

## Message Passing Interface and Hadoop

**Arafat Hussain, Joshua Fermin**

[ar.shezan@gmail.com](mailto:ar.shezan@gmail.com) , [joshuaf91@gmail.com](mailto:joshuaf91@gmail.com)

### 1. I. Message Passing Interface

**Abstract:** I choose this topic while doing research on parallel computing. What really caught my eye and sparked my interest was how during World War II, scientists used the same method of Parallel Computing to solve mathematical problems in the Manhattan Project for the atomic bomb. During World War II, well before the advent of the electronic computer, a similar technique was used for carrying out long calculations associated with the design of the atomic bomb for the Manhattan Project. To significantly reduce the amount of time it took to solve a large mathematical problem, each part of the problem was performed by a different person. Interestingly enough, the people who performed these calculations were called computers. Today electronic computers can work in harmony to solve scientific problems not dreamed of even a decade ago.

(<http://www.climate modeling.org/~forrest/osdj-2000-11/>) After reading that excerpt from [climate modeling.org](http://www.climate modeling.org) the topic was solidified. I also did some research on how costly it will be to replicate a Beowulf Cluster and found it would not be too costly to make a small 4 units replication of a cluster. The main research question is what is a Beowulf Cluster and its possible applications. The key ideas/concept behind the project would be performance comparison, what are the pros and cons to using a Beowulf Cluster compared to using a super computer, what are the difference between using a single computer to solve an issue and using a Beowulf cluster, how difficult and costly is it to build and implement working cluster. A few other related topics that I found related to Beowulf clusters are Finite element method, computational fluid dynamics, distributed computations, computational mechanics.

**Introduction:** MPI stands for Message Passing Interface. In my research I will be giving you step by step explanation on how to set up a Beowulf Cluster with inexpensive Raspberry Pi that implants a Message Passing Interface for Python (MPI4PY). I will also be going over how to set up static IP addresses to avoid having to constantly update your Machine file and how to Secure Socket Shell (SSH) into a pi and the ease of using WinScp to transfer data. I will also be walking you through some basic applications of MPI.

#### Background or Literature review

My main source is Youtube user [Rasim Muratovic](https://www.youtube.com/watch?v=JtX9IVDsqzg) (<https://www.youtube.com/watch?v=JtX9IVDsqzg>) who does have a walkthrough for setting up MPI4PY on a Raspberry Pi B+. In Rasims guild he does not go over how to use winSCP a program that makes transferring data very easy. Also Rasim does not talk about Static ips something that I found very useful for many reasons like remotely SSHing into the Raspberry Pi and also not having to constantly set up the Machine file which is used for on the master computer to know who to communicate with. I myself am using Raspberry Pi 3 but the setup is very similar to that of a Pi B+. I will also be installing more up to date software on the Pis. For the documentation on MPI I used (<http://pythonhosted.org/mmpi4py/usrman/tutorial.html#point-to-point-communication>) .

**Problem Statement:** Getting a working MPI using Raspberry Pi 3.

#### Proposed work:

#### Hardware needed for the setup.

##### Mandatory Hardware.

- 2 or more Raspberry pi 3
- X micro SD cards 8gb with adapter (x being the number of Raspberry pis you are setting up)
- X micro USB power cords (x being the number of Raspberry pis you are setting up)

##### Option Hardware

- Basic Network switch and x+1 CAT5 (Ethernet cables) (in this guild we will be setting up the cluster using the switch but it can also be done using a Wi-Fi connection)
- X Raspberry Pi 3 cases
- A multi USB power hub (not a data transfer hub) (Strongly Recommended)

##### Software

- Win32 disk Imager to install Operating system and copy your progress (<https://sourceforge.net/projects/win32diskimager/files/latest/download>)

- PuTTY to SSH into the Raspberry Pis (<https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe> )
- Win SCP to Transfer Data between the Pis with ease (<https://winscp.net/download/winscp577setup.exe>)
- Raspbian Jessi Lite (can also use the Full Desktop Version) ([https://downloads.raspberrypi.org/raspbian\\_lite\\_latest](https://downloads.raspberrypi.org/raspbian_lite_latest))

You should also have some basic knowledge of how to code in python, Linux commands, and a working internet connection.

