

Universidad Nacional  
**ARTURO JAURETCHE**

SISTEMAS DISTRIBUIDOS

LABORATORIO N° 1

---

## Paralelización MPI

---

*Autor*  
Emiliano SALVATORI

8 de diciembre de 2020

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Configuración de las máquinas virtuales</b>	<b>2</b>
2.1. Instalación las herramientas a utilizar . . . . .	2
2.2. Clonación de la máquina Maestro . . . . .	3
2.3. Configuración de las direcciones IP estáticas . . . . .	4
2.4. Configuración mediante la herramienta SSH . . . . .	5
2.5. Configuración de los nombres de los host . . . . .	6
2.6. Configuración NFS . . . . .	7
<b>3. Testeo sobre la librería MPI</b>	<b>9</b>
<b>4. Paralelización para el producto entre matrices</b>	<b>11</b>
4.1. Consideraciones generales . . . . .	11
4.2. Ejecución del código . . . . .	11
4.3. Modificación de las Dimensiones de las matrices . . . . .	14
<b>5. Implementación con NTP</b>	<b>17</b>
5.1. Configuración inicial del sistema para NTP . . . . .	17
5.2. Modificación del archivo de configuración NTP en el Maestro . . . . .	19
5.3. Modificación del archivo de configuración NTP en los Esclavos . . . . .	20
5.4. Prueba del funcionamiento distribuido de NTP . . . . .	21
<b>6. Código del programa</b>	<b>23</b>
<b>7. Conclusiones</b>	<b>25</b>

## 1. Introducción

En el siguiente informe se detalla lo realizado como parte del laboratorio de la materia **Sistemas Distribuidos** para la **Comisión n° 1**.

La implementación realizada para este trabajo se basa en los siguientes objetivos:

- Configuración de 3 máquinas virtuales bajo entorno GNU/Linux.
- Configuración de un cluster mediante SSH/NFS
- Entendimiento básico del pasajeo de mensajes mediante MPI
- Sincronización bajo Sistemas Distribuidos NTP
- Paralelización de código para calcular el resultado de la multiplicación de matrices.

## 2. Configuración de las máquinas virtuales

### 2.1. Instalación las herramientas a utilizar

Para poder realizar la construcción de un cluster, se procede a configurar tres máquinas virtuales bajo VirtualBox, bajo la disposición de un Nodo Maestro y dos Nodos Esclavos.

Para ello comprobamos que tenemos conexión a internet haciendo uso de la herramienta *ping*. Una vez confirmada la conexión a internet se procede a instalar los programas necesarios:

```
root@debian:~# ping -c 5 www.google.com
PING www.google.com(2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004)) 56 data bytes
64 bytes from 2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004): icmp_seq=1 ttl=115 time=7.07 ms
64 bytes from 2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004): icmp_seq=2 ttl=115 time=7.06 ms
64 bytes from 2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004): icmp_seq=3 ttl=115 time=7.07 ms
64 bytes from 2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004): icmp_seq=4 ttl=115 time=7.34 ms
64 bytes from 2800:3f0:4002:803::2004 (2800:3f0:4002:803::2004): icmp_seq=5 ttl=115 time=6.64 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 13ms
rtt min/avg/max/mdev = 6.639/7.035/7.341/0.237 ms
root@debian:~# apt-get install mpich
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  gfortran gfortran-8 hwloc-nox libgfortran5 libhwloc-plugins libhwloc5 libicu63
  libltdl7 libmpich-dev libmpich12 libnuma1 libpciaccess0 libxml2 ocl-icd-libopencl1
Paquetes sugeridos:
  gfortran-multilib gfortran-doc gfortran-8-multilib gfortran-8-doc libgfortran5-dbg
  libcoarrays-dev libhwloc-contrib-plugins mpich-doc opencl-icd
Se instalarán los siguientes paquetes NUEVOS:
  gfortran gfortran-8 hwloc-nox libgfortran5 libhwloc-plugins libhwloc5 libicu63
  libltdl7 libmpich-dev libmpich12 libnuma1 libpciaccess0 libxml2 mpich ocl-icd-libopencl1
0 actualizados, 16 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 23.4 MB de archivos.
Se utilizarán 86,0 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S_
```

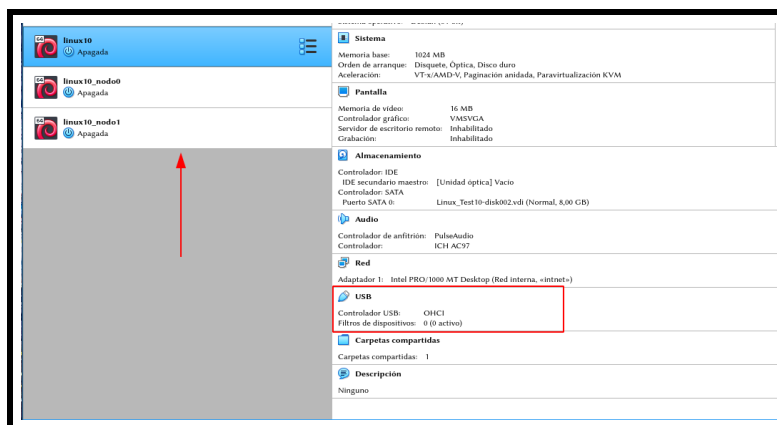
Los programas que se instalan son:

1. mpich
2. uml-utilities

3. bridge-utils
4. openssh-server
5. nfs-kernel-server
6. nfs-common
7. net-tools
8. vim : útil para poder suplantar a los editores *nano* y *vi* que vienen por defecto.

## 2.2. Clonación de la máquina Maestro

Luego de instalados los programas listados, se procede a clonar la máquina virtual:



Se puede observar que se tiene una carpeta compartida, la cual será necesaria para poder editar el código en la máquina host, y poder de esta manera intercambiar los archivos que querramos con mayor facilidad.

### 2.3. Configuración de las direcciones IP estáticas

```
13 auto enp0s3
14 iface enp0s3 inet dhcp
15 #address 192.168.2.156
16 #netmask 255.255.255.0
17
18 #Modificado por mi
19
20 address 192.168.0.10
21 netmask 255.255.255.0
22
```

Se procede a configurar para cada máquina las direcciones IP, para ello se utiliza el editor *vim*, asignándosele las siguientes direcciones a cada una de las máquinas:

- **Maestro:** 192.168.0.10
- **Esclavo n° 1:** 192.168.0.11
- **Esclavo n° 2:** 192.168.0.12

Como ejemplo, vemos en la máquina Esclavo n° 1, la configuración correspondiente:

```
root@debian:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug enp0s3
#iface enp0s3 inet dhcp
auto enp0s3
#iface enp0s3 inet dhcp
#address 192.168.2.156
#netmask 255.255.255.0

#Modificado por mi
iface enp0s3 inet static
address 192.168.0.11
netmask 255.255.255.0

root@debian:~# ifquery enp0s3
address: 192.168.0.11
netmask: 255.255.255.0
broadcast: 192.168.0.255
root@debian:~# _
```

