CS380 Final Project

Evaluating the Scalability of Hadoop in a Real and Virtual Environment

James Devine

December 15, 2008

Abstract

Mapreduce has been a very successful computational technique that has greatly helped Google grow. While Google does not release their source code, there are projects such as Hadoop that allow anyone to implement a MapReduce system. The power of mapreduce lies in its ability to break up a very computationally intense task into a number of much small pieces that can be worked on by computers in a cluster. This paper seeks to provide background on Hadoop and outlines the process of getting a Hadoop Cluster running. It then presents the results of an empirical study conducted on the scalability of Hadoop in a real and virtual environment.

1 What is Hadoop

Hadoop is an open source Java based distributed file system (dfs) with built in support for MapReduce. The distributed file system is referred to as the Hadoop Distributed File System (HDFS).[**?**]

1.1 How it works

The Hadoop dfs service stores files on the local hard disk of each node of the cluster. The HDFS capacity is equal to the sum of the free space on all nodes of the cluster. The default configuration provides three way replication so each file created is stored on three separate nodes. The MapReduce service runs ontop of the HDFS. A user submits a program and input to Hadoop to be processed. Hadoop then, using MapReduce, runs the program on the specified input. The MapReduce process is broken into two parts- map and reduce.

During the map process the input file(s) are split up into many smaller pieces. Each piece is sent to a node in the cluster to be computed. The map process creates *(key, value)* pairs and then uses a mapper task. The mapper task is written in the program that is being run and specifies what should be done to the *(key, value)* pairs. A combine

function can be specified by the program as part of of the map process to decrease the amount of output by combining *(key, value)* pairs. The reduce process collects all of the solutions from each of the nodes in the cluster are combines it into one file.[**?**]

1.2 Getting Hadoop

Hadoop can be downloaded from http://hadoop.apache.org/core/releases.html.

2 Hadoop Setup

The following sections outline the steps required to get a Hadoop Cluster Up and Running. [**?**]

2.1 Prerequisites

- Java 1.6
- JAVA_HOME Environment Variable set to the java installation location
- SSHD
- SSH
- SSH key generated for each machine in the cluster

Generating an SSH key for each machine on the cluster is an important step. The scripts to start Hadoop will need to be able to log into the other machines in the cluster without entering a password. This can be done with the following command:

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

After running the commands above, each machine in the cluster should be connected to with ssh.

The JAVE_HOME environment variable can be set using the a setenv statement in the .cshrc file (under a user's home directory) if using a C shell.

```
setenv JAVA_HOME=location_of_java
```

2.2 Extracting the Software

Once the Hadoop software has been downloaded it needs to be extracted to a directory on the local hard drive. Then a temporary directory needs to be created so Hadoop has a place to store the distributed files. In this example the folder tmp was created in the Hadoop installation directory.

2.3 Configuring Hadoop

The main settings for Hadoop reside in the hadoop-site.xml file under the conf directory. Initially the file is empty and needs to be configured. There are many values that can be specified, if a specific value is not specified the default value is used (the default values reside in the hadoop-default.xml file in the conf directory). The hadoop.tmp.dir parameter specifies the temporty directory that Hadoop uses to save dfs files. The fs.default.name parameter is set to the address of the master server in the cluster that is running the NameNode service. The mapred.job.tracker parameter is set to the server that is running the JobTracker service. The dfs.block.size parameter specifies the block size that the HDFS uses for file storage. The following is the configuration file from this experiment.

```
<?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>/opt/devinej/hadoop/tmp/\${user.name}</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://aldenv167:54310</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>

<property>
  <name>mapred.job.tracker</name>
  <value>aldenv167:54311</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>

<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.block.size</name>
 <value>2097152</value>
 <description>Sets the block size to 2 MB since most of the files used in
 the empirical study are between 250KB-2MB
 </description>
</property>
</configuration>
```

After editing the hadoop-site.xml file, the masters and slaves files need to be edit. Each file resides in the conf folder. The masters file contains a list of master server ip address or host names. This is usually just set to one machine for simple installations; the best machine for this would be the machine that is set up with Hadoop first. The slaves files is edited next. The slaves file should contain the host name or ip address of each machine that will run Hadoop. An example slaves file is provided below.

```
141.195.226.167
141.195.226.168
141.195.226.169
141.195.226.170
141.195.226.171
141.195.226.172
141.195.226.173
141.195.226.174
```

