesadores = 0
La suma del procesador 5 de 10 procesadores = 0
La suma del procesador 6 de 10 procesadores = 0
La suma del procesador 7 de 10 procesadores = 0
La suma del procesador 8 de 10 procesadores = 0
La suma del procesador 9 de 10 procesadores = 0

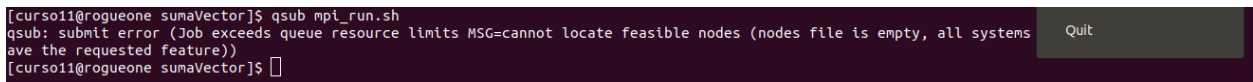
Tiempo total de ejecución: -0.874511 segundos
La suma del procesador 0 de 10 procesadores = 0
La suma del procesador 1 de 10 procesadores = 0
La suma del procesador 2 de 10 procesadores = 0
```

e. Ejecutar el script con 100 procesadores

Número de procesadores: 100.

Parámetros: array[100,000 n aleatorios entre 0 y 100,000].

No existen 100 nodos. El clúster arrojó este error: Job exceeds queue limits MSG= cannot locate feasible nodes...



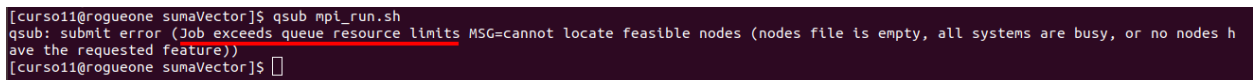
```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
qsub: submit error (Job exceeds queue resource limits MSG=cannot locate feasible nodes (nodes file is empty, all systems are busy, or no nodes have the requested feature))
[curso11@rogueone sumaVector]$
```

f. Ejecutar el script con 50 procesadores.

Mismo error que el anterior... El trabajo excede los límites Número de procesadores: 50.

Parámetros: array[100,000 n aleatorios entre 0 y 100,000].

de la cola.



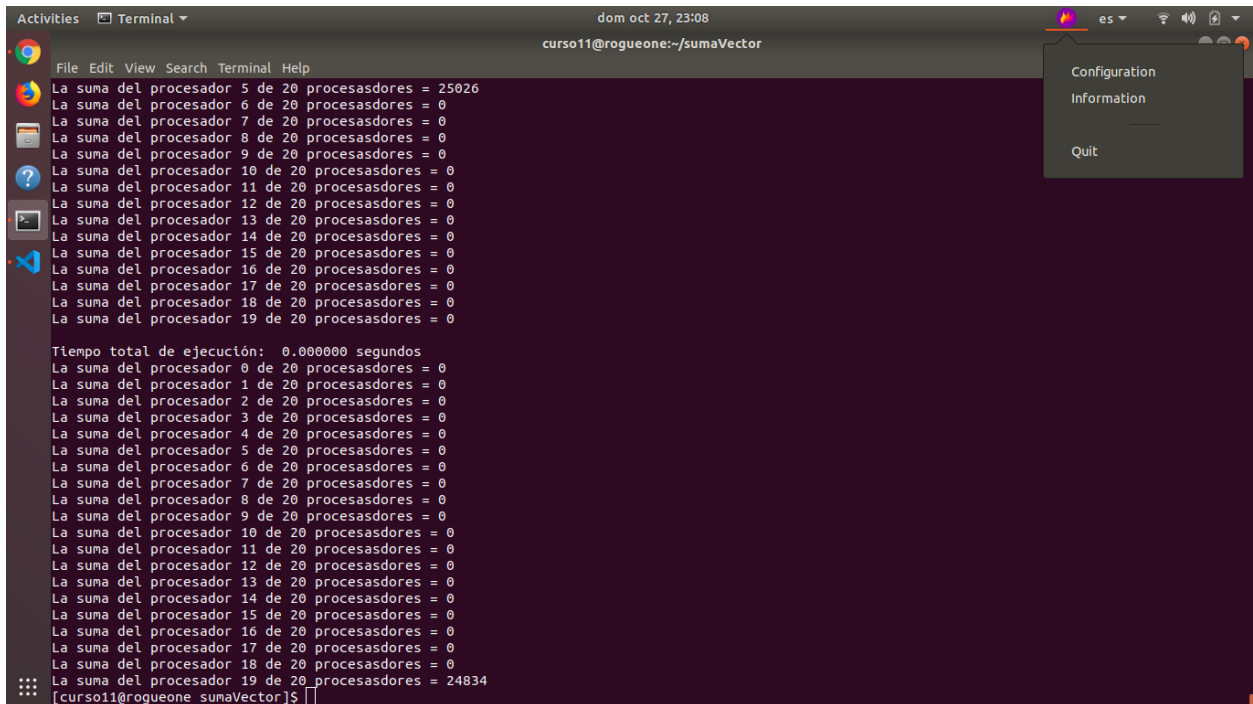
```
[curso11@rogueone sumaVector]$ qsub mpi_run.sh
qsub: submit error (Job exceeds queue resource limits MSG=cannot locate feasible nodes (nodes file is empty, all systems are busy, or no nodes have the requested feature))
[curso11@rogueone sumaVector]$
```

g. Ejecutar el script con 20 procesadores

Número de procesadores: 20. Sí permitió

Parámetros: array[100,000 n aleatorios entre 0 y 100,000].