Como se puede visualizar, desde el Nodo Maestro realizamos un ping a las dos direcciones de los nodos Esclavos:

```
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ ping -c 5 192.168.0.11  
PING 192.168.0.11 (192.168.0.11) 56(84) bytes of data.  
64 bytes from 192.168.0.11: icmp_seq=1 ttl=64 time=1.22 ms  
64 bytes from 192.168.0.11: icmp_seq=2 ttl=64 time=1.26 ms  
64 bytes from 192.168.0.11: icmp_seq=3 ttl=64 time=1.21 ms  
64 bytes from 192.168.0.11: icmp_seq=4 ttl=64 time=1.41 ms  
64 bytes from 192.168.0.11: icmp_seq=5 ttl=64 time=1.19 ms  
  
--- 192.168.0.11 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 15ms  
rtt min/avg/max/mdev = 1.193/1.260/1.412/0.090 ms  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$  
linuxtest@NodoMaestro:~$ ping -c 5 192.168.0.12  
PING 192.168.0.12 (192.168.0.12) 56(84) bytes of data.  
64 bytes from 192.168.0.12: icmp_seq=1 ttl=64 time=2.38 ms  
64 bytes from 192.168.0.12: icmp_seq=2 ttl=64 time=0.657 ms  
64 bytes from 192.168.0.12: icmp_seq=3 ttl=64 time=1.26 ms  
64 bytes from 192.168.0.12: icmp_seq=4 ttl=64 time=1.43 ms  
64 bytes from 192.168.0.12: icmp_seq=5 ttl=64 time=1.33 ms  
  
--- 192.168.0.12 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 30ms  
rtt min/avg/max/mdev = 0.657/1.409/2.376/0.553 ms  
linuxtest@NodoMaestro:~$
```

## 2.4. Configuración mediante la herramienta SSH

Para que desde el Maestro se pueda acceder a los distintos nodos de forma remota y sin tener que ingresar la contraseña del nodo accedido se hace uso del paquete *ssh*.

Para ello primero se crea una carpeta de nombre *share* en el directorio */home/linuxtest* del Mestro, haciendo esto también en cada nodo.

Luego se procede a ejecutar:

```
$ ssh-keygen -t rsa -b 2048
```

Esto hace posible que en el nodo Maestro se generen las llaves públicas y privadas a utilizar. A continuación podemos observar las llaves ya generadas dentro de la carpeta:

```
linuxtest@NodoMaestro:~$ pwd  
/home/linuxtest  
linuxtest@NodoMaestro:~$ ls -la .ssh/  
total 24  
drwx----- 2 linuxtest linuxtest 4096 sep 20 20:40 .  
drwxr-xr-x 6 linuxtest linuxtest 4096 sep 26 16:11 ..  
-rw----- 1 linuxtest linuxtest 1823 sep 20 20:24 id_rsa  
-rw-r--r-- 1 linuxtest linuxtest 398 sep 20 20:24 id_rsa.pub  
-rw-r--r-- 1 linuxtest linuxtest 444 sep 20 20:41 known_hosts
```

Copiamos en cada directorio `.ssh` de los nodos Esclavos la llave pública del Maestro de la siguiente manera:

```
$ scp id_rsa.pub linuxtest@192.168.0.11:~/.ssh/id_rsa.pub
$ scp id_rsa.pub linuxtest@192.168.0.12:~/.ssh/id_rsa.pub
```

Ahora podremos observar que se puede ingresar desde el Maestro al **Nodo Esclavo n° 1**:

```
linuxtest@NodoMaestro:~$ sudo ifquery enp0s3
sudo: unable to resolve host NodoMaestro: Fallo temporal en la resolución del nombre
address: 192.168.0.10
netmask: 255.255.255.0
broadcast: 192.168.0.255
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$ ssh linuxtest@192.168.0.11
Linux LinuxNodo1 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 26 21:36:56 2020 from 192.168.0.10
linuxtest@LinuxNodo1:~$ sudo ifquery enp0s3
sudo: unable to resolve host LinuxNodo1: Fallo temporal en la resolución del nombre
[sudo] password for linuxtest:
address: 192.168.0.11
netmask: 255.255.255.0
broadcast: 192.168.0.255
linuxtest@LinuxNodo1:~$
```

Como también se puede ingresar desde el Maestro al **Nodo Esclavo n° 2**:

```
linuxtest@NodoMaestro:~$ sudo ifquery enp0s3
sudo: unable to resolve host NodoMaestro: Fallo temporal en la resolución del nombre
address: 192.168.0.10
netmask: 255.255.255.0
broadcast: 192.168.0.255
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$ ssh linuxtest@192.168.0.12
Linux LinuxNodo2 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 26 21:27:20 2020
linuxtest@LinuxNodo2:~$ sudo ifquery enp0s3
sudo: unable to resolve host LinuxNodo2: Fallo temporal en la resolución del nombre
[sudo] password for linuxtest:
address: 192.168.0.12
netmask: 255.255.255.0
broadcast: 192.168.0.255
linuxtest@LinuxNodo2:~$ _
```

## 2.5. Configuración de los nombres de los host

En cada uno de los nodos, será necesario modificar sus direcciones IP estáticas por un pseudónimo que apunte a sus nombres de host. Primero, en cada uno de los nodos, será necesario ejecutar el siguiente comando:

```
$ sudo hostnamectl set-hostname <NOMBRE_NODO>
```

Donde *NOMBRE\_NODO* será el nombre que nosotros le querramos dar a cada máquina. Se optó por los siguientes nombres:

- **NodoMaestro:** Será el Nodo Maestro que controlará los otros dos nodos Esclavos
- **LinuxNodo1:** Nodo Esclavo nº 1
- **LinuxNodo2:** Nodo Esclavo nº 1

Una vez modificado cada uno de los nombres de los hosts, lo siguiente será editar en los tres nodos el archivo */etc/hosts* para poder relacionar cada dirección con las máquinas que forman el cluster:

```
linuxtest@NodoMaestro:~$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      debian

# The following lines are desirable for IPv6 capable hosts
::1           localhost ip6-localhost ip6-loopback
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

# Modificado por Emiliano
192.168.0.10  NodoMaestro
192.168.0.11  LinuxNodo1
192.168.0.12  LinuxNodo2
linuxtest@NodoMaestro:~$
```

Ahora podemos observar que con la nueva configuración es posible conectarse a los distintos nodos ingresando solamente el nombre del host:

```
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$ ssh linuxtest@LinuxNodo1
Linux LinuxNodo1 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
last login: Sun Sep 27 11:56:46 2020
linuxtest@LinuxNodo1:~$
```

## 2.6. Configuración NFS

Mediante la herramienta NFS es posible tener disponible una carpeta mediante la cual se compartirá el código entre las máquinas que conforman el Cluster, para que luego sea ejecutado por los distintos nodos.

