Worcester Polytechnic Institute
CS 585/DS 503 Big Data Management
Project 2: Advanced big data operations and analytics

Group 9: Xiaoting Cui, Allison Rozet, Abhinav Singh

October 6, 2017

Problem 1: Spatial Join

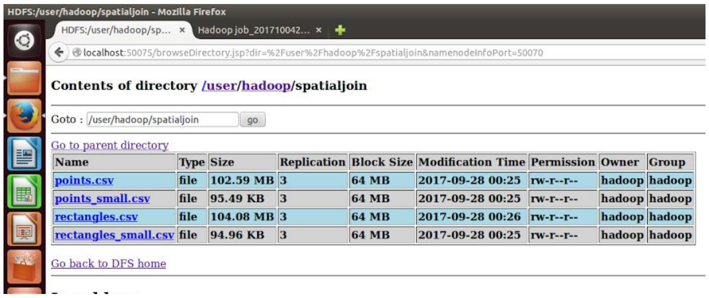
We solved this problem three ways based on three different assumptions: (1) the rectangles data set was small and the points data set was large, (2) points was small and rectangles was large, and (3) both points and rectangles were large.

Step 1 (Create the Two Datasets)

First, we do not assume that each point must be unique. This is because in the real, 3D world two points (or rectangles) might have the same x- and y-coordinates, but say, be on different floors of a building. We assume that points and edges can lie on the edge of the space, i.e. at 1 or 10000. Rectangles are generated so that both corners are within the space and are at most 20x20 in size. Finally, we generated integer coordinates. We discuss at the end of this section how to relax this final assumption.

To execute, run datacreation.jar group9.Main1. Use arguments such as: `-P points.csv -R rectangles.csv -np 11000000 -nr 4000000`

After creating the data sets, we loaded them into Hadoop as in Project 1. Our large points file has 11 million points, our large rectangles file has 4 million rectangles, our small points file has 10 thousand points, and our small rectangles file has 4 thousand rectangles. The screenshot below verifies that our large files exceed 100MB.



localhost:50075/browseDirectory.jsp?dir=%2Fuser%2Fhadoop%2Fspatialjoin&namenodeInfoPort=50070

Contents of directory /user/hadoop/spatialjoin

Goto : /user/hadoop/spatialjoin

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
points.csv	file	102.59 MB	3	64 MB	2017-09-28 00:25	rw-r--r--	hadoop	hadoop
points_small.csv	file	95.49 KB	3	64 MB	2017-09-28 00:25	rw-r--r--	hadoop	hadoop
rectangles.csv	file	104.08 MB	3	64 MB	2017-09-28 00:26	rw-r--r--	hadoop	hadoop
rectangles_small.csv	file	94.96 KB	3	64 MB	2017-09-28 00:25	rw-r--r--	hadoop	hadoop

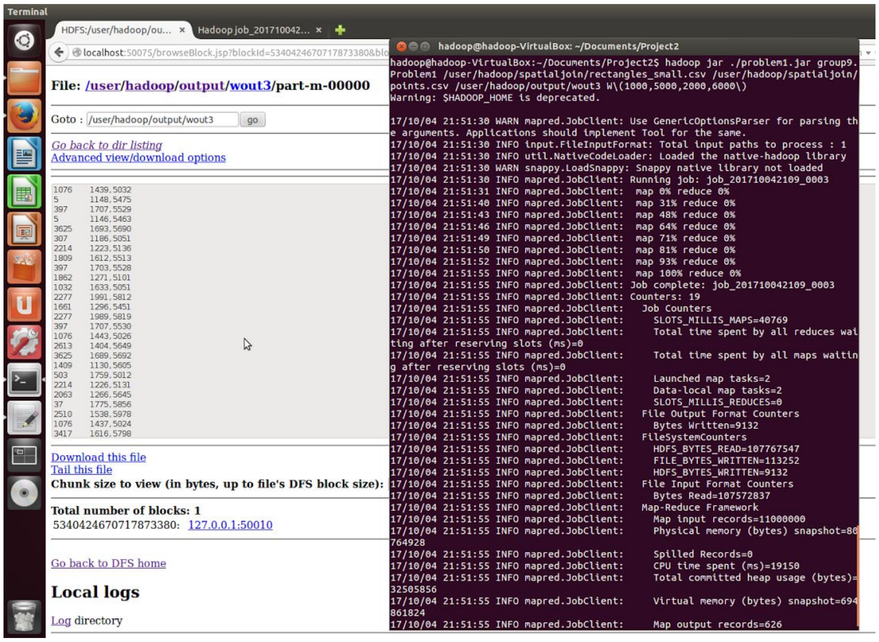
Go back to DFS home

Step 2 (MapReduce Job for Spatial Containment Join)

Small rectangles file; large points file. This is a map-only job and essentially a broadcast-join. We cache the small rectangles file so that we can broadcast it to every mapper, along with the optional window. In the setup() method, we retrieve the window corner coordinates and read in the rectangles file. In the map() method, if a point falls within or on the window, we iterate through each rectangle and write out the (rectangle id, point) if the point falls within or on the rectangle.

Please use code similar to the following to execute the jar file.

```
hadoop jar ./problem1.jar group9.Problem1 <small rectangles input>
<large points input> <output> <optional W\(xbl,ybl,xtr,ytr\)>
```



Terminal
HDFS:/user/hadoop/ou... ×
Hadoop job_201710042... ×
localhost:50075/browseBlock.jsp?blockId=5340424670717873380&blo

File: /user/hadoop/output/wout3/part-m-00000

Goto : /user/hadoop/output/wout3 go

Go back to dir listing
Advanced view/download options

```
1076    1439,5032
5       1148,5475
397     1707,5529
5       1146,5463
3625    1693,5690
307     1186,5051
2214    1223,5136
1809    1612,5513
397     1703,5528
1862    1271,5101
1032    1633,5051
2277    1991,5812
1661    1296,5451
2277    1989,5819
397     1707,5530
1076    1443,5026
2613    1404,5649
3625    1689,5692
1409    1130,5605
503     1759,5012
2214    1226,5131
2063    1266,5645
37      1775,5856
2510    1538,5978
1076    1437,5024
3417    1616,5798
```

Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block size):

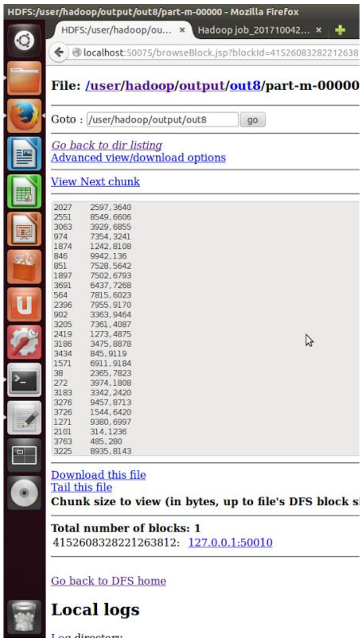
Total number of blocks: 1
5340424670717873380: 127.0.0.1:50010

Go back to DFS home

Local logs

Log directory

```
hadoop@hadoop-VirtualBox:~/Documents/Project2$ hadoop jar ./problem1.jar group9.
Problem1 /user/hadoop/spatialjoin/rectangles_small.csv /user/hadoop/spatialjoin/
points.csv /user/hadoop/output/wout3 W\(1000,5000,2000,6000\)
Warning: $HADOOP_HOME is deprecated.

17/10/04 21:51:30 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
17/10/04 21:51:30 INFO input.FileInputFormat: Total input paths to process : 1
17/10/04 21:51:30 INFO util.NativeCodeLoader: Loaded the native-hadoop library
17/10/04 21:51:30 WARN snappy.LoadSnappy: Snappy native library not loaded
17/10/04 21:51:30 INFO mapred.JobClient: Running job: job_201710042109_0003
17/10/04 21:51:31 INFO mapred.JobClient:  map 0% reduce 0%
17/10/04 21:51:40 INFO mapred.JobClient:  map 31% reduce 0%
17/10/04 21:51:43 INFO mapred.JobClient:  map 48% reduce 0%
17/10/04 21:51:46 INFO mapred.JobClient:  map 64% reduce 0%
17/10/04 21:51:49 INFO mapred.JobClient:  map 71% reduce 0%
17/10/04 21:51:50 INFO mapred.JobClient:  map 81% reduce 0%
17/10/04 21:51:52 INFO mapred.JobClient:  map 93% reduce 0%
17/10/04 21:51:55 INFO mapred.JobClient:  map 100% reduce 0%
17/10/04 21:51:55 INFO mapred.JobClient: Job complete: job_201710042109_0003
17/10/04 21:51:55 INFO mapred.JobClient: Counters: 19
17/10/04 21:51:55 INFO mapred.JobClient:   Job Counters
17/10/04 21:51:55 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=40769
17/10/04 21:51:55 INFO mapred.JobClient:     Total time spent by all reduces wai
ting after reserving slots (ms)=0
17/10/04 21:51:55 INFO mapred.JobClient:     Total time spent by all maps waitin
g after reserving slots (ms)=0
17/10/04 21:51:55 INFO mapred.JobClient:     Launched map tasks=2
17/10/04 21:51:55 INFO mapred.JobClient:     Data-local map tasks=2
17/10/04 21:51:55 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=0
17/10/04 21:51:55 INFO mapred.JobClient:   File Output Format Counters
17/10/04 21:51:55 INFO mapred.JobClient:     Bytes Written=9132
17/10/04 21:51:55 INFO mapred.JobClient:   FileSystemCounters
17/10/04 21:51:55 INFO mapred.JobClient:     HDFS_BYTES_READ=107767547
17/10/04 21:51:55 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=113252
17/10/04 21:51:55 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=9132
17/10/04 21:51:55 INFO mapred.JobClient:   File Input Format Counters
17/10/04 21:51:55 INFO mapred.JobClient:     Bytes Read=107572837
17/10/04 21:51:55 INFO mapred.JobClient:   Map-Reduce Framework
17/10/04 21:51:55 INFO mapred.JobClient:     Map input records=11000000
17/10/04 21:51:55 INFO mapred.JobClient:     Physical memory (bytes) snapshot=80
764928
17/10/04 21:51:55 INFO mapred.JobClient:     Spilled Records=0
17/10/04 21:51:55 INFO mapred.JobClient:     CPU time spent (ms)=19150
17/10/04 21:51:55 INFO mapred.JobClient:     Total committed heap usage (bytes)=
32505856
17/10/04 21:51:55 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=694
861824
17/10/04 21:51:55 INFO mapred.JobClient:     Map output records=626
```



