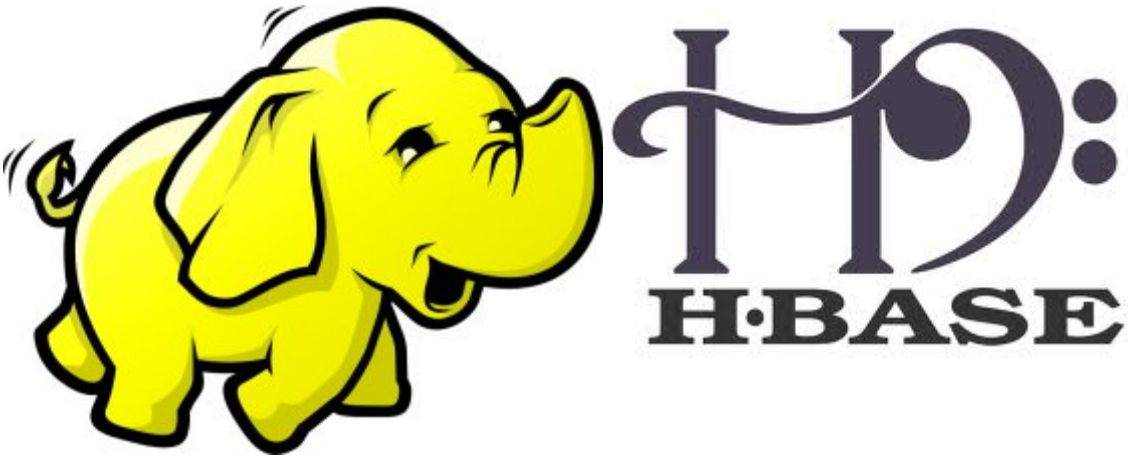

Hadoop and HBase

Set-up guide

Ng Zhi An

Based on Cloudera's CDH3 Update 3

Contents

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Setting up a fully-distributed Hadoop

1. Java

Download and install **Java 6 OpenJDK** if you do not have it yet:

```
$ sudo apt-get java-6-openjdk
```

It will be installed under either /usr/lib/jvm/java-6-openjdk or /usr/java.
This step has to be repeated on all servers

2. Dedicated Hadoop user

Create a new user specially for using/testing Hadoop

```
$ sudo addgroup hadoop // creates the user-group hadoop
$ sudo adduser --ingroup hadoop hduser // adds hduser to the group hadoop
```

The following step might not be needed, try doing without it. But if permissions error ensues, then use this solution, this will add sudo permissions to hduser:

```
$ sudo visudo
// under user privilege specification add in
hduser ALL=(ALL) ALL
```

More info on adduser and addgroup, visudo and how to edit the sudoers file in visudo.
This step has to be repeated on all servers

3. Download and Install Hadoop

Download from Apache mirror and extract contents to a location, in this example we'll use /home/hduser/hadoopand change owner of all hadoop files to hduser. Assuming your downloaded Hadoop archive file is in the folder/home/hduser/Downloads

```
# change to your Downloads directory
$ cd /home/hduser/Downloads

# extract the files
$ tar xzf hadoop-[versionnumber].tar.gz

# move the folder to the appropriate directory
$ mv hadoop-[versionnumber] /home/hduser/hadoop

# change owner of files
$ cd /home/hduser
$ sudo chown -R hduser:hadoop hadoop
```

More info on chown
This step has to be repeated on all servers
An alternative to proceeding with the following steps on each and every server is to complete the steps on a single server, then copy the hadoop folder to the other servers via rcp or rsync

4. System Setup

Configure $HOME/.bashrc

```
$ nano $HOME/.bashrc
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Nano is a simple CLI editor. You can use any text editor you like, e.g. vi, vim, emacs, gedit, by replacing nano with the appropriate program name.
add the following lines to the end of the file

```
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
export HADOOP_HOME=/home/hduser/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

Change your JAVA_HOME value to the proper directory where your Java is in.
Configure /etc/hosts.

