Bitácora de trabajo
Gustavo Ramírez Hidalgo
Proyecto módulo 4 - SP2118

---

1. Sobre la programación distribuida y MPI:

2. Sobre la conexión con el cluster:

Para llevar a cabo la conexión con el Cluster CNCA, se hizo a través de ssh (ingresando la clave en el momento en el que se pide):

```
ssh gramirez@163.178.80.34 -X
```

y la bandera agregada al final sirve para desplegar un sistema de ventanas desde el cluster y hacia la computadora local en la que se está trabajando. Por ejemplo, en este caso se quiere editar el archivo PVM.c (en el cual se programará el módulo para testeo del desempeño del cluster), con lo cual, una vez logeado en el cluster, se ejecuta:

```
[gramirez@meta curso]$ gedit PVM.c
```

Por último, para llevar a cabo la conexión específica con uno de los nodos, se hizo:

```
./interactivo.sh xeon cadejos-0.cnca 2:00:00
```

3. Sobre el uso de MPI y C:

El primer punto importante es que, para evitar problemas de validación de ciertos aspectos del programa, la dimensionalidad del espacio en el cual se están trabajando estos productos matriciales (que al final terminan siendo productos vectoriales, por lo cual se está hablando aquí de dimensionalidad, refiriéndose esta a la dimensionalidad de los vectores multiplicados) debe ser un múltiplo de la cantidad de procesos.

Es importante notar además que las matrices se están representando aquí como arreglos lineales (esto para evitar saltos en la memoria, lo que haría al programa más lento).

Las bibliotecas empleadas en el programa son prácticamente las mismas empleadas en cualquier otro programa de C; pero hay un par de ellas que requieren un poco más de profundización:

- time.h: esta biblioteca se empleará en las mediciones de tiempo que se quieren hacer, para obtener así cuánto tiempo tarda el programa (para distintas cantidades de procesadores y distintos tamaños de matrices a multiplicar).
- mpi.h: esta es la biblioteca que permite el uso de MPI desde un punto de vista computacional, para la paralelización del producto matricial.

4. Sobre la compilación y ejecución:

Para llevar a cabo la compilación, se ejecuta lo siguiente:

```
$mpicc.openmpi PVM.c -o [nombre binario]
```

y para ejecutar el binario generado, se utiliza lo siguiente:

```
$mpiexec.openmpi -n [np] ./[nombre binario]
```

en donde *np* es el número de procesadores; en el caso de este proyecto, se variará este parámetro, tomando los valores 2, 4 y 8.

5. Sobre la generación de las familias de curvas:

Para automatizar un poco el proceso de testeo, se hizo un script bastante corto, que ayudó en la generación de las familias de curvas con solo una ejecución (la de dicho script, llamado compileANDrunPVM.sh):
AQUI EL SCRIPT