HDFS:/user/hadoop/output/out8/part-m-00000 - Mozilla Firefox
HDFS:/user/hadoop/ou... ×
Hadoop job_201710042... ×
localhost:50075/browseBlock.jsp?blockId=41526083282212638

File: /user/hadoop/output/out8/part-m-00000

Goto : /user/hadoop/output/out8 go

Go back to dir listing
Advanced view/download options

View Next chunk

```
2027    2597,3640
2551    8549,6606
3063    3929,6855
974     7354,3241
1874    1242,8108
846     9942,136
851     7528,5642
1897    7502,6793
3691    6437,7268
564     7815,6023
2396    7955,9170
902     3363,9464
3205    7361,4087
2419    1273,4875
3186    3475,8878
3434    845,9119
1571    6911,9184
38      2365,7823
272     3974,1808
3183    3342,2420
3276    9457,8713
3726    1544,6420
1271    9380,6997
2101    314,1236
3763    485,280
3225    8935,8143
```

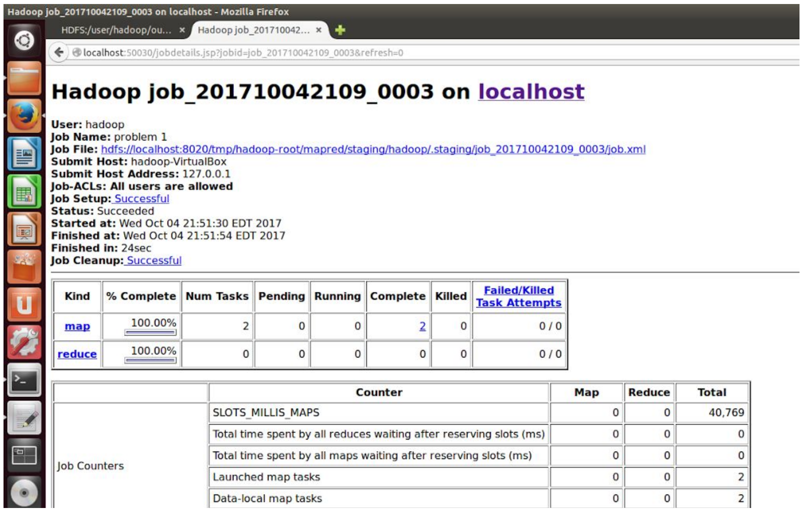
Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block si

Total number of blocks: 1
4152608328221263812: 127.0.0.1:50010

Go back to DFS home

Local logs

The screenshots above are sample outputs with a window (left) and without a window (right).



Hadoop job_201710042109_0003 on localhost - Mozilla Firefox
HDFS:/user/hadoop/ou... ×
Hadoop job_201710042... ×
localhost:50030/jobdetails.jsp?jobid=job_201710042109_0003&refresh=0

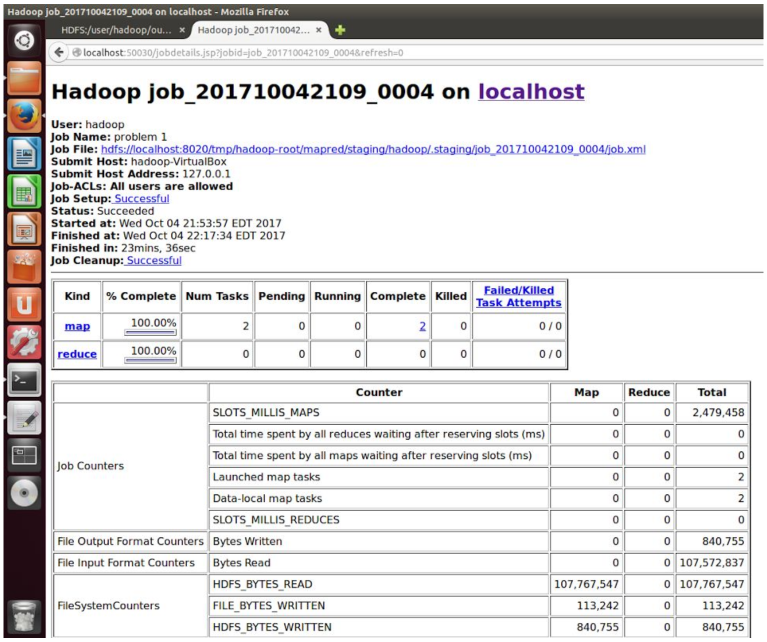
Hadoop job_201710042109_0003 on localhost

User: hadoop
Job Name: problem 1
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710042109_0003/job.xml
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 04 21:51:30 EDT 2017
Finished at: Wed Oct 04 21:51:54 EDT 2017
Finished in: 24sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	0	0	0	0	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	40,769
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	2
	Data-local map tasks	0	0	2

The screenshot on the left verifies that our program with a window was successful. The screenshot on the next page verifies it without a window.



Hadoop job_201710042109_0004 on localhost - Mozilla Firefox
HDFS:/user/hadoop/ou...
Hadoop job_201710042...
localhost:50030/jobdetails.jsp?jobid=job_201710042109_0004&refresh=0

Hadoop job_201710042109_0004 on localhost

User: hadoop
Job Name: problem 1
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710042109_0004/job.xml
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 04 21:53:57 EDT 2017
Finished at: Wed Oct 04 22:17:34 EDT 2017
Finished in: 23mins, 36sec
Job Cleanup: Successful

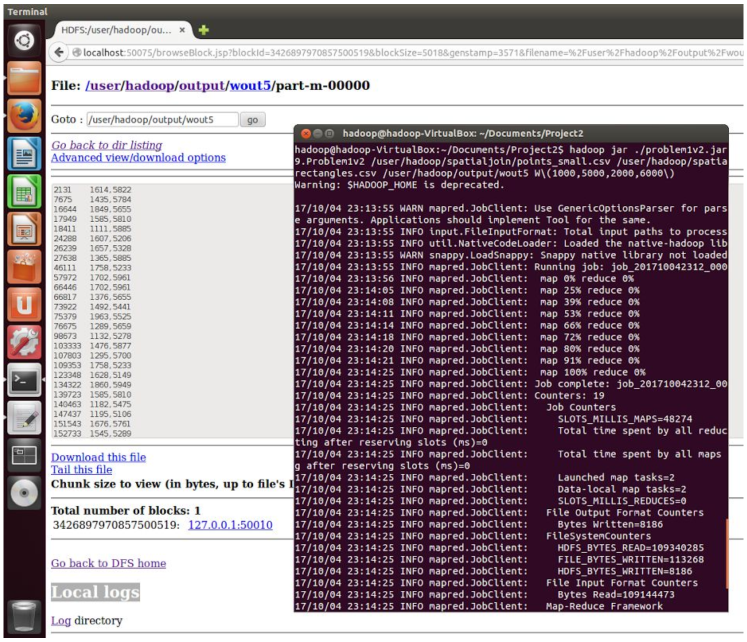
Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	0	0	0	0	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	SLOTS_MILLIS_MAPS	0	0	2,479,458
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Launched map tasks	0	0	2
	Data-local map tasks	0	0	2
	SLOTS_MILLIS_REDUCES	0	0	0
File Output Format Counters	Bytes Written	0	0	840,755
File Input Format Counters	Bytes Read	0	0	107,572,837
FileSystemCounters	HDFS_BYTES_READ	107,767,547	0	107,767,547
	FILE_BYTES_WRITTEN	113,242	0	113,242
	HDFS_BYTES_WRITTEN	840,755	0	840,755

Small points file; large rectangles file. This is also a map-only job and analogous to the previous solution. The main difference is that we perform the check for the points that fall within the window in the setup() method rather than the map() method so that it only has to be done once rather than every time a new rectangle is read.

