Setting you Hbase Cluster on Hadoop (YARN). Ubuntu 12.04 LTS

`HBase` is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of `HDFS` (Hadoop Distributed Filesystem), providing BigTable-like capabilities for Hadoop. That is, it provides a fault-tolerant way of storing large quantities of sparse data (small amounts of information caught within a large collection of empty or unimportant data, such as finding the 50 largest items in a group of 2 billion records, or finding the non-zero items representing less than 0.1% of a huge collection).

Hardware Configuration

All Machine are as below configuration. We have 4 Servers in the Cluster. 1 Master and 3 Slaves.

```
Server RAM  : 16GB
Server CPU  : [Virtual] Single Core 64bit.
```

Services Running on each Server.

```
1. Master Node  [`AHMD-HBASE-MASTER`][`16.114.26.95`]: `NameNode`, `HMaster`, `ResourceManager`.
2. Slave Node   [`AHMD-HBASE-RS01`] [`16.114.26.99`]: `NN`, `RegionServer`, `DN`, `ZK`,`SecNN`.
3. Slave Node   [`AHMD-HBASE-RS02`] [`16.114.26.94`]: `NN`, `RegionServer`, `DataNode`, `ZK`.
4. Slave Node   [`AHMD-HBASE-RS03`] [`16.114.22.192`]: `NN`, `RegionServer`, `DN`, `ZooKeeper`.
```

Prerequisites - Before we start

Update All servers with below `/etc/hosts` file.

Copy the below content in `/etc/hosts` file. Make sure there are no Duplicate entries.

```
#HBASE Nodes
16.114.26.95    AHMD-HBASE-MASTER   # HMaster, NNode, RManager.
16.114.26.99    AHMD-HBASE-RS01     # NodeManager, RServer, DNode, ZKeeper, SNN.
16.114.26.94    AHMD-HBASE-RS02     # NodeManager, RServer, DNode, ZKeeper.
16.114.22.192   AHMD-HBASE-RS03     # NodeManager, RServer, DNode, ZKeeper.
```

Passwordless Entry from `master` to all `slaves`.

Simple 3 steps to make passwordless entry. on Ubuntu, should be similar on other linux variants.

```
ahmed@AHMD-HBASE-MASTER:~> ssh-keygen -t rsa

ahmed@AHMD-HBASE-MASTER:~> ssh ahmed@AHMD-HBASE-RS01 mkdir -p .ssh
ahmed@AHMD-HBASE-MASTER:~> cat ~/.ssh/id_rsa.pub | ssh ahmed@AHMD-HBASE-RS01 \
                                                             'cat >> .ssh/authorized_keys'

ahmed@AHMD-HBASE-MASTER:~> ssh ahmed@AHMD-HBASE-RS02 mkdir -p .ssh
ahmed@AHMD-HBASE-MASTER:~> cat ~/.ssh/id_rsa.pub | ssh ahmed@AHMD-HBASE-RS02 \
                                                             'cat >> .ssh/authorized_keys'

ahmed@AHMD-HBASE-MASTER:~> ssh ahmed@AHMD-HBASE-RS03 mkdir -p .ssh
ahmed@AHMD-HBASE-MASTER:~> cat ~/.ssh/id_rsa.pub | ssh ahmed@AHMD-HBASE-RS03 \
                                                             'cat >> .ssh/authorized_keys'
```

Testing. . .

```
ahmed@AHMD-HBASE-MASTER:~> ssh AHMD-HBASE-RS01
ahmed@AHMD-HBASE-RS01:~>
```

Setting Up Hadoop Cluster (YARN)

Extracting and creating required directories.

Do the below commands on all the servers.

```
tar xvzf hadoop-2.3.0-cdh5.1.2.tar.gz -C /opt
ln -s /opt/hadoop-2.3.0-cdh5.1.2 /opt/hadoop
mkdir -p /data/hadoop/nn
mkdir -p /data/hadoop/dn
mkdir -p /data/zookeeper
mkdir -p /data/1/yarn/local
mkdir -p /data/2/yarn/local
mkdir -p /data/3/yarn/local
mkdir -p /data/1/yarn/logs
mkdir -p /data/2/yarn/logs
mkdir -p /data/3/yarn/logs
```

Setting up HDFS.