### Setup

1. Set up Raspberry pis by connecting them to the switch and your switch to a working internet connection do not power the Raspberry pi up.
2. Extract Raspbian Jessi Lite on your desktop or laptop computer
3. Load Raspbian Jessi Lite into your SD Card using Win32 disk imager.
  1. Open Win32 disk imager click on the folder and find the directory Raspbian Jessi Lite is located in make sure the device is set to your Micro SD card and select write
  2. Once done burning the image to Micro SD card Insert SD Card into the Raspberry pi (on the bottom of the PI) and power it on.
  3. Open CMD and PuTTY to avoid issues make sure only one pi is connected.
    1. In CMD enter "ping raspberrypi" take note of the ip address given should look something like 192.xxx.x.xx
    2. Open PuTTY and insert the Ip address and open.
    3. Default Username is "pi" Password is "raspberry"
  4. In step 6 we will be expanding the File system, changing the hostname, and splitting the memory for the GPU.
    1. Type "sudo raspi-config"
    2. Select option 1 expand filesystem
    3. Now go to 9 advance options then A2 Hostname and change the host name to that of your choosing I will be naming and referring to this pi pi01
    4. Now go back into 9 advanced options and A3 Memory Split and give it a minimum of 16 (since we are not using much of the GPU)
    5. Tab over to finish and reboot
  5. In step 7 we will be setting up MPICH for the following steps you can copy and paste by first copying to your clipboard then right clicking in PuTTY to paste.
    1. Launch PuTTY and enter your pi01 ip address.
    2. Enter the following commands
      - i. sudo apt-get update
      - ii. sudo mkdir mpich2
      - iii. cd ~/mpich2
      - iv. sudo wget <http://www.mpich.org/static/downloads/3.2/mpich-3.2.tar.gz>
      - v. sudo tar xzf mpich-3.2.tar.gz
      - vi. sudo mkdir /home/rpimpi/
      - vii. sudo mkdir /home/rpimpi/mpi-install
      - viii. mkdir /home/pi/mpi-build
      - ix. sudo apt-get install gfortran
      - x. sudo /home/pi/mpich2/mpich-3.2/configure -prefix=/home/rpimpi/mpi-install
      - xi. sudo make
      - xii. sudo make install
      - xiii. cd .
      - xiv. nano .bashrc
    1. copy the following line to the end of the file. "PATH=\$PATH:/home/rpimpi/mpi-install/bin" press ctrl x to exit and y to save.

- xv. `sudo reboot`
  - xvi. Re-launch PuTTY and enter your pi01 ip address.
  - xvii. `mpixec -n 1 hostname` (if it returns your hostname then the installation is complete).
1. In step 8 we will be setting up MPI4PY
    1. SSH into your pi01
    2. Enter the following commands
      - i. `wget https://bitbucket.org/mpi4py/mpi4py/downloads/mpi4py-2.0.0.tar.gz`
      - ii. `sudo tar -zxf mpi4py-2.0.0.tar.gz`
      - iii. `cd mpi4py-2.0.0`
      - iv. `sudo aptitude install python-dev`
      - v. `python setup.py build`
      - vi. `sudo python setup.py install`
      - vii. `export PYTHONPATH=/home/pi/mpi4py-2.0.0`
      - viii. `mpixec -n 5 python demo/helloworld.py`
    1. if you are shown a message like this it means that your install was successful.
1. Rather than repeating this process on every pi we will now just copy an image of the SD card into different SD cards.
    1. Open win32 Disk imager.
    2. Insert your Micro SD card.
    3. Copy an image to your computer using the read button
    4. Now insert a new SD card and copy that image to your SD card using the write button. Repeat Step d for each new SD card.
    5. In this step we will be using nmap to get the ip addresses of the rest of your raspberry pis
      1. SSH into Pi01
      2. Enter the following commands
        - i. `sudo apt-get update`
        - ii. `sudo apt-get install nmap`
        - iii. `sudo nmap -sn 192.168.1.*` (the following IP Addresses will be shown as followed copy them down)
  1. Now we need to change the host names of the rest of the pis
    1. SSH into Pi01 (your original pi or what even pi you would like to set up as your master)
    2. Enter the following commands
      - i. `ssh pi@"enter ip address"`
      - ii. `password: "raspberrry"`
      - iii. `sudo raspi-config`
    1. change the name Pi02 and
    2. Press Tab until Finish is selected and reset
    3. Repeat on each pi and change the name to the next number
    4. Now we are going to create a machinefile this tells the master who it will be communicating with.
      1. SSH into pi01
      2. Enter the following command
        - i. `nano machinefile`
      1. enter the ip address for pi01
      2. enter the ip address for pi02
      3. enter the ip address for pi03