Please use code similar to the following to execute the jar file.

```
hadoop jar ./problem1v2.jar group9.Problem1v2 <small points input>
<large rectangles input> <output> <optional W\(xbl,ybl,xtr,ytr\)>
```



Terminal
HDFS:/user/hadoop/ou...
localhost:50075/browseBlock.jsp?blockId=3426897970857500519&blockSize=5018&genstamp=3571&filename=%2Fuser%2Fhadoop%2Foutput%2Fwou
go
Go back to dir listing
Advanced view/download options
Download this file
Tail this file
Go back to DFS home
Local logs
Log directory

File: /user/hadoop/output/wout5/part-m-00000

Goto : /user/hadoop/output/wout5

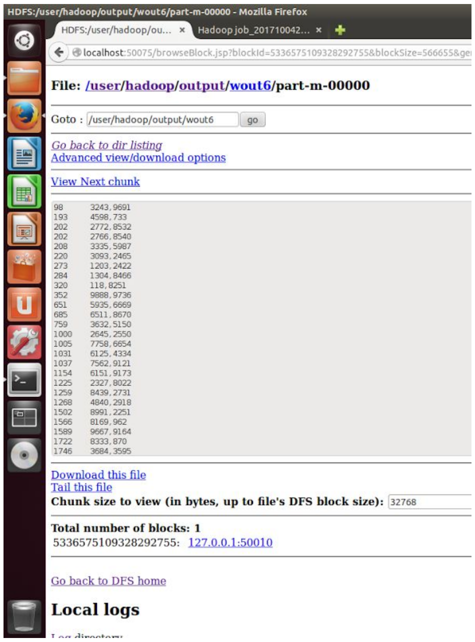
```
2131    1614,5822
7675    1435,5784
16644   1849,5655
17949   1585,5810
18411   1111,5885
24288   1607,5206
26239   1657,5328
27638   1365,5885
46111   1758,5233
57972   1702,5961
66446   1702,5961
66817   1376,5655
73922   1492,5441
75379   1963,5525
76675   1289,5659
98673   1132,5278
103333  1476,5877
107803  1295,5700
109353  1758,5233
123348  1628,5149
134322  1860,5949
139723  1585,5810
140463  1182,5475
147437  1195,5106
151543  1676,5761
152733  1545,5289
```

Chunk size to view (in bytes, up to file's I

Total number of blocks: 1
3426897970857500519: 127.0.0.1:50010

hadoop@hadoop-VirtualBox: ~/Documents/Project2

```
hadoop@hadoop-VirtualBox:~/Documents/Project2$ hadoop jar ./problem1v2.jar
9.Problem1v2 /user/hadoop/spatialjoin/points_small.csv /user/hadoop/spatia
rectangles.csv /user/hadoop/output/wout5 W\(1000,5000,2000,6000\)
Warning: $HADOOP_HOME is deprecated.

17/10/04 23:13:55 WARN mapred.JobClient: Use GenericOptionsParser for pars
e arguments. Applications should implement Tool for the same.
17/10/04 23:13:55 INFO input.FileInputFormat: Total input paths to process
17/10/04 23:13:55 INFO util.NativeCodeLoader: Loaded the native-hadoop lib
17/10/04 23:13:55 WARN snappy.LoadSnappy: Snappy native library not loaded
17/10/04 23:13:55 INFO mapred.JobClient: Running job: job_201710042312_000
17/10/04 23:13:56 INFO mapred.JobClient:  map 0% reduce 0%
17/10/04 23:14:05 INFO mapred.JobClient:  map 25% reduce 0%
17/10/04 23:14:08 INFO mapred.JobClient:  map 39% reduce 0%
17/10/04 23:14:11 INFO mapred.JobClient:  map 53% reduce 0%
17/10/04 23:14:14 INFO mapred.JobClient:  map 66% reduce 0%
17/10/04 23:14:18 INFO mapred.JobClient:  map 72% reduce 0%
17/10/04 23:14:20 INFO mapred.JobClient:  map 80% reduce 0%
17/10/04 23:14:21 INFO mapred.JobClient:  map 91% reduce 0%
17/10/04 23:14:25 INFO mapred.JobClient:  map 100% reduce 0%
17/10/04 23:14:25 INFO mapred.JobClient: Job complete: job_201710042312_00
17/10/04 23:14:25 INFO mapred.JobClient: Counters: 19
17/10/04 23:14:25 INFO mapred.JobClient:   Job Counters
17/10/04 23:14:25 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=48274
17/10/04 23:14:25 INFO mapred.JobClient:     Total time spent by all reduc
ting after reserving slots (ms)=0
17/10/04 23:14:25 INFO mapred.JobClient:     Total time spent by all maps
g after reserving slots (ms)=0
17/10/04 23:14:25 INFO mapred.JobClient:     Launched map tasks=2
17/10/04 23:14:25 INFO mapred.JobClient:     Data-local map tasks=2
17/10/04 23:14:25 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=0
17/10/04 23:14:25 INFO mapred.JobClient:   File Output Format Counters
17/10/04 23:14:25 INFO mapred.JobClient:     Bytes Written=8186
17/10/04 23:14:25 INFO mapred.JobClient:   FileSystemCounters
17/10/04 23:14:25 INFO mapred.JobClient:     HDFS_BYTES_READ=109340285
17/10/04 23:14:25 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=113268
17/10/04 23:14:25 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=8186
17/10/04 23:14:25 INFO mapred.JobClient:   File Input Format Counters
17/10/04 23:14:25 INFO mapred.JobClient:     Bytes Read=109144473
17/10/04 23:14:25 INFO mapred.JobClient:   Map-Reduce Framework
```



HDFS:/user/hadoop/output/wout6/part-m-00000 - Mozilla Firefox
HDFS:/user/hadoop/ou...
Hadoop job_201710042...
localhost:50075/browseBlock.jsp?blockId=5336575109328292755&blockSize=566655&ge
go
Go back to dir listing
Advanced view/download options
View Next chunk
Download this file
Tail this file
Go back to DFS home
Local logs

File: /user/hadoop/output/wout6/part-m-00000

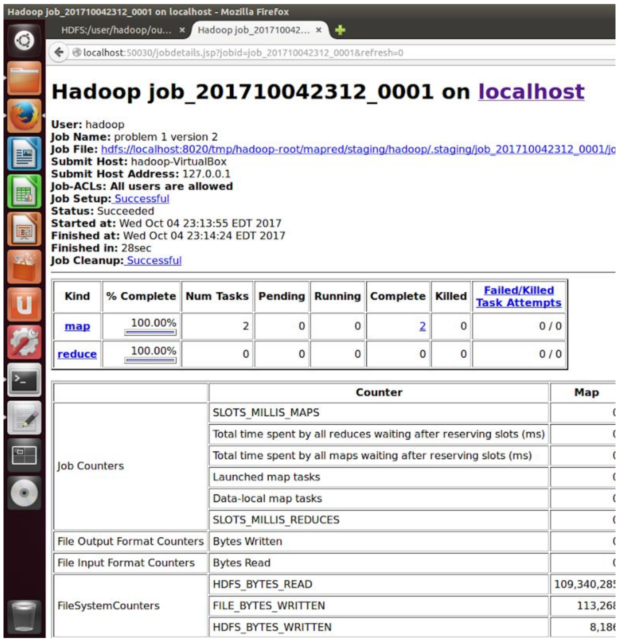
Goto : /user/hadoop/output/wout6

```
98      3243,9691
193     4598,733
202     2772,8532
202     2766,8540
208     3335,5987
220     3093,2465
273     1203,2422
284     1304,8466
320     118,8251
352     9888,9736
651     5935,6669
685     6511,8670
759     3632,5150
1000    2645,2550
1005    7758,6654
1031    6125,4334
1037    7562,9121
1154    6151,9173
1225    2327,8022
1259    8439,2731
1268    4840,2918
1502    8991,2251
1566    8169,962
1589    9667,9164
1722    8333,870
1746    3684,3595
```

Chunk size to view (in bytes, up to file's DFS block size): 32768

Total number of blocks: 1
5336575109328292755: 127.0.0.1:50010

Again, two screenshots to verify successes.



Hadoop job_201710042312_0001 on localhost - Mozilla Firefox

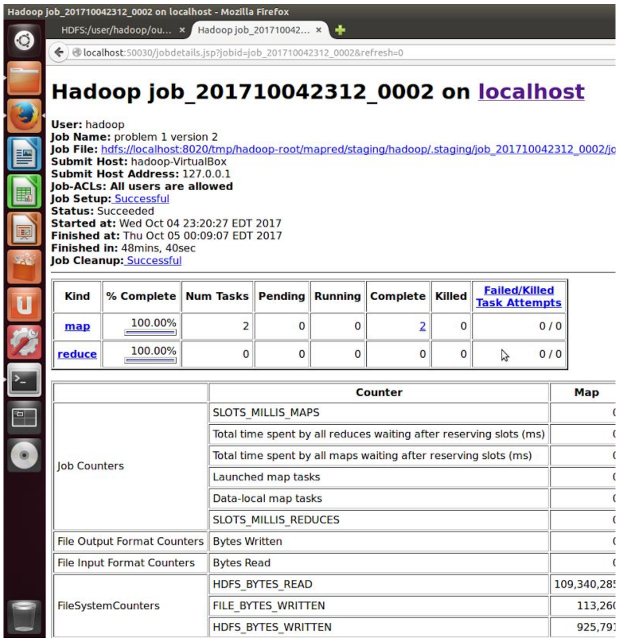
localhost:50030/jobdetails.jsp?jobid=job_201710042312_0001&refresh=0

Hadoop job_201710042312_0001 on localhost

User: hadoop
Job Name: problem 1 version 2
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710042312_0001/jo
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 04 23:13:55 EDT 2017
Finished at: Wed Oct 04 23:14:24 EDT 2017
Finished in: 28sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	0	0	0	0	0	0 / 0

	Counter	Map
Job Counters	SLOTS_MILLIS_MAPS	0
	Total time spent by all reduces waiting after reserving slots (ms)	0
	Total time spent by all maps waiting after reserving slots (ms)	0
	Launched map tasks	0
	Data-local map tasks	0
	SLOTS_MILLIS_REDUCES	0
File Output Format Counters	Bytes Written	0
File Input Format Counters	Bytes Read	0
FileSystemCounters	HDFS_BYTES_READ	109,340,285
	FILE_BYTES_WRITTEN	113,268
	HDFS_BYTES_WRITTEN	8,186



Hadoop job_201710042312_0002 on localhost - Mozilla Firefox

localhost:50030/jobdetails.jsp?jobid=job_201710042312_0002&refresh=0

Hadoop job_201710042312_0002 on localhost

User: hadoop
Job Name: problem 1 version 2
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710042312_0002/jo
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 04 23:20:27 EDT 2017
Finished at: Thu Oct 05 00:09:07 EDT 2017
Finished in: 48mins, 40sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	0	0	0	0	0	0 / 0

