Cluster Computing

Name: Konstantinos Papadopoulos

Keywords: Raspberry Pi, Cluster, MPI, MPICH, RASPBIAN.

Table of Contents

1 Introduction

The way which allows you to have several computers connected and have them working together so as to be seen as a single system, we call it computer cluster. In addition, every single node of a computer cluster is set to do the same task, as long as software control and schedule them. All of the elements of the cluster are connected to each other through fast local area networks in order to communicate with each other as they also use the same OS(Operating System). The main reason of building a cluster is the need of improving high performance speed and availability over that performance, to be able to use low-cost microprocessors, high-speed networks and software when we have to work on distributed computing.[4] To sum up, we have built a small compute cluster using several Raspberry Pi modules which will use the same operating system and particularly Raspbian. Furthermore, a Raspberry Pi module is a single-board computer with an ARM processor. To power it on we can use a USB or a mobile's phone charger, it has a built-in 100Mb/s Ethernet and everything used by the module, even the operating system is stored in an SD card (usually 4GB or 16GB capacity).

As for the computer cluster we could add that, the running parallel programs on the one of the nodes uses the processing power of all the nodes and produce the output(the result). Regarding to the motherboards, they will be stacked into a single cabinet and connected using some interconnection network. They will also share RAM, Hard Disk and the other peripherals. The Operating System operates on one of the nodes and manipulate the activities of the other nodes.[11]

1.1 Building a small compute cluster

In this particular subsection we will enumerate the steps which have to be done, in order to build a small compute cluster using several Raspberry Pi modules. The aforementioned steps are the following[12]:

1. Download SD Memory Card Formatter from [1]
2. Download Raspbian Stretch.zip from[2]
3. Download Etcher.io for Windows x64 (63-bit) (Portable) from[3]
4. Boot on pi and input username and password.

   Username = "pi"

   Password = "raspberry"
5. In order to update and upgrade files we type these commands:

   Sudo apt-get update

   Sudo apt-get upgrade
6. We install MPICH[12]:

   Sudo apt-get install mpich

7. Just to check our ip address on raspberry pi even if we don't need to know it or the dns server because our ip addresses are in a private range and we do not have access to the internet. Those, clarifying the default gateway and dns servers in dhcpcd.conf are just for the requirements of the program that parses the file.
   Ifconfig

8. Now we have to set a static ip address. Static addresses were needed to make certain aspects of the configuration stability, and for the hostname to ip address mapping in /etc/hosts and the automatic mounting of the /mirror shared folder. We type these commands:

   Sudo nano /etc/dhcpcd.conf

   Interface eth0

   Static ip_address=10.0.1.x/24

   Static routers=10.0.1.32

   Static domain_name_servers=8.8.8.8 /*DNS of google*/

   Sudo systemctl reboot (to reboot)

9. Now we have to delineate a hostname in etc/hosts/. As for the /etc/hostname file it defines the hostname for our own machine. The /etc/hosts file maps hosts to ip addresses.

   Sudo nano /etc/hostname

   and we replace raspberry pi to slave x:

   Sudo nano /etc/hosts

   We change 127.0.1.1 raspberry into slave x.

   We now add all the other nodes

   10.0.1.30 master

   10.0.1.31 slave1

   10.0.1.32 slave2

   10.0.1.33 slave3

   10.0.1.34 slave4

   10.0.1.35 slave5

   10.0.1.36 slave6

   Sudo systemctl reboot

10. We now have to install NFS(Network File System), OpenSSH and the gcc compiler (which build-essential acquires it).

    Sudo apt-get install nfs-server nfs-client openssh-server build-essential

    All machines in the cluster need to afford to log in to each other. Thus, we need the ssh server running on all Raspberry Pi modules. We do it using the following commands:

    sudo systemctl enable ssh /*enables the ssh server at boot time*/

    sudo systemctl start ssh /*starts the ssh server*/

11. Sharing Master Folder. We now have to share the Master folder as we need a shared folder in order all to have for all the machines on the network, the same ssh key. It is an important step to ensure passwordless ssh because mpi programs won't work if the ssh connections are interrupted with questions for password.

    Sudo mkdir /mirror .We have to create the mirror folder in all of the modules.