4. enter the ip address for pi04
- ii. press ctrl x and save
  1. if you enter "mpirun -f machinefile -n hostname" you will get it works for the first Pi but you get a permission denied lets fix that.
  2. SSH into pi01
  3. Enter the following commands.
    - i. Pi01
      1. ssh-keygen
      2. cd ~
      3. cd .ssh
      4. cp id\_rsa.pub Pi01
    - ii. Pi02
      1. ssh pi@(enter pi02 ipaddress)
      2. ssh-keygen
      3. cd .ssh
      4. cp id\_rsa.pub Pi02
      5. scp (enter pi01 ipaddress):/home/pi/.ssh/Pi01 .
      6. cat Pi01 >> authorized\_keys
      7. exit
    - iii. Pi03
      1. ssh pi@(enter pi01 ipaddress)
      2. ssh-keygen
      3. cd .ssh
      4. cp id\_rsa.pub Pi03
      5. scp (enter pi01 ipaddress):/home/pi/.ssh/Pi01 .
      6. cat Pi01 >> authorized\_keys
      7. exit
    - iv. Pi04
      1. ssh pi@(enter pi01 ipaddress)
      2. ssh-keygen
      3. cd .ssh
      4. cp id\_rsa.pub Pi04
      5. scp (enter pi01 ipaddress):/home/pi/.ssh/Pi01 .
      6. cat Pi01 >> authorized\_keys
      7. exit
    - v. scp (enter pi02 ipaddress):/home/pi/.ssh/Pi02 .
    - vi. cat Pi02 >> authorized\_keys
    - vii. scp (enter pi03 ipaddress):/home/pi/.ssh/Pi03 .
    - viii. cat Pi03 >> authorized\_keys
    - ix. scp (enter pi04 ipaddress):/home/pi/.ssh/Pi04 .
    - x. cat Pi04 >> authorized\_keys
    1. Test that this was done correctly.
    - i. cd ~

ii. `mpiexec -f machinefile -n 4 hostname`

1. If the last command returns all the host names, then your setup is complete.

Now that you have successfully set up your cluster let's go over some basic things you need to make working with MPI easy.

`winSCP` is used for transferring information from one computer to the next in our case from either your desktop /laptop to the raspberry pi. Very simple to use program `hostname` is the ip address of the raspberry pi and the username and password are both the default pi raspberry. From here you can transfer data with ease. All the programs that we will write must be in python so they must end with `.py`, also you must have a copy of the `.py` doc in the same directory on each pi. Now let's work with some basic commands.

1. To import use `"from mpi4py import MPI"`

2. Create an instance of `MPI.COMM_WORLD` to use the following functions I will be creating it using `comm = MPI.COMM_WORLD`

3. `comm.size` will give me the size of computers

4. `comm.rank` will give you the rank of the processor

5. `MPI.Get_processor_name()` will give you the host name of your node

6. `comm.send(data(a variable),dest=x(where the data will be going to))`

1. if I use the below `comm.send` I will be hardcode where I send the data in this case rank 1

i. `comm.send(data,dest=1)`

1. If I use the below `comm.send` I will be dynamically choosing where the data goes.

i. `comm.send(data,dest=(rank+1)%size)`

1. `comm.recv(source=x)` is used to receive data from a `.send` or `.bcast`

1. if I use the below `comm.recv` I will be hardcoding where I receive the data from.

i. `comm.recv(source=0)`

1. if I use the below `comm.recv` I will be dynamically choosing where I receive the data from.

i. `comm.recv(source=(rank-1)%size)`

1. `comm.bcast(data, root=x)` data will be the data you will broadcast across all of your systems, root will be where the data is coming from.

