

DePaul University

CSC555 - Mining Big Data

Course Project by Bill Qualls
Dr. Alexander Rasin, Instructor
November 2013

Outline

- Objectives
- About the Data
- Loading the Data to HDFS
- The Map Reduce Program
- My Experiment: The Plan
- My Experiment: The Results
- Things I Do Not Want to Forget

OBJECTIVES

Objectives

- To gain experience in setting up a multi-node Hadoop cluster.
- To gain experience in loading data to HDFS.
- To gain experience in programming within the Map/Reduce paradigm.
- To witness first hand the benefits of distributed processing; in particular, in reducing runtime.

ABOUT THE DATA

About the Data

- Sensor data from 52 different railroad cars
- 509,013 data files with a total of 28 million rows.
- Lots of bad data, 1.8% of rows dropped.
- Each file has (usually) 60 rows of data
- Each row has (usually) 60 variables
- Data was made available as a single zip file
 - Zip file contained other zipped and unzipped files
 - Zip file was just over 1GB
 - Used *wget* command to download to master node

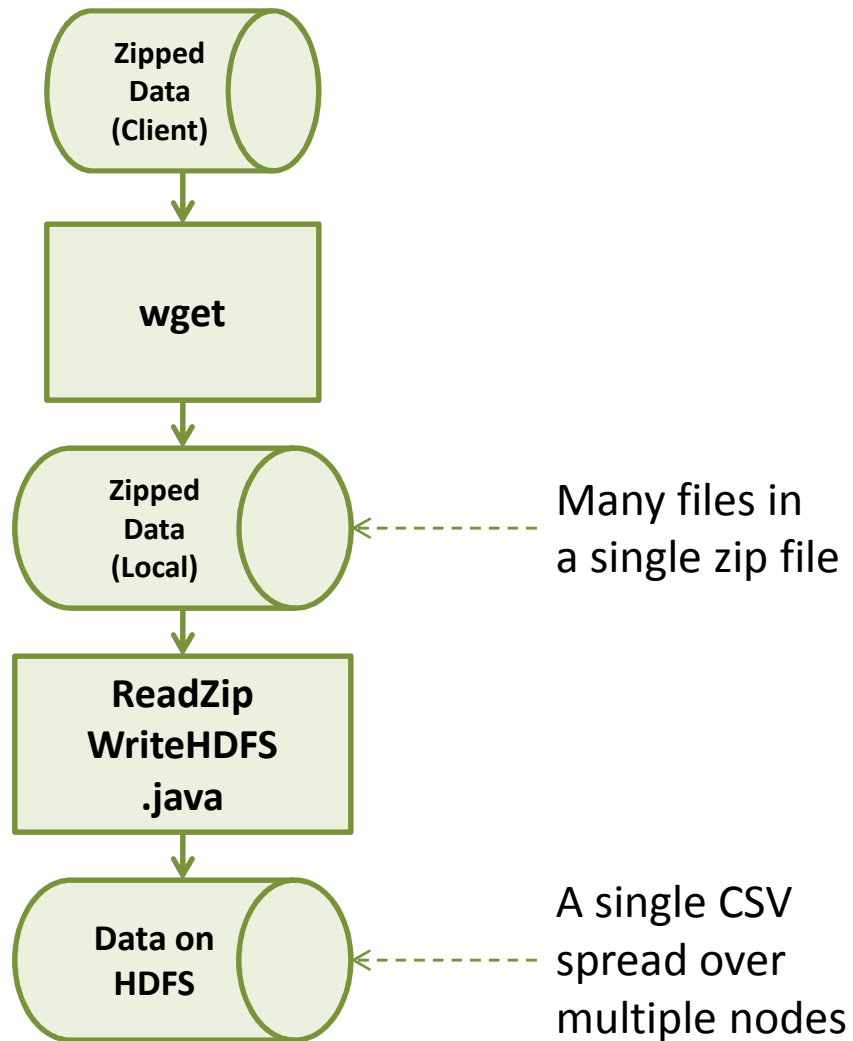
LOADING THE DATA TO HDFS

Loading the Data to HDFS

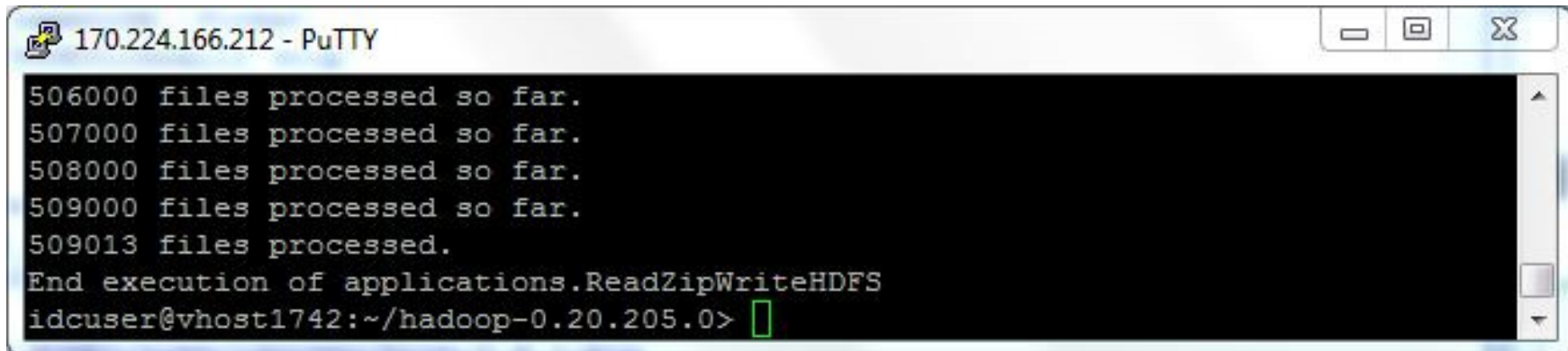
- I wrote program ReadZipWriteHDFS.java
- How to write to HDFS as an output stream:

```
// see Hadoop in Action page 44
Configuration conf = new Configuration();
FileSystem hdfs = FileSystem.get(conf);
Path hdfsFile = new Path(HDFS_FILE_OUT);
FSDataOutputStream outputStream =
    hdfs.create(hdfsFile);
```

Loading the Data to HDFS



Loading the Data to HDFS



A screenshot of a PuTTY terminal window titled "170.224.166.212 - PuTTY". The terminal displays the following output:

```
506000 files processed so far.  
507000 files processed so far.  
508000 files processed so far.  
509000 files processed so far.  
509013 files processed.  
End execution of applications.ReadZipWriteHDFS  
idcuser@vhost1742:~/hadoop-0.20.205.0> █
```