```
6488.rogueone      mpi_sumaVector  curso11      00:00:02 C staff
[curso11@rogueone sumaVector]$
```



```
Activities  Terminal  dom oct 27, 23:08
curso11@rogueone:~/sumaVector

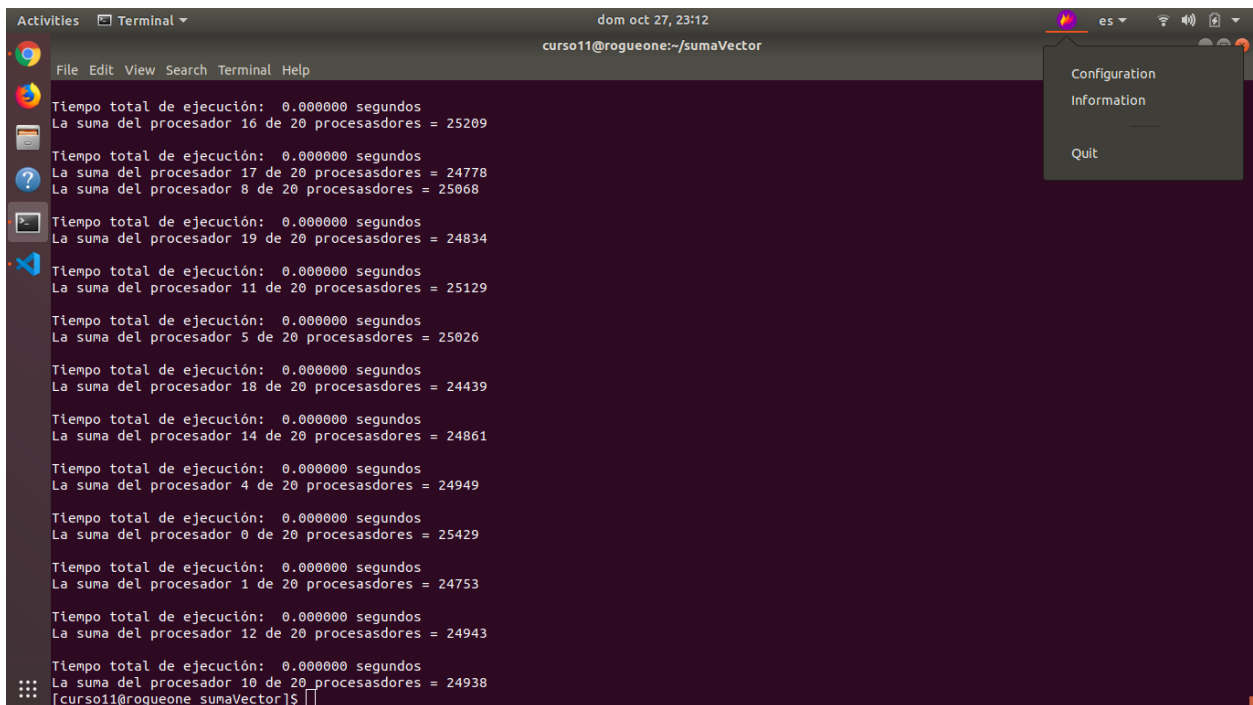
File Edit View Search Terminal Help

La suma del procesador 5 de 20 procesadores = 25026
La suma del procesador 6 de 20 procesadores = 0
La suma del procesador 7 de 20 procesadores = 0
La suma del procesador 8 de 20 procesadores = 0
La suma del procesador 9 de 20 procesadores = 0
La suma del procesador 10 de 20 procesadores = 0
La suma del procesador 11 de 20 procesadores = 0
La suma del procesador 12 de 20 procesadores = 0
La suma del procesador 13 de 20 procesadores = 0
La suma del procesador 14 de 20 procesadores = 0
La suma del procesador 15 de 20 procesadores = 0
La suma del procesador 16 de 20 procesadores = 0
La suma del procesador 17 de 20 procesadores = 0
La suma del procesador 18 de 20 procesadores = 0
La suma del procesador 19 de 20 procesadores = 0

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 20 procesadores = 0
La suma del procesador 1 de 20 procesadores = 0
La suma del procesador 2 de 20 procesadores = 0
La suma del procesador 3 de 20 procesadores = 0
La suma del procesador 4 de 20 procesadores = 0
La suma del procesador 5 de 20 procesadores = 0
La suma del procesador 6 de 20 procesadores = 0
La suma del procesador 7 de 20 procesadores = 0
La suma del procesador 8 de 20 procesadores = 0
La suma del procesador 9 de 20 procesadores = 0
La suma del procesador 10 de 20 procesadores = 0
La suma del procesador 11 de 20 procesadores = 0
La suma del procesador 12 de 20 procesadores = 0
La suma del procesador 13 de 20 procesadores = 0
La suma del procesador 14 de 20 procesadores = 0
La suma del procesador 15 de 20 procesadores = 0
La suma del procesador 16 de 20 procesadores = 0
La suma del procesador 17 de 20 procesadores = 0
La suma del procesador 18 de 20 procesadores = 0
La suma del procesador 19 de 20 procesadores = 24834

[curso11@rogueone sumaVector]$
```

Captura mejorando la salida



