The Creation of a Low-cost Raspberry Pi Cluster for Teaching

Kevin Doucet
Mathematics and Computer Science Department
Texas Woman's University
Denton, Texas
kdoucet@twu.edu

Jian Zhang
Mathematics and Computer Science Department
Texas Woman's University
Denton, Texas
jzhang@twu.edu

ABSTRACT

Parallel programming is a computing model in which the computations are run on multiple processors simultaneously. In order to teach or learn parallel computing, there is a need for a computer system on which parallel programs can be executed. One such computing system is cluster computer in which a group of individual computers networked together to run multiple processes concurrently. Providing step-by-step instructions on building a cluster computer using 20 Raspberry Pi 3s, this experience report describes a low-cost cluster for teaching parallel computing in the undergraduate and high-school computing classrooms. The main contribution of this report is the detailed explanation on the software configuration including MPI (Message Passing Interface) when setting up the cluster. The MPI library described in this paper is Open-MPI.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**.

KEYWORDS

Cluster-Computer, MPI, Open-MPI, Raspberry Pi

ACM Reference Format:
Kevin Doucet and Jian Zhang. 2019. The Creation of a Low-cost Raspberry Pi Cluster for Teaching. In *Western Canadian Conference on Computing Education (WCCCE '19), May 3–4, 2019, Calgary, AB, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3314994.3325088

1 INTRODUCTION

With the increasing demand in solving intensive problems in data mining, business analytics, and data science, High Performance Computing (HPC) is becoming a topic of teaching interest in higher education institutions [5, 9]. To teach High Performance Computing, instructors should not only have a good understanding of the HPC algorithms but also the access to a HPC platform in order to execute those algorithms. Traditionally, HPC platforms, for example, Graphics Processing Units (GPU) and the Intel Xeon Phi have been used to achieve reliable results in solving problems that require huge computing power [7]. However, most of the current parallel computing machines need a high amount of energy to operate and are usually expensive for a small to medium size computer science department to obtain for teaching purpose.

A cluster computer is a computing system which can increase computational power of a computing system by connecting multiple computing units to run parallel algorithms. With the high costs associated with creating a large production cluster or even renting time on one, the ability to build a small cluster which is energy efficient and at a low cost can be beneficial to a wide range of users from students in high schools to instructors and researchers in higher education institutions [5].

Many publications can be found on the topic of Raspberry Pi-based mini cluster's performance compared to conventional clusters built from desktop computers or GPUs [4, 7, 8, 10]. However, there is very little information available on how these mini cluster computers were set up at the hardware level and their software configurations, especially the Message Passing Interface (MPI) configuration.

In this experience report, we describe the detailed setup of a low-cost cluster using 20 Raspberry Pis for computer science education which enable students and instructors to execute and test parallel programs.

2 RASPBERRY PI CLUSTER HARDWARE SETUP

Because of the low cost of an individual Raspberry Pi ($35.00), a cluster with 80 nodes (20 Raspberry PIs with quad-core processors) can be built for less than $900.00. This allows an organization such as a high school or a small computer science department in a college to set up a versatile cluster at a reasonable cost.

Figure 1: Raspberry Pi 3.

An individual Raspberry Pi (Figure 1) consists of a quad core ARMv8 1.2 Gigahertz CPU and 1gigabit of onboard RAM. It has four USB ports, an HDMI port, and a mini-USB charging port.

In this project, 20 Raspberry Pis are housed in a mobile cart which is equipped with pre-wired charging cables and a 24-port

WCCCE '19, May 3–4, 2019, Calgary, AB, Canada

ACM ISBN 978-1-4503-6715-8/19/05…$15.00
https://doi.org/10.1145/3314994.3325088

network switch. The mobile cart has twenty shelves which are arranged in two columns of ten shelves. The first step in creating the cluster was using this mobile cart to physically connect all twenty Raspberry Pis. Since a Raspberry Pi is only three inches long by two inches wide, we were able to fit 5 Raspberry on one shelf in the cart (Figure 2).

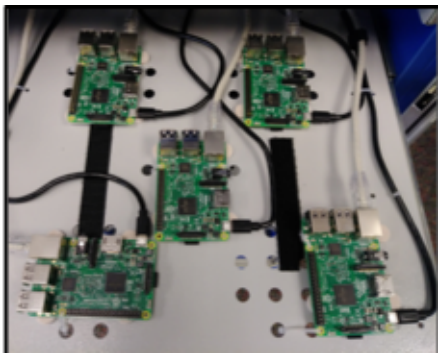

Figure 2: Five Raspberry Pis on one shelf.