12. We have to mount the master's mirror folder onto the mirror folder in each machine.
13. Sudo mount master:/mirror /mirror
14. Ls -l /mirror /*a quick check in the mirror folder-it has been mounted correctly or not*/
15. Sudo nano /etc/fstab
16. Master:/mirror /mirror nfs /*mounting the mirror master folder automatically*/

    Sudo systemctl reboot

17. Afterwards we have to mark a user for running the MPI programs and this process must be done on all nodes.

    sudo useradd -d /mirror -s /bin/bash mpiuser /*create a user with home directory as /mirror, default shell as /bin/bash and username mpiuser*/

    sudo passwd mpiuser /*set a password onto mpiuser so that we can log in to it when it is required*/

18. We now have to generate the ssh key in the masters machine(Raspberry Pi module). Before generating the key we have to log into mpi user on the master's node.
19. Ssh-keygen -t rsa

20. After that, we have to confirm among the nodes the passwordless ssh.
21. ssh mpiuser@hostname /*Replace "hostname" either with "master" if we're try to log in to master or "slavex".The number of the slave machine is "x". If we log in without being asked for a password everything is working fine.*/

22. Setting up a machine file. We have to create a file in master's home directory which is called "machine file", with node names that are followed by a colon and a number of processes to spawn.

    Slave1:4 /*this will spawn 4 processes on slave 1*/

    Slave2:3 /*this will spawn 3 processes on slave 2*/

    Slave4:1 /*this will spawn 1 process on slave 4*/

23. Now we have to do the testing.

    mpicc programname.c -o program /* programname.c → name of the file which contains the source code and program → name of the executable file created by the compiler*/

    mpiexec -n number_of_processes -f machinefile ./program /* -n →number_of_processes and shows how many processes we run in total. This is the sum of all processes which have been given to all of the nodes in the machine file. -f →machinefile states the machinefile that we use. ./program → specifies that we run the executable file which is called program.

24. Lastly, we have to prefix the mpiexec with “time” in order to for us to get running time for every call:

    time mpiexec -n number_of_processes -f machinefile ./program

    Our small compute cluster is now ready.

1.2 OS

We could choose the Linux operating system called RASPBIAN Stretch with Desktop or the RASPBIAN Stretch LITE version. We have chosen to install and use in particular the Linux operating system called RASPBIAN Stretch with Desktop because unlike the LITE version, it has a graphical user interface(GUI) which is user friendly and better than the LITE version. Additionally, it is an updated version of RASPBIAN, maybe faster than the former version and more adequate for the use of an advanced user. Thus, it is much more promising than the former version.