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

More info on .bashrc and export
In our case, we have three servers, master, slave1 and slave2. Our /etc/hosts file will then look like this:

```
127.0.0.1    localhost
10.2.41.78   master nadal
10.2.41.101          slave1  UbuntuHadoop01
10.2.41.85   slave2  UbuntuHadoop02
```

More info on /etc/hosts
There will probably be a few more lines in the file, but comment them out. Pay special attention to comment lines that looks like this:

```
#127.0.0.1  nadal
#127.0.1.1  localhost
```

Note the 127.0.0.1 references to your machine name, nadal in my case, and the different locahost ip, 127.0.1.1 (with two 1s). I have found that the above lines cause problems when trying to run HBase, so comment it out.
This step has to be repeated on all servers

5. SSH

Configure ssh access to localhost for hduser on master

```
// change current user to hduser
$ su hduser
// generate a SSH key for hduser, creates an rsa key pair with an empty password
$ ssh-keygen -t rsa -P ""
// enable ssh access to local machine with this new key
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
// test ssh setup
$ ssh localhost
$ ssh master
// if there are any problems, debug with
$ ssh -vvv localhost
```

hduser on the master must be able to connect to itself (ssh master) and also to hduser on the slaves via a password-less ssh login
Add the huser@master public ssh key to autorized_keys of hduser@slave in the hduser@slave's$HOME/.ssh/authorized_keys

```
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave1
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave2
```

More info on ssh and passwordless SSH logins
This step has to be repeated on all servers

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

6. Configuration

conf/masters (on master machine only) defines on which machines Hadoop will start secondary NameNodes in our multi-node cluster
The primary NameNode and the JobTracker will always be the machines on which you run the bin/start-dfs.shand bin/start-mapred.sh scripts,
On master machine, update conf/masters, add the following line

```
master
```

The conf/slaves file lists the hosts, one per line, where the Hadoop slave daemons (DataNodes and TaskTrackers) will be run. On master, update conf/slaves so that it looks like this:

```
slave1
slave2
```

This will run the Secondary NameNode on master, and DataNodes and TaskTrackers on slave1 and slave2.
You have to change the configuration files conf/core-site.xml, conf/mapred-site.xml and conf/hdfs-site.xml on ALL machines as follows.

```
<!-- In: conf/core-site.xml -->
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://master:54310</value>
    <description>The name of the default file system.  A URI whose scheme and authority determine the FileSystem implementation.  The uri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class.  The uri's authority is used to determine the host, port, etc. for a filesystem.
    </description>
  </property>
</configuration>
<!-- In: conf/mapred-site.xml -->
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>master:54311</value>
    <description>The host and port that the MapReduce job tracker runs at.  If "local", then jobs are run in-process as a single map and reduce task.
    </description>
  </property>
</configuration>
<!-- In: conf/hdfs-site.xml -->
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
    <description>Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.
    </description>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>/home/hduser/hadoop/tmp/dfs/name</value>
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
    <description> Determines where on the local filesystem the DFS name node should store
  the name table(fsimage). If this is a comma-delimited list of directories then the name table is
  replicated in all of the directories, for redundancy.
    </description>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hduser/hadoop/tmp/dfs/data</value>
    <description> Determines where on the local filesystem an DFS data node should store
  its blocks. If this is a comma-delimited list of directories, then data will be stored in all named
  directories, typically on different devices. Directories that do not exist are ignored.
    </description>
  </property>
</configuration>
```

More info on the configuration parameters: important parameters, and all parameters in the file core-site.xml, hdfs-site.xml, mapred-site.xml
You can use rcp or rsync to copy your configuration files from one server to another.
The information in the links might be outdated due to frequent updates on Hadoop
This step has to be repeated on all servers

7. First-run

Format namenode on NameNode (master)

```
$ bin/hadoop namenode -format
```

You might be able to omit the bin/ because you have added the $HADOOP_HOME/bin to your .bashrc path, so bashwill automatically be able to locate the hadoop script, and be able to execute the namenode command properly.So for the above input, and all other inputs into the script below, you can omit the bin/ prefixes, as such:

```
$ hadoop namenode -format
```

