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

About the Data

```

XXXX4300,1,11/10/13,19:07:04,42.87409,105.24813,1603.59998,22.30473,1241,18.75641,13230
6,7.8,12,9.2,15.3,-2.4,1.5,-0.3,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,4,2,32.17,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,74151,0,0,0,0,0,NULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNUL
XXXX4300,1,11/10/13,19:08:04,42.8698,105.24569,1603.59998,22.54334,1241,20.02218,132306
,7.7,12.1,9.2,15.3,-2.4,1.4,-0.3,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,4,2,32.17,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,74151,0,0,0,0,0,NULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
```

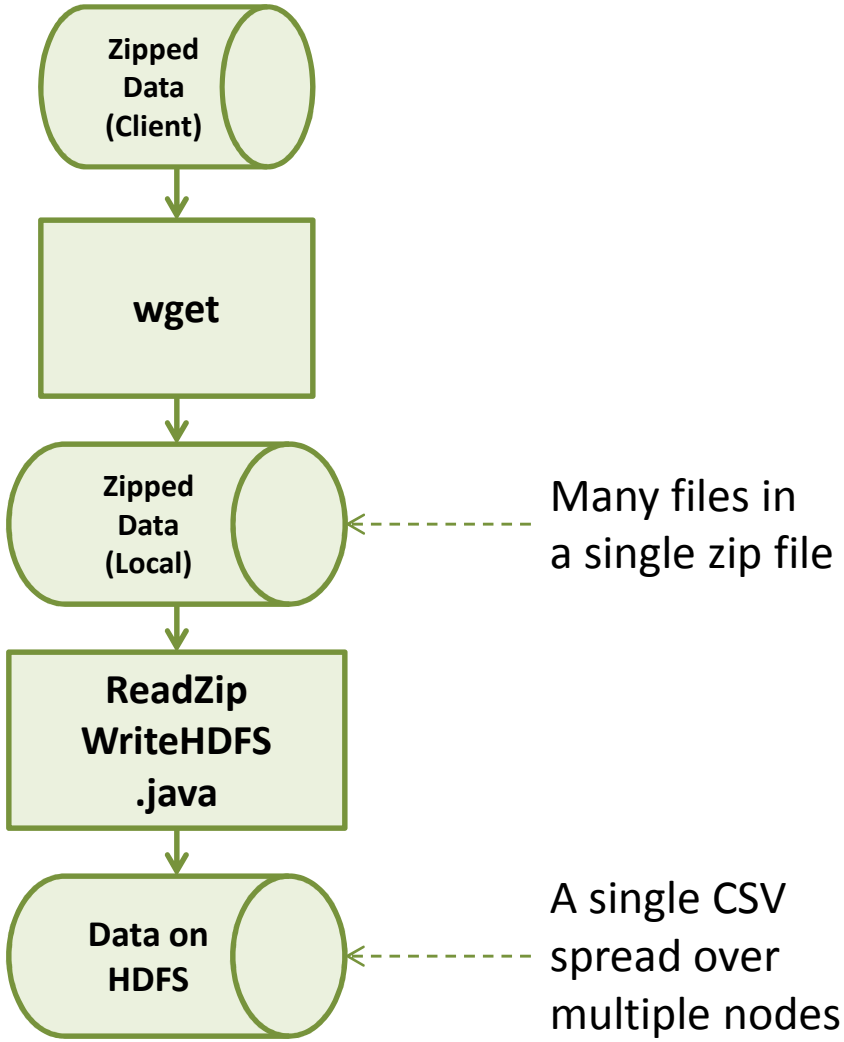
LOADING THE DATA TO HDFS

Loading the Data to HDFS

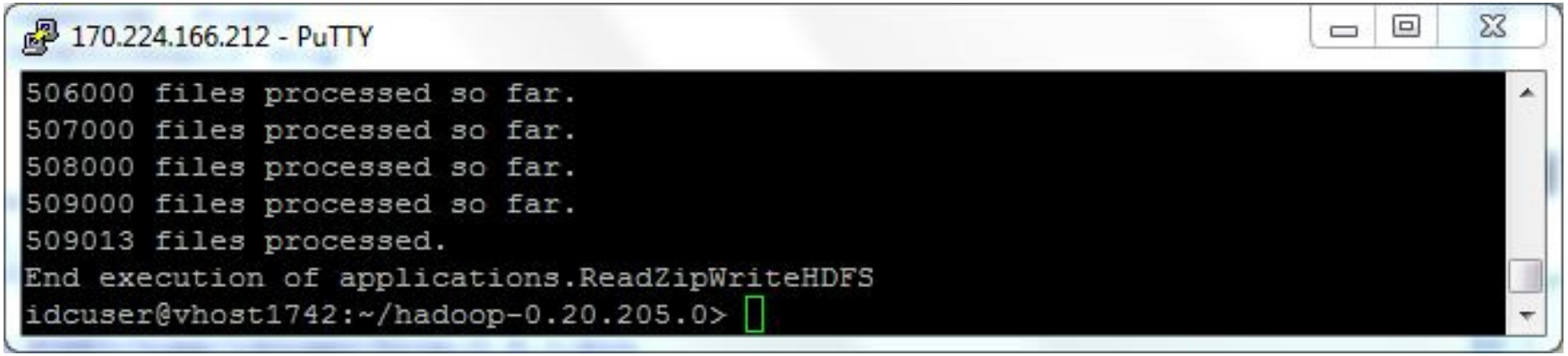
- I wrote program ReadZipWriteHDFS.java
- How to write to HDFS as an output stream:

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

Loading the Data to HDFS



Loading the Data to HDFS



THE MAP REDUCE PROGRAM

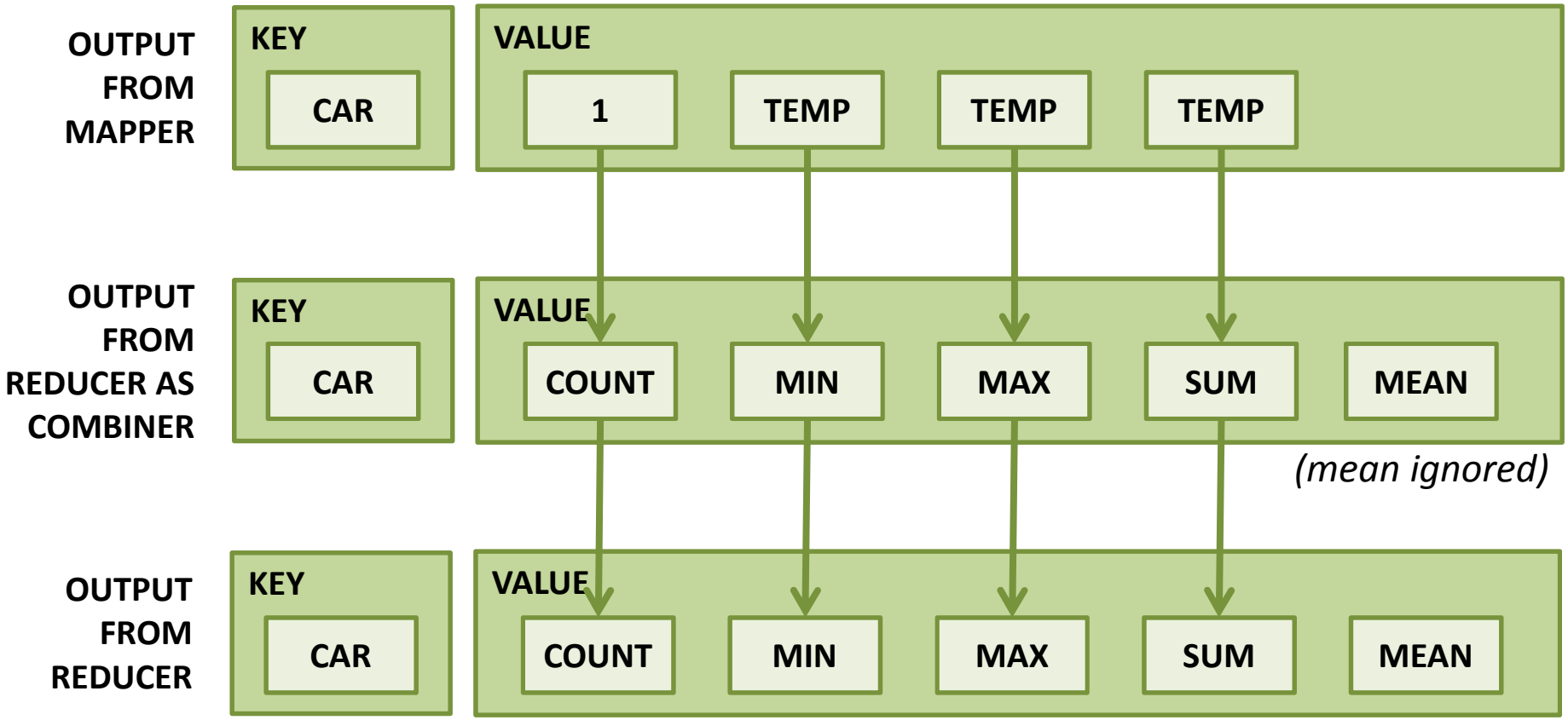
The Map Reduce Program

- 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.