2. `comm.scatter(data,root=0)` data will be exploded and sent across all the nodes. Keep in mind that if data contains 5 different sets of information ie. [1,2,3,4,5] you will need at least 5 processes to be run or manipulate the data to fit the number of processes you will be running.

3. `comm.gather (data,root=0)` sends data back to root device

Some advantages of using MPI are that you can use it to do multi-threading and get full usage of your CPU. If a problem can be solved in parallel, you can use MPI to solve it. [Message Passing Interface.docx](#)

## 1. II. Hadoop

Hadoop is a framework of tools based on Java that supports running applications on big data. It is an open source set of tools and it is distributed under the Apache License. Hadoop basically deals with a large volume of variety data in a relatively short time. Hadoop is consisting of two main components, Hadoop Distributed File System (HDFS) and MapReduce. Hadoop was originally a file system and MapReduce implementation. First it was used by internet giants like Google, Yahoo, and Facebook. Recently it has grown into a complex ecosystem that includes a wider range of software systems, such as HBase (a distributed table store), Zookeeper (a reliable coordination service), Pig and the Hive high-level language that compile down to MapReduce [1]. In this research paper I only looked at HDFS and MapReduce just to figure out how Hadoop works in a cluster.

1. Big Data

According to IBM, 90% of the big data in the world today has been generated in the past six years alone and this accelerating trend is going to continue. Due to the expansion of technologies, people are engaging more than ever before and businesses feeling the needs to store all those data accurately. Big data consider of size more than 10TB of data.



Fig: 1- Data Center.

In early years, companies like Oracle, IBM, Microsoft etc. used to provide the database management systems to the small companies. In 2000's when companies like Google were dealing with their enormous amount of data, it became inconvenient for them to store in single database bottleneck. To solve this issue, Google labs team had come up with an algorithm that allowed large data files to split into smaller blocks and mapped into many computers, later after doing the calculations, data brought back together to produce the resulting data set. This algorithm process called MapReduce. This algorithm was later used to develop an open source project called Hadoop which allows applications to run using the MapReduce algorithm.

#### 1. Hadoop Distributive File System(HDFS)

Hadoop has a master and slave architecture. The master components of Hadoop are the NameNode and JobTracker and the slave components of Hadoop are DataNode and TaskTracker.

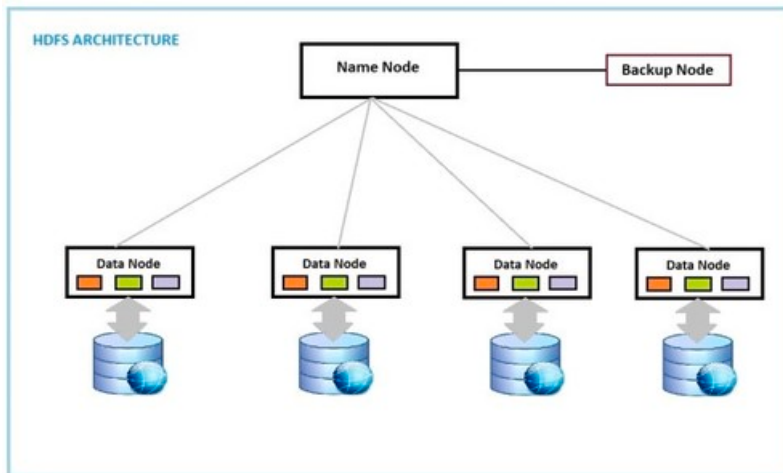


Fig: 2 – HDFS Architecture

NameNode is the centerpiece of Hadoop and also keep directory of all the files, Secondary NameNode assist NameNode and DataNode store data in the Hadoop file system. There is only one NameNode running in the master node and multiple DataNodes running in the slave nodes throughout the cluster. Hadoop has a built-in Fault Tolerance System. While storing data into DataNodes, Hadoop by default creates three replicas of the data, so if any nodes crush there should be other nodes that have the copy of the file.

