



Universidad Católica
San Pablo

CIENCIA DE LA COMPUTACIÓN

Configuración MPI Cluster

Laboratorio - 3

Rodrigo Alonso Torres Sotomayor

CCOMP 8-1

"El alumno declara haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

Configuración MPI Cluster

Introducción

Un clúster MPI (Message Passing Interface) es un conjunto de computadoras interconectadas que colaboran en tareas de procesamiento paralelo y distribuido. MPI permite que estas máquinas se comuniquen entre sí mediante mensajes para resolver problemas científicos y de ingeniería de manera eficiente. Cada nodo en el clúster realiza cálculos independientes y comparte información a través de la red para lograr una mayor capacidad de procesamiento. En este trabajo se verá como poder configurar un MPI cluster, para ello se utilizará máquinas virtuales para realizar dicha conexión.

Requisitos de la PC

En este trabajo se utilizó una máquina virtual y las características de dicha máquina son:

- Sistema Operativo: Linux.
- Memoria base (RAM): 4 GB
- Procesador: 4CPUs
- Memoria VRAM: 17mb
- Sistema Operativo: Ubuntu 20

Configuración

0.1 SSH

Permite la comunicación remota entre master y cliente. Los mensajes estan encriptados por ello se necesita compartir la clave con los clientes para no realizar una verificación de identidad.

```
sudo apt-get install openssh-server
```

```
sudo apt-get install openssh-client
```

0.2 NFS

El NFS (Network File System) es un protocolo que permite el acceso a archivos a través de la red. Esto permite a un usuario poder leer, escribir, ejecutar o modificar archivos como si estuvieran en su propia máquina. Es por ello necesario instalar el nfs tanto para el master como para el cliente:

```
sudo apt-get install nfs-server
```

```
sudo apt-get install nfs-client
```

0.3 Pasos siguientes

Una vez instalados procederemos a ingresar estos comandos en el cliente:

```
sudo mkdir /mirror
```

```
echo "/mirror-*(rw,sync)" | sudo tee -a /etc/exports
```

```
sudo service nfs-kernel-server restart
```

```
sudo touch /mirror/sampleFileForMySlaves
```

```
sudo mount master:/mirror /mirror
```

```
echo "master:/mirror----/mirror----nfs" | sudo tee -a /etc/fstab
```

```
sudo apt-get install openssh-server
```

```
sudo adduser --home /mirror --uid 1100 mpi
```

En la terminal del master se ingresarán estos comandos, con los que se dará la autorización de las claves para el cliente.

```
su - mpi
```

```
ssh-keygen -t rsa
```

```
cd .ssh
```

```
cat id_rsa.pub >> authorized_keys
```

```
sudo apt-get install build-essential
```

```
sudo apt-get install mpich
```

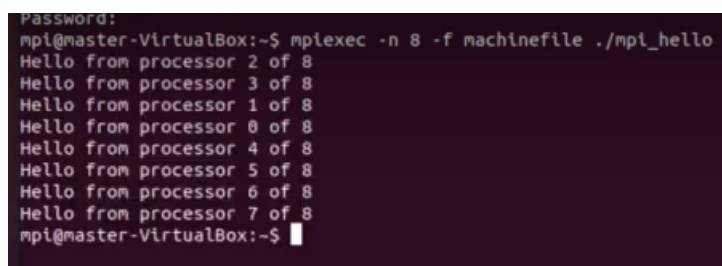
Probar la configuración del cluster

Para probar que la configuración está completa, se utilizó un pequeño código donde solo se realizará una impresión de que los procesos del master y el cliente se están ejecutando. Para esto se ingresarán estos comandos en el terminal del master:

```
su - mpi
```

```
mpicc mpi_hello.c -o mpi_hello
```

```
mpiexec -n 8 -f machinefile ./mpi_hello
```



```
Password:
mpi@master-VirtualBox:~$ mpiexec -n 8 -f machinefile ./mpi_hello
Hello from processor 2 of 8
Hello from processor 3 of 8
Hello from processor 1 of 8
Hello from processor 0 of 8
Hello from processor 4 of 8
Hello from processor 5 of 8
Hello from processor 6 of 8
Hello from processor 7 of 8
mpi@master-VirtualBox:~$
```

Figure 1: Compilación mpi-hello.c

Los resultados se pueden observar en la Figura 1.

Aplicación Odd Even Sort

En esta prueba se utilizó el algoritmo de ordenamiento Odd Even Sort y a este se le hará la prueba a través del MPI cluster comparándolo con una versión que no es MPI. Para se obtuvieron los siguientes resultados:

Procesos	200	400	800
1	0.000079	0.000253	0.000949
2	0.000016	0.000022	0.000038
4	0.000035	0.000059	0.000075
8	0.000110	0.000154	0.000058

	200	400	800
1	0.00004	0.000181	0.000909

Como se observa en ambas tablas cuando se trabaja sin el cluster MPI de las máquinas virtuales puede ser rápido al inicio pero conforme aumente la cantidad de elementos a ordenar el tiempo va aumentando y no es muy eficiente. Por otro lado al usar MPI y vemos como los tiempos al trabajar con un proceso no es muy óptimo pero al utilizar 8 procesos y una mayor cantidad de datos se refleja en el tiempo como está optimización del código ahorra tiempo y es eficiente.

Link del repositorio

<https://github.com/RodATS/ComputacionParalela.git>