	Counter	Map
Job Counters	SLOTS_MILLIS_MAPS	0
	Total time spent by all reduces waiting after reserving slots (ms)	0
	Total time spent by all maps waiting after reserving slots (ms)	0
	Launched map tasks	0
	Data-local map tasks	0
	SLOTS_MILLIS_REDUCES	0
File Output Format Counters	Bytes Written	0
File Input Format Counters	Bytes Read	0
FileSystemCounters	HDFS_BYTES_READ	109,340,285
	FILE_BYTES_WRITTEN	113,260
	HDFS_BYTES_WRITTEN	925,791

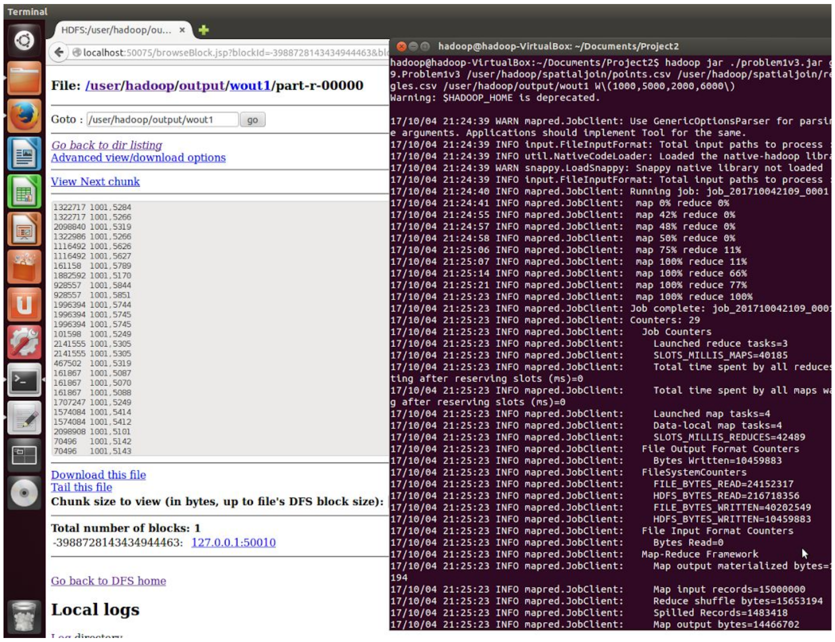
Both files are large. This is a one map-reduce job with a separate Mapper class for points and rectangles. We employ the "striping" method similar to the lecture on matrix multiplication. Our first attempt was to "stripe" by x-value, but Hadoop failed with the two large datasets. We further divided our "stripe" into "top" (y>5000) and "bottom" (y<=5000).

We join the points and rectangles with the key <x-value:top/bottom>. In addition, rectangles were marked with their id for the secondary sort (see next paragraph). The map output value for a point was simply the y-value. In the point mapper, we also filtered only the points that fell within or on the window. The map output value for a rectangle was the y-values for the bottom-left and top-right corners. Although not required for the correct result (since the point mapper filtered points in/on the window), we chose to also filter rectangles that overlapped the window to speed up the sort/shuffle and read/write to disk in the reducer.

After performing a few experiments, we determined that each key receives about twice as many rectangles than points in the reducer. Thus, we want to ensure that the reducer reads all points first and stores them in main memory. Next, as each rectangle arrives, it checks each point, similar to our previous solution (small points file, large rectangles file). We had to overwrite the partitioner, grouping comparator, and sort comparator in order to implement the secondary sort.

Please use code similar to the following to execute the jar file.

```
hadoop jar ./problem1v3.jar group9.Problem1v3 <small points input>
<large rectangles input> <output> <optional W\(xbl,ybl,xtr,ytr\)>
```



Terminal

HDFS:/user/hadoop/ou...

localhost:50075/browseBlock.jsp?blockId=-3988728143434944463&bl

File: /user/hadoop/output/wout1/part-r-00000

Goto : /user/hadoop/output/wout1 go

Go back to dir listing
Advanced view/download options

View Next chunk

```
1322717 1001,5284
1322717 1001,5266
2098840 1001,5319
1322986 1001,5266
1116492 1001,5626
1116492 1001,5627
161158  1001,5789
1882592 1001,5170
928557  1001,5844
928557  1001,5851
1996394 1001,5744
1996394 1001,5745
1996394 1001,5745
101598  1001,5249
2141555 1001,5305
2141555 1001,5305
467502  1001,5319
161867  1001,5087
161867  1001,5070
161867  1001,5088
1707247 1001,5249
1574084 1001,5414
1574084 1001,5412
2098908 1001,5101
70496   1001,5142
70496   1001,5143
```

Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block size):

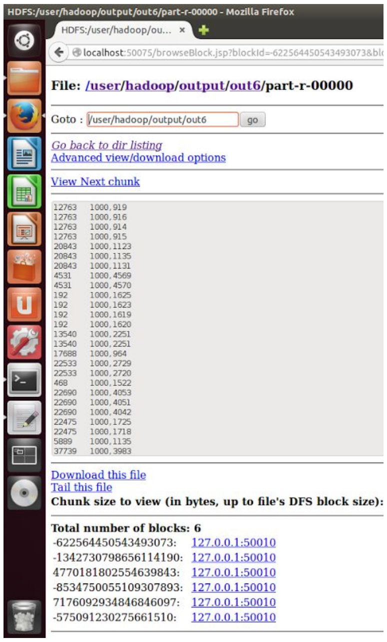
Total number of blocks: 1
-3988728143434944463: 127.0.0.1:50010

Go back to DFS home

Local logs

hadoop@hadoop-VirtualBox: ~/Documents/Project2

```
hadoop@hadoop-VirtualBox:~/Documents/Project2$ hadoop jar ./problem1v3.jar g
9.Problem1v3 /user/hadoop/spatialjoin/points.csv /user/hadoop/spatialjoin/re
gles.csv /user/hadoop/output/wout1 W\(1000,5000,2000,6000\)
Warning: $HADOOP_HOME is deprecated.

17/10/04 21:24:39 WARN mapred.JobClient: Use GenericOptionsParser for parsin
e arguments. Applications should implement Tool for the same.
17/10/04 21:24:39 INFO input.FileInputFormat: Total input paths to process :
17/10/04 21:24:39 INFO util.NativeCodeLoader: Loaded the native-hadoop libra
17/10/04 21:24:39 WARN snappy.LoadSnappy: Snappy native library not loaded
17/10/04 21:24:39 INFO input.FileInputFormat: Total input paths to process :
17/10/04 21:24:40 INFO mapred.JobClient: Running job: job_201710042109_0001
17/10/04 21:24:41 INFO mapred.JobClient:  map 0% reduce 0%
17/10/04 21:24:55 INFO mapred.JobClient:  map 42% reduce 0%
17/10/04 21:24:57 INFO mapred.JobClient:  map 48% reduce 0%
17/10/04 21:24:58 INFO mapred.JobClient:  map 50% reduce 0%
17/10/04 21:25:06 INFO mapred.JobClient:  map 75% reduce 11%
17/10/04 21:25:07 INFO mapred.JobClient:  map 100% reduce 11%
17/10/04 21:25:14 INFO mapred.JobClient:  map 100% reduce 66%
17/10/04 21:25:21 INFO mapred.JobClient:  map 100% reduce 77%
17/10/04 21:25:23 INFO mapred.JobClient:  map 100% reduce 100%
17/10/04 21:25:23 INFO mapred.JobClient: Job complete: job_201710042109_0001
17/10/04 21:25:23 INFO mapred.JobClient: Counters: 29
17/10/04 21:25:23 INFO mapred.JobClient:   Job Counters
17/10/04 21:25:23 INFO mapred.JobClient:     Launched reduce tasks=3
17/10/04 21:25:23 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=40185
17/10/04 21:25:23 INFO mapred.JobClient:     Total time spent by all reduces
ting after reserving slots (ms)=0
17/10/04 21:25:23 INFO mapred.JobClient:     Total time spent by all maps wa
g after reserving slots (ms)=0
17/10/04 21:25:23 INFO mapred.JobClient:     Launched map tasks=4
17/10/04 21:25:23 INFO mapred.JobClient:     Data-local map tasks=4
17/10/04 21:25:23 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=42489
17/10/04 21:25:23 INFO mapred.JobClient:   File Output Format Counters
17/10/04 21:25:23 INFO mapred.JobClient:     Bytes Written=10459883
17/10/04 21:25:23 INFO mapred.JobClient:   FileSystemCounters
17/10/04 21:25:23 INFO mapred.JobClient:     FILE_BYTES_READ=24152317
17/10/04 21:25:23 INFO mapred.JobClient:     HDFS_BYTES_READ=216718356
17/10/04 21:25:23 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=40202549
17/10/04 21:25:23 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=10459883
17/10/04 21:25:23 INFO mapred.JobClient:   File Input Format Counters
17/10/04 21:25:23 INFO mapred.JobClient:     Bytes Read=0
17/10/04 21:25:23 INFO mapred.JobClient:   Map-Reduce Framework
17/10/04 21:25:23 INFO mapred.JobClient:     Map output materialized bytes=1
194
17/10/04 21:25:23 INFO mapred.JobClient:     Map input records=15000000
17/10/04 21:25:23 INFO mapred.JobClient:     Reduce shuffle bytes=15653194
17/10/04 21:25:23 INFO mapred.JobClient:     Spilled Records=1483418
17/10/04 21:25:23 INFO mapred.JobClient:     Map output bytes=14466702
```



HDFS:/user/hadoop/output/out6/part-r-00000 - Mozilla Firefox

HDFS:/user/hadoop/ou...