#### 1. MapReduce

Data in a Hadoop cluster is broken down into smaller pieces that later distributed throughout the cluster. In this way, the MapReduce function can be implemented in parallel on a smaller subset of data.

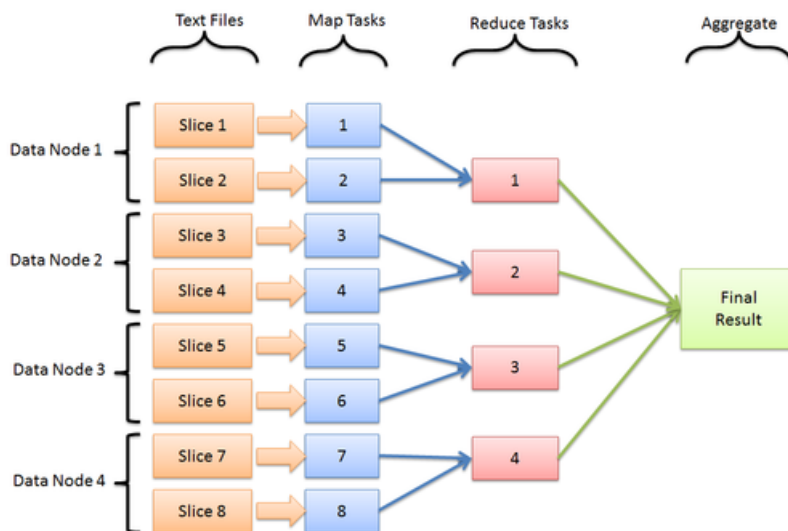


Fig: 3 – MapReduce Task

JobTracker is the service that send MapReduce on different nodes in the cluster. TaskTracker is a node which perform map, reduce and shuffle operation from the JobTracker. Not only DataNodes have duplicate data in them but also TaskTrackers have duplicates throughout the cluster.

### 1. III. Installing/Configuring Hadoop in Linux

#### 1. Configuring Linux

Before starting Hadoop installation in Linux there are few things need to be done. First of all, I would recommend updating the Linux version by typing this command-  
`"sudo apt-get update"`

The only prerequisite to install Hadoop in Linux is to have Java Development Kit (JDK) of version 6 or above. The command to check the JDK version is `"java -version"`.

`"sudo apt-get install default-jdk"`

JDK could be installed by the above command. Even if there is already an older version of JDK installed in the machine, the version can be updated by running the first update command.

In order to use Hadoop, you can dedicate a user account to run Hadoop.

`"sudo addgroup Hadoop"`

`"sudo adduser -ingroup Hadoop hduser"`

`"sudo adduser hduser sudo"`

Now we have to generate an SSH key for the hduser user. Hadoop requires SSH access to manage the nodes. To move to the hduser user account-

`"su hduser"`

`"ssh-keygen -t rsa -P "" "`

Press Enter until a random image appear. This will create an RSA key with an empty password; this allows accessing other nodes without any password. And to enable SSH access to the local machine, the following command need to be typed.