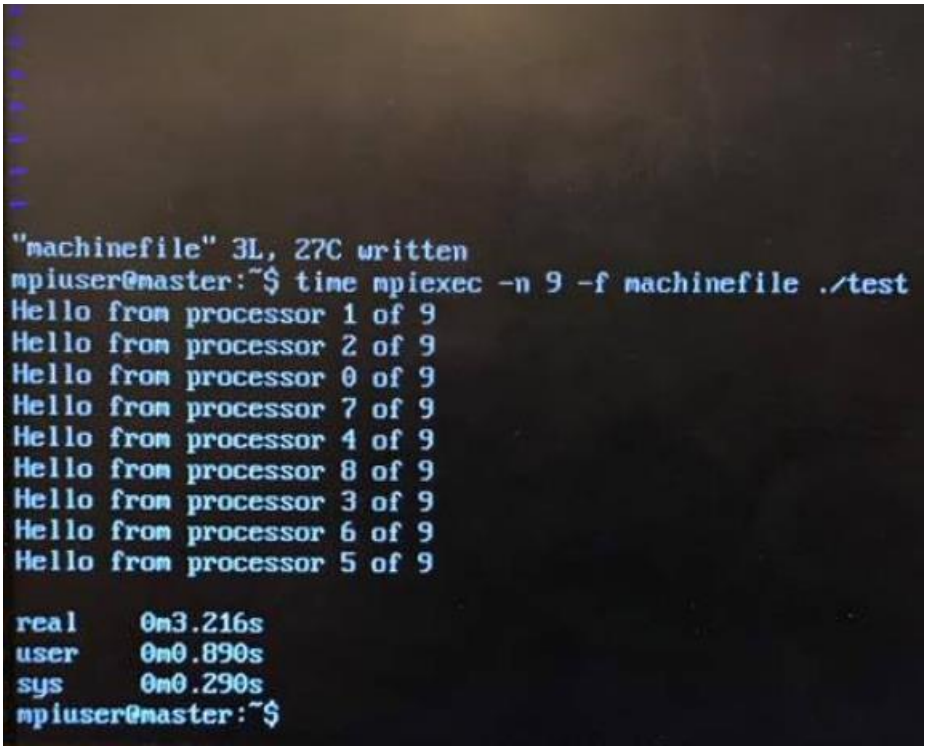
1.3 MPICH

During the procedure of building the compute cluster we could choose MPI(the Message Passing Interface software for cluster computing) or MPICH. We have chosen to install and use the MPICH over MPI because MPICH has highly broad network support, including both InfiniBand and proprietary interconnects like CraySeastar. In addition, MPICH and its derivatives create the most widely used performances of MPI in the world. [5][6][7][8][9]

1.4 Results

The small compute cluster that we have built works sufficiently. In the picture which we can observe below we can see that nodes indicate how many slaves(modules) we had online when we did the test, and we can observe that not all of the nodes were necessarily used as well. Additionally, it is important to know that the processes are always used per node, except the situation where there is an indication for a specific split. Last but not least, based on the result of this build, it best fits for big computational tasks, rather than small ones.

```
Nodes|Nodes_used|Processes|Duration
3,1,1,0.117
3,1,8,1.092
3,1,3,0.398
3,1,18,2.514
3,1,60,9.602
3,1,30,4.336
3,3,1,2.854
3,3,3,3.216
3,3,6,3.784
3,3,12,4.585
3,3,18/6/12,5.115
3,3,18,5.649
3,3,34/1/1,7.247
3,3,20,5.965
3,3,10,4.227
5,1,50,7.04
5,1,5,0.674
5,1,25,3.471
5,1,40,5.977
5,1,30,4.411
5,1,100,55.54
5,1,80,13.641
5,1,90,34.82
5,5,10,5.25
5,5,1,3.44
```



References

[1] SD Association, “SD Memory Card Formatter”. [Online]. Available at: https://www.sdcard.org/downloads/formatter_4/ . Last accessed: 09/11/2017.
[2] Raspberry Pi, “Raspbian’’.[Online]. Available at: https://www.raspberrypi.org/downloads/raspbian/ . Last accessed: 09/11/2017.
[3] ETCHER, “ETCHER”. [ONLINE]. Available at: https://etcher.io/ .Last accessed: 09/11/2017.
[4] Wikipedia, “Computer cluster”. [Online]. Available at: https://en.wikipedia.org/wiki/Computer_cluster Last accessed: 09/11/2017.
[5] Wikipedia, “MPICH”. [Online]. Available at: https://en.wikipedia.org/wiki/MPICH Last accessed: 09/11/2017.
[6] MPICH, “MPICH”. [Online]. Available at: https://www.mpich.org/ Last accessed: 09/11/2017.
[7] FUTURE ELECTRONICS, “Various interconnect”. [Online]. Available at: http://www.futureelectronics.com/en/interconnect-connectors/various-interconnect.aspx Last accessed: 09/11/2017.
[8] Wikipedia, “Infini Band”. [Online]. Available at: https://en.wikipedia.org/wiki/InfiniBand Last accessed: 09/11/2017.
[9] Stack overflow, “MPICH VS OPENMPI”. [ONLINE]. Available at: https://stackoverflow.com/questions/2427399/mpich-vs-openmpi Last accessed: 09/11/2017.
[10] Mahnoor Khan and Munam Ali Shah, “Inter-process communication, MPI and MPICH in microkernel environment: A comparative analysis”, in the proceedings of 23rd International Conference on Automation and Computing (ICAC), Huddersfield, United Kingdom 7-8 Sept. 2017.
[11] LINUX.COM, “Building a beowolf cluster in just 13 steps”. [Online]. Available at: https://www.linux.com/blog/building-beowulf-cluster-just-13-steps Last accessed: 09/11/2017.
[12] Ubuntu, “MPICH Cluster”. [Online]. Available at: https://help.ubuntu.com/community/MpichCluster Last accessed: 09/11/2017.