MPI cluster sobre una red LAN

Luigy Machaca Arcana, Gabriel Aparicio Tony, Luis Caceres Zegarra

12th April 2017

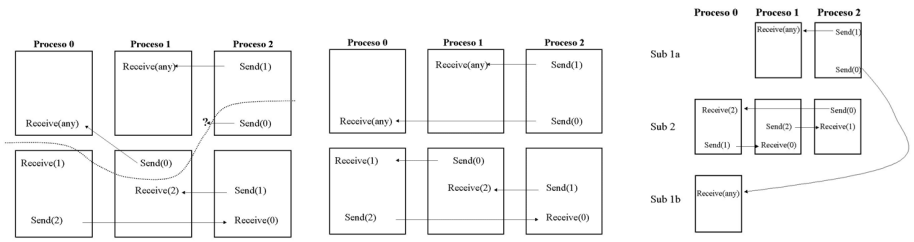
1 ¿Que es MPI?

Message Passing Interface, API[1] más utilizado en sistemas de paso de mensajes. Esta librería puede ser utilizada desde programas escritos en **C** o **FORTAN**. Como se menciono MPI está basado en un modelo de paso de mensajes. Otro dato importante a resaltar es que, fue desarrollada por una comunidad de programación paralela con el proposito de estandarizar las subrutinas de cómputo masivo.
Es usado en arquitecturas **MIMD** [2] de ordenadores de memoria compartida.

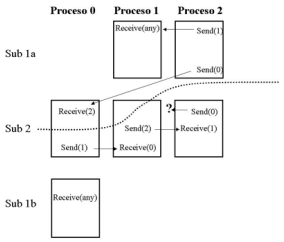
1.1 Estructura básica

- La unidad básica son los procesos (ver Figura 1.1.2)
- No existe mecanicos para asignar en tiempo de ejecución; procesos a procesadores. Ni tampoco creación y destrucción (ver Figura 1(b))
- Soporta agrupación dinámica de procesos. (ver Figura 1(c))
  - Grupos puede ser creados y destruidos
  - Relacion de pertenencia es estática
  - Existe intersección entre grupos

[1] Application Programming Interface
[2] Multiple Instruction, Multiple Data



(a) a (b) b (c) c



(d) c

Figure 1: Estructura básica (**1**)

1.1.1 Procesos y mensajes

- Un proceso es especificado por:
  - Grupo al cual pertence
  - Ranking relativo al grupo
- Un mensaje es especificado por:
  - Contexto de mensajes
  - Identificador asociados al contexto

1.1.2 Comunicators

- Usado para crear diferente *universos* de mensajes
- Agrupan *Grupos de procesos* y *Contextos de mensajes*
- evita *Deadlocks*[3] en el envio y recepcion de mensajes, mediante:
  - mensajes asincronos

[3]tiempos muertos

– Sincronía, pero hay mensajes pendientes

Figure 2: Objeto (**1**)

2 Configurando MPI sobre una red LAN

2.1 pre-requisitos:

Se debe tener instalado:

- **MPICH2** (http://www.mpich.org/)
- **SSH** http://www.openssh.com/

Seguir los siguientes pasos:

Paso: 1

Configuracion del archivo "hosts": se debe de agregar al final del archivo ubicado en "/etc/hosts", en modo super usuario de la siguiente manera:

```
#MPI CLUSTER SETUP
192.168.1.58 master
192.168.1.61 client2
192.168.1.37 client3
192.168.1.49 client4
...
```

Paso: 2

Creando Usuarios: Para esta configuración deberemos crear un mismo usuario *"client"* en todos los ordenadores involucrados; esto tanto para el *master* como para **client2,client3, ... etc**. Para esto usaremos los siguientes comandos en el terminal.

- usuario: client
- password:client

```
$ sudo adduser client
```

Agregar *client* al grupo **sudo**, con lo siguiente:

```
$ sudo adduser client sudo
```

Paso: 3

Configurando SSH: Los siguiente pasos se deben dar en el ordenador que hará de **servidor**.

- Logearse con la cuenta **client**

```
$ su client
```

- Instalar *openssh-server*.

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

- Generar una clave de tipo **dsa**; para cada una de los ordenadores involucrados(*master,client2,client3,etc.*).

```
$ ssh-keygen -t dsa
```

- Ejecutar solo en el ordenador **master**.

```
$ ssh-copy-id client2
$ ssh-copy-id client3
...
$ ssh-copy-id clientN
```

- Ejecutar solo en el ordenador lo siguiente **master**.

```
$ eval `ssh-agent`
$ ssh-add ~/.ssh/id_dsa
```

- A manera de comprobación en el ordenador **master**.

```
$ ssh client2
$ ssh client3
$ ssh client4
...
$ ssh clientN
```

El resultado en cada una de las ejecuciones debe de ser acceso al ordenador en cuestión sin necesidad de algun *password*.