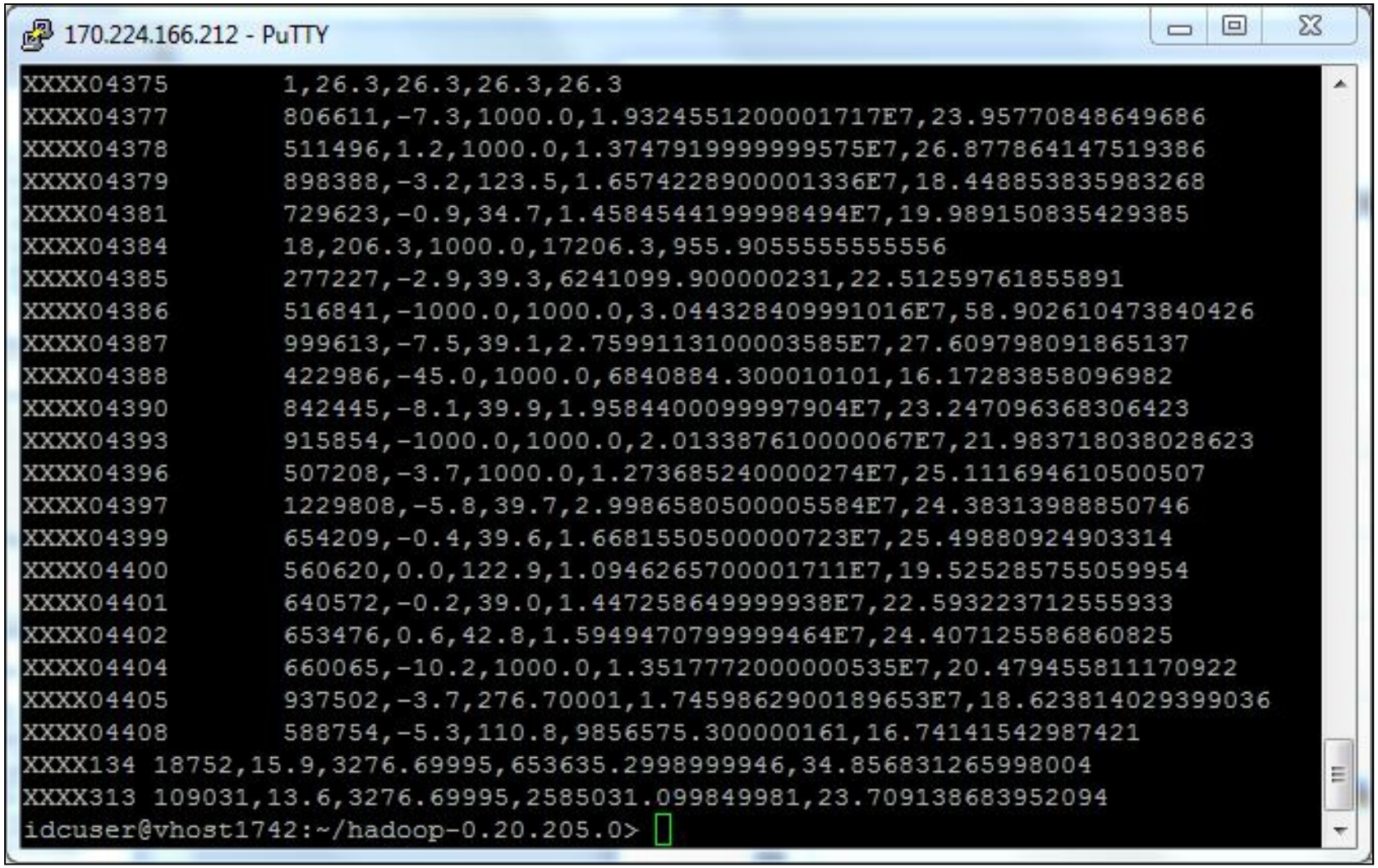
The Map Reduce Program

```
conf.setMapperClass(Map.class);

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

conf.setReducerClass(Reduce.class);
```