THE MAP REDUCE PROGRAM

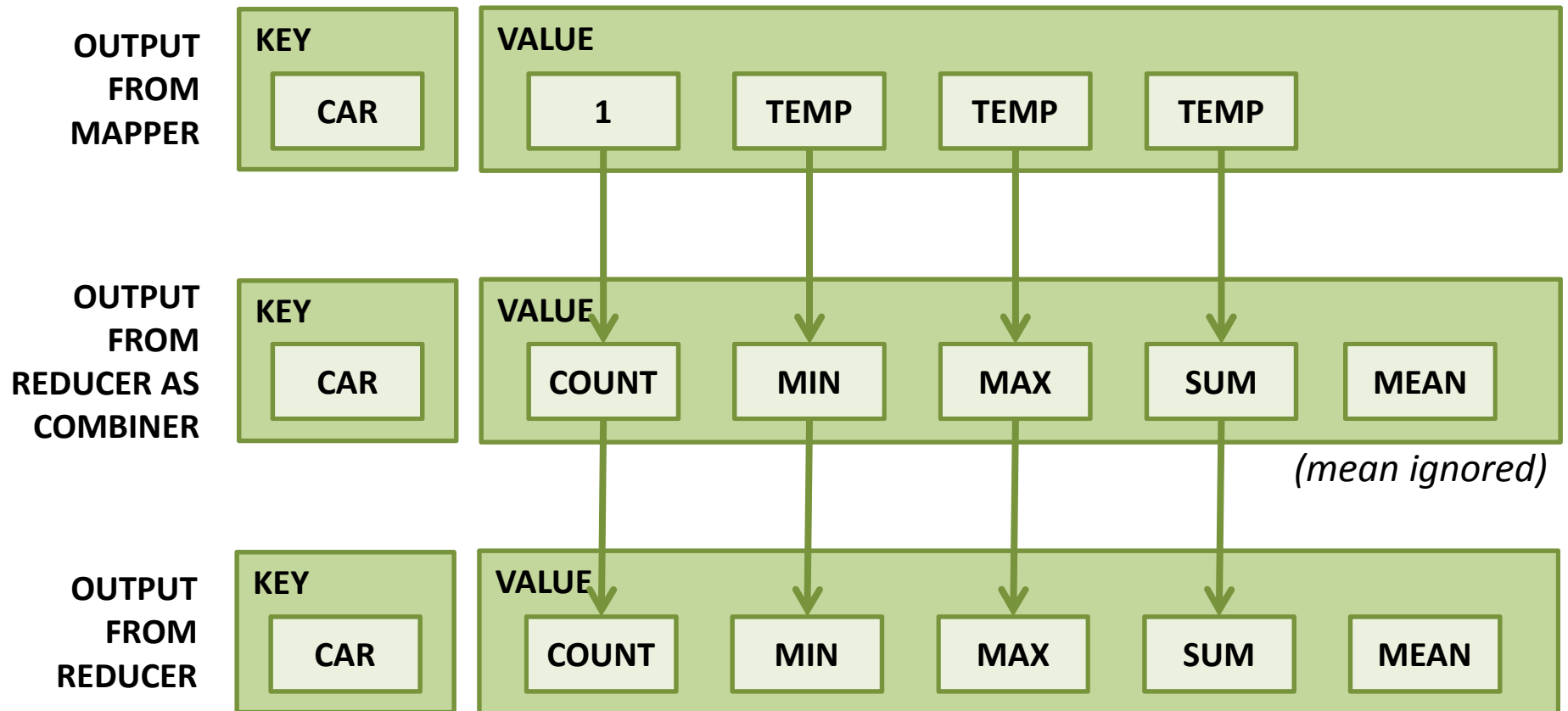
The Map Reduce Program

- I wrote program SensorMapReduce.java
- The program would determine, by rail car, the count, min, max, sum, and mean of one of the temperature values contained in each row of data.
- My program had a mapper, reducer, and combiner.
- A runtime parameter – Y or N – would indicate if the reducer was to be used as a combiner also.
- This facilitated running the program with and without the combiner to evaluate the combiner's effect on runtime.