Para llevar a cabo esto, primero es necesario crear una carpeta con el nombre *share* en la ubicación */home/linuxtest*; y luego editar el archivo denominado *exports*.



```
linuxtest@NodoMaestro:~$ pwd
/home/linuxtest
linuxtest@NodoMaestro:~$ mkdir share
linuxtest@NodoMaestro:~$ ls -la
total 56
drwxr-xr-x 7 linuxtest linuxtest 4096 sep 26 21:29 .
drwxr-xr-x 3 root      root      4096 sep 27 2019 ..
-rw-r----- 1 linuxtest linuxtest 1993 sep 26 18:59 .bash_history
-rw-r--r-- 1 linuxtest linuxtest 220 sep 27 2019 .bash_logout
-rw-r--r-- 1 linuxtest linuxtest 3526 sep 27 2019 .bashrc
drwx----- 3 linuxtest linuxtest 4096 sep 20 20:24 .gnupg
drwxr-xr-x 3 linuxtest linuxtest 4096 abr 19 03:31 .local
drwxr-x--- 2 root      root      4096 sep 19 17:39 mnt
-rwxr--r-- 1 linuxtest linuxtest 68 abr 19 03:32 mount.sh
-rw-r--r-- 1 linuxtest linuxtest 807 sep 27 2019 .profile
drwxr-xr-x 2 linuxtest linuxtest 4096 sep 26 21:29 share
drwx----- 2 linuxtest linuxtest 4096 sep 20 20:40 .ssh
-rw-r----- 1 linuxtest linuxtest 1541 sep 20 20:55 .viminfo
-rw-r--r-- 1 linuxtest linuxtest 33 sep 20 20:47 .vimrc
linuxtest@NodoMaestro:~$
```

Este archivo permite indicarle al SO el recurso que será accedido por los distintos nodos, proporcionando información sobre quienes podrán acceder, el nivel de permisos que tendrán los nodos (como leer y escribir), si el servidor responderá a peticiones cuando se complete la tarea del disco, y además se debe indicar el nivel de acceso del superusuario (root) de los clientes al sistema de archivos.

```
1 # /etc/exports: the access control list for filesystems which may be exported
2 # to NFS clients. See exports(5).
3 #
4 # Example for NFSv2 and NFSv3:
5 # /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
6 #
7 # Example for NFSv4:
8 # /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9 # /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
10 #
11 /home/linuxtest/share *(rw,sync,no_subtree_check,no_root_squash)
```

Y luego utilizamos el siguiente comando:

```
$ sudo exportfs -a
```

Una vez que tenemos los esto procedemos dentro de cada uno de los nodos Esclavos, a montar la carpeta compartida con el siguiente comando:

```
$ sudo mount -t nfs NodoMaestro:/home/linuxtest/share /share
```

Y Nos fijamos si se montó de forma correcta mediante:

```
$ df -h
```

Efectivamente, observamos que se monta de forma correcta:

```
linuxtest@LinuxNodo1:~$ sudo mount -t nfs NodoMaestro:/home/linuxtest/share ~/share/
[sudo] password for linuxtest:
linuxtest@LinuxNodo1:~$
linuxtest@LinuxNodo1:~$
linuxtest@LinuxNodo1:~$ df -h
S.ficheros          Tamaño Usados  Disp Uso% Montado en
udev                480M      0  480M   0% /dev
tmpfs               99M       3,0M   96M   3% /run
/dev/sda1           6,9G     1,7G   4,8G  27% /
tmpfs              494M      0  494M   0% /dev/shm
tmpfs              5,0M      0   5,0M   0% /run/lock
tmpfs              494M      0  494M   0% /sys/fs/cgroup
tmpfs              99M       0   99M   0% /run/user/1000
NodoMaestro:/home/linuxtest/share 6,9G     1,7G   4,8G  27% /home/linuxtest/share
linuxtest@LinuxNodo1:~$ _
```

Como nos aseguramos que montó correctamente, entonces ahora procedemos a generar una entrada en el archivo `/etc/fstab` para que se monte de forma automática cada vez que se inicien los nodos:

```
1 # /etc/fstab: static file system information.
2 #
3 # Use 'blkid' to print the universally unique identifier for a
4 # device; this may be used with UUID= as a more robust way to name devices
5 # that works even if disks are added and removed. See fstab(5).
6 #
7 # <file system> <mount point> <type> <options> <dump> <pass>
8 # / was on /dev/sda1 during installation
9 UUID=c39d52af-966e-4c0c-99c6-55443763ff5c / ext4 errors=remount-ro 0 1
10 # swap was on /dev/sda5 during installation
11 UUID=569c66d8-3de1-42d8-9412-6b9b0f4a36a4 none swap sw 0 0
12 /dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
13 # MPI CLUSTER
14 NodoMaestro:/home/linuxtest/share /home/linuxtest/share nfs
```

### 3. Testeo sobre la librería MPI

Ahora que ya se encuentra configurado correctamente el cluster, procedemos a realizar un ejemplo con un código sencillo para saber si es posible paralelizar el código compilado.

Para esto primero editamos un archivo que contenga los nombres de los host que estaremos utilizando para paralelizar el código y luego compilamos el código utilizando el siguiente comando:

```
$ mpicc ejemplo.c -o ejemplo
```

```
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$ pwd  
/home/linuxtest/share  
linuxtest@NodoMaestro:~/share$ cat host  
NodoMaestro  
LinuxNodo1  
LinuxNodo2  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$ cat ejemplo.c  
#include <stdio.h>  
#include <mpi.h>  
  
int main (int argc, char *argv[]) {  
  
    int rank, size;  
  
    MPI_Init (&argc, &argv); /* starts MPI */  
    MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id */  
    MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */  
  
    printf( "Hello world from process %d of %d\n", rank, size );  
  
    MPI_Finalize();  
  
    return 0;  
}  
linuxtest@NodoMaestro:~/share$ mpicc ejemplo.c -o ejemplo  
linuxtest@NodoMaestro:~/share$ ls -la  
total 36  
drwxr-xr-x 2 linuxtest linuxtest 4096 sep 27 14:37 .  
drwxr-xr-x 7 linuxtest linuxtest 4096 sep 27 14:35 ..  
-rwxr-xr-x 1 linuxtest linuxtest 16784 sep 27 14:37 ejemplo  
-rw-r--r-- 1 linuxtest linuxtest 369 sep 27 12:47 ejemplo.c  
-rw-r--r-- 1 linuxtest linuxtest 34 sep 27 14:35 host  
linuxtest@NodoMaestro:~/share$ _
```

Como podemos observar en la imagen anterior, la compilación nos genera un ejecutable denominado “ejemplo”, el cual ejecutamos

```
$ mpiexec -f host -n 4 ./ejemplo
```

Luego de ello se obtiene:

```
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$ ls -la  
total 36  
drwxr-xr-x 2 linuxtest linuxtest 4096 sep 27 14:37 .  
drwxr-xr-x 7 linuxtest linuxtest 4096 sep 27 14:35 ..  
-rwxr-xr-x 1 linuxtest linuxtest 16784 sep 27 14:37 ejemplo  
-rw-r--r-- 1 linuxtest linuxtest 369 sep 27 12:47 ejemplo.c  
-rw-r--r-- 1 linuxtest linuxtest 34 sep 27 14:35 host  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$ mpiexec -f host -n 4 ./ejemplo  
Hello world from process 2 of 4  
Hello world from process 1 of 4  
Hello world from process 3 of 4  
Hello world from process 0 of 4  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$
```

## 4. Paralelización para el producto entre matrices

Para la implementación de las librerías MPI, se utilizará el cálculo de multiplicación entre matrices. Para poder paralelizar la ejecución y poder distribuir la carga de cálculos entre distintos nodos, se utilizará la librería `mpi.h`, junto con la configuración desarrollada en la sección 3.