The Map Reduce Program



Program Output (Partial)



MY EXPERIMENT: THE PLAN

My Experiment: The Plan

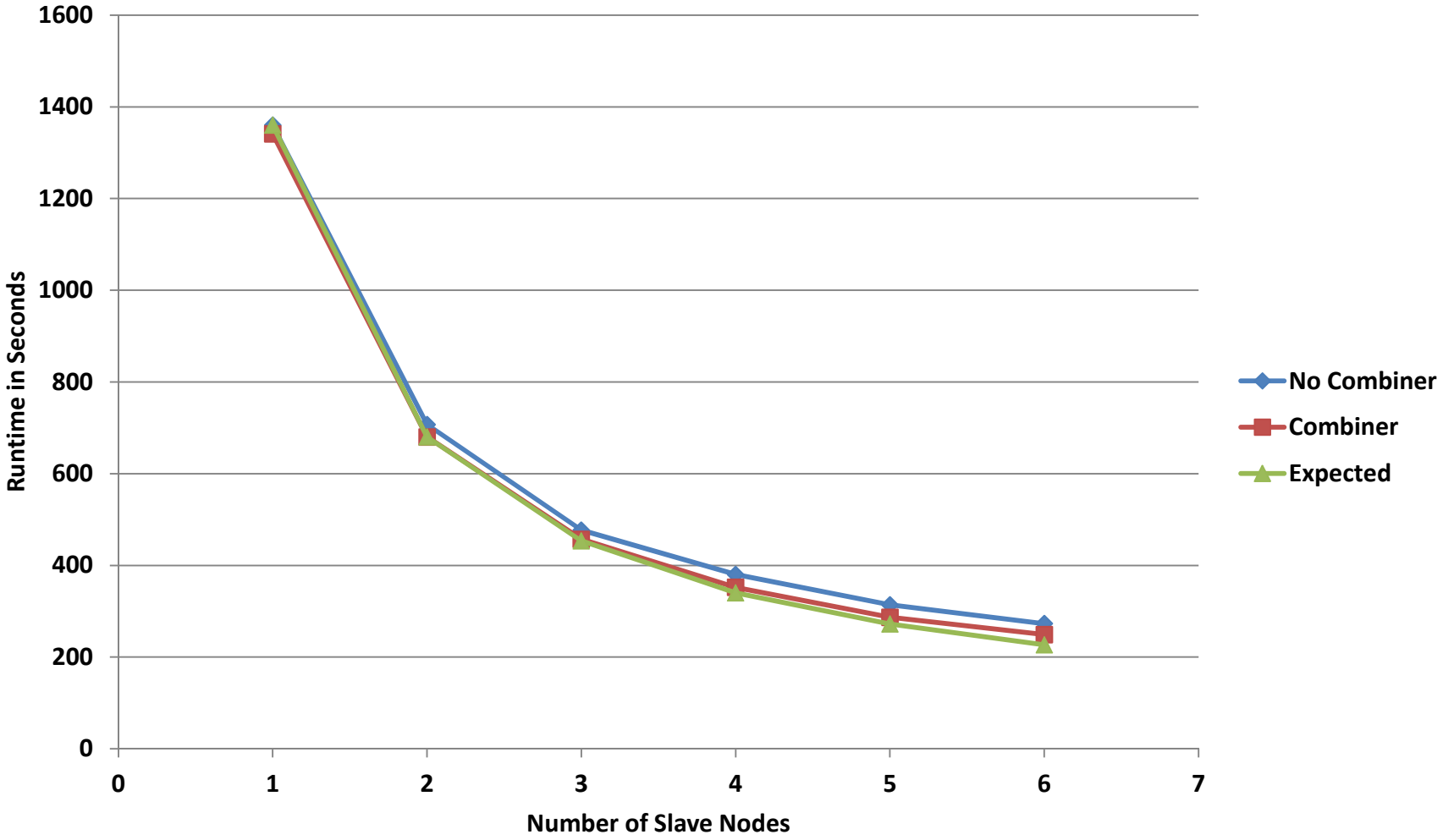
- Objectives
  - To confirm that reduction in runtime is approximately linear as more nodes are introduced.
  - To evaluate the effect of a combiner on runtime.
- The Plan
  - Run my program a total of 24 times
  - Using 1, 2, 3, 4, 5, and 6 slave nodes
  - Each time running twice with combiner and twice 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 exectuted on slave nodes from master

This cost me many hours!

THANK YOU!