Start the multi-node cluster by running bin/start-dfs.sh on the machine that you want NameNode to run on, this will bring up HDFS with NameNode on the machine this script is ran and DataNodes on the machines listed inconf/slaves

```
$ bin/start-dfs.sh
```

Examine the success/failure of this command in the logs folder.
Examine the logs folder on master to troubleshoot the starting of NameNode and SecondaryNameNode. Examing the logs on each slave to debug the its respective DataNode.
Use jps to look at the Java processes running, it should be something like this:

```
hduser@master:/home/hduser/hadoop$ jps
14799 NameNode
15314 Jps
14880 DataNode
14977 SecondaryNameNode

hduser@slave1:/home/hduser/hadoop$ jps
15183 DataNode
15616 Jps

hduser@slave2:/home/hduser/hadoop$ jps
15225 DataNode
15985 Jps
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Run the command /bin/start-mapred.sh on the machine you want the JobTracker to run on This will bring up the MapReduce cluster with the JobTracker running on the machine you ran the previous command on, and TaskTrackers on the machines listed in the conf/slaves file.
Examine the logs folder on master to troubleshoot the starting of JobTracker. Examing the logs on each slave to debug the its respective TaskTracker.
Use jps to look at the Java processes running, it should be something like this:

```
$ bin/start-mapred.sh

hduser@master:/home/hduser/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode

hduser@slave1:/home/hduser/hadoop$ jps
15183 DataNode
15897 TaskTracker
16284 Jps

hduser@slave2:/home/hduser/hadoop$ jps
15225 DataNode
15985 Jps
15897 TaskTracker
```

Hadoop has web interfaces at the following URL to check/track the various processes:

```
http://master:50030/ - web UI for MapReduce job tracker(s)
http://master:50060/ - web UI for task tracker(s)
http://master:50070/ - web UI for HDFS name node(s)
```

Run a sample MapReduce test to test that your cluster is up.

```
$ cd ~/hadoop

# Copy the input files into the distributed filesystem
# (there will be no output visible from the command):
$ bin/hadoop fs -put conf input

# Run some of the examples provided:
# (there will be a large amount of INFO statements as output)
# grep will search files in input folder for strings
# that matches the expression 'dfs[a-z.]+',
# and output the results to output folder
$ bin/hadoop jar hadoop-*-examples.jar grep input output 'dfs[a-z.]+'

# Examine the output files:
$ bin/hadoop fs -cat output/part-00000
```

Stop the cluster using the he command

```
$ bin/stop-all.sh
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Setting up a replicated ZooKeeper

1. Download and Install ZooKeeper

Download from Apache mirror and extract contents to a location, in this example we'll use/ home/hduser/zookeeper and change owner of all zookeeper files to hduser:

```
# change to your Downloads directory
$ cd /home/hduser/Downloads

# extract the files
$ tar xzf zookeeper-[versionnumber].tar.gz

# move the folder to the appropriate directory
$ mv zookeeper-[versionnumber] /home/hduser/zookeeper

# change owner of files
$ cd /home/hduser
$ sudo chown -R hduser:hadoop zookeeper
```

More info on chown
This step has to be repeated on all servers
An alternative to proceeding with the following steps on each and every server is to complete the steps on a single server, then copy the zookeeper folder to the other servers via rcp or rsync

2. System Setup

Configure $HOME/.bashrc

```
$ nano $HOME/.bashrc
```

Nano is a simple CLI editor. You can use any text editor you like, e.g. vi, vim, emacs, gedit, by replacing nano with the appropriate program name.
Append the following lines, or update if similar exists to the end of the file

```
export ZK_HOME=/home/hduser/zookeeper
export PATH=$PATH:$HADOOP_HOME/bin:$ZK_HOME/bin
```

If you followed the previous section on Setting up Hadoop, your file will look like this

```
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
export HADOOP_HOME=/home/hduser/hadoop
export ZK_HOME=/home/hduser/zookeeper
export PATH=$PATH:$HADOOP_HOME/bin:$ZK_HOME/bin
```

