Lab 2 Report

Anuj Sampat
asampat@bu.edu

1. Problem

Process the Apache log and output IPs that have hit the server more than hundred times. Additionally, sort the results to display the IPs in descending order of hit count.

2. System Configuration

OS: Ubuntu 14.04 LTS 64-bit
Home Directory: /home/anuj
Hadoop Version: hadoop-0.20.2
Hadoop Installation Directory: /home/anuj/hadoop-0.20.2
Java Installation Directory: /home/anuj/jdk1.8.0_05

3. Solution

The solution was implemented by running two MapReduce (MR) jobs in the Hadoop MapReduce framework as explained below:

MR Job 1:

The map step of this job extracts an IP from a line in the Apache log and outputs it as a key with a value of one for its hit count to give the output (key<IP>, value<1>). The reduce step aggregates the total hit count for a given IP and if it turns out to be greater than hundred, the IP is output as a key with the total hit count as the value, to give (key<IP>, value<total count>).

MR Job 2:

The map step of this job takes the (key<IP>, value<total count>) output from the Job 1 Reducer as its input and outputs it as (key<total count>, value<IP>) to obtain a descending sort on the count. The reduce step for this job simply takes the (key<total count>, value<IP>) output from the Mapper as its input and outputs it as (key<IP>, value<total count>).

3.1 Classes Used

ApacheIPHitCnt.java: Main program/class to configure and execute Job 1 and Job 2.
IpMapper: Internal class serving as the mapper for Job 1.
IpReducer: Internal class serving as the reducer for Job 1.
IpSorterMapper: Internal class serving as the mapper for Job 2.
IpSorterReducer: Internal class serving as the reducer for Job 2.

Please refer to the comments in ApacheIPHitCnt.java for additional implementation details.

3.2 Running The Program in Hadoop

ApacheIPHitCnt reads the Apache log to be analyzed from an input directory specified by the first command line argument. The sorted output is stored in an output directory specified by the second command line argument.

3.3 Compiling And Packaging

The following steps were followed:

1. Created a work folder 'work/apacheip' and copied ApacheIPHitCnt.java to that folder. Created a sub-directory called as 'classes' in this folder. Then compiled using the command:

```
$ javac -classpath ~/hadoop-0.20.2/hadoop-0.20.2-core.jar -d classes/ ApacheIPHitCnt.java
```

2. Created a package called 'ApacheIPHitCnt.jar' using the command:

```
$ jar cvf ApacheIPHitCnt.jar -C classes/ .
```

This created the jar file in the work folder.

3.4 Running Using The Local File System:

1. Created a sub-directory called as 'input' and copied the provided sample Apache log (access_log) to that directory. Created another sub-directory called as 'output'. Then executed ApacheIPHitCnt.jar through Hadoop using the command:

```
$ bin/hadoop jar ApacheIPHitCnt.jar org.myorg.ApacheIPHitCnt input output
```

Two sub-directories called as 'ipCnt' and 'ipCntSorted' were created by ApacheIPHitCnt in the 'output' directory, each containing the following files:

output/ipCnt/part-0000: File containing the output of Job 1 (IPs and their total hit count if > 100).

output/ipCntSorted/part-0000: File containing the output of Job 2 (IPs sorted in descending hit count).

3.5 Running Using HDFS in Psuedo-Distributed Mode

1. The following XML was added to conf/hdfs-site.xml to configure Hadoop HDFS:

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/anuj/hadoop-0.20.2/hadoop-tmp</value>
    </property>
    <property>
        <name>fs.default.name</name>
       <value>hdfs://localhost:54310</value>
    </property>
    <property>
        <name>mapred.job.tracker</name>
        <value>hdfs://localhost:54311</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>8</value>
    </property>
    <property>
        <name>mapred.child.java.opts</name>
        <value>-Xmx512m</value>
    </property>
</configuration>
```

2. The following XML was added to conf/mapred-site.xml to configure the Hadoop MapReduce JobTracker:

```
<configuration>
    <property>
        <name>mapred.job.tracker</name>
       <value>hdfs://localhost:54311</value>
    </property>
</configuration>
```

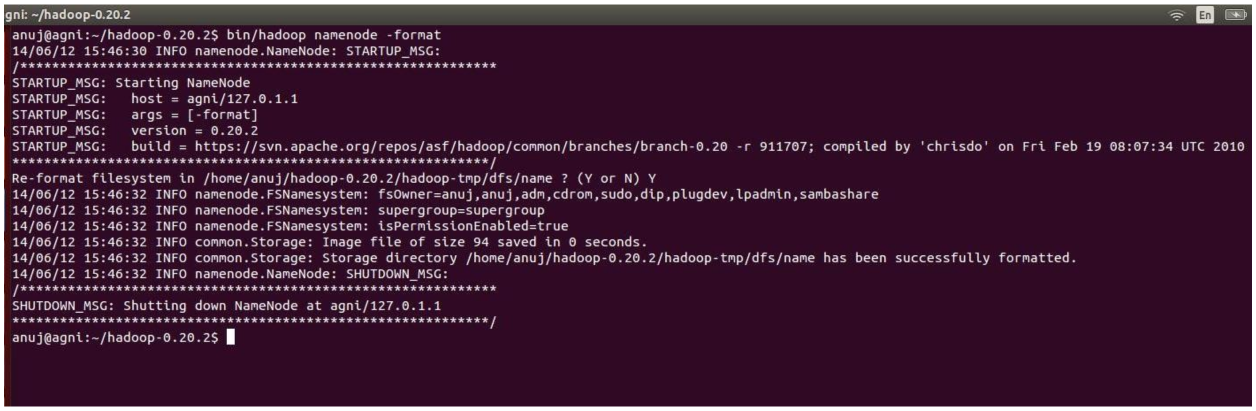
3. The Hadoop tmp directory was then created:

