Building the Cluster with Raspberry Pi

1700889

10th November 2017

Contents

1 Introduction 2
2 Building the Cluster 2
2.1 Software Installation 2
2.2 Hardware Installation 2
2.3 Network Configuration 3
3 Performance Testing 6
3.1 Testing 6
3.2 Result 7
4 References 7

Lists of Figures

Figure 1: Series of Raspberry Pi 3

Lists of Tables

Table 1 List of IP Address and Node Name 4
Table 2 The Result of Master Node Performance 7
Table 3 The Result of Cluster Performance 7

Abstract

The purpose of this project is to create a small computing cluster from several raspberry pis. It was involved of five raspberry pis, one of them became the main node which shares its files into the other node. There are three main processes involved in creating a cluster from setting software, hardware as well as network. After that, testing with different numbers of processes and nodes to measure the performance of the cluster.

1 Introduction

The cluster was built by assigning one raspberry pi called "master" and the other four raspberry pi named "slave1", "slave2", "slave3" and "slave4". The workflow started with installing software, installing hardware, setting network configuration [1], testing and obtaining the result.

Important elements are needed to construct the cluster are the operating system, hardware, MPI. There are many operating systems can run on raspberry, for example, Raspbian, Ubuntu MATE. The reason behind choosing Raspbian is it the official and supported platform for raspberry pi [2]. Also, [2] states it is the easiest and more likely the fastest to update and fix its problems. In addition, stretch lite version of raspbian was opted as operating system because its size only 1,72 GB and without GUI (Graphical User Interface). It might be faster rather than the other types, such as Raspbian with desktop. MPI (Message Passing Interface) is the software of parallel programming which employs TPC/IP to split a task to process on multiple machines simultaneously [3]. There are two options for MPI, OpenMPI and MPICH. MPICH (Message Passing Interface Chameleon) was chosen for this project because [4] claims it is a high-performance and widely portable implementation of the Message Passing Interface (MPI) that equips an efficiently support for different computation and communication platforms.

2 Building the Cluster

To build the cluster, it should start from configuring the main node, later will be called as "master", then after finishing configuration on the master, it will continue to the other nodes.

2.1 Software Installation

1 Formatting memory card before using it to ensure there is no other file still stored. Download SD Memory Card Formatter from https://www.sdcard.org/downloads/formatter_4/. Format it with overwrite option.

2 Download operating system, Raspbian Stretch Lite from https://www.raspberrypi.org/downloads/raspbian/. Unzip this file.

3 Burning operating system image file of using etcher.io. Download portable version of Etcher.io from https://etcher.io/. After selecting this file, choose the drive of SD Card and flash it into the memory card. Also, this step must be done for the other nodes.

2.2 Hardware Installation

1 Preparing the component.

- 5 Raspberry Pi
- 5 SD Card 4 GB

- 5 HDMI Cable
- 5 Keyboard
- 5 LAN Cable
- 5 Power Supply

2 Install the component for each raspberry as shown in Figure 1.

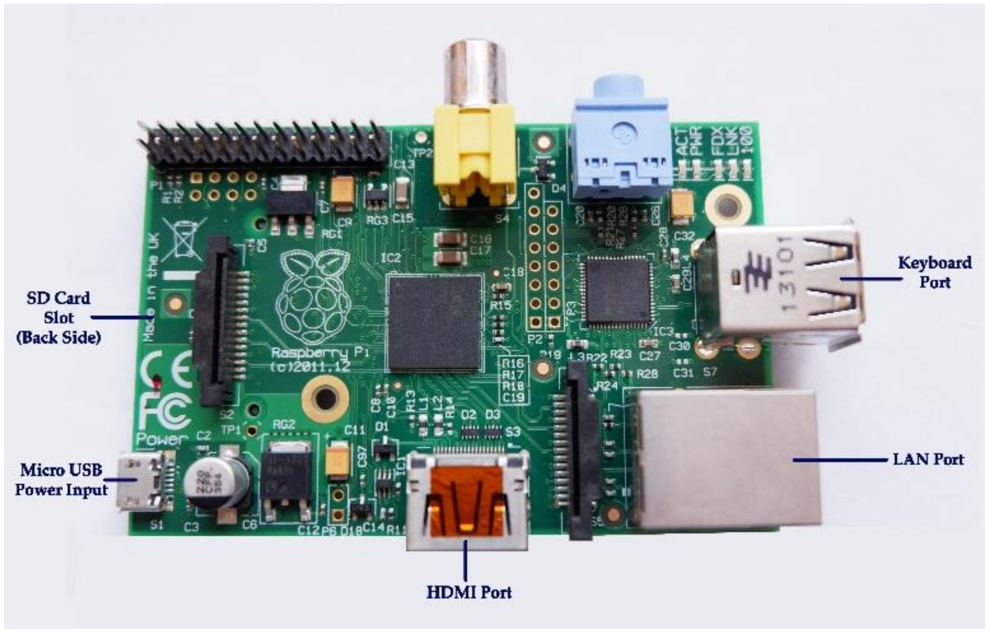

Figure 1: Series of Raspberry Pi

2.3 Network Configuration

1 Login into raspberry pi.
In the first time, the system will ask username and password as authentication.
username : pi
password : raspberry

2 Updating and upgrading files.
Assuring the system has been updated to the latest version, so it can be installed with the other settings, such as MPICH and NFS.

```
sudo apt-get update
sudo apt-get upgrade
```

3 Installing MPICH.
MPICH as message passing interface

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

4 Checking IP address on raspberry pi.

```
ifconfig
```

5 Setting a static IP address.
Determining IP address and name for each node.

No.	Node name	IP address
1	master	10.0.1.30
2	slave1	10.0.1.31
3	slave2	10.0.1.32
4	slave3	10.0.1.33
5	slave4	10.0.1.34

Table 1 List of IP Address and Node Name

IP address should be on the static network to make a stable configuration.

```
sudo nano /etc/dhcpcd.conf
```

Adding some commands in dhcpcd.conf.

```
interface eth0
static ip_address=10.0.1.30/24
static routers=10.0.1.30
static domain_name_servers=8.8.8.8
```

Later, for configuring the other nodes, input the IP address that had been decided before as shown in table 1.

