CprE 419 Lab 1: Using the Cloud and HDFS

Department of Electrical and Computer Engineering
Iowa State University
Spring 2013

Purpose

We have setup an environment on the cloud for large-scale data analysis. The first goal of this lab is get familiar with this environment, called “Hadoop”. The second goal is to get familiar with a distributed file system for storing large data sets, called “HDFS”, which is short for “Hadoop Distributed File System”. Specifically, at the end of this lab, you will know how to:

- Login to the cloud with your user id
- Transfer files between the local machine and the cloud
- Use HDFS and the HDFS Programming Interface
- Write, compile and execute a Hadoop program on the Cloud

Submission

Create a zip (or tar) archive with the following and hand it in through blackboard**.**

- A write-up answering questions for each experiment in the lab. For output, cut and paste results from your terminal and summarize when necessary.
- Commented Code for your program. Include all source files needed for compilation.

Resources

You will need the following before you start the lab. If you don’t have these, please contact the instructors.

- IP Address of the Hadoop Master Node
- Login id and private key file
- URL for the Hadoop Namenode

Logging Into the Cloud

For Windows:

You can use Putty to login to the cloud. If in case you do not have Putty (and PuttyGen) installed in your system, you can download it from the link below:
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

You will also need PuttyGen to generate your Private key in Putty format (one time) as follows:

- Start PuttyGen.exe.
- Click *Load*, locate your private key file in your local disk, and click *Open*.
- Leave your key passphrase blank. Click *Save private key*.
- Select the location where you want to save the PuTTY key file and the file name (for example key.ppk) and then click *Save*. Make sure this location is on your U Drive. Note that on the lab machines, your Desktop is on the U Drive.
- Click *Save public key*, specify a different name if you want, and then click *Save*.

Now use Putty.exe to connect to the Cloud as follows:

- Start PuTTY.
- In the Host name (or IP address) field, type the IP address of the Hadoop Master Node.
- Select *Connection -> Data* from the left category panel and type <User Name> in the Auto-login username field.
- Select *Connection -> SSH -> Auth* from the left category panel and select *Allow attempted changes of username* in SSH-2 in Authentication parameters.
- Click *Browse*, select the PuTTY private key file you generated by PuTTYgen, and click *Open*.
- Click *Open* and you will see a page that tells you that you are now connected to the Hadoop Master Node.

For more details check the support link as provided below:
http://www.ibm.com/developerworks/cloud/library/cl-cloudstart.html

For UNIX-like systems:

Save your private key in *~/.ssh/*

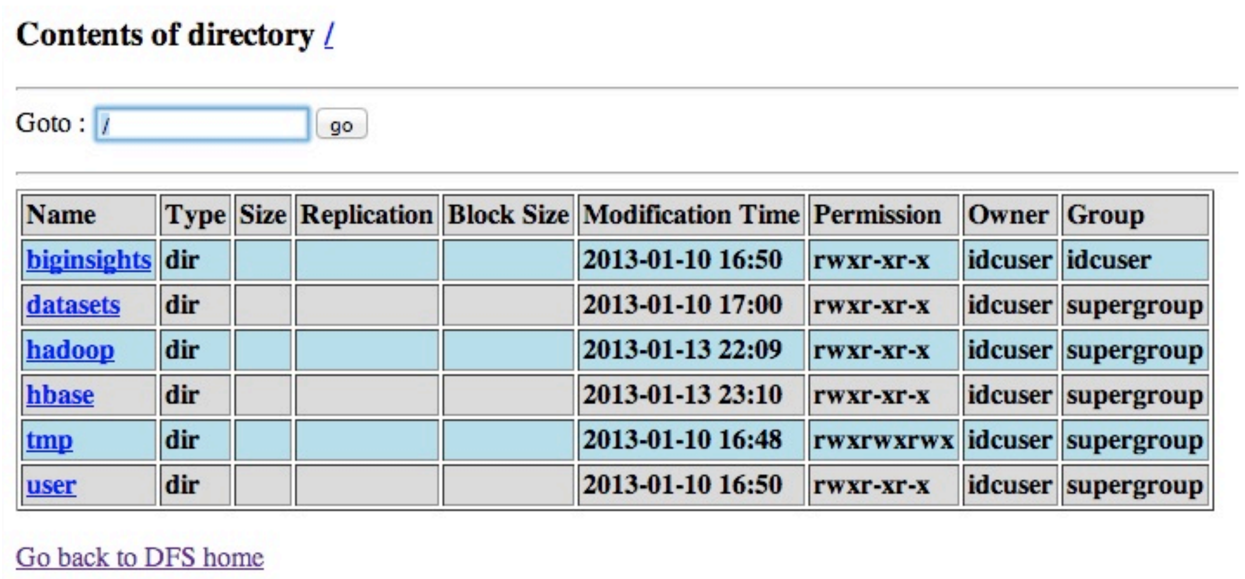
The following command helps you to log in to the server using ssh:

```
$ ssh -i ~/.ssh/<private_key_file> <user ID>@<Hadoop Master IP Address>
```

HDFS Usage

HDFS is a distributed file system that is used by Hadoop to store large files. It automatically splits a file into many "blocks" of no more than 64MB and stores the blocks on separate machines. This helps the system to store huge files, much larger than any single machine can store. The HDFS also creates replicas for each block (by default 3). This helps to make the system more fault-tolerant. HDFS is an integral part of the Hadoop eco-system. HDFS is an Open Source implementation of the Google File System.

In addition to the command line, the HDFS file-system can be viewed through a web browser using the "NameNode" link provided to you. Go to the link provided and click on *Browse the filesystem*. It would similar to the screenshot provided below. You can now explore the Distributed File System in the server through your web browser!