localhost:50075/browseBlock.jsp?blockId=-6225644505434930738&bl

File: /user/hadoop/output/out6/part-r-00000

Goto : /user/hadoop/output/out6 go

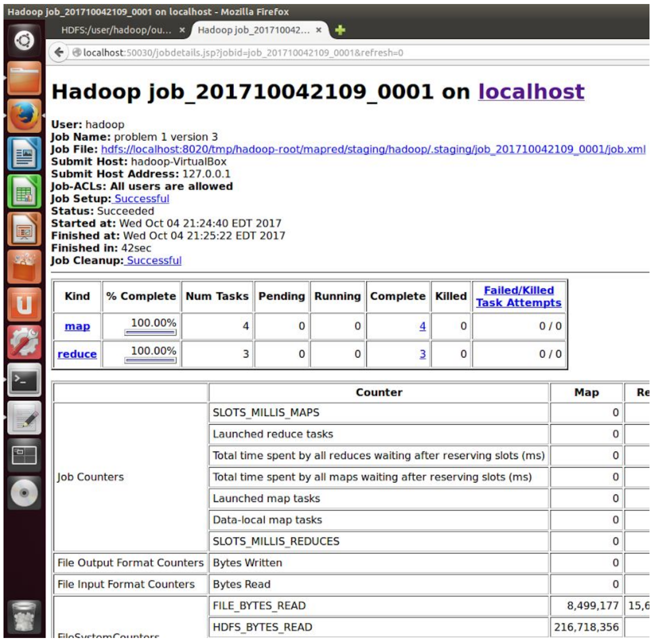
Go back to dir listing
Advanced view/download options

View Next chunk

```
12763   1000,919
12763   1000,916
12763   1000,914
12763   1000,915
20843   1000,1123
20843   1000,1135
20843   1000,1131
4531    1000,4569
4531    1000,4570
192     1000,1625
192     1000,1623
192     1000,1619
192     1000,1620
13540   1000,2251
13540   1000,2251
17688   1000,964
22533   1000,2729
22533   1000,2720
468     1000,1522
22690   1000,4053
22690   1000,4051
22690   1000,4042
22475   1000,1725
22475   1000,1718
5889    1000,1135
37739   1000,3983
```

Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block size):

Total number of blocks: 6
-622564450543493073: 127.0.0.1:50010
-1342730798656114190: 127.0.0.1:50010
4770181802554639843: 127.0.0.1:50010
-8534750055109307893: 127.0.0.1:50010
7176092934846846097: 127.0.0.1:50010
-575091230275661510: 127.0.0.1:50010



Hadoop job_201710042109_0001 on localhost - Mozilla Firefox

HDFS:/user/hadoop/ou... Hadoop job_201710042...

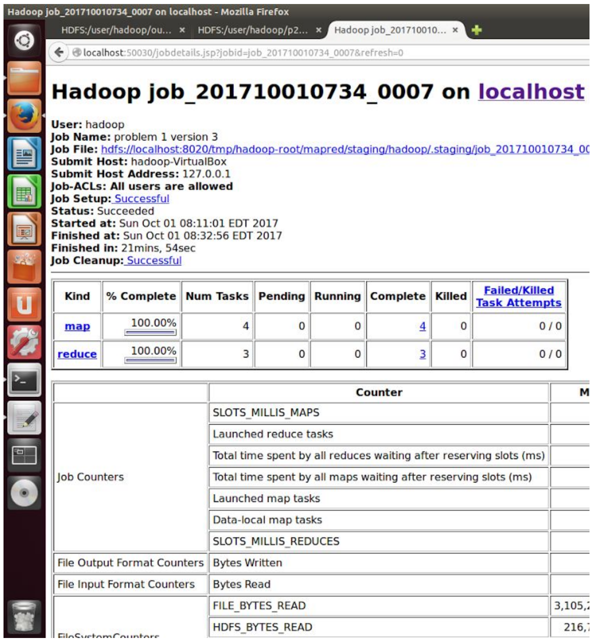
localhost:50030/jobdetails.jsp?jobid=job_201710042109_0001&refresh=0

Hadoop job_201710042109_0001 on localhost

User: hadoop
Job Name: problem 1 version 3
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710042109_0001/job.xml
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Wed Oct 04 21:24:40 EDT 2017
Finished at: Wed Oct 04 21:25:22 EDT 2017
Finished in: 42sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	4	0	0	4	0	0 / 0
reduce	100.00%	3	0	0	3	0	0 / 0

	Counter	Map	Re
Job Counters	SLOTS_MILLIS_MAPS	0	
	Launched reduce tasks	0	
	Total time spent by all reduces waiting after reserving slots (ms)	0	
	Total time spent by all maps waiting after reserving slots (ms)	0	
	Launched map tasks	0	
	Data-local map tasks	0	
	SLOTS_MILLIS_REDUCES	0	
File Output Format Counters	Bytes Written	0	
File Input Format Counters	Bytes Read	0	
FileSystemCounters	FILE_BYTES_READ	8,499,177	15,6
	HDFS_BYTES_READ	216,718,356	



Hadoop job_201710010734_0007 on localhost - Mozilla Firefox

HDFS:/user/hadoop/ou... HDFS:/user/hadoop/p2... Hadoop job_201710010...

localhost:50030/jobdetails.jsp?jobid=job_201710010734_0007&refresh=0

Hadoop job_201710010734_0007 on localhost

User: hadoop
Job Name: problem 1 version 3
Job File: hdfs://localhost:8020/tmp/hadoop-root/mapred/staging/hadoop/.staging/job_201710010734_00
Submit Host: hadoop-VirtualBox
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sun Oct 01 08:11:01 EDT 2017
Finished at: Sun Oct 01 08:32:56 EDT 2017
Finished in: 21mins, 54sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	4	0	0	4	0	0 / 0
reduce	100.00%	3	0	0	3	0	0 / 0

	Counter	M
Job Counters	SLOTS_MILLIS_MAPS	
	Launched reduce tasks	
	Total time spent by all reduces waiting after reserving slots (ms)	
	Total time spent by all maps waiting after reserving slots (ms)	
	Launched map tasks	
	Data-local map tasks	
	SLOTS_MILLIS_REDUCES	
File Output Format Counters	Bytes Written	
File Input Format Counters	Bytes Read	
FileSystemCounters	FILE_BYTES_READ	3,105,2
	HDFS_BYTES_READ	216,7

Final screenshots are above. The table below summarizes the performance of each job.

Job	W(1000,5000,2000,6000)	No window input
Small rectangles; large points	24 sec	23 min, 36 sec
Small points; large rectangles	28 sec	48 min, 40 sec
Large points; large rectangles	42 sec	21 min, 54 sec

We conclude that the “striping” method for large points and rectangles is best because it scales well, even though it had the slowest performance for a window. Indeed, if the data was event larger, we could continue striping the y-coordinates into smaller sections. We might generalize that as the window increases, the performance of the map-only broadcast join methods deteriorates much more rapidly.

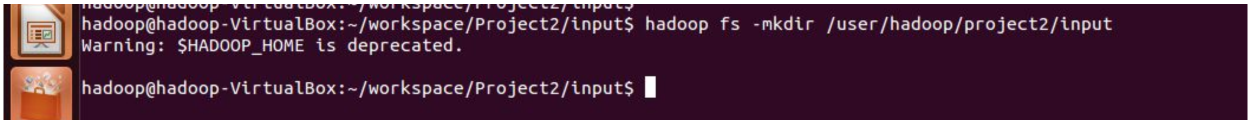
Finally, we discuss how our code for this problem would be adjusted for real number coordinates rather than integers. First, the data generation requires the Random instance to generate nextFloat() rather than nextInt() with some simple arithmetic to ensure that the domain and range is still [1, 10000]. In the Hadoop programs, we would parseFloat() rather than parseInt() and change the types of all applicable data structures.

Problem 2: Custom Input Format

Step 1 (Upload the Dataset)

We uploaded the dataset provided on Canvas. First, we created a folder on the Hadoop file system:

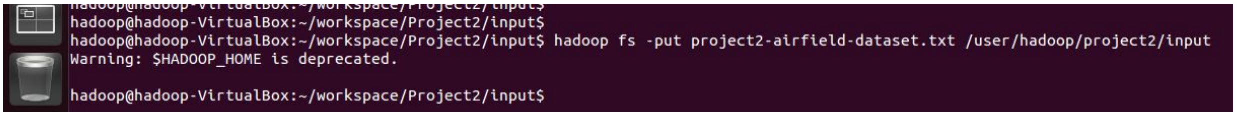
```
hadoop fs -mkdir /user/hadoop/project2/input
```



```
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$ hadoop fs -mkdir /user/hadoop/project2/input
Warning: $HADOOP_HOME is deprecated.

hadoop@hadoop-VirtualBox:~/workspace/Project2/input$ 
```

For loading dataset into hadoop, type:

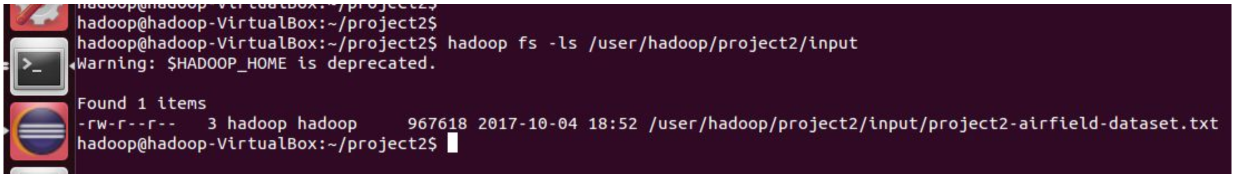
```
hadoop fs -put project2-airfield-dataset.txt
/user/hadoop/project2/input
```



```
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$ hadoop fs -put project2-airfield-dataset.txt /user/hadoop/project2/input
Warning: $HADOOP_HOME is deprecated.

hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
```

