Appendix A

Multi Node Apache Hadoop Installation

```
Step 1. Installing Java

       Java is the main prerequisite for Hadoop. Install latest java.
       \$sudo apt-get update
       \$sudo apt-get install default-jdk
       Then, you should verify the existence of java using the command 'java - version'.
       \$java -version

Step 2. Note the IP address of master machine and slave machine
        \$ifconfig

Step 3. In the /etc/hostname file of master add the name of the name-node system.
        \$sudo gedit /etc/hostname
          master
        In the /etc/hostname file of slave add the name of the data-node system.
        slave1

Step 4. In the /etc/hosts file add the name-node(ip-address, name) and data-nodes(ip-address, name).
        Name-node is the master and data-node is the slave.
        \$sudo gedit /etc/hosts
          master-IP  master
          slave-IP slave1

          Restart the system

Step 5 Create a dedicated user account for hadoop
       \$sudo addgroup hadoop
       \$sudo adduser --ingroup hadoop hduser
       \$sudo usermod -a -G sudo hduser
       \$su hduser
        Restart the system & Login to hduser

Step 5. Configure ssh
        Hadoop requires SSH access to manage its nodes, therefore we need to install ssh on
both master and slave systems.

\$ sudo apt-get install openssh-server
          Now, we have to generate an SSH key on master machine. When it asks you to enter
a file name to save the key, do not give any name, just press enter.

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

```
Second, you have to enable SSH access to your master machine with this newly created key.
        \$ cat \$HOME/.ssh/id_rsa.pub >> \$HOME/.ssh/authorized_keys
        Now test the SSH setup by connecting to your master.

        \$ ssh master
           Now run the below command to send the public key generated on master to slave.

        \$ ssh-copy-id -i \$HOME/.ssh/id_rsa.pub hduser@slave1
           Now that both master and slave have the pubic key, you can connect master to master
and master to slave as well.
        \$ ssh master
        \$ ssh slave1
        \$ exit
         On Master, Edit the masters file as below.
        \$ sudo gedit hadoop-2.7.3/etc/hadoop/masters
           master

         Edit the slaves file as below.
        \$ sudo gedit hadoop-2.7.3/etc/hadoop/slaves
           slave1

        On Slave, Edit the masters file as below.
        \$ sudo gedit hadoop-2.7.3/etc/hadoop/masters
           master

Step 6. Disable IPV6 by including the following lines in /etc/sysctl.conf file
        \$sudo gedit /etc/sysctl.conf

          net.ipv6.conf.all.disable_ipv6 = 1
          net.ipv6.conf.default.disable_ipv6 = 1
          net.ipv6.conf.lo.disable_ipv6 = 1

         Reboot the machine to make the changes and logon to hduser

Step 7. To find the java path
        \$ sudo update-alternatives --config javac

Step 8. Install Hadoop
        \$cd /usr/local

        \$sudo tar xvzf \$HOME/Downloads/hadoop-2.7.3.tar.gz
        \$sudo chmod 777 hadoop-2.7.3

Step 9. Set the hadoop environment variables.
        \$sudo gedit \$HOME/.bashrc
        Include the following lines in the \$HOME/.bashrc file

       # Set Hadoop-related environment variables export HADOOP_HOME=/usr/local/hadoop-2.7.3
       # Set JAVA home directory
         export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
       #Set aliases and functions for running Hadoop-related commands
        unalias fs &> /dev/null
        alias fs="hadoop fs"
        unaliash ls&> /dev/null
        alias hls="fs -ls"

       #Add Hadoop bin/ directory to PATH
```

```
        export PATH=\$PATH:\$HADOOP_HOME/bin

Step 10. Set hadoop environment variables.
         \$sudo gedit /etc/profile
         Include the following lines /etc/profile file
         # Insert JAVA_HOME JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
         # Insert HADOOP_PREFIX HADOOP_PREFIX=/usr/local/hadoop-2.7.3
         #--in PATH variable just append at the end of the line
 PATH=\$PATH:\$JAVA_HOME/bin:\$HADOOP_PREFIX/bin
         #--Append HADOOP_PREFIX at end of the export statement
         export PATH JAVA_HOME HADOOP_PREFIX

Step 11. Run the .bashrc& profile files from the \$ prompt for updating the changes
         \$ source \$HOME/.bashrc
         \$ source /etc/profile

Step 12. Check java & hadoop installation using
         \$ java -version
         \$ echo \$HADOOP_PREFIX
         \$ cd \$HADOOP_PREFIX
         \$ bin/hadoop version

Step 13. Configuration of the Hadoop files:
         hadoop-env.sh, core-site.xml, mapred-site.xml, hdfs-site.xml and yarn-site.xml
         \$cd etc/hadoop
          verify the path : /usr/local/hadoop-2.7.3/etc/hadoop
         13.1. Configuration of the hadoop-env.sh file
               \$sudo gedit hadoop-env.sh
                 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
                 export HADOOP_PREFIX=/usr/local/hadoop-2.7.3