On `AHMD-HBASE-MASTER`. Path : `/opt/hadoop/etc/hadoop/`

File to update : `core-site.xml` Add below configuration.

```
<configuration>
<property>
        <name>fs.default.name</name>
        <value>hdfs://AHMD-HBASE-MASTER:9000</value>
</property>
</configuration>
```

File to update : `hdfs-site.xml`

```
<configuration>
<property>
        <name>dfs.replication</name>
        <value>3</value>
</property>
```

```
<property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///data/hadoop/nn</value>
</property>
<property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///data/hadoop/dn</value>
</property>
<property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>AHMD-HBASE-RS01:50090</value>
</property>
</configuration>
```

File to update : `slaves`

```
AHMD-HBASE-RS01
AHMD-HBASE-RS02
AHMD-HBASE-RS03
```

Setting up YARN

File to update : `mapred-site.xml`

```
<configuration>
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
</configuration>
```

File to update : `yarn-site.xml`

```
<configuration>

<!-- Site specific YARN configuration properties -->
  <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>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>AHMD-HBASE-MASTER</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>file:///data/1/yarn/local,file:///data/2/yarn/local,file:///data/3/yarn/local</value>
  </property>
  <property>
```

```
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:///data/1/yarn/logs,file:///data/2/yarn/logs,file:///data/3/yarn/logs</value>
  </property>
  <property>
  </property>
    <name>yarn.log.aggregation-enable</name>
    <value>true</value>
  <property>
    <description>Where to aggregate logs</description>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>hdfs://AHMD-HBASE-MASTER:9000/var/log/hadoop-yarn/apps</value>
  </property>
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>6144</value>
  </property>
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>1024</value>
  </property>

</configuration>
```

Copy all the configuration to `slaves`

Note : Here using login as `root` as hadoop is running as `root` in /opt. If a specific user is created for `hadoop` then we transfer configuration using that user.

```
scp -r /opt/hadoop/etc/hadoop/* root@AHMD-HBASE-RS01:/opt/hadoop/etc/hadoop/
scp -r /opt/hadoop/etc/hadoop/* root@AHMD-HBASE-RS02:/opt/hadoop/etc/hadoop/
scp -r /opt/hadoop/etc/hadoop/* root@AHMD-HBASE-RS03:/opt/hadoop/etc/hadoop/
```

Starting Services HDFS

Step 1. First the first time we have to format `HDFS` before we start any services.

```
/opt/hadoop/bin/hadoop namenode -format
```

Step 2. Next we start the `namenode`, we are using the script `hadoop-daemon.sh` which will start a single service on the current machine.

```
/opt/hadoop/sbin/hadoop-daemon.sh start namenode
```

Step 3. Next we start `datanode`, using `hadoop-daemons.sh` (there is a `s` in `daemon`). This script will start the required service on all the nodes using the configuration data which we have already updated earlier.

```
/opt/hadoop/sbin/hadoop-daemons.sh start datanode
```

Here above `datanode` service is started on all the `slaves`, this information is got from the `slaves` file which has this information.

Step 4. Next we start `secondary-namenode`, Here though we are running `hadoop-daemons.sh`, this will only run on the `AHMD-HBASE-RS01` machine, since we have mentioned it in the configuration above in `hdfs-site.xml`.

```
/opt/hadoop/sbin/hadoop-daemons.sh start secondarynamenode
```

Here what the configuration looks like.

```
<property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>AHMD-HBASE-RS01:50090</value>
</property>
```

Starting Service YARN.

Step 1. Next we start the `resourcemanager`, we are using the script `yarn-daemon.sh` which will start a single service on the current machine.

```
/opt/hadoop/sbin/yarn-daemon.sh start resourcemanager
```

Step 2. Next we start `nodemanager`, using `yarn-daemons.sh` (there is a `s` in `daemon`). This script will start the required service on all the nodes using the configuration data which we have already updated earlier.

```
/opt/hadoop/sbin/yarn-daemons.sh start nodemanager
```

Testing Our Hadoop Cluster.

There are multiple ways to test it. But we will use the existing example programs to test it. Example jars are located in the below directory.

To run `yarn` we can use the below command.

```
/opt/hadoop/hadoop/yarn jar <path to example jar> pi <map to run> <samples>
```

Here are all the examples which can be run.

```
root@AHMD-HBASE-MASTER:~# /opt/hadoop/bin/yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-e
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in t
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input files.
  wordmedian: A map/reduce program that counts the median length of the words in the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the wo
```

Parameter for pi calculation.

```
root@AHMD-HBASE-MASTER:~# /opt/hadoop/bin/yarn jar \
            /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.3.0-cdh5.1.2.jar pi
Usage: org.apache.hadoop.examples.QuasiMonteCarlo <nMaps> <nSamples>
Generic options supported are
-conf <configuration file>     specify an application configuration file
-D <property=value>            use value for given property
-fs <local|namenode:port>      specify a namenode
-jt <local|jobtracker:port>    specify a job tracker
-files <comma separated list of files>    specify comma separated files to be copied to the map reduce c
-libjars <comma separated list of jars>    specify comma separated jar files to include in the classpath
-archives <comma separated list of archives>    specify comma separated archives to be unarchived on the

The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
```