Para que los nodos tengan acceso al ejecutable, es necesario que sea suministrado en una dirección donde todo el conjunto de máquinas que conforman el cluster, puedan acceder a él. Para ello se utiliza la dirección configurada en el apartado 2.6

### 4.1. Consideraciones generales

Para la ejecución del producto entre matrices, se deben tener en cuenta algunas consideraciones especiales del programa:

1. Las dimensiones de las matrices son estáticas; es decir, que los elementos de la **matriz A** serán determinados en el código, sin posibilidad de ser modificadas en tiempo de ejecución. Esto es así ya que de lo contrario, sería necesario utilizar las funciones `malloc()` y `calloc()` y leer las entradas desde el teclado, implementaciones que excenden al alcance del trabajo actual.

Lo mismo sucede con la cantidad de elementos de la **matriz B**.

2. El producto entre matrices se hace entre una **matriz A** cuyas dimensiones pueden ser modificadas en el código (es decir que la cantidad de elementos de sus filas y columnas pueden variar); y una **matriz B** con una cantidad de elementos por columna también alterable por el usuario, pero con sus filas **igual a las Columnas de A**.

Esto se implementó de esta manera ya que de lo contrario, habría que manejar el ingreso de datos desde el teclado, labor que excedía al tiempo estimado para la entrega del trabajo. De esta forma, siempre será posible el cálculo del producto entre las matrices que se busquen.

3. La inicialización de los elementos de las matrices se realiza mediante una estructura de control, con un `for`, cuyos contadores comienzan en 0 y se incrementan de uno a 1 por cantidad de columnas en A, y en Filas por B.

### 4.2. Ejecución del código

Para editar el código se utilizó el editor `vim` en un SO anfitrión de tipo GNU/Linux. El archivo se colocó en la carpeta compartida entre el SO anfitrión y el SO huésped (Debian 10), como se puede ver a continuación en la siguiente imagen:

```
talino ~/Distribuidos/TPs/LaboratorioI/compartida/matrices.c
calculoPi.c      2.21 K
Ejecutar.sh      1.29 K
ejemplo.c        369 B
matrices.c       5.67 K

*****
* ARCHIVO: matrices.c
* DESCRIPCION: Multiplicación por Matrices -
* El nodo Maestro distribuye las operaciones
* Las tareas se separan segun la cantidad de
* paralelizacion, que sera siempre n-1
* AUTOR: Blaise Barney. Adaptado por Emiliano
* FECHA: 01/10/2020
*****
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>

#define NCA 10      /* Numero de Columnas
#define NFA 5       /* Numero de Fil
#define NCB 5       /* Numero de Co
#define MAESTRO 0   /* ID para la
#define DEL_MAESTRO 1 /* Se configura
#define DEL_NODO 2  /* Se configura un

int main (int argc, char *argv[]) {

    int numtareas, /* Numero de t
        taskid,    /* Identificado
        numprocesos, /* Numero de pr
```

Se puede observar, que el contenido es el mismo en la carpeta compartida de Debian:

```
linuxtest@NodoMaestro:~/share$ ls -la ../mnt/
total 28
drwxr-xr-x 1 linuxtest linuxtest 4096 oct  1 16:23 .
drwxr-xr-x 7 linuxtest linuxtest 4096 oct  1 16:49 ..
-rw-r--r-- 1 linuxtest linuxtest 2268 sep 28 01:11 calculoPi.c
-rwxr--r-- 1 linuxtest linuxtest 1320 oct  1 14:23 Ejecutar.sh
-rw-r--r-- 1 linuxtest linuxtest 369 sep 27 14:34 ejemplo.c
-rw-r--r-- 1 linuxtest linuxtest 5804 oct  1 16:23 matrices.c
linuxtest@NodoMaestro:~/share$
```

Ahora procedemos a ingresar el siguiente comando para copiar desde la carpeta compartida (mnt) a la compartida entre el cluster (share):

```
$ cp ../mnt/matrices.c
```

Una vez copiada a la carpeta share lo compilamos mediante el siguiente comando:

```
$ mpicc matrices.c -o matrices
```

```
linuxtest@NodoMaestro:~/share$
linuxtest@NodoMaestro:~/share$
linuxtest@NodoMaestro:~/share$ cp ../mnt/matrices.c .
linuxtest@NodoMaestro:~/share$ mpicc matrices.c -o matrices
linuxtest@NodoMaestro:~/share$ _
```

Vemos que compiló sin ningún tipo de problemas. Procedemos luego a ejecutarlo indicando que se repartirá la carga de cálculos entre los nodos que se encuentran en el archivo `host` y que se requiere paralelizar entre 4 procesos. Cabe resaltar que se utiliza el programa `less` el cual permita *paginar* la terminal, visualizando todo el contenido volcado:

```
$ mpiexec -f host -n 4 ./matrices | less
```

Visualizando la ejecución del proceso de forma correcta:

```
Se inicia MPI con un numero de 4 de procesos.
Dimensiones de Matriz A: 5x10
Contenido de la Matriz A de :
0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0
2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0  11.0
3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0  11.0  12.0
4.0  5.0  6.0  7.0  8.0  9.0  10.0  11.0  12.0  13.0

Dimensiones de Matriz B: 10x5
Contenido de la Matriz B de :
0.0  0.0  0.0  0.0  0.0
0.0  1.0  2.0  3.0  4.0
0.0  2.0  4.0  6.0  8.0
0.0  3.0  6.0  9.0  12.0
0.0  4.0  8.0  12.0  16.0
0.0  5.0  10.0  15.0  20.0
0.0  6.0  12.0  18.0  24.0
0.0  7.0  14.0  21.0  28.0
0.0  8.0  16.0  24.0  32.0
0.0  9.0  18.0  27.0  36.0

*****
Estatus de la paralelizacion:

Enviando 2 filas al proceso n° 1 offset=0
Enviando 2 filas al proceso n° 2 offset=2
Enviando 1 filas al proceso n° 3 offset=4
Recibiendo resultado del proceso numero: 1
Recibiendo resultado del proceso numero: 2
Recibiendo resultado del proceso numero: 3
:
```

```
Dimensiones de Matriz B:      10x5
Contenido de la Matriz B de :

0.0  0.0  0.0  0.0  0.0
0.0  1.0  2.0  3.0  4.0
0.0  2.0  4.0  6.0  8.0
0.0  3.0  6.0  9.0  12.0
0.0  4.0  8.0  12.0  16.0
0.0  5.0  10.0  15.0  20.0
0.0  6.0  12.0  18.0  24.0
0.0  7.0  14.0  21.0  28.0
0.0  8.0  16.0  24.0  32.0
0.0  9.0  18.0  27.0  36.0

*****
Estatus de la paralelizacion:

Enviando 2 filas al proceso n° 1 offset=0
Enviando 2 filas al proceso n° 2 offset=2
Enviando 1 filas al proceso n° 3 offset=4
Recibiendo resultado del proceso numero: 1
Recibiendo resultado del proceso numero: 2
Recibiendo resultado del proceso numero: 3

*****
Resultado de la Matriz C de 5x5:

0.0  285.0  570.0  855.0  1140.0
0.0  330.0  660.0  990.0  1320.0
0.0  375.0  750.0  1125.0  1500.0
0.0  420.0  840.0  1260.0  1680.0
0.0  465.0  930.0  1395.0  1860.0

*****
Finalizado.

(END)
```