2.4 Deploying the Hadoop Installation to the Servers in the Cluster

The installation directory needs to be copied to each of the other servers in the cluster. This can be easily done by using the *scp* command. A sample of the command is given below. This will copy all of the installation and configuration files to the specified remote machine.

```
scp -r /hadoop_install_directory remote_computer:/hadoop_install_directory
```

Through the course of the experimentation it was necessary to write a short script to deploy the hadoop config files to all the computer in the cluster. It could very easily by modified to deploy the entire Hadoop install folder. The script can be found below.

```
#!/bin/sh

install_dir=/opt/devinej/hadoop
```

```
#Copy the configuration files to each of the machines in the cluster
for((i =167; i<=174; i++))
do
scp $install_dir/conf/masters aldenv$i:$install_dir/conf
scp $install_dir/conf/slaves aldenv$i:$install_dir/conf
scp $install_dir/conf/hadoop-site.xml aldenv$i:/$install_dir/conf
done
```

2.5 Formatting the Namenode

On the master server the namenode must be formated before the first run of Hadoop. Formatting the namenode prepares the temp folder specified in the config file for storing data as part of the HDFS. If there is already data in the HDFS it will be erased. The command below is a sample on how to format the namenode.

```
install_dir/bin/hadoop dfs namenode -format
```

2.6 Starting the DFS Service

The DFS service is the main distributed file system service. The script starts the dfs service on each server in the cluster; it resides in bin directory and is called start-dfs.sh. The script will reads in the list of servers from the slaves file in the conf directory. The following command demonstrates the execution of the services.

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

Once the dfs service is started, typing the jps command on each slave server in the cluster should show *DataNode*. On the master server typing in jps should show *DataNode, NameNode,* and *SecondaryNameNode*. This indicates that the dfs service is running correctly.

2.7 Starting the MapReduce Service

The MapReduce service supports the use of MapReduce. The script to execute the service on each node in the cluster is called start-mapred.sh and resides in the bin directory. The command to execute the command is shown below.

```
hadoop_install_directory/bin/start-mapred.sh
```

Once the mapred service, typing the jps command on each slave server in the cluster should show *TaskTaracker* (in addition to *DataNode*). On the master server typing in jps should show *JobTracker*, (in addition to *DataNode, NameNode,* and *SecondaryNameNode*). This indicates that the MapReduce Service is correctly running. Note that to properly use Hadoop the mapred service must be running.

2.8 Accessing the HDFS

The HDFS can be accessed though the hadoop executable. A variety of typical Linux comands can be passed. The HDFS can also be accessed through the web interface at `http://master_node:50070`. A sample HDFS command is given below that will list everything in the top directory of the HDFS.

```
install_dir/bin/hadoop dfs -ls
```

2.9 Sending a MapReduce Job to Cluster

The hadoop executable allows for MapReduce job to be sent to the cluster. The syntax is provided below. The status of the job can be viewed at `http://master_node:50030`

```
install_dir/bin/hadoop jar program.jar org.myorg.ProgramName /input /output
```

3 Running Hadoop In a Virtual Environment

Hadoop was set up on a cluster of virtual machines in VMware Workstation. The computer had 8 GB of RAM with a Q6700 Core 2 Quad Core processor and was running Windows Vista.

3.1 Virtual Machine Configuration

Table 1: Configuration of each virtual machine

Operating System	Ubuntu Server 8.10
Processor	2.6 GHz
RAM	512 MB
Hard Drive	10 GB

3.2 Setting up the Environment

To create the virtual environment a Ubuntu Server 8.10 virtual machine was set up and Hadoop was installed. The virtual machine was then cloned 7 times to create the remaining machines in the environment.

4 Running Hadoop In a Real Environment

This section outlines how Hadoop was run in a real environment.

4.1 Real Machine Configuration

.

Table 2: Configuration of each real machines

Operating System	Fedora 8
Processor	Pentium 4 1.7 GHz
RAM	1 GB
Hard Drive	40 GB

4.2 Setting up the Environment

To begin setting up the environment a local directory was created on 8 machines in the lab by the Systems Administrator. There was initially an issue with the network configuration and a conflict arising from the iptables that had to be resolved. After the issue was resolved the configuration files were written and deployed to each machine in the cluster using the methodology outlined in the previous section.