More info on .bashrc and export
This step has to be repeated on all servers

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

3. Configuration

For configuring the zookeeper goto conf folder in where you installed zookeeper, we used/ home/hduser/zookeeper, you may or may not find zoo.cfg. If you find it, make sure the contents match below, if not do a file copy:

```
cp zoo_sample.cfg zoo.cfg
```

More info on cp

If no zoo_sample.cfg exists, just make a new file named zoo.cfg with the following contents:

```
# The number of milliseconds of each tick
tickTime=2000

# The directory where the snapshot is stored
dataDir=/home/hduser/zookeeper/dataDir

# The port at which clients will connect
clientPort=2181

# The number of ticks that the initial
# synchronization phase can take
initLimit=5

# The number of ticks that can pass
# between sending a request and
# getting an acknowledgement
syncLimit=2

server.3=master:2888:3888
server.1=slave1:2888:3888
server.2=slave2:2888:3888
```

More info on the attributes in the file here and here.

The most important thing is the lines in the form of server.N. The entries list the servers that make up the ZooKeeper service. When the server starts up, it knows which server it is by looking for the file myid in the data directory, which as specified is /home/hduser/zookeeper/ dataDir

The myid file in /home/hduser/zookeeper/dataDir has a single line containing that machine's id. This id is unique within the ensemble and should have a value of between 1 and 255

As such, the myid file of master will be:

```
3
```

The myid file of slave1 will be:

```
1
```

The myid file of slave2 will be:

```
2
```

The server with the highest myid will be elected the leader

4. First-run

Execute this command on all machines:

```
$ zkServer.sh start
```

or for a more verbose version:

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
$ zkServer.sh start-foreground
```
Check the status of the machines with
```
$ zkServer.sh status
# you'll get either of the two reponses below
Mode: follower
Mode: leader
# depending on the machines myid
```

It doesn't really matter which machine is the leader or the follower, the NameNode does not have to be the leader. The important thing is that ZooKeeper is running in all the machines that you have stated.
If you followed the Setting up Hadoop section before this, running jps will show you the following output:
```
hduser@master:/home/hduser/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
3269 QuorumPeerMain

hduser@slave1:/home/hduser/hadoop$ jps
15183 DataNode
15897 TaskTracker
16284 Jps
1878 QuorumPeerMain

hduser@slave2:/home/hduser/hadoop$ jps
15225 DataNode
15985 Jps
15897 TaskTracker
1258 QuorumPeerMain
```

Setting up a fully-distributed HBase

1. Download and Install HBase

Download from Apache Mirror and unpack it to a location, in this example we'll use /home/hduser/hbase:
```
# change to your Downloads directory
$ cd /home/hduser/Downloads

# extract the files
$ tar xzf hbase-[versionnumber].tar.gz
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
# move the folder to the appropriate directory
$ mv hbase-[versionnumber] /home/hduser/hbase

# change owner of files
$ cd /home/hduser
$ sudo chown -R hduser:hadoop hbase
```

More info on chown
This step has to be repeated on all servers
An alternative to proceeding with the following steps on each and every server is to complete

2. System Setup

Configure $HOME/.bashrc

```
$ nano $HOME/.bashrc
```

Nano is a simple CLI editor. You can use any text editor you like, e.g. vi, vim, emacs, gedit, by replacing nano with the appropriate program name.
Append the following lines, or update if similar exists to the end of the file

```
export HBASE_HOME=/home/hduser/hbase
export PATH=$PATH:$HADOOP_HOME/bin:$ZK_HOME/bin:$HBASE_HOME/bin
```

If you followed the previous section on Setting up Hadoop, your file will look like this

```
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
export HADOOP_HOME=/home/hduser/hadoop
export ZK_HOME=/home/hduser/zookeeper
export HBASE_HOME=/home/hduser/hbase
export PATH=$PATH:$HADOOP_HOME/bin:$ZK_HOME/bin:$HBASE_HOME/bin
```

More info on .bashrc and export
This step has to be repeated on all servers

3. Configuration

You have to edit 3 configuration files on ALL MACHINES.
First, edit the hbase-env.sh file in the conf directory of HBase:

```
$ nano hbase-env.sh
```

Scroll to the bottom of the file, edit the last line of the file such that it now says:

```
export HBASE_MANAGES_ZK=false
```

Next, the hbase-site.xml file. Add the following lines:

```
<!-- In: conf/hbase-site.xml -->
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://master:54310/hbase</value>
    <description>The directory shared by region servers and into which HBase persists.
  The URL should be 'fully-qualified' to include the filesystem scheme. For example, to
  specify the HDFS directory '/hbase' where the HDFS instance's namenode is running at
  namenode.example.org on port 9000, set this value to: hdfs://namenode.example.org:9000/
  hbase. By default HBase writes into /tmp. Change this configuration else all data will be lost
  on machine restart.</description>
  </property>
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
    <description>The mode the cluster will be in. Possible values are false for standalone
mode and true for distributed mode. If false, startup will run all HBase and ZooKeeper
daemons together in the one JVM.</description>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>master,slave1,slave2</value>
    <description>Comma separated list of servers in the ZooKeeper Quorum. For
example, "host1.mydomain.com,host2.mydomain.com,host3.mydomain.com". By
default this is set to localhost for local and pseudo-distributed modes of operation. For
a fully-distributed setup, this should be set to a full list of ZooKeeper quorum servers. If
HBASE_MANAGES_ZK is set in hbase-env.sh this is the list of servers which we will start/
stop ZooKeeper on.</description>
  </property>
  <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
    <description>Property from ZooKeeper's config zoo.cfg. The port at which the clients will
connect.</description>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/hduser/zookeeper/dataDir</value>
    <description>Property from ZooKeeper's config zoo.cfg. The directory where the snapshot
is stored.</description>
  </property>
</configuration>
```

You'll notice that some properties in hbase-site.xml are similar to the ones defined in zoo.cfg. According to documentation, HBase will prefer the configuration found in zoo.cfg over any settings in hbase-site.xml, as long aszoo.cfgis in HBase's CLASSPATH. So if you copy zoo.cfg to the conf directory of HBase, those properties should not be needed, but to be safe, just leave them inside, but make sure that the settings match.
Lastly, the regionservers file in the conf directory. If that file does not exist, create it. This file lists all the hosts to you would like RegionServers to be running on, one host per line:

```
slave1
slave2
```

You also have to make sure HBase knows your HDFS configuration, the easiest way to do it is to create a soft symbolic link to Hadoop's hdfs-site.xml in HBase's conf directory.

```
$ cd /home/hduser/hbase/conf
$ ln -s ../../hadoop/conf/hdfs-site.xml
```

More information on ln

4. First-run

Ensure that you HDFS is running first, MapReduce is not required to be running at this stage, but if it is, leave it.

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
$ start-hdfs.sh
```

Since we have told HBase not to manage its ZooKeeper, we'll have to start it manually. If you didn't not follow the previous section on Setting up ZooKeeper try:

```
$ hbase-daemons.sh start zookeeper
# check the status of each machine by running the following
$ zkServer.sh status
```

If the above command doesn't start the ZooKeeper instances, we'll have to start it manually on each machine. Run on each machine the following command:

```
$ zkServer.sh Start
# examine the status by running
$ zkServer.sh status
# Each machine will either be a follower or a leader
```

Once the ZooKeeper instances are up in the machines specified in your zoo.cfg file (which is the same as thehbase.zookeeper.quorum property), you can start HBase:

```
$ start-hbase.sh
```

This will bring up RegionMaster on the machine with NameNode running RegionServers on the machines with DataNodes running

Examine the success/failure of this command in the /home/hduser/hbase/logs folder.

Examine the logs folder on master to troubleshoot the starting of HMaster. Examine the logs on each slave to debug the its respective HRegionServer.