To verify if the file is inside the correct path, type:

```
hadoop fs -ls /user/hadoop/project/input
```



```
hadoop@hadoop-VirtualBox:~/project2$
hadoop@hadoop-VirtualBox:~/project2$
hadoop@hadoop-VirtualBox:~/project2$ hadoop fs -ls /user/hadoop/project2/input
Warning: $HADOOP_HOME is deprecated.

Found 1 items
-rw-r--r--   3 hadoop hadoop     967618 2017-10-04 18:52 /user/hadoop/project2/input/project2-airfield-dataset.txt
hadoop@hadoop-VirtualBox:~/project2$ 
```

Step 2 (MapReduce Job with a Custom Input Format)

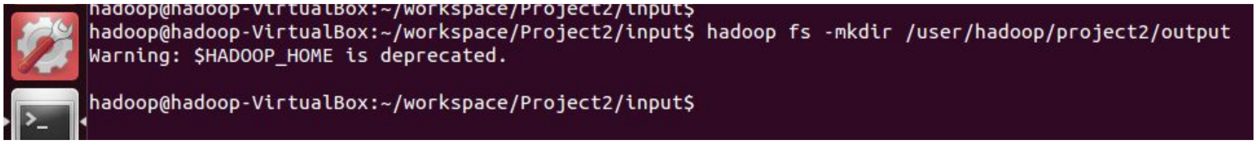
Our JsonInputFormat class extends the FileInputFormat class. (To view the code, please open JsonInputFormat.java.) We have overridden the RecordReader class. The overridden method nextKeyValue() has our core logic for reading JSON input file. We read in each line in the text file from “`{`“ symbol to “`}`” symbol and output the whole record as one Text line to the mapper.

We have one mapper and one reducer for this problem. The input for our mapper is <Text,Text>, and the output key-value pair is <IntWritable,IntWritable>. For the reducer (and combiner), the input is <IntWritable,IntWritable> and the output is <IntWritable,IntWritable>.

Making the key an IntWritable sorts our results numerically by elevation, which will be helpful to the user reading the output. The code is straightforward and analogous to a simple word count program. Our mapper, reducer, and `main()` code can be found in JsonOverider.java.

To execute the jar file, we created an output folder in Hadoop.

```
hadoop fs -mkdir /user/hadoop/project2/output
```



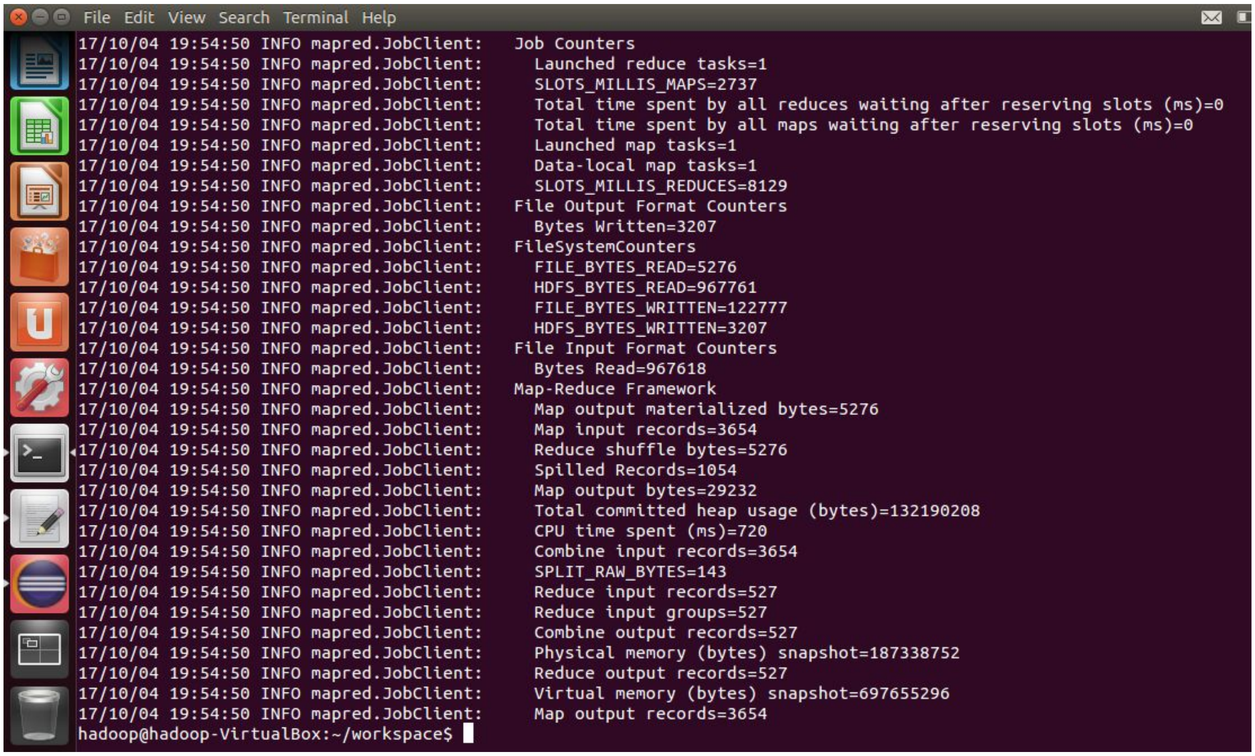
```
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
hadoop@hadoop-VirtualBox:~/workspace/Project2/input$ hadoop fs -mkdir /user/hadoop/project2/output
Warning: $HADOOP_HOME is deprecated.

hadoop@hadoop-VirtualBox:~/workspace/Project2/input$
```

Please use code similar to the following to execute the jar file.

```
hadoop jar problem2.jar project2.JsonOverider
/user/hadoop/project2/input /user/hadoop/project2/output
```

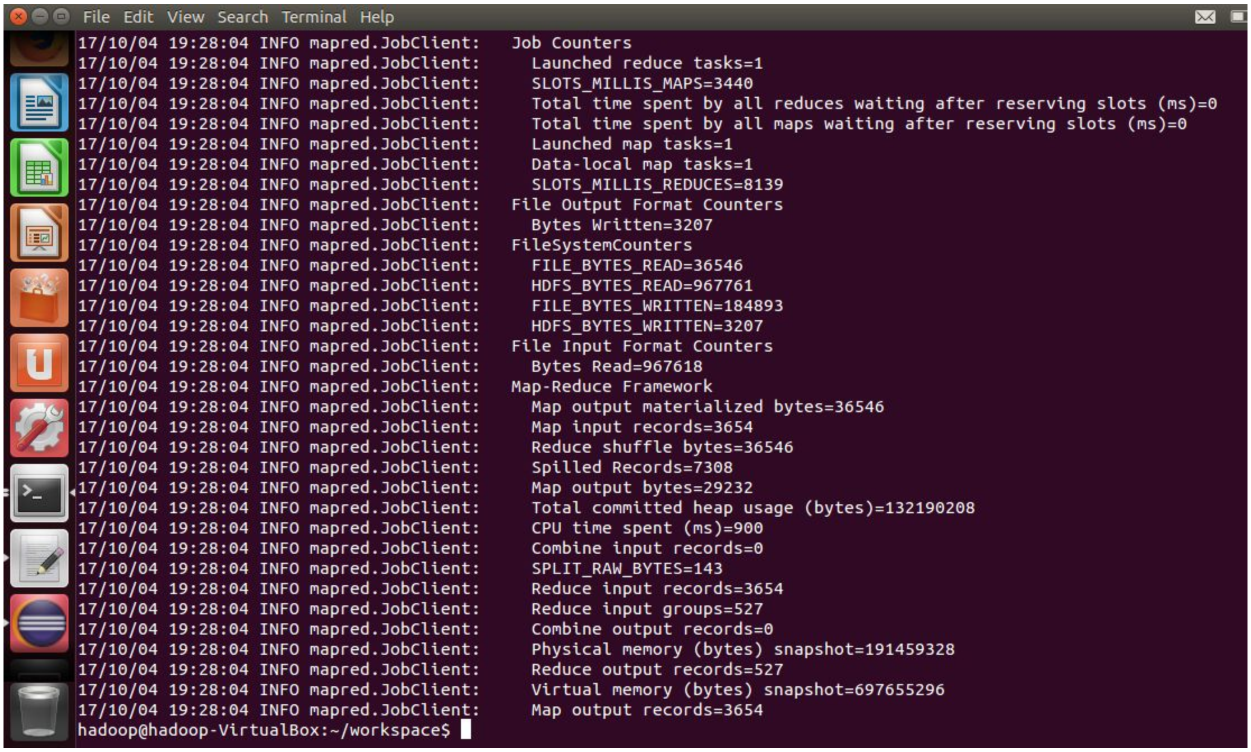
We executed the program both with and without a combiner to compare the performance. In the case of executing the job with a combiner, it takes about **13 seconds** to finish the job. In case of executing the job without the combiner, it takes about **20 seconds**. The combiner improves performance by decreasing shuffle/sort and I/O costs.

The following screenshot is the log on the terminal after executing the job *with* a combiner.