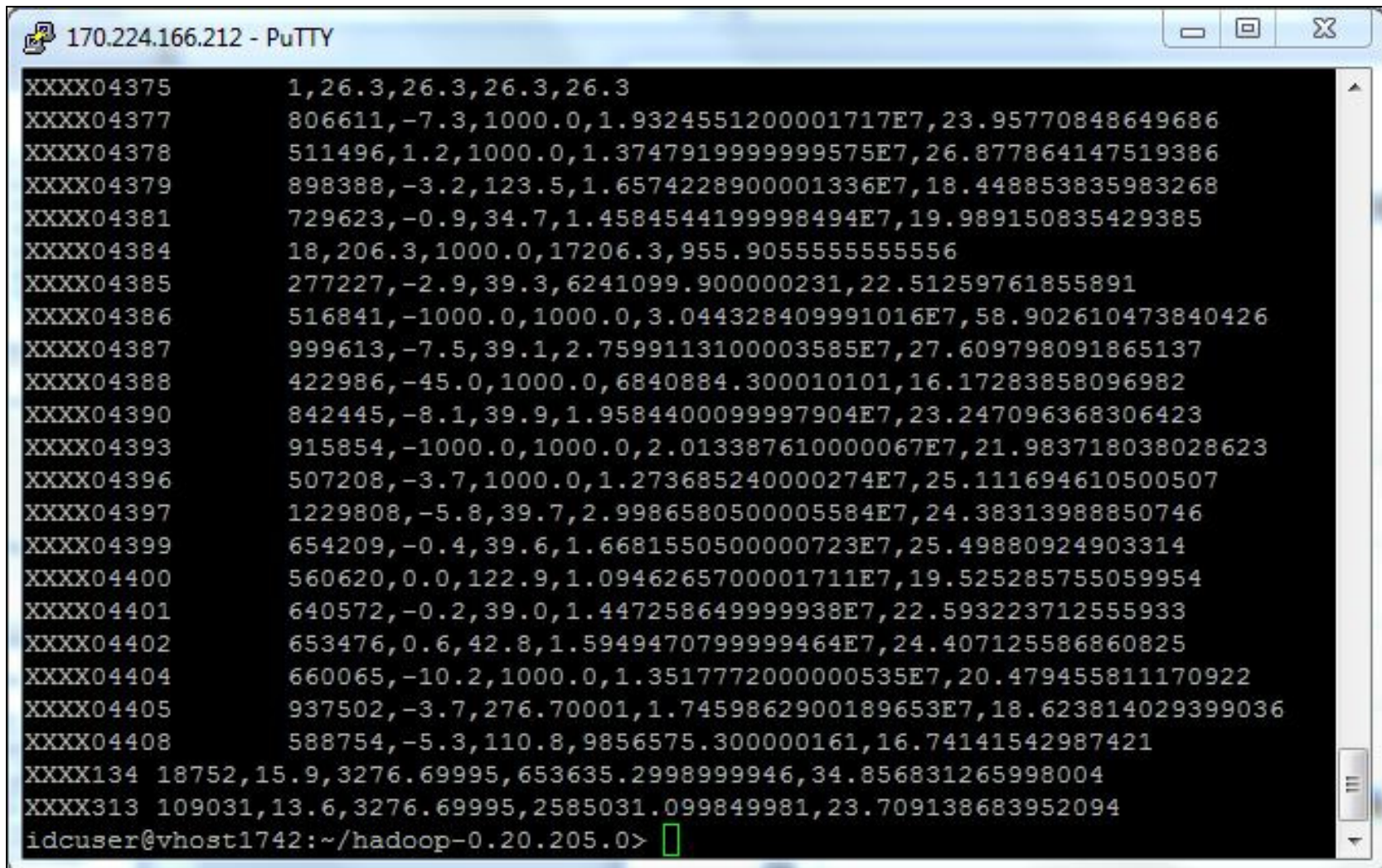
The Map Reduce Program

```
conf.setMapperClass(Map.class);  
  
if ("Y".equals(args[2]))  
{  
    conf.setCombinerClass(Reduce.class);  
}  
  
conf.setReducerClass(Reduce.class);
```

The Map Reduce Program



Program Output (Partial)



```
170.224.166.212 - PuTTY
XXXX04375      1,26.3,26.3,26.3,26.3
XXXX04377      806611,-7.3,1000.0,1.9324551200001717E7,23.95770848649686
XXXX04378      511496,1.2,1000.0,1.3747919999999575E7,26.877864147519386
XXXX04379      898388,-3.2,123.5,1.6574228900001336E7,18.448853835983268
XXXX04381      729623,-0.9,34.7,1.4584544199998494E7,19.989150835429385
XXXX04384      18,206.3,1000.0,17206.3,955.9055555555556
XXXX04385      277227,-2.9,39.3,6241099.900000231,22.51259761855891
XXXX04386      516841,-1000.0,1000.0,3.044328409991016E7,58.902610473840426
XXXX04387      999613,-7.5,39.1,2.7599113100003585E7,27.609798091865137
XXXX04388      422986,-45.0,1000.0,6840884.300010101,16.17283858096982
XXXX04390      842445,-8.1,39.9,1.958440099997904E7,23.247096368306423
XXXX04393      915854,-1000.0,1000.0,2.013387610000067E7,21.983718038028623
XXXX04396      507208,-3.7,1000.0,1.273685240000274E7,25.111694610500507
XXXX04397      1229808,-5.8,39.7,2.9986580500005584E7,24.38313988850746
XXXX04399      654209,-0.4,39.6,1.6681550500000723E7,25.49880924903314
XXXX04400      560620,0.0,122.9,1.0946265700001711E7,19.525285755059954
XXXX04401      640572,-0.2,39.0,1.447258649999938E7,22.593223712555933
XXXX04402      653476,0.6,42.8,1.5949470799999464E7,24.407125586860825
XXXX04404      660065,-10.2,1000.0,1.3517772000000535E7,20.479455811170922
XXXX04405      937502,-3.7,276.70001,1.7459862900189653E7,18.623814029399036
XXXX04408      588754,-5.3,110.8,9856575.300000161,16.74141542987421
XXXX134 18752,15.9,3276.69995,653635.2998999946,34.856831265998004
XXXX313 109031,13.6,3276.69995,2585031.099849981,23.709138683952094
idcuser@vhost1742:~/hadoop-0.20.205.0>
```