              Now we will create NameNode and DataNode directories.
              \$cd
              \$mkdir -p \$HADOOP_HOME/data/hdfs/namenode
              \$mkdir -p \$HADOOP_HOME/data/hdfs/datanode

         13.2. Configuration of the core-site.xml file
               \$sudo gedit core-site.xml
                 Include the following lines:
                 <property>
                   <name>fs.defaultFS</name>
                   <value>hdfs://master:9000</value>
                 </property>

         13.3 Configuration of the mapred-site.xml
              \$sudo cp mapred-site.xml.template mapred-site.xml
              \$sudo gedit mapred-site.xml
                Include the following lines in mapred-site.xml file:
                <property>
                   <name>mapreduce.framework.name</name>
                   <value>yarn</value>
                </property>

13.4 Configuration of the hdfs-site.xml
     \$sudo gedit hdfs-site.xml
       Include the following lines in hdfs-site.xml file:
```

```
     <property>
        <name>dfs.replication</name>
        <value>2</value>
     </property>
     <property>
        <name>dfs.permissions</name>
        <value>false</value>
     </property>
     <property>
        <name>dfs.namenode.name.dir</name>
        <value>/usr/local/hadoop-2.7.3/data/hdfs/namenode</value>
     </property>
     <property>
        <name>dfs.datanode.data.dir</name>
        <value>/usr/local/hadoop-2.7.3/data/hdfs/datanode</value>
     </property>

13.5 Configuration of the yarn-site.xml
      \$sudo gedit yarn-site.xml
      Include the following lines in yarn-site.xml file:
      <property>
          <name>yarn.nodemanager.aux-services</name>
          <value>mapreduce_shuffle</value>
      </property>
      <property>
          <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
          <value>org.apache.hadoop.mapred.ShuffleHandler</value>
      </property>

Step 14. Format the Hadoop File system implemented on top of the local file system using
         \$cd ..
         \$cd ..
         \$cd bin
           Verify the path :/usr/local/hadoop-2.7.3/bin
         \$hadoop namenode -format

Step 15. Start Hadoop using
         \$cd ..
         \$cd sbin
         Verify the path : /usr/local/hadoop-2.7.3/sbin
         \$./start-all.sh
         \$ jps

Step 15. Open Hadoop GUI in browser
         https://master:50070/
```

Appendix B

Multi Node Apache Spark Installation

```
Step 1: Install Java
        Java is the main prerequisite for Apache Spark. Install latest java.
        \$ sudo apt-get update
        \$ sudo apt-get install default-jdk
        Then, verify the existence of java in your system using the command 'java - version'.
        \$ java -version

Step 2. Set the Host Name as 'master' in /etc/hostname file

Step 3. Set the Known Hosts in /etc/hosts file
        \$ sudo gedit /etc/hosts

       192.168.0.1 master

        Restart the system \& Login

Step 4 Create a dedicated user account for hadoop
       \$ sudo addgroup hadoop
       \$ sudo adduser --ingroup hadoop hduser
       \$ sudo usermod -a -G sudo hduser
       \$ su hduser

        Restart the system \& Login to hduser

Step 5. Configure ssh

     5.1. Generate private and public key pair at terminal using

         \$ sudo apt-get install ssh
         \$ ssh-keygen

     5.2. To enable ssh to the local machine

         \$ cat \$HOME/.ssh/id_rsa.pub >>\$HOME/.ssh/authorized_keys
         \$ ssh master

Step 6. Disable IPV6 by including the following lines in /etc/sysctl.conf file
        \$ sudo gedit /etc/sysctl.conf

         net.ipv6.conf.all.disable_ipv6 = 1
         net.ipv6.conf.default.disable_ipv6 = 1
         net.ipv6.conf.lo.disable_ipv6 = 1
```

```
       Reboot the machine to make the changes and logon to hduser

Step 7. To find the java path
        \$sudo update-alternatives --config javac

Step 8. Install Scala
        \$sudo apt-get install scala
        or
        Download Scala and extract
        \$cd /usr/local
        \$sudo tar xvzf \$HOME/Downloads/scala-2.12.1.tar.gz
        \$sudo chmod 777 scala-2.12.1

Step 9. Download and Install Spark

        http://spark.apache.org/downloads.html

        Install Spark:
          \$cd /usr/local
          \$sudo tar xvzf \$HOME/Downloads/spark-2.1.0-bin-hadoop2.7.tar
          \$sudo chmod 777 spark-2.1.0-bin-hadoop2.7

Step 10. Set the environment variables.
         \$sudo gedit \$HOME/.bashrc

         Include the following lines in the \$HOME/.bashrc file

         # Set JAVA home directory
         export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

         # Set Scala-related environment variables
         export SCALA_HOME=/usr/local/scala-2.12.1

         # Set Spark-related environment variables
         export SPARK_HOME=/usr/local/spark-2.1.0-bin-hadoop2.7

         #Add Spark bin/ directory to PATH
         export PATH=\$PATH:\$SCALA_HOME/bin:\$SPARK_HOME/bin