Executing pi example program.

```
root@AHMD-HBASE-MASTER:~# /opt/hadoop/bin/yarn jar \
        /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.3.0-cdh5.1.2.jar pi 5 1000
Number of Maps  = 5
Samples per Map = 1000
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Starting Job
15/02/13 05:38:39 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:38:39 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:38:39 INFO client.RMProxy: Connecting to ResourceManager at
                                                        AHMD-HBASE-MASTER/16.114.26.95:8032
15/02/13 05:38:39 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:38:39 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:38:39 INFO input.FileInputFormat: Total input paths to process : 5
15/02/13 05:38:40 INFO mapreduce.JobSubmitter: number of splits:5
15/02/13 05:38:40 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1423765442485_0003
15/02/13 05:38:40 INFO impl.YarnClientImpl: Submitted application application_1423765442485_0003
15/02/13 05:38:40 INFO mapreduce.Job: The url to track the job:
                          http://AHMD-HBASE-MASTER:8088/proxy/application_1423765442485_0003/
15/02/13 05:38:40 INFO mapreduce.Job: Running job: job_1423765442485_0003
15/02/13 05:38:48 INFO mapreduce.Job: Job job_1423765442485_0003 running in uber mode : false
15/02/13 05:38:48 INFO mapreduce.Job:  map 0% reduce 0%
15/02/13 05:38:54 INFO mapreduce.Job:  map 80% reduce 0%
15/02/13 05:38:55 INFO mapreduce.Job:  map 100% reduce 0%
15/02/13 05:39:00 INFO mapreduce.Job:  map 100% reduce 100%
15/02/13 05:39:00 INFO mapreduce.Job: Job job_1423765442485_0003 completed successfully
```

```
15/02/13 05:39:00 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:39:00 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:39:00 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:39:00 WARN conf.Configuration: bad conf file: element not <property>
15/02/13 05:39:00 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=116
                FILE: Number of bytes written=544587
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1360
                HDFS: Number of bytes written=215
                HDFS: Number of read operations=23
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
        Job Counters
                Launched map tasks=5
                Launched reduce tasks=1
                Data-local map tasks=5
                Total time spent by all maps in occupied slots (ms)=20066
                Total time spent by all reduces in occupied slots (ms)=3998
                Total time spent by all map tasks (ms)=20066
                Total time spent by all reduce tasks (ms)=3998
                Total vcore-seconds taken by all map tasks=20066
                Total vcore-seconds taken by all reduce tasks=3998
                Total megabyte-seconds taken by all map tasks=20547584
                Total megabyte-seconds taken by all reduce tasks=4093952
        Map-Reduce Framework
                Map input records=5
                Map output records=10
                Map output bytes=90
                Map output materialized bytes=140
                Input split bytes=770
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=140
                Reduce input records=10
                Reduce output records=0
                Spilled Records=20
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=108
                CPU time spent (ms)=3600
                Physical memory (bytes) snapshot=1472352256
                Virtual memory (bytes) snapshot=6874152960
                Total committed heap usage (bytes)=1209401344
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=590
        File Output Format Counters
                Bytes Written=97
Job Finished in 21.618 seconds
Estimated value of Pi is 3.14160000000000000000
root@AHMD-HBASE-MASTER:~#
```

Setting up Zookeeper on `AHMD-HBASE-RS01`,`AHMD-HBASE-RS02`,`AHMD-HBASE-RS03`

Do the below process in all the slave nodes as we will be running zookeeper on these machines. Thumb rule for `zookeeper` is the have odd number of `zookeeper` in a cluster, so that its easier to elect a leader.

```
AHMD-HBASE-RS01
AHMD-HBASE-RS02
AHMD-HBASE-RS03
```

Initial setup on all `zookeeper` servers.

Extracting `zookeeper`

```
tar xvzf zookeeper-3.4.5-cdh5.1.2.tar.gz -C /opt/
ln -s /opt/zookeeper-3.4.5-cdh5.1.2 /opt/zookeeper
```

Create a directory for zookeeper.

```
mkdir -p /data/zookeeper
```

Setting Configuration for zookeeper `zoo.cfg`

```
cp /opt/zookeeper/conf/zoo_sample.cfg /opt/zookeeper/conf/zoo.cfg
```

Change the path in `zoo.cfg` for `dataDir`.

```
sed -i -- 's/tmp\/zookeeper/data\/zookeeper/g' /opt/zookeeper/conf/zoo.cfg
```

Next we Add the `zookeeper` cluster nodes.

```
echo "server.1=AHMD-HBASE-RS01:2888:3888" >> /opt/zookeeper/conf/zoo.cfg
echo "server.2=AHMD-HBASE-RS02:2888:3888" >> /opt/zookeeper/conf/zoo.cfg
echo "server.3=AHMD-HBASE-RS03:2888:3888" >> /opt/zookeeper/conf/zoo.cfg
```

In the above line `server.1` where `1` is the id for server `AHMD-HBASE-RS01` and `2` for `AHMD-HBASE-RS02` and `3` for `AHMD-HBASE-RS03`.