Paso: 4

Configurando NFS-server: Los siguiente pasos se deben dar en el ordenador que hará de **master**.

- Instalar en **nfs** solo en **master**, con el siguiente comando.

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

- Para cada ordenador involucrado crear una carpeta con el mismo nombre, en este caso usaremos el nombre de **cloud**.

```
$ mkdir ~/cloud
```

- Ahora debemos dar permiso para el acceso a dicha carpeta (en **master**), para agregaremos lo siguiente, en el archivo */etc/exports* (al final del archivo).

```
1  /home/mpiuser/cloud 192.168.0.61(rw,sync,
       no_root_squash,no_subtree_check)
2  /home/mpiuser/cloud 192.168.0.37(rw,sync,
       no_root_squash,no_subtree_check)
3  /home/mpiuser/cloud 192.168.0.49(rw,sync,
       no_root_squash,no_subtree_check)
4   ...
```

Para cada uno de los ordenadores que harán de **client** (*client2, client3, etc.*)

- Una vez terminado debemos de reiniciar servicios y actualizar el archivo anterios. Para esto utilizaremos los sigueintes comandos.

```
$ sudo exportfs -a
$ sudo service nfs-kernel-server restart
```

Paso: 5

Configurando NFS-client: Los siguiente pasos se deben dar en el los ordenadores que harán de **client** (*client2, client3, etc.*).

- Instalar **NFS**

```
$ sudo apt-get install nfs-common
```

- En cada ordenador debemos montar la carpeta compartida por **master**.

```
$ sudo mount -t nfs master:/home/mpiuser/cloud ~/cloud
```

- Para comprobar que dicha sincronización con la carpeta *cloud* se completo, ejecutaremos lo siguiente:

```
$ df -h
```

- El resultado de la ejecución anterior debe de ser lo siguiente

```
Filesystem          Size  Used Avail Use\% Mounted on
master:             49G   15G   32G  32\% /home/client/cloud
/home/client/cloud
```

Paso: 6

textbfEjecutando un ejemplo: Ubicandonos en **master**, estando logeado con el usuario *client*.

- Ejecutaremos el sigueinte codigo, para probar que el MPI cluster sobre una LAN. Que estará ubicado en */home/client/cloud*. Con el nombre de main.c
- El ejemplo para calcular el promedio de un vector de grandes dimencion (en el ejemplo: **10 000 000** de datos)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <mpi.h>
4
5  float compute_avg(float* p,int size)
6  {
7          float avg = 0;
8          int i;
```

```
9           for(i=0;i<size;i++)
10          {
11                  avg+=p[i];
12          }
13          return avg/size;
14  }
15
16  int main()
17  {
18          int world_rank,world_size;
19          MPI_Init(NULL,NULL);
20          MPI_Comm_size(MPI_COMM_WORLD,&world_size
               );
21          MPI_Comm_rank(MPI_COMM_WORLD,&world_rank
               );
22
23          int elements_per_proc = 10000000;
24          float* rand_nums = NULL;
25          int i,arr_size = world_size*
               elements_per_proc;
26
27
28          if (world_rank == 0) {
29
30                  rand_nums = malloc(sizeof(float)
                       *arr_size);
31                  for(i=1;i<=arr_size;i++)
32                  {
33                          rand_nums[i-1]=i*1.0;
34                  }
35          }
36
37
38          float *sub_rand_nums = malloc(sizeof(
               float) * elements_per_proc);
39
40
41          MPI_Scatter(rand_nums, elements_per_proc
               , MPI_FLOAT, sub_rand_nums,
               elements_per_proc, MPI_FLOAT, 0,
               MPI_COMM_WORLD);
42
43
```

```
44          float sub_avg = compute_avg(
               sub_rand_nums, elements_per_proc);
45
46          float *sub_avgs = NULL;
47          if (world_rank == 0) {
48            sub_avgs = malloc(sizeof(float) *
                 world_size);
49          }
50          MPI_Gather(&sub_avg, 1, MPI_FLOAT,
               sub_avgs, 1, MPI_FLOAT, 0,
51                      MPI_COMM_WORLD);
52
53
54          if (world_rank == 0) {
55            float avg = compute_avg(sub_avgs,
                 world_size);
56            printf("The average is: %f\n",avg);
57          }
58
59          MPI_Finalize();
60  }
```

- mpicc

```
$ mpicc -o mpi_example main.c
```

- mpiexec

```
$ mpiexec -np 19 -hosts master,client2,client3,etc ./
    mpi_example
```

References

[1] http://users.dcc.uchile.cl/ ynakanis/pagina/presenta.html

[2] http://www.hpcc.ecs.soton.ac.uk/EandT/courseware/MPI/introduction.html