5 Creating a Performance Metric for Measuring the Scalability Hadoop

Hadoop's example MapReduce Java program was used to measure the performance of each cluster. The wordcount program counts the occurrence of each word in the input. To create a dataset for the wordcount program to use, 438 ebooks totaling 289MB were randomly download from Project Gutenberg. The main metric used to judge performance was the time it took each cluster size to run wordcount on the collection of ebooks.

5.1 Building the Wordcount Sample Program

The WordCount.java files was downloaded from Hadoop's website. It was then packed into a jar file. The commands are shown below.[**?**]

```
mkdir wordcount_classes
javac -classpath install_dir/hadoop-0.10.9-core.jar -d wordcount_classes WordCount.java
jar -cvf /install_dir/wordcount.jar -C wordcount_classes/
```

5.2 Running the Experiment

To run the experiment a script was written that ran wordcount 5 times and saved the output to a file. It then decreased the size of the cluster by one, formatted the namenode, and then copied the ebooks into the HDFS. The script was set to start out with a cluster of 8 nodes and run until it had reduced the cluster size to 2 nodes.

```
#!/bin/bash
install_dir=/opt/devinej/hadoop

for((i =7; i>2; i--))
do
for((j=5; j>0;j--))
do
#run the mapred and save contents to a file
$install_dir/bin/hadoop jar $install_dir/wordcount.jar org.myorg.WordCount
/ebooks /out$j &> $install_dir/output/$i"real-"$j".txt"

#copy result to the hdd
$install_dir/bin/hadoop dfs "-copyToLocal" /out$j /$install_dir/output/$i"real-"$j

done

#take a node out of the cluster and restart cluster
$install_dir/remove_data.sh
done
```

The script relies on another script that is called remove_data.sh that stops the Hadoop services on all the machines in the cluster, removes a machine from the cluster, formats the name node, and then brings the cluster back online. The script is provided below.

```
#!/bin/bash

install_dir=/opt/devinej/hadoop

"$install_dir/bin/stop-all.sh"

#remove the last entry from the slaves file
sed -n '$! {p}' < $install_dir/conf/slaves | cat > $install_dir/conf/slaves

#deploy the conf files
$install_dir/deploy_conf.sh

#Remove the local hadoop file from each node
for((i =167; i<=174; i++))
do
ssh aldenv$i "rm -r $install_dir/tmp/devinej"
done

#format the namenode
$install_dir/bin/hadoop namenode -format

#start the dfs and mapred services
```

```
"$install_dir/bin/start-all.sh"
$install_dir/bin/hadoop dfs -copyFromLocal \$install_dir/ebooks /ebooks
```

6 The Scalability of Hadoop

The results from the experiment showed that adding more nodes to the cluster greatly decreased the time required to run the wordcount program in both the virtual and real environments. The graph of the real and virtual runtime both begin to show signs of diminishing returns. This suggests that at some point adding more nodes to the cluster will not improve the runtime. In the context of what Hadoop was designed for, the clusters and data set used in the experiment are both considered small. Hadoop was built to handle much larger data sets running on clusters with many more nodes. Given the relatively under power of the machines used in the real cluster the results were fairly significant. Using 8 nodes to run the wordcount program nearly reduced the runtime by 75% when compared to using 2 nodes. Assuming that this trend could be achieved in other MapReduce programs, a job that would take a month to compute on a single machine could be executed in about week on the cluster thart was configured. The additions of more machines in the cluster could lead to an even greater reduction in runtime.

The curve for the virtual clusters has an interesting shape. The results showed that as the cluster size increased the runtime continued to decrease. However the results of diminishing returns can be clearly seen as a result of the unavailability of free resources. Running 8 virtual machines put a considerable load on the host computer running the virtualization software and pushed the CPU utilization to 100%. Perhaps the most interesting results from the study was that adding more virtual machines decreased the time required to run wordcount on the data set. This indicates that the use of virtualization helped better utilize the resources of the host computer.