Contents of directory /

Goto : / go

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
biginsights	dir				2013-01-10 16:50	rwxr-xr-x	idcuser	idcuser
datasets	dir				2013-01-10 17:00	rwxr-xr-x	idcuser	supergroup
hadoop	dir				2013-01-13 22:09	rwxr-xr-x	idcuser	supergroup
hbase	dir				2013-01-13 23:10	rwxr-xr-x	idcuser	supergroup
tmp	dir				2013-01-10 16:48	rwxrwxrwx	idcuser	supergroup
user	dir				2013-01-10 16:50	rwxr-xr-x	idcuser	supergroup

Go back to DFS home

You can also use the terminal to navigate the HDFS. However, you cannot use the shell commands to do so (one way to overcome this is to use the FUSE project @ http://wiki.apache.org/hadoop/MountableHDFS). HDFS is not compatible with the shell commands and hence you will need the utilities provided by the Hadoop project to navigate the HDFS filesystem.

Log in to the server as described above.

To try out different HDFS utilities, try the command:

```
$ hadoop fs –help
```

You will notice that the HDFS utilities are very similar to that of the default UNIX filesystem utilities and hence should be easy to learn.

Experiment 1 (10 points):

Create a directory called /user/<your login id>/Lab1 under HDFS and create a new file called "afile.txt" in this directory, and write something into this file. Show this to the instructor. The file will be visible through the web browser.

Writing, Compiling, and Executing Programs using Hadoop

Next we will see how to compile a Hadoop program in the Cloud. We will use the Java programming language, as that is the default language used with Hadoop.

```
import java.io.*;
import java.lang.*;
import java.util.*;
import java.net.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.util.*;


public class HDFSWrite  {

    public static void main ( String [] args ) throws Exception {

        // The system configuration
        Configuration conf = new Configuration();

        // Get an instance of the Filesystem
        FileSystem fs = FileSystem.get(conf);

        String path_name = "/user/<Your ID>/Lab1/newfile";

        Path path = new Path(path_name);

        // The Output Data Stream to write into
        FSDataOutputStream file = fs.create(path);

        // Write some data
        file.writeChars("The first Hadoop Program!");

        // Close the file and the file system instance
        file.close();
        fs.close();

    }

}
```

The above Java program uses the HDFS Application Programming Interface (API) provided by Hadoop to write into a HDFS file.

Paste the code into your favorite Java editor and change the following:
String path_name = "/user/<Your ID>/Lab1/newfile"; to reflect your actual User ID.

We will transfer this program to the Cloud, compile it and then run it to check the output.
You can also compile the program on your local machine if you have the Hadoop libraries downloaded.

Transferring Files from/to the Cloud

For Windows:

WinSCP is an open source SFTP and FTP client for Microsoft Windows. It can be used to transfer files securely between a remote and a local machine. You can get the WinSCP client from http://winscp.net/eng/index.php

To connect to the Hadoop Master node:

- Use the converted private key in Putty format that you created using PuttyGen
- Start WinSCP.
- Type the IP address or host name of the Hadoop Master and type the user name <User ID>. Leave the password blank.
- Locate the private key file.
- Click **Login**.
- During the first login, you might get a warning message if the server's key is not cached. Click **Yes** to continue.

For UNIX-like systems:

Save your private key in *~/.ssh/*

Issue the commands as described below from your local UNIX-like system to transfer files between the Cloud and your local system.

To transfer a file from your local machine to the Hadoop Master Server:

```
$ scp -i ~/.ssh/<private_key_file> <filename> <user ID>@<server_IP>:/home/<user ID>/<filename>
```

To transfer a file from the server to your local machine:

```
$ scp -i ~/.ssh/<private_key_file> <user ID>@<server_IP>:/home/<user ID>/<filename> <filename>
```

Compiling and Running Hadoop Programs

Compile the Java program on the server as follows.

```
$ mkdir class
$ javac -classpath $HADOOP_HOME/hadoop-core-1.0.0.jar -d class/ HDFSWrite.java
$ jar -cvf HDFSWrite.jar -C class/ .
$ rm -rf class
```

Ececute the Hadoop program on the server as follows:

```
$ hadoop jar HDFSWrite.jar HDFSWrite
```

Check the HDFS path *"/user/<Your ID>/Lab1/newfile"* to verify the output of the program. You should see the file created with the contents as given in the program.

You can also compile the program and generate the jar file on a local machine using Eclipse. For further instructions, see the manual on “Compiling Hadoop With Eclipse” that we have linked from the class website.

HDFS API

As seen in the last section, HDFS can also be accessed using a Java API, which allows us to read/write into the file system. For documentation, see:
http://hadoop.apache.org/docs/r1.0.0/api/org/apache/hadoop/fs/FileSystem.html

The FSDataOutputStream and FSDataInputStream can be used to read / write files in HDFS respectively. Check out their class documentations as well.

Read the source code of the program that you just compiled. That would help you to better understand how the HDFS API works. Then, look at the different classes that were used and learn more about the various methods they provide. All the methods can be found in the Hadoop API documentation. Once you have browsed through the HDFS API, go onto the following experiment.

Experiment 2 (40 points):

Write a program using the Java HDFS API that reads the contents of the HDFS file "/datasets/Lab1/bigdata" and computes the 8-bit XOR checksum of all bytes whose offsets range from 5000000000 till 5000000999, both endpoints inclusive. Print out the 8-bit checksum.

Attach the Java code in your submission, as well as the XOR output.

For instance, the XOR checksum of the bitstring "0000000111111110000000011111111" is "00000000".