`"cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys"`

## 1. Downloading Hadoop

After doing all the configurations, now it's time to download and install Hadoop into the local machine.

```
" wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz "
```

The above command is to download Hadoop package in a .tar file format. After doing so the .tar file needs to be unzipped by the following command.

```
"tar xvzf hadoop-2.7.1.tar.gz"
```

Let's move the hadoop-2.7.1 folder to local directory.

```
"sudo mv -r hadoop-2.7.1 /usr/local/Hadoop"
```

Now the ownership of the directory needs to be changed to Hadoop dedicated user hduser.

```
" sudo chown -R hduser:hadoop /usr/local/Hadoop "
```

### 1. Configuring Hadoop

First file we want to configure is .bashrc which is executed for interactive non-login shells.

```
sudo nano /.bashrc
```

Append the following lines to the file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

```
export HADOOP_INSTALL=/usr/local/hadoop
```

```
export PATH=$PATH:$HADOOP_INSTALL/bin
```

```
export PATH=$PATH:$HADOOP_INSTALL/sbin
```

```
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
```

```
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
```

```
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
```

```
export YARN_HOME=$HADOOP_INSTALL
```

```
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
```

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
```

Now save and close the file and run this command to load the file to the script.

```
" source ~/.bashrc "
```

Changing the file configuration of hadoop-env.sh

```
" sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh "
```

Change the following line

```
export JAVA_HOME=${JAVA_HOME}
```

to this

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Now change the core-site.xml file configuration.

```
" sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml "
```

All the way to the end, in between two <configuration> tag add the following lines and save the changes.

```
<property>
```

```
<name>fs.default.name</name>
```

```
<value>hdfs://localhost:9000</value>
```

```
</property>
```

Now make a copy of the mapred-site.xml and rename it.

```
cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Now open the copied file to make some changes.

```
" sudo nano /usr/local/hadoop/etc/hadoop/mapred-site.xml "
```

Again all the way to the end, in between two <configuration> tag add the following lines and save the changes

```
<property>
```

```
<name>mapred.job.tracker</name>
```

```
<value>localhost:54311</value>
```

```
</property>
```

In this part we are going to create some directories for HDFS's namenode and datanode.

```
"sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode"
```

```
"sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode"
```

```
"sudo chown -R hduser /usr/local/hadoop_tmp"
```

Now we can configure the hdfs-site.xml file.



```
" sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml "
```

In between two <configuration> tag add the following lines and save the changes

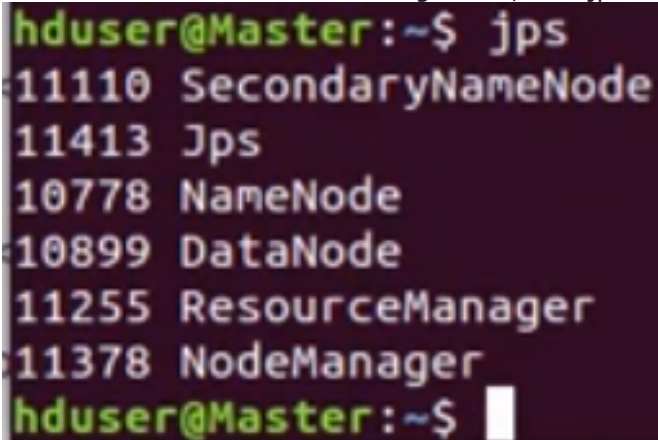
```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
```

Now it's time to format the namenode and start all services.

```
"hdfs namenode -format"
```

```
"start-all.sh"
```

To check if all the nodes are running or not, run "jps" command in the terminal.



```
hduser@Master:~$ jps
11110 SecondaryNameNode
11413 Jps
10778 NameNode
10899 DataNode
11255 ResourceManager
11378 NodeManager
hduser@Master:~$
```

Fig: 4 – jps shows all running Nodes

If something like this appears, that means the single node Hadoop cluster has been installed and running properly.

#### Conclusion

In this project we have created a step by step list of how to create a Beowulf cluster implementing a MPI and Hadoop. We have also gone over how to use MPI and the documentation of the commands I have used in the attached file. I would like to set up MPI MapReduce and compare it to Hadoop MapReduce on raspberry pis.

#### REFERENCES

[1]. Rabkin, Ariel and Randy Howard Katz (July 2013). *How Hadoop Clusters Break*. *IEEE Software* [online], vol. 30, no. 4, pp. 88-94. Available: Academic Search Complete, EBSCOhost