Running Hadoop On OS X 10.5 64-bit (Single-Node Cluster)

These instructions are for installing and running Hadoop on a OS X single node cluster (MacPro). This tutorial follows the same format and largely the same steps of the incredibly thorough and well-written tutorial by Michael Noll about Ubuntu Cluster Setup. This is pretty much his procedure with changes made for OS X users. I also added other things that I was able to piece together after looking up things from the Hadoop Quickstart and the forums /archives.

Step 1: Creating a designated hadoop user on your system

This isn't ~~entirely~~ necessary, but it's a good idea for security reasons. To add a user, go to:

```
System Preferences > Accounts
```

Click the "+" button near the bottom of the account list. You may need to unlock this ability by hitting the lock icon at the bottom corner and entering the admin username and password.

When the New account window comes out enter a name, as short name and a password. I entered the following:

```
Name: hadoop
Short name: Hadoop
Password: MyPassword (well you get the idea)
```

Once you are done, hit "create account". Now, log in as the hadoop user. You are ready to set up everything!

Step 2: Install/Configure Preliminary Software

Before installing Hadoop, there are a couple things that you need make sure you have on your system.

1. Java, and the latest version of the JDK 2. SSH

Because OS X is awesome, you actually don't have to install these things. However, you will have to enable and update what you have. Let's start with Java:

Updating Java

Open up the Terminal application. If it's not already on your dock, you can access it through

```
Applications > Utilities > Terminal
```

Next check to see the version of Java that's currently available on the system:

```
$:~ java -version
java version "1.5.0_13"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)
Java HotSpot(TM) Client VM (build 1.5.0_13-119, mixed mode, sharing)
```

You may want to update this to Java Sun 6, which is available as an update for OS X 10.5 (Update 1). It's currently only available for 64-bit machines though. You can download it here.

After you download and install the update, you are going to need to configure Java on your system so the default points to this new update. Go to:

```
Applications > Utilities > Java > Java Preferences
```

Under "Java Version" hit the radio button next to "Java SE 6" Down by "Java Application Runtime Settings" change the order so Java SE 6 (64 bit) is first, followed by Java SE 5 (64 bit) and so on. Hit "Save" and close this window.

Now, when you go to the terminal, and type in "java -version" you should get the following:

```
$:~ java -version
java version "1.6.0_05"
Java(TM) SE Runtime Environment (build 1.6.0_05-b13-120)
Java HotSpot(TM) 64-Bit Server VM (build 1.6.0_05-b13-52, mixed mode)
```

and for "javac -version":

```
$:~ javac -version
javac 1.6.0_05
```

Onto ssh!

SSH: Setting up Remote Desktop and Enabling Self-Login

SSH also comes installed on your Mac. However, you need to enable access to your own machine (so hadoop doesn't ask you for a password at inconvenient times). To do this, go to

```
System Preferences > Sharing (under Internet & Network)
```

Under the list of services, check "Remote Login". For extra security, you can hit the radio button for "Only these Users" and select **hadoop**

Now, we're going to configure things so we can log into localhost without being asked for a password. Type the following into the terminal:

```
$:~ ssh-keygen -t rsa -P ""
$:~ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Now try:

```
$:~ ssh localhost
```

You should be able to log in without a problem.

You are now ready to install Hadoop. Let's go to step 3!

Step 3: Downloading and Installing Hadoop

So this actually involves a few smaller steps:

1. Downloading and Unpacking Hadoop 2. Configuring Hadoop

After we finish these, you should be ready to go! So let's get started:

Downloading and Unpacking Hadoop

Download Hadoop. Make sure you download the latest version (as of this post, 0.17.2 and 0.18.0 are the latest versions). We call our generic version of hadoop hadoop-* in this tutorial.

Unpack the hadoop-*.tar.gz in the directory of your choice. I placed mine in /Users/hadoop. You may also want to set ownership permissions for the directory:

```
$:~ tar -xzvf hadoop-*.tar.gz
$:~ chown -R hadoop hadoop-*
```

Configuring Hadoop

There are two files that we want to modify when we configure Hadoop. The first is conf/hadoop-env.sh . Open this in nano or your favorite text editor and do the following:

- uncomment the export JAVA_HOME line and set it to /Library/Java/Home
- uncomment the export HADOOP_HEAPSIZE line and keep it at 2000

You may want to change other settings as well, but I chose to leave the rest of hadoop-env.sh the same. Here is an idea of what part of mine looks like:

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.  Required.
 export JAVA_HOME=/Library/Java/Home

# Extra Java CLASSPATH elements.  Optional.
# export HADOOP_CLASSPATH=

# The maximum amount of heap to use, in MB. Default is 1000.
 export HADOOP_HEAPSIZE=2000
```

The next part that we need to set up is hadoop-site.xml. The most important parts to set here are hadoop.tmp.dir (which should be set to the directory of your choice) and to add mapred.tasktracker.maximum property to the file. This will effectively set the maximum number of tasks that can simulataneously run by a task tracker. You should also set dfs.replication 's value to one.