### 4.3. Modificación de las Dimensiones de las matrices

Ahora procedemos a realizar la modificación de las dimensiones de A y de B. Para ello editamos directamente en el Nodo Maestro, el archivo *matrices.c*, y modificamos el valor de NCA (número de columnas de A) y de NFA (número de filas de A), llevándola a una dimensión de  $D_A = 12 \times 6$ ; modificamos también el valor de NCB (número de columnas de B), llevando a la matriz a una dimensión de  $D_B = 7 \times 12$ :

```

1  /******
2  * ARCHIVO: matrices.c
3  * DESCRIPCION: Multiplicación por Matrices - Implementado en Lenguaje
4  *   El nodo Maestro distribuye las operaciones para multiplicar la
5  *   Las tareas se separan segun la cantidad de clusters asignados
6  *   paralelizacion, que sera siempre n-1
7  * AUTOR: Blaise Barney. Adaptado por Emiliano Salvatori
8  * FECHA: 01/10/2020
9  *****/
10 #include "mpi.h"
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 #define NCA 12          /* Numero de Columnas en la Matriz A */
15 #define NFA 6           /* Numero de Filas en la Matriz A <--*/
16 #define NCB 7           /* Numero de Columnas en la Matriz B <--
17 #define MAESTRO 0       /* ID para la primer tarea */
18 #define DEL_MAESTRO 1   /* Se configura un tipo de mensaje del
19 #define DEL_NODO 2      /* Se configura un tipo de mensaje de los
20
21 int main (int argc, char *argv[]) {
22

```

Nuevamente procedemos a compilar el archivo con el siguiente comando:

```
$ mpicc matrices.c -o matrices
```

```

31         l, j, k, rc;      /* Contadores */
32         double a[NFA][NCA], /* Matriz A que sera multiplicada
33         b[NCA][NCB],       /* Matriz B que sera multiplicada
34         c[NFA][NCB];       /* Resultado de la matriz C */
35
36         /* Permite instanciar una estructura que contiene informacion del
"matrices.c" 185L, 6335C escritos
linuxtest@NodoMaestro:~/share$ mpicc matrices.c -o matrices
linuxtest@NodoMaestro:~/share$ _

```

Vemos que compiló sin ningún tipo de problemas. Procedemos luego a ejecutarlo indicando que se repartira la carga de cálculos entre los nodos que se encuentran en el archivo `host` y que se requiere paralelizar entre 7 procesos. Nuevamente, como toda la pantalla de la terminal no alcanza para visualizar la salida de los proceso, entonces se procede a utilizar la herramienta `less`:

```
$ mpiexec -f host -n 7 ./matrices | less
```



```
Se inicia MPI con un numero de 7 de procesos.
Dimensiones de Matriz A: 6x12
Contenido de la Matriz A de :
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0
3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0
4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0
5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0

Dimensiones de Matriz B: 12x7
Contenido de la Matriz B de :
0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 1.0 2.0 3.0 4.0 5.0 6.0
0.0 2.0 4.0 6.0 8.0 10.0 12.0
0.0 3.0 6.0 9.0 12.0 15.0 18.0
0.0 4.0 8.0 12.0 16.0 20.0 24.0
0.0 5.0 10.0 15.0 20.0 25.0 30.0
0.0 6.0 12.0 18.0 24.0 30.0 36.0
0.0 7.0 14.0 21.0 28.0 35.0 42.0
0.0 8.0 16.0 24.0 32.0 40.0 48.0
0.0 9.0 18.0 27.0 36.0 45.0 54.0
0.0 10.0 20.0 30.0 40.0 50.0 60.0
0.0 11.0 22.0 33.0 44.0 55.0 66.0

*****
Estatus de la paralelizacion:
Enviando 1 filas al proceso n° 1 offset=0
Enviando 1 filas al proceso n° 2 offset=1
Enviando 1 filas al proceso n° 3 offset=2
Enviando 1 filas al proceso n° 4 offset=3
:
```

```
0.0 6.0 12.0 18.0 24.0 30.0 36.0
0.0 7.0 14.0 21.0 28.0 35.0 42.0
0.0 8.0 16.0 24.0 32.0 40.0 48.0
0.0 9.0 18.0 27.0 36.0 45.0 54.0
0.0 10.0 20.0 30.0 40.0 50.0 60.0
0.0 11.0 22.0 33.0 44.0 55.0 66.0

*****
Estatus de la paralelizacion:

Enviando 1 filas al proceso n° 1 offset=0
Enviando 1 filas al proceso n° 2 offset=1
Enviando 1 filas al proceso n° 3 offset=2
Enviando 1 filas al proceso n° 4 offset=3
Enviando 1 filas al proceso n° 5 offset=4
Enviando 1 filas al proceso n° 6 offset=5
Recibiendo resultado del proceso numero: 1
Recibiendo resultado del proceso numero: 2
Recibiendo resultado del proceso numero: 3
Recibiendo resultado del proceso numero: 4
Recibiendo resultado del proceso numero: 5
Recibiendo resultado del proceso numero: 6

*****
Resultado de la Matriz C de 6x7:

0.0 506.0 1012.0 1518.0 2024.0 2530.0 3036.0
0.0 572.0 1144.0 1716.0 2288.0 2860.0 3432.0
0.0 638.0 1276.0 1914.0 2552.0 3190.0 3828.0
0.0 704.0 1408.0 2112.0 2816.0 3520.0 4224.0
0.0 770.0 1540.0 2310.0 3080.0 3850.0 4620.0
0.0 836.0 1672.0 2508.0 3344.0 4180.0 5016.0

*****
Finalizado.

(END)
```

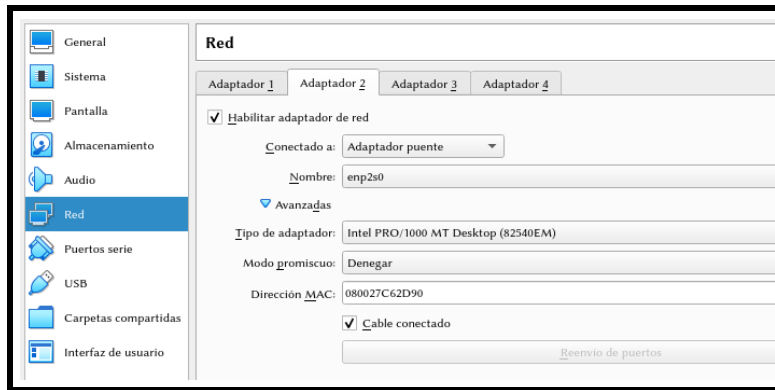
## 5. Implementación con NTP

### 5.1. Configuración inicial del sistema para NTP

Para el siguiente ejercicio es necesario modificar los nodos del sistema para que tengan salida a internet y con ello poder instalar todos los paquetes necesarios para el correcto funcionamiento de NTP. Cabe resaltar que estas modificaciones en los nodos esclavos serán *momentáneas*. La configuración en cada uno de los nodos que conforman el sistema, consistirá en:

- Habilitar un nuevo adaptador en la configuración de Red, como *Adaptador Puente*.
- Configurar el archivo *interfaces* para que tome y configure el nuevo adaptador.