Use jps to look at the Java processes running, it should be something like this:

```
hduser@master:/home/hduser/hadoop$ jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
3269 QuorumPeerMain
6238 HMaster


hduser@slave1:/home/hduser/hadoop$ jps
15183 DataNode
15897 TaskTracker
16284 Jps
1879 QuorumPeerMain
2001 HRegionServer

hduser@slave2:/home/hduser/hadoop$ jps
15225 DataNode
15985 Jps
15897 TaskTracker
1258 QuorumPeerMain
2158 HRegionServer
```

HBase has web interfaces at the following URL to check/track the various processes:

```
http://master:60000/ - web UI listing vital atttributes
```

Run some simple Shell exercises to test that your HBase is set up correctly:

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
$ hbase shell
# this will allow you to connect to hbase via the shell
# you will see a prompt on the same window as such
hbase(main):001:>
```

Type help and then <RETURN> to see a listing of shell commands and options.

We will create a test table called 'test' with a single column family named 'cf'. We then enter 3 different values, in 3 different rows, under 3 different columns.

```
hbase(main):003:0> create 'test', 'cf'
0 row(s) in 1.2200 seconds
hbase(main):003:0> list 'test'
..
1 row(s) in 0.0550 seconds
hbase(main):004:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.0560 seconds
hbase(main):005:0> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0370 seconds
hbase(main):006:0> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0450 seconds
```

This enters first value 'value1' at the row 'row1', at column 'a' under column family 'cf'; the value 'value2' at the row 'row2', at column 'b' under column family 'cf'; the value 'value3' at the row 'row3', at column 'c'under column family 'cf'.

Columns in HBase are comprised of a column family prefix - cf in this example - followed by a colon and then a column qualifier suffix.

We then run a scan on the table to verify that the values have been inserted properly

```
hbase(main):007:0> scan 'test'
ROW       COLUMN+CELL
row1      column=cf:a, timestamp=1288380727188, value=value1
row2      column=cf:b, timestamp=1288380738440, value=value2
row3      column=cf:c, timestamp=1288380747365, value=value3
3 row(s) in 0.0590 seconds
```

We can also retrieve a single row as such:

```
hbase(main):008:0> get 'test', 'row1'
COLUMN      CELL
cf:a        timestamp=1288380727188, value=value1
1 row(s) in 0.0400 seconds
```

Since our HBase is runs on HDFS, we can browse the tables on HDFS's web UI:

```
# On your browser go to
http://master:50070
# look for the "Brow the filesystem" link
# you will see a listing for folders which is simlar to the below
hbase       dir     2012-05-12 03:04     rwxr-xr-x       hduser supergroup
tmp dir     2012-05-12 01:42     rwxr-xr-x       hduser supergroup
user        dir     2012-05-12 01:35     rwxr-xr-x       hduser supergroup
# click on the hbase link to access the hbase directory
# you should be able to see your a directory named after your table, 'test'
-ROOT-      dir     2012-05-10 05:06     rwxr-xr-x       hduser supergroup
-META-      dir     2012-05-12 01:41     rwxr-xr-x       hduser supergroup
.logs       dir     2012-05-12 08:12     rwxr-xr-x       hduser supergroup
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
test dir 2012-05-12 00:29 rwxr-xr-x hduser supergroup
```

This will confirm that your setup of HBase on HDFS is successful. We will keep the table because we want to use it to test our MapReduce set-up, but if you want try removing the table, you can now disable and drop your test table this way:

```
hbase(main):012:0> disable 'test'
0 row(s) in 1.0930 seconds
hbase(main):013:0> drop 'test'
0 row(s) in 0.0770 seconds
```

And finally exit the HBase shell:

```
hbase(main):014:0> exit
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

MapReduce on HBase

1. Configuration

By default MapReduce does not have access to the HBase jars, so we either have to copy the HBase jars to Hadoop'slib folder, or we can edit the CLASSPATH of Hadoop, which is demonstrated below:

```
# open the hadoop script file
$ gedit /home/hduser/hadoop/bin/hadoop
# look for the comment saying "add user-specified CLASSPATH last"
# add the following line above the if statement
HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath`
```

The back ticks ` will tell the script to execute the expression inside. In this case ${HBASE_HOME}/bin/hbase classpath resolves to a representation of the entire classpath of HBase. We then add the representation to the variable HADOOP_CLASSPATH. Subsequently, this value will be added to Hadoop's classpath, meaning that Hadoop will look at these directories for jar files.