MY EXPERIMENT: THE PLAN

My Experiment: The Plan

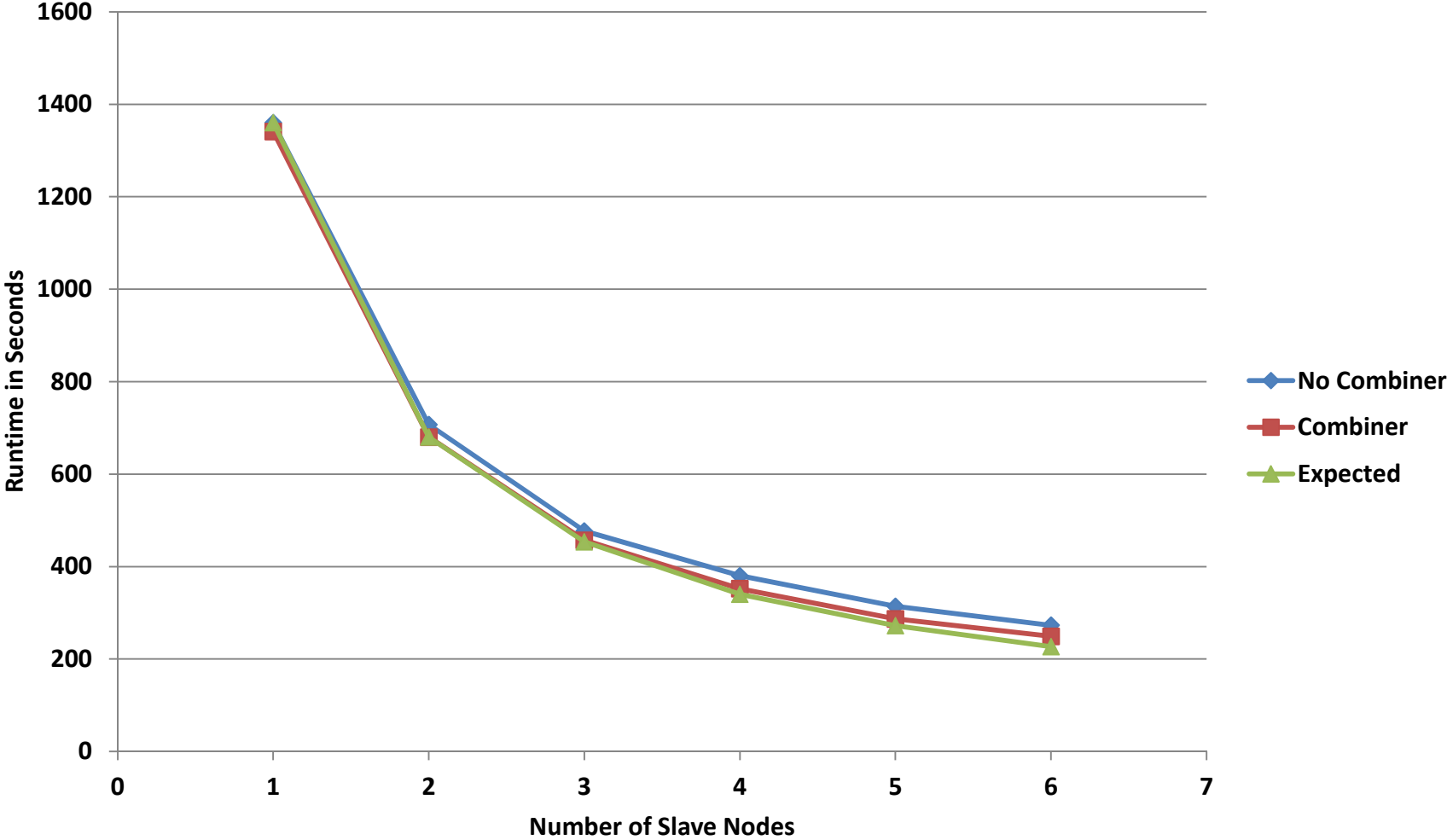
- Objectives
 - To confirm that reduction in runtime is approximately linear as more nodes are introduced.
 - To evaluate the effect of a combiner on runtime.
- The Plan
 - Run my program a total of 24 times
 - Using 1, 2, 3, 4, 5, and 6 slave nodes
 - Each time running twice with combiner and twice without
 - Reload HDFS each time a node is added / dropped
 - Record runtime in seconds