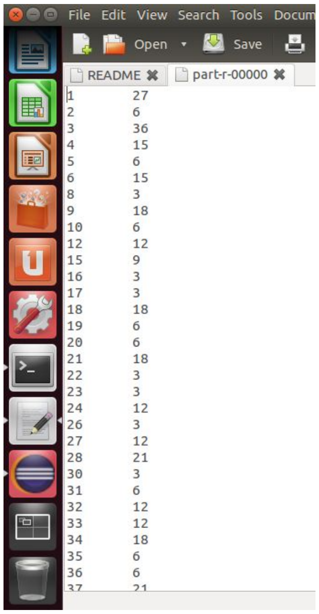
```
17/10/04 19:54:50 INFO mapred.JobClient:   Job Counters
17/10/04 19:54:50 INFO mapred.JobClient:     Launched reduce tasks=1
17/10/04 19:54:50 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=2737
17/10/04 19:54:50 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
17/10/04 19:54:50 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
17/10/04 19:54:50 INFO mapred.JobClient:     Launched map tasks=1
17/10/04 19:54:50 INFO mapred.JobClient:     Data-local map tasks=1
17/10/04 19:54:50 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=8129
17/10/04 19:54:50 INFO mapred.JobClient:   File Output Format Counters
17/10/04 19:54:50 INFO mapred.JobClient:     Bytes Written=3207
17/10/04 19:54:50 INFO mapred.JobClient:   FileSystemCounters
17/10/04 19:54:50 INFO mapred.JobClient:     FILE_BYTES_READ=5276
17/10/04 19:54:50 INFO mapred.JobClient:     HDFS_BYTES_READ=967761
17/10/04 19:54:50 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=122777
17/10/04 19:54:50 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=3207
17/10/04 19:54:50 INFO mapred.JobClient:   File Input Format Counters
17/10/04 19:54:50 INFO mapred.JobClient:     Bytes Read=967618
17/10/04 19:54:50 INFO mapred.JobClient:   Map-Reduce Framework
17/10/04 19:54:50 INFO mapred.JobClient:     Map output materialized bytes=5276
17/10/04 19:54:50 INFO mapred.JobClient:     Map input records=3654
17/10/04 19:54:50 INFO mapred.JobClient:     Reduce shuffle bytes=5276
17/10/04 19:54:50 INFO mapred.JobClient:     Spilled Records=1054
17/10/04 19:54:50 INFO mapred.JobClient:     Map output bytes=29232
17/10/04 19:54:50 INFO mapred.JobClient:     Total committed heap usage (bytes)=132190208
17/10/04 19:54:50 INFO mapred.JobClient:     CPU time spent (ms)=720
17/10/04 19:54:50 INFO mapred.JobClient:     Combine input records=3654
17/10/04 19:54:50 INFO mapred.JobClient:     SPLIT_RAW_BYTES=143
17/10/04 19:54:50 INFO mapred.JobClient:     Reduce input records=527
17/10/04 19:54:50 INFO mapred.JobClient:     Reduce input groups=527
17/10/04 19:54:50 INFO mapred.JobClient:     Combine output records=527
17/10/04 19:54:50 INFO mapred.JobClient:     Physical memory (bytes) snapshot=187338752
17/10/04 19:54:50 INFO mapred.JobClient:     Reduce output records=527
17/10/04 19:54:50 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=697655296
17/10/04 19:54:50 INFO mapred.JobClient:     Map output records=3654
hadoop@hadoop-VirtualBox:~/workspace$ █
```

After executing the job *without* a combiner:



```
17/10/04 19:28:04 INFO mapred.JobClient:   Job Counters
17/10/04 19:28:04 INFO mapred.JobClient:     Launched reduce tasks=1
17/10/04 19:28:04 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=3440
17/10/04 19:28:04 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
17/10/04 19:28:04 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
17/10/04 19:28:04 INFO mapred.JobClient:     Launched map tasks=1
17/10/04 19:28:04 INFO mapred.JobClient:     Data-local map tasks=1
17/10/04 19:28:04 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=8139
17/10/04 19:28:04 INFO mapred.JobClient:   File Output Format Counters
17/10/04 19:28:04 INFO mapred.JobClient:     Bytes Written=3207
17/10/04 19:28:04 INFO mapred.JobClient:   FileSystemCounters
17/10/04 19:28:04 INFO mapred.JobClient:     FILE_BYTES_READ=36546
17/10/04 19:28:04 INFO mapred.JobClient:     HDFS_BYTES_READ=967761
17/10/04 19:28:04 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=184893
17/10/04 19:28:04 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=3207
17/10/04 19:28:04 INFO mapred.JobClient:   File Input Format Counters
17/10/04 19:28:04 INFO mapred.JobClient:     Bytes Read=967618
17/10/04 19:28:04 INFO mapred.JobClient:   Map-Reduce Framework
17/10/04 19:28:04 INFO mapred.JobClient:     Map output materialized bytes=36546
17/10/04 19:28:04 INFO mapred.JobClient:     Map input records=3654
17/10/04 19:28:04 INFO mapred.JobClient:     Reduce shuffle bytes=36546
17/10/04 19:28:04 INFO mapred.JobClient:     Spilled Records=7308
17/10/04 19:28:04 INFO mapred.JobClient:     Map output bytes=29232
17/10/04 19:28:04 INFO mapred.JobClient:     Total committed heap usage (bytes)=132190208
17/10/04 19:28:04 INFO mapred.JobClient:     CPU time spent (ms)=900
17/10/04 19:28:04 INFO mapred.JobClient:     Combine input records=0
17/10/04 19:28:04 INFO mapred.JobClient:     SPLIT_RAW_BYTES=143
17/10/04 19:28:04 INFO mapred.JobClient:     Reduce input records=3654
17/10/04 19:28:04 INFO mapred.JobClient:     Reduce input groups=527
17/10/04 19:28:04 INFO mapred.JobClient:     Combine output records=0
17/10/04 19:28:04 INFO mapred.JobClient:     Physical memory (bytes) snapshot=191459328
17/10/04 19:28:04 INFO mapred.JobClient:     Reduce output records=527
17/10/04 19:28:04 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=697655296
17/10/04 19:28:04 INFO mapred.JobClient:     Map output records=3654
hadoop@hadoop-VirtualBox:~/workspace$
```

Finally, we have included a screenshot of sample output after executing the jar file. The number on the right (key) is the elevation, and the number on the left (value) is the count. For example, there were 36 records that listed an elevation of 3.



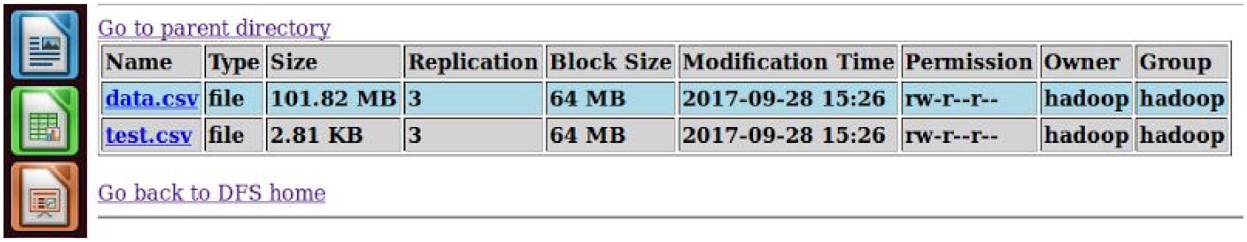
```
1       27
2       6
3       36
4       15
5       6
6       15
8       3
9       18
10      6
12      12
15      9
16      3
17      3
18      18
19      6
20      6
21      18
22      3
23      3
24      12
26      3
27      12
28      21
30      3
31      6
32      12
33      12
34      18
35      6
36      6
37      21
```

Problem 3: K-Means Clustering

Step 1 (Creation of Dataset) :

We created a dataset of 2D points based on some randomly chosen centers. This way, the data is more likely to be “clustered” than the uniform distribution of points we generated in Problem 1. Our data set contains 11 million points and is over 100MB (see the screenshot below). We also tested our code on a smaller set of 300 points.

To execute, run datacreation.jar group9.Main3. Use arguments such as: `-P kmeans.csv -k 110 -m 100000`



Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
data.csv	file	101.82 MB	3	64 MB	2017-09-28 15:26	rw-r--r--	hadoop	hadoop
test.csv	file	2.81 KB	3	64 MB	2017-09-28 15:26	rw-r--r--	hadoop	hadoop

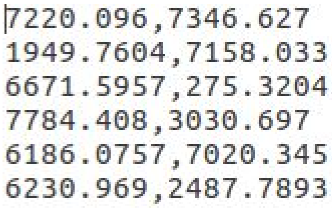
Go back to DFS home

To put data into HDFS:

```
hadoop fs -put /home/yifan/Desktop/data.txt /home/yifan/hw2/
```

We created another file that stores k random initial centers, put it into HDFS, and updated it during each iteration. This file is created each time we run a Map-Reduce Java class (when iteration is 0).

The following screenshot are initial k centers. Each line is a 2D point with real number coordinates.



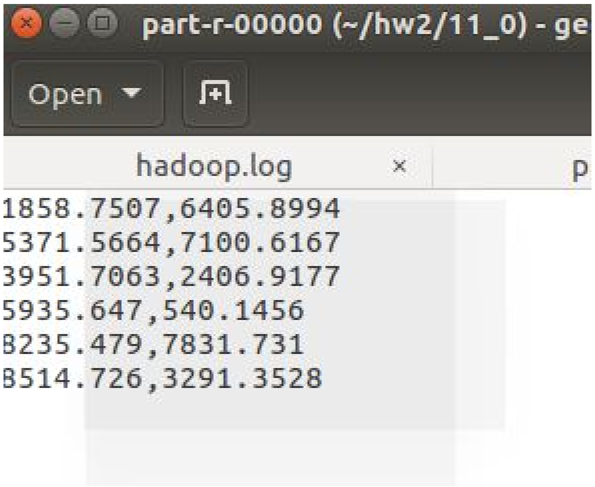
```
7220.096,7346.627
1949.7604,7158.033
6671.5957,275.3204
7784.408,3030.697
6186.0757,7020.345
6230.969,2487.7893
```

Step 2 (Clustering the Data)

1.single-iteration: We set the number of centers to 6. First, we created a file that contains the initial k centers. In the mapper, the points in data.csv are read in line by line. We use the k center files to find the nearest center for each point, returning Key (center)-Pair (point that belongs to this center). The reducer takes the output of the mapper, calculating mean values of all the points that belong to a certain center. The reducer returns the mean value (the centroid) as key.