The Raspberry Pi were zip tied onto the shelves so they would stay in place when the shelves were pulled out. Five Raspberry Pis were placed on each of the top the two shelves in each column (Figure 3). Each Raspberry Pi was connected to the networking switch inside the cart using an Ethernet cable.

Figure 3: 20 Raspberry Pis in the mobile cart and a wireless router on top of the mobile cart.

To power the Raspberry Pis, we used 2 ten-port USB charging hubs which were also placed on the top shelf of each column in the mobile cart. The ten Raspberry Pis in each column were connected to the USB hub in their respective columns. The network structure for this cluster is a star topology with the main node serving as the hub for the network.

Once the cluster was physically connected, the initial software set up of the master node can be started.

3 RASPBERRY PI CLUSTER SOFTWARE SETUP

In the next few sections, we will explain how to configure one Raspberry Pi as the Master Node and the setup of the other 19 Raspberry Pi as Slave nodes, including how to configure Open-MPI, IP addresses, and passwordless ssh setup.

The operating system installed in Raspberry Pi is the Raspberry Pi version of Linux (Raspbian) and an implementation of the MPI (Message Passing Interface) library from Open-MPI [6]. We chose to use Open-MPI over MPICH [3] because Open-MPI allows the use of more computing languages. For example, Open-MPI has bindings for C, C++, FORTRAN, and Java. Since the goal for our project is to create a test bed for people learning how to write and run parallel programs, we chose Open-MPI for its support to the above programming languages which are commonly taught in the current Computer Science curriculum across the world.

Raspbian is a version of Debian Linux created specifically for the Raspberry Pi and can be downloaded from the Raspberry Pi website [2]. Installer called Noobs is available from the website and it allows for the easy installation of Raspbian on the Raspberry Pi.

User can download Noobs and save it on a micro-SD card. Once the micro-SD card was installed with Noobs, it can be placed into a Raspberry Pi which has been hooked to a monitor and mouse and keyboard. This step is needed only for the initial set up of the Raspbian operating system.

Once Raspbian was successfully installed, ssh has to be enables in the system settings so that later on user can interact with the Raspberry Pi using ssh. Also, the name of this Raspberry Pi should be changed to PiMaster. After that, the Raspberry Pi can be powered down and the monitor, mouse and keyboard can be unhooked.

User can connect a wireless router to the switch on the mobile cart so that a computer can be connected wirelessly to the cluster through ssh (Figure 3). Once the Raspberry Pi PiMaster was turned on and access was gained through the ssh connection, system updates are done on the Raspberry Pi master by using the command apt-get update, then apt-get upgrade at the command line.

There are four programming languages FORTRAN, C, C++, and Java which Open-MPI has bindings for. Since Open-MPI requires a specific file from the Open JDK (Open Java Development Kit), Open-Java was installed instead of the Oracle Version. The current installations of C and C++ were upgraded to the newest versions. Gfortran was installed on the Raspberry Pi as well.

3.1 Open-MPI configuration

Once the programming languages are installed or updated, we downloaded the newest version of Open-Mpi from Open-Mpi website [1]. This tarball was untarred and then installed on the Raspberry Pi being used during the initial system setup. To activate the C++, FORTRAN, and Java bindings while installing Open-MPI, user should set the flags for activating the bindings.

The following commands set the flags for running on arm8 (Raspberry Pi) hardware which also activates C++, FORTRAN, and Java bindings.

CFLAGS=-march=armv7-a CCASFLAGS=-march=armv7-a ./configure --prefix=/usr/local --enable-mpi-fortran --enable-mpi-cxx --enable-mpi-java --with-jdk-bindir=/usr/bin/ --with-jdk-headers=/usr/lib/jvm/java-8-openjdk-armhf/include

The above command also points to a specific include file in the JDK which Open-MPI needs to use for the Java bindings.

To speed up the compilation of the make files, user can run the following command which allows the use all four cores of a Raspberry Pi's when the make files are created.

make -j 4

The creation of the make files is an automated process which can take quite a long time. The next command uses the make files that were just created to install Open-MPI on the system.

sudo make install

3.2 IP Address Configuration

In this section, we will explain the process to set up the static IP address for the Raspberry Pis. Although it's possible to use DHCP, we chose to use the static IP setup for its simplicity and less cost by using a single switch.

The process we explain here includes the setup of a temporary static IP address for the PiMaster, the copying of PiMaster software to all Raspberry Pis in the cluster, and the setup of final static IP addresses for PiMaster and all other Raspberry Pis in the cluster.