MY EXPERIMENT: THE RESULTS

Run Time Data

Slaves	without Combiner		with Combiner	
1	1370	1349	1344	1338
2	715	699	678	682
3	470	480	458	456
4	382	379	352	353
5	313	315	286	288
6	275	272	251	248

Run Time vs. Number of Slave Nodes



Observations

- The green line shows the expected run time, determined as follows: If I treat the run time for one data node (1360 seconds) as a baseline, then the expected run time for two nodes would be half that, for three nodes one third, etc. So in general, the expected run time for n nodes would be $1360/n$. As the curve shows, it's pretty close!
- I was surprised that the combiner had such a small impact on run time. Perhaps this is because the relatively small number of keys (rail car numbers).

(Feel free to skip this part!)

THINGS I DON'T WANT TO FORGET

Commands I Used

- Verify that cluster is running:
`http://vhost1742.site1.compute.ihost.com:50070/dfshealth.jsp`
...or...
`/usr/java/default/bin/jps`
- If first time...create directory in HDFS:
`bin/hadoop fs -mkdir /data`
- Forcefully leave safemode (sometimes after start-all.sh):
`bin/hadoop dfsadmin -safemode leave`

Adding / Removing Slave Nodes

1. Remove datafile(s) from HDFS
2. bin/stop-all.sh
3. Edit conf/slaves. Add or remove hash signs:

```
vhost1812  
vhost1722  
vhost1678  
vhost1817  
#vhost1521  
#vhost1742
```

4. bin/start-all.sh
5. Reload datafile(s) to HDFS
6. Run test

Commands I Used

- Remove old data from HDFS:

```
bin/hadoop fs -rmr /data/hugefile.txt
```

- Run program to copy nested zip files to HDFS:

```
bin/hadoop jar csc555.jar applications.ReadZipWriteHDFS  
/home/idcuser/mydata/xxxxxx.zip /data/hugefile.txt
```

- Remove old results file from HDFS:

```
bin/hadoop fs -rmr /data/output_mine
```

Commands I Used

- Execute MapReduce program:
`time bin/hadoop jar csc555.jar applications.SensorMapReduce /data/hugefile.txt /data/output_mine Y`
- View results:
`bin/hadoop fs -cat /data/output_mine/part-00000`
- Copy results from HDFS to local file system:
`bin/hadoop fs -get /data/output_mine/part-00000 results.txt`
- View results on local file system:
`cat results.txt`

Commands I Used

- Consider adding these to ~/.bashrc to avoid having to set them manually...

```
export JAVA_HOME=/usr/java/default
```

```
export HADOOP_HOME=/home/idcuser/hadoop-0.20.205.0/
```

```
export HIVE_HOME=/home/idcuser/hive-0.8.1-bin
```

```
export PATH=$HIVE_HOME/bin:$PATH
```

"No datanode to stop"

1. bin/stop-all.sh
2. ssh vhost????
3. `rm -rf /tmp/hadoop-idcuser/dfs/`
4. `cd hadoop-0.20.205.0`
5. `bin/hadoop namenode -format`
6. `exit`
7. (repeat 2-6 as necessary for other slave nodes)
8. bin/start-all.sh

Instructions
executed on
slave nodes
from master

This cost me many hours!

THANK YOU!