Finally the portion of the code that you edited should look like this:

```
# add user-specified CLASSPTH last
HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath`
if [ "$HADOOP_USER_CLASSPATH_FIRST" = "" ] && [ "$HADOOP_CLASSPATH" != "" ];
then
  CLASSPATH=${CLASSPATH}:${HADOOP_CLASSPATH}
fi
```

This method is easy to implement, but will pollute your Hadoop installation. Another method is as such, to run the sample HBase RowCounter MapReduce job:

```
$ HADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath` ${HADOOP_HOME}/
bin/hadoop jar ${HBASE_HOME}/hbase-[version].jar rowcounter usertable
```

This does the same thing, it makes HBase's classpath known to Hadoop. However you have to enterHADOOP_CLASSPATH=`${HBASE_HOME}/bin/hbase classpath` everytime you wish to run a MapReduce job on HBase. Which is why I prefer the first method.

More info on official HBase API

2. First-run

We shall test that our we can run MapReduce jobs on our HBase setup now. The test is a rowcounter test that counts the number of rows in a table. Run it in this matter:

```
$ cd /home/hduser/hbase
hadoop jar hbase-[version].jar rowcounter test
# there will be a very verbose output of INFO but look out for this line
org.apache.hadoop.hbase.mapreduce.RowCounter$RowCounterMapper$Counters
ROWS=3
```

Or if you did not modify the hadoop file:

hadoop jar hbase-[version].jar rowcounter test

```
$ cd /home/hduser/hbase
# there will be a very verbose output of INFO but look out for this line
```

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

```
    org.apache.hadoop.hbase.mapreduce.RowCounter$RowCounterMapper$Counters
    ROWS=3
```

As you can see our test table has 3 rows. Which is correct.

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

Reference

Hadoop Reference

Hadoop Homepage
Hadoop Releases Download Page
Official Wiki for Hadoop
Hadoop Documentation Landing Page (r1.0.2)
Hadoop 1.0.2 API
Make sure you look at your Hadoop version number and read the documentation and API for the appropriate version
Going through the MapReduce tutorial will ease your entry into MapReduce on HBase. You can still try MapReduce on HDFS even when HBase is set up, by following the link above.

Quickstarts

Hadoop Single Node Setup Quickstart Guide
Hadoop Cluster Setup Quickstart Guide

Tutorials

Basic MapReduce Tutorial
Yahoo! Hadoop Tutorial
Running Hadoop On Ubuntu Linux (Single-Node Cluster)
Running Hadoop On Ubuntu Linux (Multi-Node Cluster)
Ankit Jain's blog: Installation of hadoop in the cluster - A complete step by step tutorial

In-depth

HDFS Architecture

Book

Hadoop: The Definitive Guide

ZooKeeper Reference

ZooKeeper Homepage
ZooKeeper Releases Download Page
ZooKeeper Wiki Page
ZooKeeper 3.4 Documentation Landing Page
ZooKeeper 3.4.3 API
ZooKeeper Overview

Quickstart

ZooKeeper Standalone QuickStart
ZooKeeper Replicated Quickstart

HBase Reference

HBase Homepage

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.

HBase Releases Download Page
HBase 0.93-SNAPSHOT API
HBase 0.95-SNAPSHOT Reference (Source Code)
API specifically for MapReduce on HBase

Quickstart

HBase Single Node Quickstart
HBase Distributed Quickstart

Tutorial

HBase Pseudo-Distributed Guide
HBase Distributed Setup Guide
HBase/Hadoop on Mac OS X
Shekhar Gulati:Installing HBase over HDFS on a Single Ubuntu Box

Example Code

Example code for hooking up Java Client to HBase
example code for HFileOutputFormat

In-depth

HBase Acid Semantics
HBase Bulk Loading
HFile Structure

Book

HBase: The Definitive Guide

Note: A multi-page HTML version is available in the guide folder. You can find the guide folder in the parent directory of where this current file is located.