Para ello vamos a realizar el primer punto en los tres nodos, como se puede ver a continuación:



Luego de ello se configura en las tres máquinas el archivo, de la siguiente forma:

```
1 # This file describes the network interfaces available on your system
2 # and how to activate them. For more information, see interfaces(5).
3
4 source /etc/network/interfaces.d/*
5
6 # The loopback network interface
7 auto lo
8 iface lo inet loopback
9
10 # The primary network interface
11 #allow-hotplug enp0s3
12 #iface enp0s3 inet dhcp
13
14 #Modificado por Emiliano
15 auto enp0s8
16 iface enp0s8 inet dhcp
17
18 #address 192.168.2.156
19 #netmask 255.255.255.0
20
21 #Modificado por Emiliano
22 auto enp0s3
23 iface enp0s3 inet static
24 address 192.168.0.10
25 netmask 255.255.255.0
```

Y se procede a instalar en cada una de las máquinas, lo requerido, mediante el siguiente comando:

```
$ sudo apt-get install ntp ntp-doc
```

Se puede ver que todo funciona correctamente cuando se lanza el siguiente comando:

```
$ ntpq -p
```

El programa *ntpq* se utiliza para supervisar las operaciones *ntpd* del demonio NTP y determinar su rendimiento. Cuando se ingresa el parámetro *-q* se imprime una lista de los pares conocidos por el servidor, así como un resumen de su estado.

```
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$ ntpq -p
=====
remote               refid          st t when poll reach  delay  offset  jitter
=====
0.debian.pool.n .POOL.        16 p -   64    0   0.000   0.000   0.000
1.debian.pool.n .POOL.        16 p -   64    0   0.000   0.000   0.000
2.debian.pool.n .POOL.        16 p -   64    0   0.000   0.000   0.000
3.debian.pool.n .POOL.        16 p -   64    0   0.000   0.000   0.000
+time.cloudflare 10.44.9.236    3 u 12   64   37   5.763  -2.062  28.059
+evlbi.aggo-coni 10.10.10.5     2 u 41   64   37   8.240  -33.596  10.479
+aggo2.aggo-coni .PTPO.         1 u 41   64   77   8.299  -14.640  25.624
+time.cloudflare 10.44.9.236    3 u 46   64   37   6.224  -24.499   9.458
+2a01:3e0:701:0: 195.219.14.21 2 u 41   64   77  232.121 -11.297  25.680
*168.96.251.227 168.96.251.196 2 u 39   64   37   6.656  -26.980  10.273
linuxtest@NodoMaestro:~$ _
```

## 5.2. Modificación del archivo de configuración NTP en el Maestro

Para el correcto funcionamiento de NTP, es necesario modificar en el nodo Maestro, el archivo *ntp.conf*, para que este nodo pueda encargarse de la difusión a los demás nodos del sistema con la opción *broadcastclient*. Asimismo, con la configuración ingresada se restringe la modificación del sistema sólo al maestro. La modificación queda de la siguiente manera:

```
17 # You do need to talk to an NTP server or two (or three).
18 #server ntp.your-provider.example
19
20 # pool.ntp.org maps to about 1000 low-stratum NTP servers. Your server will
21 # pick a different set every time it starts up. Please consider joining the
22 # pool: <http://www.pool.ntp.org/join.html>
23 pool 0.debian.pool.ntp.org iburst
24 pool 1.debian.pool.ntp.org iburst
25 pool 2.debian.pool.ntp.org iburst
26 pool 3.debian.pool.ntp.org iburst
27
28 #Modificado por Emiliano
29 restrict 192.168.0.10 mask 255.255.255.0 nomodify notrap
30 broadcastclient
31
32 # Access control configuration; see /usr/share/doc/ntp-doc/html/accpt.html for
33 # details. The web page <http://support.ntp.org/bin/view/Support/AccessRestrictions>
34 # might also be helpful.
35 #
36 # Note that "restrict" applies to both servers and clients, so a configuration
28,24 Comienzo
```

Luego de ello, se ingresa un comando para que se permita la comunicación mediante UDP por el puerto 123:

```
$ sudo iptables -A INPUT -m state --state NEW -m udp -p udp --dport 123 -j ACCEPT
```

Si todo sale de forma correcta, se vuelve al prompt:

```
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$
linuxtest@NodoMaestro:~$ sudo iptables -A INPUT -m state --state NEW -m udp -p udp --dport 123 -j ACCEPT
linuxtest@NodoMaestro:~$
```

### 5.3. Modificación del archivo de configuración NTP en los Esclavos

Para la configuración de los nodos Esclavos, se debe modificar también el archivo *ntp.conf* pero comentando todas las líneas de los pools, y especificando que el servidor será el nodo Maestro, quedando de la siguiente forma:

```
20 # pool.ntp.org maps to about 1000 low-stratum NTP servers. Your server will
21 # pick a different set every time it starts up. Please consider joining the
22 # pool: <http://www.pool.ntp.org/join.html>
23 # pool 0.debian.pool.ntp.org iburst
24 # pool 1.debian.pool.ntp.org iburst
25 # pool 2.debian.pool.ntp.org iburst
26 # pool 3.debian.pool.ntp.org iburst
27 server 192.168.0.10
28
29 # Access control configuration; see /usr/share/doc/ntp-doc/html/acopt.html for
30 # details. The web page <http://support.ntp.org/bin/view/Support/AccessRestrictions>
31 # might also be helpful.
32 #
33 # Note that "restrict" applies to both servers and clients, so a configuration
34 # that might be intended to block requests from certain clients could also end
35 # up blocking replies from your own upstream servers.
```

Se procede a reiniciar el servicio de NTP en cada uno de los nodos esclavos con el siguiente comando:

```
$ sudo /etc/init.d/ntp restart
```

Y se visualiza el correcto reinicio en el nodo Esclavo nº 1:

```
linuxtest@LinuxNodo1:~$
linuxtest@LinuxNodo1:~$ sudo /etc/init.d/ntp restart
[sudo] password for linuxtest:
[ ok ] Restarting ntp (via systemctl): ntp.service.
linuxtest@LinuxNodo1:~$ _
```

Y en el Nodo Esclavo nº 2:

```
linuxtest@LinuxNodo2:~$
linuxtest@LinuxNodo2:~$ sudo /etc/init.d/ntp restart
[sudo] password for linuxtest:
[ ok ] Restarting ntp (via systemctl): ntp.service.
linuxtest@LinuxNodo2:~$
```

A continuación visualizamos que todo esté funcionando correctamente ingresando tanto en el Nodo Esclavo nº 1 como en el nº 2 el siguiente comando:

```
$ ntpq -p
```

En el nodo Esclavo nº 1:

```
linuxtest@LinuxNodo1:~$  
linuxtest@LinuxNodo1:~$ ntpq -p  
=====  
remote      refid      st t when poll reach  delay  offset jitter  
=====  
NodoMaestro 168.96.251.197 2 u  6  64  1  1.374 -0.062  0.000  
linuxtest@LinuxNodo1:~$
```

Y en el nodo Esclavo nº 2:

```
linuxtest@LinuxNodo2:~$  
linuxtest@LinuxNodo2:~$ ntpq -p  
=====  
remote      refid      st t when poll reach  delay  offset jitter  
=====  
NodoMaestro 168.96.251.197 2 u 16  64  3  1.311 318879. 35.970  
linuxtest@LinuxNodo2:~$
```

En este punto se debe volver atrás las modificaciones realizadas en la sección 5.1, es decir que se quitan de los nodos Esclavos el segundo Adaptador de Red, quedando sólo el primer Adaptador como Red Interna y se vuelven para atrás las modificaciones establecidas en el archivo *interfaces*.