Step 11. Set environment variables.
         \$sudo gedit /etc/profile


         Include the following lines /etc/profile file

         # Insert JAVA_HOME
           JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

         #--in PATH variable just append at the end of the line
 PATH=\$PATH:\$JAVA_HOME/bin:\$SCALA_HOME/bin:\$SPARK_HOME/bin

         #--Append SPARK_HOME at end of the export statement
         export PATH JAVA_HOME SCALA_HOME SPARK_HOME

Step 12. Run the .bashrc& profile files from the \$ prompt for updating the changes
         \$source \$HOME/.bashrc
```

```
        \$source /etc/profile

Step 13. Check Spark Installation
        \$ echo \$SPARK_HOME
        \$ bin/spark -version

Step 14. Configure Spark

        Edit configuration file spark-env.sh (in \$SPARK_HOME/conf/) and
 set following parameters: (Create a copy of template of spark-env.sh and rename it)

        \$ sudo cd \$SPARK_HOME
        \$ cd conf
        \$ sudo cp spark-env.sh.template spark-env.sh
        \$ sudo gedit spark-env.sh

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

Step 15. Add Slaves
        Create configuration file slaves (in \$SPARK_HOME/conf/) and add following entries:
       \$ sudo gedit slaves
          slave1
          slave2

Step 16: Install Spark on all slaves

        16.1 Setup Pre-requisites on all the slaves:

             Run following steps on all the slaves (or worker nodes):

             16.1.1 Add Entries in hosts file
             16.1.2 Install Java 7
             16.1.3 Install Scala
             16.1.4 Setup environment variables
             16.2 Copy setups from master to all the slaves
                16.2.1 Create tar-ball of configured setup:
                       \$ tar czf spark.tar.gz spark-2.1.0-bin-hadoop2.7
                          NOTE: Run this command on Master
                16.2.2 Copy the configured tar-ball on all the slaves
                       \$ scp spark.tar.gz slave01:~
                          NOTE: Run this command on Master
             16.3 Un-tar configured spark setup on all the slaves
                 \$ cd /usr/local
                 \$ sudo tar xzf /home/hduser/spark.tar.gz
                   NOTE: Run this command on all the slaves

Step 17. Start Spark Cluster
        17.1 Start Spark Services
             \$ sbin/start-all.sh
             Note: Run this command on Master
        17.2 Check whether services have been started
             17.2.1 Check daemons on Master
             \$ jps
                Master
             17.2.2 Check daemons on Slaves
                \$ jps
```

```
                    Worker

Step 18. Spark Web UI
         18.1 Spark Master UI
               Browse the Spark UI for information about the cluster resources (like CPU, memory),
details of worker nodes, running application, segments, etc.

http://MASTER-IP:8080/

        18.2 Spark application UI
             http://MASTER-IP:4040/

Step 19. Stop the Cluster

        Once all the applications have finished, you can stop the spark services
(master and slaves daemons) running on the cluster

        \$ sbin/stop-all.sh
           Note: Run this command on Master
```

Appendix C

Multi Node Torque Installation

```
1. Installation of the development tools:
    Necessary development tools are installed including various gCC compiler, assemblers
and interpreters for various languages.

2. Install Development Tools:
  yum groupinstall 'Development Tools'

3. Installation of Dependencies:

    TCL:
        yum install tcl

    LIBSSL:
     yum install libssl-devel

  LIBXML:
     yum install libxml2-devel

  OpenSSL:
     yum install libtool openssl-devel libxml2-devel boost-devel gcc gcc-c++

4. Intall and configure cpuset:
    yum install cpuset
    yum install hwloc-devel
    mkdir /dev/cpuset
    mount -t cpuset none /dev/cpuset
    ./configure --enable-cpuset
    sudo yum install libcgroup
    sudo service cgconfig start

5. Install Torque Server and configure on HPC server host
    [root]# tar -xzvf torque-6.1.1.tar.gz
    [root]# cd torque-6.1.1/
    [root]# ./configure --enable-cgroups --with-hwloc-path=/usr/local
    [root]# make
    [root]# make install

6. Install and Configure Compute Nodes
    ./torque-package-mom-linux-i686.sh --install

7. Install and configure client nodes
   ./torque-package-clients-linux-i686.sh --install
```

8. Test the cluster: Give a test job from a valid client who is registered
    at server in /etc/hosts.equiv directory

Echo sleep 10 | qsub #press enter 10 times

Each time qsub command is typed one job is submitted, in job01 it only sleep the compute node
for given amount of time and returns. After getting successful return EXIT CODE = 0
for a particular tarcejob command, we can assure that server and compute node are
cooperating for this job.