Once this configuration has been successfully saved, reboot the system.

```
sudo systemctl reboot
```

6 Defining a hostname to machine.

```
sudo nano /etc/hostname
```

Replace the contain of hostname file which is “raspberry” with “nodename”.

```
master
```

7 Mapping all nodes.

```
sudo nano /etc/hosts
```

Change default IP address name from “raspberry” into node name.

```
127.0.1.1    master
```

Adding some commands in hosts file to register the other nodes.

```
10.0.1.31    slave1
10.0.1.33    slave2
10.0.1.33    slave3
10.0.1.34    slave4
10.0.1.35    slave5
10.0.1.36    slave6
```

If assign the other nodes, for instance mapping all nodes for slave1, just change default IP address “raspberry” into “slave1” and register the rest of nodes.

Thus, reboot the system.

```
sudo systemctl reboot
```

8 Installing NFS (Network File System), OpenSSH and GCC compiler.

NFS provides the access to create a folder on the master node and sync to other nodes. The function of GCC compiler is to compile all the code on the master node that located in build-essential.

```
sudo apt-get install nfs-server nfs-client openssh-server build-essential
sudo systemctl enable ssh
sudo systemctl start ssh
```

9 Sharing master folder.

To make the node can communicate each other, master folder must be mounted on the other nodes.

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

Checking if the /mirror folder has already mounted properly.

```
ls -l /mirror
```

Editing fstab file for automatically mounting.

```
sudo nano /etc/fstab
```

Adding a command in fstab file.

```
master:/mirror /mirror nfs
```

Rebooting the system after making few changes.

```
sudo systemctl reboot
```

10 Defining a user for MPI.

Giving the same username for each node to share the same SSH key, so it passwordless.

Setting /mirror folder as home directory as well as password for mpiuser.

```
sudo useradd -d /mirror -s /bin/bash mpiuser
sudo passwd mpiuser
```

11 Generating the SSH key.

This step is only required for master node, so the other nodes do not need to follow this step.

Login into mpiuser on the master.

```
mpiuser
```

Generating the ssh key.

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

12 Checking SSH is working between the nodes.

Login from the master or the other node to slave1.

```
ssh mpiuser@slave1
```

Thus, if the other node wants to login into the master, it need to type this command.

```
ssh mpiuser@master
```

13 Setting up a machine file.
Creating a file named “machinefile” in mpiuser’s home directory, define the name and number of processes.

```
master:10
 slave1:10
  slave2:10
  slave3:10
  slave4:10
  slave5:10
```

Number 10 on each node represents how many processes that will be assign.

3 Performance Testing

3.1 Testing

Changing directory to mirror folder and write these commands in a new file named mpi.hello.c.

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char** argv){
    int myrank, nprocs;
    MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    printf(“hello from processor %d of %d\n”, myrank, nprocs);
    MPI_Finalize();
    return 0;
}
```

Testing and obtaining duration of the process with these commands

```
mpicc mpi_hello.c -o mpi_hello
time mpiexec -n 5 -f machinefile ./mpi_hello
```

Number 5 in the command can be changed into others, it means the total processes that allowed to run this processing. Another wat to test only one or not all nodes, uncomment the node to deactivate and assign new number of process for each node.

3.2 Result

No.	Number of processes Master	 Each Node	 Total	Time (seconds)
1	5	-	5	0.674
2	25	-	25	3.471
3	30	-	30	4.411
4	40	-	40	5.977
5	50	-	50	7.040
6	80	-	80	13.641
7	90	-	90	34.820

Table 2 The Result of Master Node Performance

No.	Number of processes Master	 Each Node	 Total	Time (seconds)
1	1	1	5	3.440
2	5	5	25	4.390
3	6	6	30	4.531
4	8	8	40	4.861
5	10	10	50	5.250
6	20	20	100	7.400
7	80	80	400	21.400

Table 3 The Result of Cluster Performance

Based on the result of this experiment, it is better to use a cluster for big computation rather than small computation.

4 References

[1] Help.ubuntu.com. (2015). "Setting Up an MPICH2 Cluster in Ubuntu". [online] Available: https://help.ubuntu.com/community/MpichCluster [Accessed 9 Nov. 2017].

[2] P. Membrey and D. Hows, *Learn Raspberry Pi with Linux.* Apress, 2013

[3] J. Kiepert, "Creating a Raspberry Pi-Based Beowulf Cluster", pp. 1-2, May 2013. [online]. Available: http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf [Accessed 3 Nov. 2017].

[4] Mpich.org (2017). "MPICH Overview". [online] Available: https://www.mpich.org/about/overview/ [Accessed 3 Nov. 2017].