```
$ mkdir -p /home/anuj/hadoop-0.20.2/hadoop-tmp/dfs/name
```

4. The file system was then formatted using the command:

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

(screen shot on next page)



```
anuj@agni:~/hadoop-0.20.2$ bin/hadoop namenode -format
14/06/12 15:46:30 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = agni/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.2
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20 -r 911707; compiled by 'chrisdo' on Fri Feb 19 08:07:34 UTC 2010
************************************************************/
Re-format filesystem in /home/anuj/hadoop-0.20.2/hadoop-tmp/dfs/name ? (Y or N) Y
14/06/12 15:46:32 INFO namenode.FSNamesystem: fsOwner=anuj,anuj,adm,cdrom,sudo,dip,plugdev,lpadmin,sambashare
14/06/12 15:46:32 INFO namenode.FSNamesystem: supergroup=supergroup
14/06/12 15:46:32 INFO namenode.FSNamesystem: isPermissionEnabled=true
14/06/12 15:46:32 INFO common.Storage: Image file of size 94 saved in 0 seconds.
14/06/12 15:46:32 INFO common.Storage: Storage directory /home/anuj/hadoop-0.20.2/hadoop-tmp/dfs/name has been successfully formatted.
14/06/12 15:46:32 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at agni/127.0.1.1
************************************************************/
anuj@agni:~/hadoop-0.20.2$
```

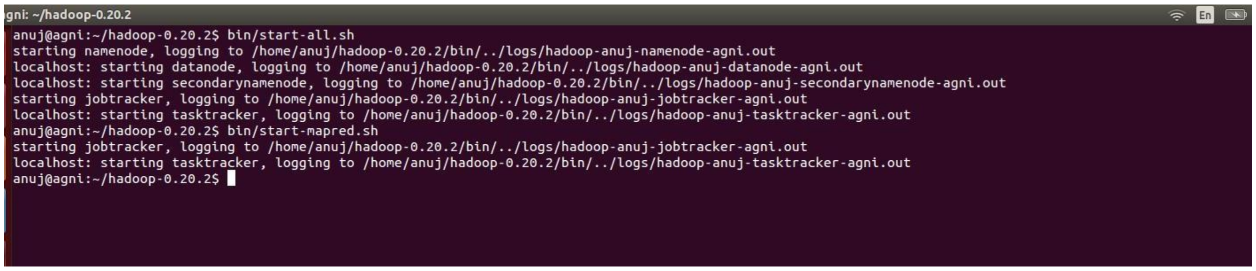
Figure 1: Formatting HDFS

5. The name node, data node, job tracker and a task tracker were then started using the command:

$bin/start-all.sh

MapReduce was also started using the command:

$ bin/start-mapred.sh



```
anuj@agni:~/hadoop-0.20.2$ bin/start-all.sh
starting namenode, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-namenode-agni.out
localhost: starting datanode, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-datanode-agni.out
localhost: starting secondarynamenode, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-secondarynamenode-agni.out
starting jobtracker, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-jobtracker-agni.out
localhost: starting tasktracker, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-tasktracker-agni.out
anuj@agni:~/hadoop-0.20.2$ bin/start-mapred.sh
starting jobtracker, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-jobtracker-agni.out
localhost: starting tasktracker, logging to /home/anuj/hadoop-0.20.2/bin/../logs/hadoop-anuj-tasktracker-agni.out
anuj@agni:~/hadoop-0.20.2$
```

Figure 2: Hadoop Start

6. The same job as described in step (1) of 'Running Using The Local File System' was executed using the same command to get the sorted IP hit output.

3.6 Output

The following files (provided with this report in Lab2Output.tar.gz) show the successful configuration of Hadoop and the output generated by running ApacheIPHitCnt in Hadoop on analyzing the sample 'access_log':
HadoopAdmin.html: Localhost Hadoop Map/Reduce administration site.
HadoopNameNode.html: The NameNode administration site.
Job1_HDFS_RunStatus.html: Run status of MR Job 1.
Job2_HDFS_RunStatus.html: Run status of MR Job 2.
HadoopLocalRun.txt: Execution output generated by Hadoop when running ApacheIPHitCnt using the local file system.
HadoopHDFSRun.txt: Execution output generated when using the local file system.
ipCnt-part-00000: IPs in access_log with hit count > 100.
ipCntSorted-part-00000: The IPs sorted in descending order.
Additional testing was also done using the Apache Log at:
http://ita.ee.lbl.gov/html/contrib/Sask-HTTP.html
4. Limitations
The ApacheIPHitCnt program does not recognize Apache log entries in which the client is identified by its hostname rather than IP address.