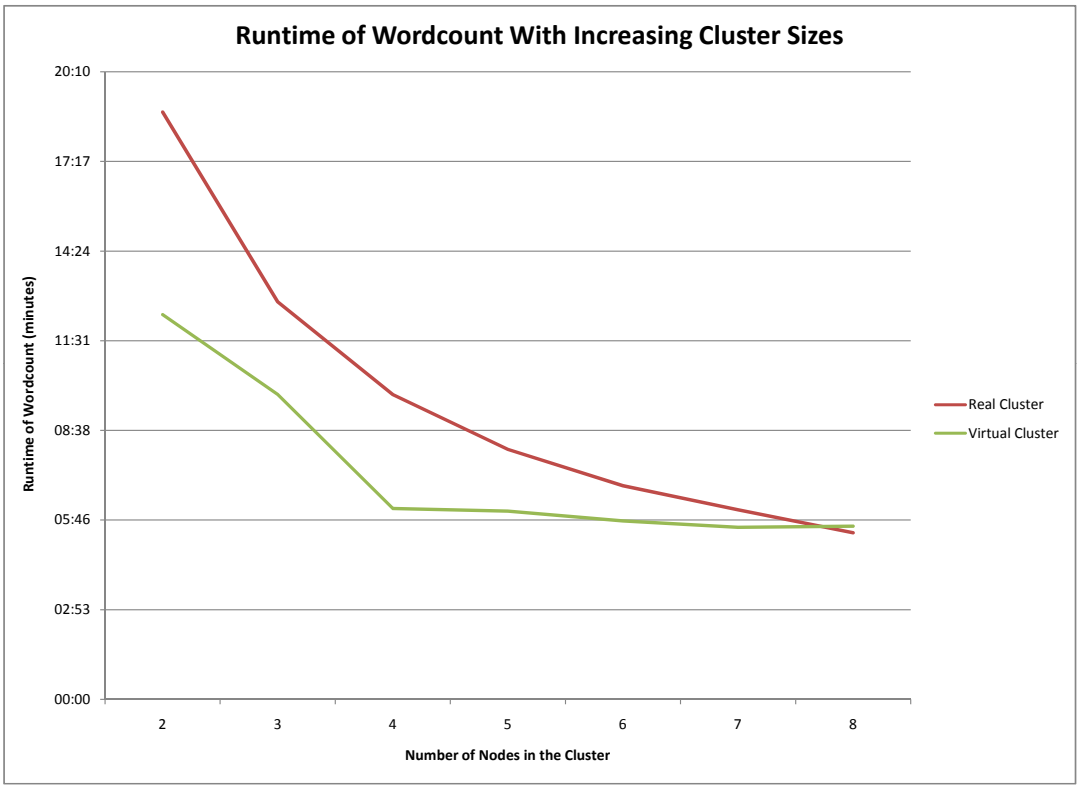

Table 3: Results from Running Wordcount On Various Cluster Sizes

Number of Nodes	Real Cluster Runtime	Virtual Cluster Runtime
2	0:18:52	0:12:22
3	0:12:46	0:09:48
4	0:09:47	0:06:08
5	0:08:02	0:06:03
6	0:06:52	0:05:44
7	0:06:05	0:05:31
8	0:05:21	0:05:33

7 Future Research

There is much future research that could come out of this project. The potential research can be broken up into two main areas, that of experimentation and exploration.

In the area of experimentation a more comprehensive performance study could be done. There are many parameters in Hadoop that can be customized that could potentially increase the performance of the MapReduce process. Experiments could also be conducted with larger clusters and more demanding MapReduce tasks that require much larger data sets.

In the area of exploration a more in depth study of MapReduce could be conducted. This could involve writing programs that make use of the MapReduce process that Hadoop provides. An attempt to make an "efficient" solution to an NP-complete problem would be an intersting application. Another area of exploration could involve setting up a permanent cluster in Alden Hall.

References

[1] The Apache Software Foundation. *Hadoop*. `http://hadoop.apache.org`. 2008.

[2] The Apache Software Foundation. *Hadoop*. `http://hadoop.apache.org/core/docs/r0.19.0/mapred_tutorial.html`. 2008.

[3] Kurdyumov, Andrey. *HadoopMapReduce*. `http://wiki.apache.org/hadoop/HadoopMapReduce`. 2008.

[4] Noll, Michael G. *Running Hadoop On Ubuntu Linux (Multi-Node Cluster)*. `http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_(Multi-Node_Cluster)`. 2008.