## 5.4. Prueba del funcionamiento distribuido de NTP

A continuación se pueden apreciar el tiempo transcurrido operando con NTP. Para ello volvemos a realizar lo comentado en la sección 4.2 y realizamos un cambio agregando más columnas y filas al código fuente:

```
1 /******  
2 * ARCHIVO: matrices.c  
3 * DESCRIPCION: Multiplicación por Matrices - Implementado en Lenguaje C  
4 * El nodo Maestro distribuye las operaciones para multiplicar las matrices.  
5 * Las tareas se separan según la cantidad de clusters asignados para la  
6 * paralelización, que será siempre n-1  
7 * AUTOR: Blaise Barney. Adaptado por Emiliano Salvatori  
8 * FECHA: 01/10/2020  
9 *****/  
10 #include "mpi.h"  
11 #include <stdio.h>  
12 #include <stdlib.h>  
13  
14 #define NCA 12 /* Numero de Columnas en la Matriz A */  
15 #define NFA 6 /* Numero de Filas en la Matriz A <--*/  
16 #define NCB 7 /* Numero de Columnas en la Matriz B <-- */  
17 #define MAESTRO 0 /* ID para la primer tarea */  
18 #define DEL_MAESTRO 1 /* Se configura un tipo de mensaje del nodo maestro*/  
19 #define DEL_NODO 2 /* Se configura un tipo de mensaje de los nodos*/  
20
```

A continuación volvemos a compilar utilizando los comandos:

```
$ mpicc matrices.c -o matrices
```

```
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$  
linuxtest@NodoMaestro:~/share$ mpicc matrices.c -o matrices  
linuxtest@NodoMaestro:~/share$
```

Vemos que compiló sin ningún tipo de problemas. Procedemos luego a ejecutarlo indicando que se repartirá la carga de cálculos entre los nodos que se encuentran en el archivo `host` y que se requiere paralelizar entre 4, sin antes testear el tiempo utilizando para ello la herramienta `time` procesos mediante:

```
$ time mpiexec -f host -n 4 ./matrices | less
```

```
Se inicia MPI con un numero de 4 de procesos.
Dimensiones de Matriz A:      6x12
Contenido de la Matriz A de :
  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0
  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0
  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0 13.0
  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0 13.0 14.0
  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0 13.0 14.0 15.0
  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0

Dimensiones de Matriz B:      12x7
Contenido de la Matriz B de :
  0.0  0.0  0.0  0.0  0.0  0.0  0.0
  0.0  1.0  2.0  3.0  4.0  5.0  6.0
  0.0  2.0  4.0  6.0  8.0 10.0 12.0
  0.0  3.0  6.0  9.0 12.0 15.0 18.0
  0.0  4.0  8.0 12.0 16.0 20.0 24.0
  0.0  5.0 10.0 15.0 20.0 25.0 30.0
  0.0  6.0 12.0 18.0 24.0 30.0 36.0
  0.0  7.0 14.0 21.0 28.0 35.0 42.0
  0.0  8.0 16.0 24.0 32.0 40.0 48.0
  0.0  9.0 18.0 27.0 36.0 45.0 54.0
  0.0 10.0 20.0 30.0 40.0 50.0 60.0
  0.0 11.0 22.0 33.0 44.0 55.0 66.0

*****
Estatus de la paralelizacion:
Enviando 2 filas al proceso nº 1 offset=0
Enviando 2 filas al proceso nº 2 offset=2
Enviando 2 filas al proceso nº 3 offset=4
Recibiendo resultado del proceso numero: 1
:
```

```

0.0  4.0  8.0  12.0  16.0  20.0  24.0
0.0  5.0  10.0  15.0  20.0  25.0  30.0
0.0  6.0  12.0  18.0  24.0  30.0  36.0
0.0  7.0  14.0  21.0  28.0  35.0  42.0
0.0  8.0  16.0  24.0  32.0  40.0  48.0
0.0  9.0  18.0  27.0  36.0  45.0  54.0
0.0  10.0  20.0  30.0  40.0  50.0  60.0
0.0  11.0  22.0  33.0  44.0  55.0  66.0

*****
Estatus de la paralelizacion:

Enviando 2 filas al proceso n° 1 offset=0
Enviando 2 filas al proceso n° 2 offset=2
Enviando 2 filas al proceso n° 3 offset=4
Recibiendo resultado del proceso numero: 1
Recibiendo resultado del proceso numero: 2
Recibiendo resultado del proceso numero: 3

*****
Resultado de la Matriz C de 6x7:

0.0  506.0  1012.0  1518.0  2024.0  2530.0  3036.0
0.0  572.0  1144.0  1716.0  2288.0  2860.0  3432.0
0.0  638.0  1276.0  1914.0  2552.0  3190.0  3828.0
0.0  704.0  1408.0  2112.0  2816.0  3520.0  4224.0
0.0  770.0  1540.0  2310.0  3080.0  3850.0  4620.0
0.0  836.0  1672.0  2508.0  3344.0  4180.0  5016.0

*****
Finalizado.

real    0m25.635s
user    0m0.161s
sys      0m0.149s
linuxtest@NodoMaestro:~/share$ _

```

## 6. Código del programa

A continuación se visualiza el código en lenguaje C que permite obtener el producto entre dos matrices:

```

1  /* *****
2  * ARCHIVO: matrices.c
3  * DESCRIPCION: Multiplicacion por Matrices - Implementado en Lenguaje C
4  * El nodo Maestro distribuye las operaciones para multiplicar las matrices.
5  * Las tareas se separan segun la cantidad de clusters asignados para la
6  * paralelizacion, que sera siempre n-1
7  * AUTOR: Blaise Barney. Adaptado por Emiliano Salvatori
8  * FECHA: 01/10/2020
9  * ***** */
10 #include "mpi.h"
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 #define NCA 10          /* Numero de Columnas en la Matriz A */
15 #define NFA 5           /* Numero de Filas en la Matriz A <--*/
16 #define NCB 5           /* Numero de Columnas en la Matriz B <-- */
17 #define MAESTRO 0       /* ID para la primer tarea */
18 #define DEL_MAESTRO 1   /* Se configura un tipo de mensaje del nodo maestro */
19 #define DEL_NODO 2      /* Se configura un tipo de mensaje de los nodos */
20
21 int main (int argc, char *argv[]) {
22
23     int    numtareas ,          /* Numero de tareas en ejecucion */