The first thing in setting up the PiMaster's static IP address is to edit the "etc/network/interfaces" file. This file should be edited by adding the following command at the bottom of the file:

eth0 dhcp

The above code makes the system to check an dhcp file for the IP address. Next, the "etc/dhcpcd.conf" configuration file should be edited by adding the following code at the bottom of the file:

interface eth0
static ip_address = 192.168.1.121

After we set up the static IP address, PiMaster was powered down and the micro-SD card was removed. The operating system image on this micro-SD card was copied and saved on the other 19 micro-SD cards for the cluster.

In order to set up the final IP addresses for the PiMaster, the first micro-SD card was placed back into the PiMaster and the "etc/dhcpcd.conf" configuration file was edited again. At the bottom of the file the following code was changed from:

interface eth0
static ip_address = 192.168.1.121

to:
interface eth0 (this line does not change)
static ip_address = 192.168.1.101 (only this line changes)

After we set up the final static IP address for the master node, we can access the slave nodes by their temporary static IP address 192.168.1.121 since the software was copied from MasterPi. For example, you can access one of the 19 slave nodes through ssh and the change the name of this node. Since this is the first slave node, we can assign its name as PiSlave1. This change can be done by running the raspi-config command:

sudo raspi-config

This command accesses the graphical configuration tool raspi-config. To change the name of this Raspberry Pi, choose the option "9 Advanced Options." in the graphical configuration tool. This leads to a second menu and the option to choose is Hostname (which allows for changes to the hostname). Here, we change the node's name to PiSlave1 and then exit the raspi-config tool.

In order to change the static IP address for the slave nodes, we use the same method as the one used to set up the master node's final IP address. For example, for the PiSlave1 node, its "etc/dhcpcd.conf" configuration file was edited and at the bottom of the file the following code was changed from:

interface eth0
static ip_address = 192.168.1.121

to:

interface eth0 (this line does not change)
static ip_address = 192.168.1.102 (only this line changes)

User can repeat he same steps described above to change the node name and the static IP address for the other 18 nodes. The IP addresses for the twenty nodes used in this project are 192.168.1.101 (master node) and 192.168.102-192.168.120 (slave nodes).

3.3 Passwordless ssh Setup

Once all the Raspberry Pi nodes had their hostnames changed and static IP addresses set up, user can turn on the nodes in the cluster and connect to the master node through ssh.

In order for the mater node to communicate with the other slave node, we created a 'nodefile' which stores all of the IP addresses of the nodes in the cluster. This 'nodefile' is only stored on the master node. The content of this file is the IP address of each node which is placed on a separate line respectively.

After the 'nodefile' was created, we then set up passwordless ssh for the master node. The reason for setting up the wireless ssh is that when the master node sends and receives data to and from the slave nodes it does not have to send a password. This increases the speed of computations and the functionality of the cluster. To set up the passwordless ssh, the following commands can be run on the master node:

ssh-keygen

Once the above command is executed, user needs to press enter through all questions without entering a password. This sets up the ssh without a password, thus passwordless ssh.

After creating the ssh key, run the following command to move to the home folder:

cd ~

and the next command to move to the .ssh folder:

cd .ssh

Next, copy the public key to a file called PiMaster:

cp id_rsa_pub PiMaster

To set up passwordless ssh for the slave nodes, access the slave node through ssh from the master node and run the same commands as described above but change the last command to the name of the respective slave node (PiSlave1 - PiSlave19).

Also, additional commands are executed for each slave node to allow master node's access to each slave node through ssh without a password. The commands below should be run on each slave node. The following command copies the master node's public key:

scp 192.168.1.101:/home/pi/.ssh/PiMaster .
Make sure the ending space and period is present or the above command will not work

The next command takes the above copied public key and saves it in the slave node's authorized_keys file:

cat PiMaster »authorized_keys

The next command exits the current slave node and goes back to the master node:

exit

The above commands set up the passwordless ssh so that the master node is able to connect to each slave node, and send and receive data without having to enter a password.

After all commands described above are run on each of the slave nodes, the setup of the cluster is complete and the cluster is ready to be tested.

4 TESTING OF THE CLUSTER

We ran several example parallel programs utilizing the MPI library to check the functionality of the cluster. Open-MPI has specific commands which first compile the parallel program and other commands to run the compiled programs. Since the main objective of this experience report was to build a working cluster instead of solving a specific problem, we used the example programs which are included in Open-MPI to test the functionality of the cluster.

To facilitate the testing, several controlling programs were used to assist in the compiling and running of the example programs on the cluster. These controlling programs were written in Perl and were all stored on the master node. The controlling programs allowed us to just specify the example program source file and the correct language compiler command will be chosen to compile the source file based on the file type. After the source file was successfully compiled, it was distributed to all the slave nodes on the cluster and the controlling program would select the corresponding command to run it for the language (C, C++, Java, or Fortran) of which the example file was compiled in.