Below is a sample hadoop-site.xml file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
  <name>hadoop.tmp.dir</name>
  <value>/Users/hadoop/hadoop-0.17.2.1/hadoop-${user.name}</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</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>

<property>
  <name>mapred.job.tracker</name>
  <value>localhost:9001</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>

<property>
<name>mapred.tasktracker.tasks.maximum</name>
<value>8</value>
<description>The maximum number of tasks that will be run simultaneously by a
a task tracker
</description>
</property>

<property>
  <name>dfs.replication</name>
  <value>1</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>

</configuration>
```

Now to our last step!

Step 4: Formatting and Running Hadoop

Our last step involves formatting the namenode and testing our system.

```
$:~ hadoop-*/bin/hadoop namenode -format
```

This will give you output along the lines of

```
$:~ hadoop-*/bin/hadoop namenode -format
 08/09/14 21:22:14 INFO dfs.NameNode: STARTUP_MSG:
 /************************************************************
 STARTUP_MSG: Starting NameNode
 STARTUP_MSG:   host = loteria/127.0.0.1
 STARTUP_MSG:   args = [-format]
 ************************************************************/
 08/09/14 21:22:14 INFO dfs.Storage: Storage directory [...] has been successfully formatted.
 09/09/14 21:22:14 INFO dfs.NameNode: SHUTDOWN_MSG:
 /************************************************************
 SHUTDOWN_MSG: Shutting down NameNode at loteria/127.0.0.1
 ************************************************************/
```

Once this is done, we are ready to test our program.

Running Hadoop

First, start up the DFS. This will start up a TaskTracker, JobTracker, and DataNode on the machine.

```
$:~ hadoop-*/bin/start-all.sh
```

As input for our test, we are going to copy the conf folder up to our DFS.

```
$:~ hadoop-*/bin/hadoop dfs -copyFromLocal hadoop-*/conf input
```

You can check to see if this actually worked by doing an ls on the dfs as follows:

```
$:~ hadoop-*/bin/hadoop dfs -ls
Found 1 item
/user/hadoop/input    <dir>        2008-09-11 13:33    rwxr-xr-x    hadoop supergroup
```

Now, we need to compile the code. cd into the hadoop-*/ directory and do:

```
$:~ ant examples
```

This will compile the example programs found in hadoop-*/src/examples

Now, we will run the example distributed grep program on the conf program as input.

```
$:~ hadoop-*/bin/hadoop jar hadoop-*-examples.jar grep input output 'dfs[a-z.]+'
```

If this works, you'll see something like this pop up on your screen:

```
08/09/13 20:47:24 INFO mapred.FileInputFormat: Total input paths to process : 1
08/09/13 20:47:24 INFO mapred.JobClient: Running job: job_200809111608_0033
08/09/13 20:47:25 INFO mapred.JobClient:  map 0% reduce 0%
08/09/13 20:47:38 INFO mapred.JobClient:  map 13% reduce 0%
08/09/13 20:47:39 INFO mapred.JobClient:  map 16% reduce 0%
08/09/13 20:47:43 INFO mapred.JobClient:  map 22% reduce 0%
08/09/13 20:47:44 INFO mapred.JobClient:  map 24% reduce 0%
08/09/13 20:47:48 INFO mapred.JobClient:  map 33% reduce 0%
08/09/13 20:47:53 INFO mapred.JobClient:  map 41% reduce 0%
08/09/13 20:47:54 INFO mapred.JobClient:  map 44% reduce 0%
08/09/13 20:47:58 INFO mapred.JobClient:  map 50% reduce 0%
08/09/13 20:47:59 INFO mapred.JobClient:  map 52% reduce 0%
08/09/13 20:48:03 INFO mapred.JobClient:  map 61% reduce 0%
08/09/13 20:48:08 INFO mapred.JobClient:  map 69% reduce 0%
08/09/13 20:48:09 INFO mapred.JobClient:  map 72% reduce 0%
08/09/13 20:48:13 INFO mapred.JobClient:  map 78% reduce 0%
08/09/13 20:48:14 INFO mapred.JobClient:  map 80% reduce 0%
... and so on
```

The last step is to check if you have output!

You can do this by doing:

```
$:~ hadoop-*/bin/hadoop dfs -ls output
Found 2 items
/user/hadoop/output/_logs <dir>  2008-09-13 19:21 rwxr-xr-x hadoop supergroup
/user/hadoop/output/part-00000 <r 1> 2917 2008-09-13 20:10 rw-r--r-- hadoop supergroup
```

The most important part is that the number next to the <r 1> should **not** be 0.

To check the actual contents of the output do:

```
$:~ hadoop-*/bin/hadoop dfs -cat output/*
```

Alternatively, you can copy it to local disk and check/modify it:

```
$:~ hadoop-*/bin/hadoop dfs -copyToLocal output myoutput
$:~ cat myoutput/*
```

Stopping the Hadoop DFS

When you're done running jobs on the dfs, run the stop-all.sh command.

```
$:~ hadoop-*/bin/stop-all.sh
```

That's all! Happy Map Reducing!