```



```

24         taskid,          /* Identificador de las tareas */
25         numprocesos,     /* Numero de procesadores que tomaran la tarea */
26         origen,          /* Id de la tarea segun el mensaje de origen */
27         dest,             /* Id de la tarea segun el mensaje destino */
28         menstipo,         /* Tipo de mensaje */
29         filas,            /* Numero de filas de A enviadas a cada nodo */
30         averow, extra, offset, /* Determina la cantidad de filas para cada nodo */
31         i, j, k, rc;      /* Contadores */
32     double a[NFA][NCA],   /* Matriz A que sera multiplicada */
33            b[NCA][NCB],   /* Matriz B que sera multiplicada */
34            c[NFA][NCB];   /* Resultado de la matriz C */
35
36     /* Permite instanciar una estructura que contiene informacion del mensaje enviado */
37     MPI_Status status;
38
39     /* Se inicializa el entorno de ejecucion para MPI. */
40     MPI_Init(&argc,&argv);
41
42     /* Determina el rango del proceso de llamada en el controlador de seniales
43     INPUT: recibe el controlador de seniales
44     OUTPUT: el rango del proceso de llamada el el grupo del controlador de seniales */
45     MPI_Comm_rank(MPI_COMM_WORLD,&taskid);
46
47
48     /* Determina el tamaño del grupo de los nodos asociados con el programa main
49     Recibe el parametro de entrada que vendria a ser el controlador de seniales
50     OUTPUT: numero de los proceso de llamada el el grupo del controlador de seniales */
51     MPI_Comm_size(MPI_COMM_WORLD,&numtareas);
52
53     /* En caso que el numero de procesadores a ejecutar sea menor a 2, entonces abortar */
54     if (numtareas < 2 ) {
55         printf("Se requieren al menos 2 procesos para ejecutar el programa. Abortando...\n");
56         MPI_Abort(MPI_COMM_WORLD, rc);
57         exit(1);
58     }
59
60     numprocesos = numtareas -1;
61
62     /* ***** Proceso Maestro ***** */
63
64     if (taskid == MAESTRO) {
65
66         printf("\n");
67         printf("Se inicia MPI con un numero de %d de procesos.\n",numtareas);
68
69         for (i=0; i<NFA; i++)
70             for (j=0; j<NCA; j++)
71                 a[i][j]= i+j;
72         for (i=0; i<NCA; i++)
73             for (j=0; j<NCB; j++)
74                 b[i][j]= i+j;
75
76
77         /* Se tratan de imprimir los datos por pantalla de las matrices */
78
79         printf("\n");
80         printf("Dimensiones de Matriz A: \t %d x %d\n",NFA, NCA);
81         printf("Contenido de la Matriz A de :\n");
82
83         for (i=0; i<NFA; i++){
84             printf("\n");
85             for (j=0; j<NCA; j++){
86                 a[i][j]= i+j;
87                 printf("%6.1f\t", a[i][j]);
88             }
89         }
90
91         printf("\n");
92         printf("\n");
93         printf("Dimensiones de Matriz B: \t %d x %d\n",NCA, NCB);
94         printf("Contenido de la Matriz B de :\n");
95
96         for (i=0; i<NCA; i++){
97             printf("\n");
98             for (j=0; j<NCB; j++){
99                 b[i][j]= i+j;
100                 printf("%6.1f\t", b[i][j]);
101             }
102         }
103
104         printf("\n");
105         printf("\n");
106         printf("*****\n");
107         printf("Estatus de la paralelizacion: \n");
108
109         /* Se envia los datos de la matriz a los nodos */
110
111         averow = NFA/numprocesos;
112         extra = NFA%numprocesos;
113         offset = 0;
114         menstipo = DEL_MAESTRO;

```

```

115
116
117     printf("\n");
118     for (dest=1; dest<=numprocesos; dest++) {
119
120         filas = (dest <= extra) ? averow+1 : averow;
121         printf("Enviando %d filas al proceso nÂ° %d offset=%d\n",filas,dest,offset);
122         MPI_Send(&offset, 1, MPI_INT, dest, menstipo, MPLCOMM_WORLD);
123         MPI_Send(&filas, 1, MPI_INT, dest, menstipo, MPLCOMM_WORLD);
124         MPI_Send(&a[offset][0], filas*NCA, MPLDOUBLE, dest, menstipo,
125                 MPLCOMM_WORLD);
126         MPI_Send(&b, NCA*NCB, MPLDOUBLE, dest, menstipo, MPLCOMM_WORLD);
127         offset = offset + filas;
128     }
129
130     /* Se recibe los resultados de los nodos */
131     menstipo = DELNODO;
132     for (i=1; i<=numprocesos; i++) {
133
134         origen = i;
135         MPI_Recv(&offset, 1, MPI_INT, origen, menstipo, MPLCOMM_WORLD, &status);
136         MPI_Recv(&filas, 1, MPI_INT, origen, menstipo, MPLCOMM_WORLD, &status);
137         MPI_Recv(&c[offset][0], filas*NCB, MPLDOUBLE, origen, menstipo,
138                 MPLCOMM_WORLD, &status);
139         printf("Recibiendo resultado del proceso numero: %d\n",origen);
140     }
141
142     /* Se imprimen resultados */
143     printf("\n");
144     printf("..... \n");
145     printf("Resultado de la Matriz C de %dx%d:\n",NFA, NCB);
146
147     for (i=0; i<NFA; i++) {
148         printf("\n");
149         for (j=0; j<NCB; j++)
150             /* printf("%6.2f\t", c[i][j]); */
151             printf("%6.1f\t", c[i][j]);
152     }
153
154     printf("\n");
155     printf("..... \n");
156     printf("Finalizado.\n");
157     printf("\n");
158 }
159
160 /* ..... Nodos Esclavos ..... */
161
162 if (taskid > MAESTRO) {
163
164     menstipo = DELMAESTRO;
165     MPI_Recv(&offset, 1, MPI_INT, MAESTRO, menstipo, MPLCOMM_WORLD, &status);
166     MPI_Recv(&filas, 1, MPI_INT, MAESTRO, menstipo, MPLCOMM_WORLD, &status);
167     MPI_Recv(&a, filas*NCA, MPLDOUBLE, MAESTRO, menstipo, MPLCOMM_WORLD, &status);
168     MPI_Recv(&b, NCA*NCB, MPLDOUBLE, MAESTRO, menstipo, MPLCOMM_WORLD, &status);
169
170     for (k=0; k<NCB; k++)
171         for (i=0; i<filas; i++) {
172             c[i][k] = 0.0;
173             for (j=0; j<NCA; j++)
174                 c[i][k] = c[i][k] + a[i][j] * b[j][k];
175         }
176
177     menstipo = DELNODO;
178
179     MPI_Send(&offset, 1, MPI_INT, MAESTRO, menstipo, MPLCOMM_WORLD);
180     MPI_Send(&filas, 1, MPI_INT, MAESTRO, menstipo, MPLCOMM_WORLD);
181     MPI_Send(&c, filas*NCB, MPLDOUBLE, MAESTRO, menstipo, MPLCOMM_WORLD);
182 }
183
184 /* Finaliza la entorno de ejecucion MPI */
185 MPI_Finalize();
186 }

```

## 7. Conclusiones

Con este Trabajo, se pudo adentrar en las prácticas de los sistemas operativos de tipo GNU/Linux. Asimismo se pudo comprender de una forma más cercana el funcionamiento de programas cuya ejecución se realiza de forma distribuida empleando las librerías MPI.

Luego de esto, se pudo comprobar la mejora de tiempos con el uso del protocolo

NTP, el cual es un protocolo basado en un sistema cliente-servidor; el mismo provee a los clientes con tres productos fundamentales: clock offset, round-trip delay y referencia de dispersión: el Offset especifica la diferencia entre la hora del sistema local y la referencia externa de reloj; el Round-trip delay especifica las latencias de tiempo medidas durante la transferencias de paquetes dentro de la red; y la Referencia de dispersión de tiempo especifica el máximo número de errores asociados con la información de tiempo recibido de un reloj externo.

Cabe destacar que las configuraciones realizadas para establecer una correcta comunicación entre todo el sistema (tanto para la sección 2.4 como para la 5) hizo posible que se comprendiera de mejor forma el funcionamiento del File System de Linux.