In a previous pilot study which used 5 Raspberry Pis to create a cluster, a high computation problem and a low computation problem were tested and the cluster performance was analyzed and reported [5]. In this project, we were able to verify that the created Raspberry Pi cluster compiled an example program written in each of the supported languages. The communication between the master node and each of the slave nodes was completed successfully when distributing the compiled files; all compiled files were executed successfully on the cluster.

5 REFLECTION AND FUTURE WORK

This experience report describes the creation, configuration, and testing of a low-cost energy efficient Raspberry Pi cluster. The student author of this paper was motivated by his educational experience in the Texas Woman's University where no stand-alone distributed computing or parallel processing courses was available. The undergraduate computer science classes he took at the Texas Woman's University included Networking, Programming languages (Perl, Java, Assembly Language, web site development (PHP), and Databases (MySQL). He wanted to apply the textbook knowledge to a hands-on project in high-performance computing. He felt it would be beneficial to himself by gaining experience in setting up a cluster computer and also beneficial to the computer science department at Texas Woman's University by making the cluster available for teaching purpose. Although online clusters such as Amazon AWS, Google Cloud Platform, or Microsoft Azure would offer experience using the cluster functionality, they lack of the hands-on experience working with the hardware set up. This experience report is to share his experience of building a cluster computer with the broader computer science education community. After the cluster was used in a demonstration of high performance computing concept in a Applied Computational Thinking class with over 60 undergraduate students majoring Mathematics and Computer Science. An informal end of class survey reports more interest in the demo than reading the textbook description of high performance computing.

The future work involves the utilization of this Raspberry Pi cluster in several undergraduate computer science courses such as Applied Computational Thinking and Machine Learning at the Texas Woman's University. Data will be collected from the instructors who create new parallel computing topic related assignments and students from those courses and reported the findings as an experience report on the usage of a low-cost mini cluster in a small liberal arts computer science program.

ACKNOWLEDGMENTS

This work was partially supported Texas Woman's University's Research Enhancement Program.

REFERENCES

[1] last accessed April 15, 2019. Open MPI download. www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.gz

[2] Last accessed April 15, 2019. Raspbian Operating System download. www.raspberrypi.org/downloads/raspbian

[3] last accessed April 15, 2019), url = "https://www.mpich.org/",. High-Performance Portable MPI.

[4] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, and et al N. Preda. 2013. Affordable and energy-efficient cloud computing clusters: the bolzano raspberry pi cloud cluster experiment. In *Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, Vol. 2. 170–176.

[5] K. Doucet and J. Zhang. 2017. Learning Cluster Computing by Creating a Raspberry Pi Cluster. In *Proceedings of SouthEast Conference (ACM SE '17)*. 191–194. https://doi.org/10.1145/3077286.3077324

[6] Gabriel E. et al. 2004. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface. EuroPVM/MPI 2004*, D. Kranzlmüller, P. Kacsuk, and J. Dongarra (Eds.). Lecture Notes in Computer Science, Vol. 3241. Springer-Verlag. https://doi.org/10.1007/978-3-540-30218-6_19

[7] Saffran J. et al. 2017. A Low-Cost Energy-Efficient Raspberry Pi Cluster for Data Mining Algorithms. In *Euro-Par 2016: Parallel Processing Workshops*, Desprez F. et al (Ed.). Lecture Notes in Computer Science, Vol. 10104. Springer-Verlag. https://doi.org/10.1007/978-3-319-58943-5_63

[8] A. Mappuji, N. Effendy, M. Mustaghfirin, F. Sondok, R. P. Yuniar, and S. P. Pangest. 2016. Study of Raspberry Pi 2 quad-core Cortex-A7 CPU cluster as a mini supercomputer. In *Proceedings of 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*. 1–4. https://doi.org/10.1109/ICITEED.2016.7863250

[9] Jawwad Shamsi, Nouman M. Durrani, and Nadeem Kafi. 2015. Novelties in Teaching High Performance Computing. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*. 772–778. https://doi.org/10.1109/IPDPSW.2015.88

[10] Anuj Shinde, Shashank Pal, Vedant Singhvi, and Manish Ranat. 2017. Cluster Computing Using Raspberry PI. *IOSR Jornal of Computer Engineering* (2017), 6–9. http://www.iosrjournals.org/iosr-jce/papers/Conf.17014-2017/Volume-2/2.%2006-09.pdf?id=7557