```
Activities  Terminal  dom oct 27, 23:12
curso11@rogueone:~/sumaVector

File Edit View Search Terminal Help

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 16 de 20 procesadores = 25209

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 17 de 20 procesadores = 24778
La suma del procesador 8 de 20 procesadores = 25068

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 19 de 20 procesadores = 24834

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 11 de 20 procesadores = 25129

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 5 de 20 procesadores = 25026

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 18 de 20 procesadores = 24439

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 14 de 20 procesadores = 24861

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 4 de 20 procesadores = 24949

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 0 de 20 procesadores = 25429

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 1 de 20 procesadores = 24753

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 12 de 20 procesadores = 24943

Tiempo total de ejecución: 0.000000 segundos
La suma del procesador 10 de 20 procesadores = 24938

[curso11@rogueone sumaVector]$
```

OBSERVACIONES Y CONCLUSIONES

En resumen, en esta práctica repasamos las principales funciones de la librería MPI. Ya utilizamos una función de broadcast para enviar las variables a todos los procesadores implicados en el job. Otra función que aprendimos fue la de clock(). Esta función marcaba 0.000 para parámetros pequeños, pero observamos que sí contaba los milisegundos con parámetros más grandes.

Por otra parte, pienso que podría mejorar en el formato de salida, y en incorporar alguna condición para que solo se imprima una vez el array de números aleatorios. Encontré un repositorio de github (luanmv) que está más elaborado.

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>

luanmv. (2016). Suma de vectores con MPI Recupeardo de <https://gist.github.com/luanmv/dce30466d8109b0beff71806dfa7b1df>