So to assign an `id` to server we need to create a `myid` file `dataDir` path, which is currently set to `/data/zookeeper`

Assigning ids to zookeeper nodes.

So in Server `AHMD-HBASE-RS01` which has an `id` of `1`, use the below command.

```
echo "1" > /data/zookeeper/myid
```

`AHMD-HBASE-RS02` which has an `id` of `2`, use the below command.

```
echo "2" > /data/zookeeper/myid
```

`AHMD-HBASE-RS03` which has an `id` of `3`, use the below command.

```
echo "3" > /data/zookeeper/myid
```

Starting up zookeeper server

Use below command on all `zookeeper` servers.

```
/opt/zookeeper/bin/zkServer.sh start
```

Log `zookeeper.out` of the startup will in the directory were the above command was executed.

Testing zookeeper

`zookeeper` by default will run on port `2181`. To verify of the service is running on this port, use `telnet` to verify. if you get the `^]` char then we are all good.

Do this to connect to all the `zookeeper` nodes.

```
root@AHMD-HBASE-RS01:~# telnet AHMD-HBASE-RS02 2181
Trying 16.114.26.94...
Connected to AHMD-HBASE-RS02.
Escape character is '^]'.
^CConnection closed by foreign host.
root@AHMD-HBASE-RS01:~#
```

Setting up HBase

On `AHMD-HBASE-MASTER`

Extract the archive on all the servers and updating configurations.

```
tar xvzf hbase-0.98.1-cdh5.1.2.tar.gz -C /opt
ln -s /opt/hbase-0.98.1-cdh5.1.2 /opt/hbase
```

In Directory `/opt/hbase/conf/`

Update `hbase-site.xml` file with below information.

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://AHMD-HBASE-MASTER:9000/hbase_jio</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>AHMD-HBASE-RS01,AHMD-HBASE-RS02,AHMD-HBASE-RS03</value>
  </property>
</configuration>
```

Update `hbase-env.sh`.

To Enable JMX monitoring.

```
 export HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false
                                        -Dcom.sun.management.jmxremote.authenticate=false"
```

Setting 6GB for Master Heap Memory. JMX Monitoring port 10101

```
 export HBASE_MASTER_OPTS="-Xmx6g $HBASE_MASTER_OPTS $HBASE_JMX_BASE
                                        -Dcom.sun.management.jmxremote.port=10101"
```

Setting 6GB for Region servers. JMX Monitoring port 10102

```
 export HBASE_REGIONSERVER_OPTS="-Xmx6g $HBASE_REGIONSERVER_OPTS $HBASE_JMX_BASE
                                        -Dcom.sun.management.jmxremote.port=10102"
```

Setting all region servers in `regionservers` file.

```
AHMD-HBASE-RS01
AHMD-HBASE-RS02
AHMD-HBASE-RS03
```

Copy configuration to all the servers.

```
scp -r /opt/hbase/conf/* root@AHMD-HBASE-RS01:/opt/hbase/conf/
scp -r /opt/hbase/conf/* root@AHMD-HBASE-RS02:/opt/hbase/conf/
scp -r /opt/hbase/conf/* root@AHMD-HBASE-RS03:/opt/hbase/conf/
```

Starting HBase services.

To start the `HMaster`.

```
/opt/hbase/bin/hbase-daemon.sh start master
```

To start `RegionServer`, on all the slave nodes use `hbase-daemons.sh` script, similar to the one in `hadoop` called `hadoop-daemons.sh`.

```
/opt/hbase/bin/hbase-daemons.sh start regionserver
```

This will start all the `regionserver` in all the slaves.

Testing HBase Server.

Go to `hbase` shell

```
/opt/hbase/bin/hbase shell
```

This command will describe a simple status of the HBase cluster nodes:

```
status 'simple'
```

This command will create a table with one column family:

```
create 'table2', 'cf1'
```

This command will add a row to the table:

```
put 'table2', 'row1', 'column1', 'value'
```

This command will display all rows in the table:

```
scan 'table2'
```

If all the command work above then we have `hbase` ready.