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.oneiteration 6
/home/yifan/hw2/data.txt /home/yifan/hw2/1
```

Screenshot of output:

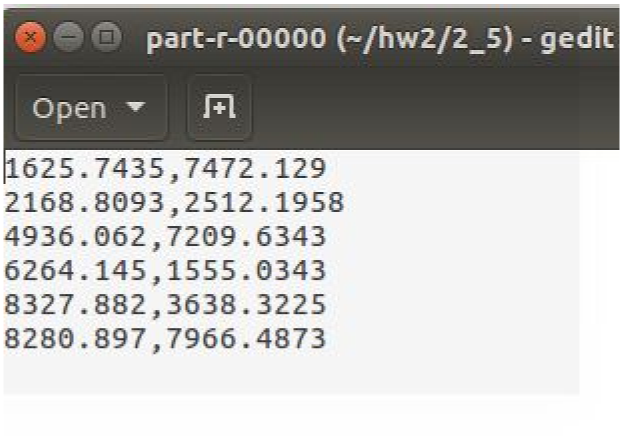


```
1858.7507,6405.8994
5371.5664,7100.6167
3951.7063,2406.9177
5935.647,540.1456
8235.479,7831.731
8514.726,3291.3528
```

2.Multi-iteration: We set the number of centers to 6. We made some changes in the DriverKmeans class. Each time a new map-reduce job starts, its input is the output from the last iteration. We set the number of iterations to 6.

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.multiiteration 6
/home/yifan/hw2/data.txt /home/yifan/hw2/2
```

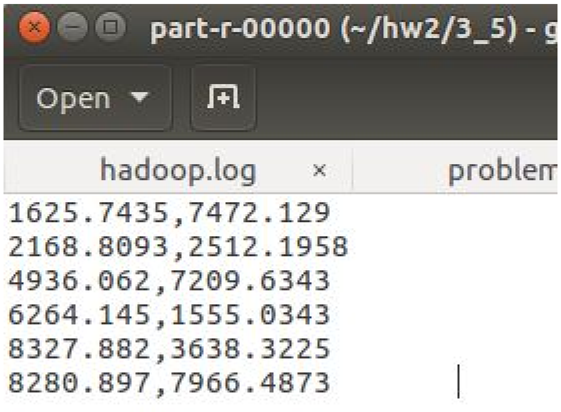
Screenshot of output:



```
1625.7435,7472.129
2168.8093,2512.1958
4936.062,7209.6343
6264.145,1555.0343
8327.882,3638.3225
8280.897,7966.4873
```

3. Same as above. We set the threshold to 100 and created a new variable which computes the difference between the latest centers and centers created from the last iteration. When the difference is under 100 or iteration comes to 6, the job will stop.

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.problem3 6
/home/yifan/hw2/data.txt /home/yifan/hw2/3
```

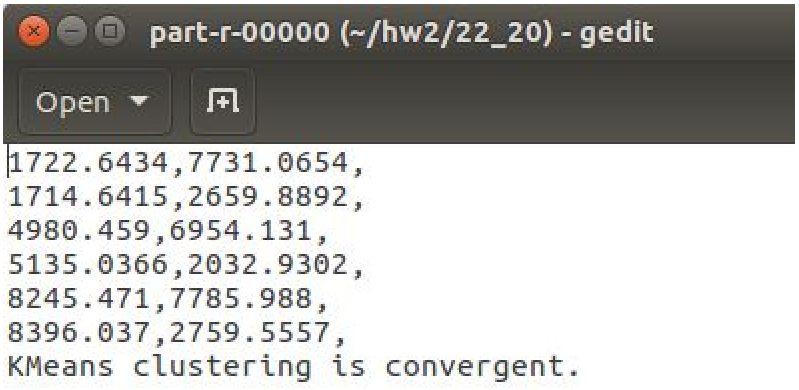


```
1625.7435,7472.129
2168.8093,2512.1958
4936.062,7209.6343
6264.145,1555.0343
8327.882,3638.3225
8280.897,7966.4873
```

We set iteration to 21 and threshold to 3000.

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.singleconvergence 6
/home/yifan/hw2/data.txt /home/yifan/hw2/22
```

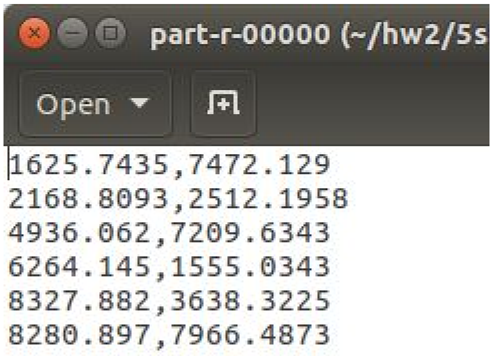
Screenshot of output:



```
1722.6434,7731.0654,
1714.6415,2659.8892,
4980.459,6954.131,
5135.0366,2032.9302,
8245.471,7785.988,
8396.037,2759.5557,
KMeans clustering is convergent.
```

4.1 SingleReducer: When doing this optimization, we first set the number of reducers to 1 and then remove the code comparing the difference between latest center and centers created from the last iteration in DriverKmeans to the Reducer class.

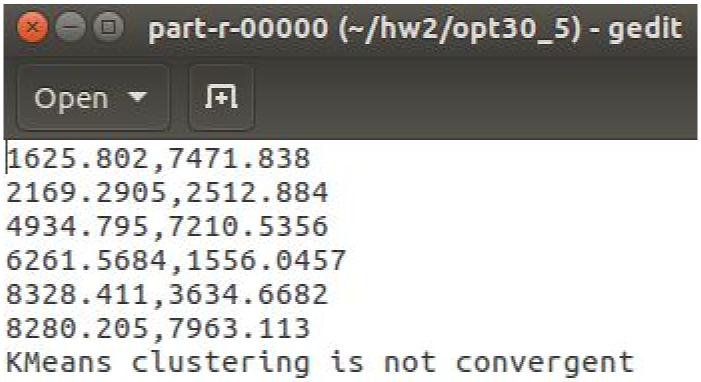
```
hadoop jar /home/yifan/hw2/hw2.jar hw2.single1 6
/home/yifan/hw2/data.txt /home/yifan/hw2/4_1
```



```
1625.7435,7472.129
2168.8093,2512.1958
4936.062,7209.6343
6264.145,1555.0343
8327.882,3638.3225
8280.897,7966.4873
```

4.2 Combiner: When doing optimization, we want to use a combiner in front of the reducer to locally combine the same key. First we created a file that contains the initial k centers. We applied one mapper and reducer. In the mapper, the point in data.csv is read in line by line. We use the k center files to find the nearest center for each point, returning Key (center)-Pair (point that belongs to this center,1). The combiner takes the output of mapper, calculating sum values of all the points and the number of points that belong to a certain center, passing Key (center)-Pair(sum,and number of points) to reducer.

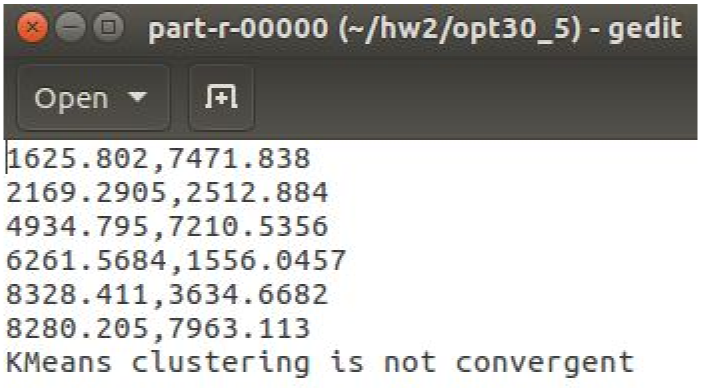
```
hadoop jar /home/yifan/hw2/hw2.jar hw2.problem3optimization 6
/home/yifan/hw2/data.txt /home/yifan/hw2/4_2
```



```
1625.802,7471.838
2169.2905,2512.884
4934.795,7210.5356
6261.5684,1556.0457
8328.411,3634.6682
8280.205,7963.113
KMeans clustering is not convergent
```

5.1(a) Convergence or Not: Using combiner, in the DriverKmeans class, if the difference between the latest center and centers created from the last iteration is below the threshold, we write “Convergence” into the file and otherwise, we write “not convergence.”

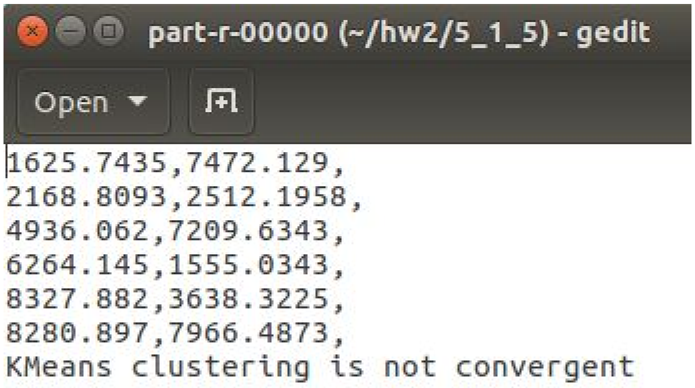
```
hadoop jar /home/yifan/hw2/hw2.jar hw2.problem3optimization 6
/home/yifan/hw2/data.txt /home/yifan/hw2/5_1
```



```
1625.802,7471.838
2169.2905,2512.884
4934.795,7210.5356
6261.5684,1556.0457
8328.411,3634.6682
8280.205,7963.113
KMeans clustering is not convergent
```

5.2(a): For a singlereducer task, based on 4.1, in the DriverKmeans class, if the difference between the latest center and centers created from last iteration is below the threshold, we write “Convergence” into the file and otherwise, we write “not convergence.”

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.singleconvergence 6
/home/yifan/hw2/data.txt /home/yifan/hw2/5_3
```



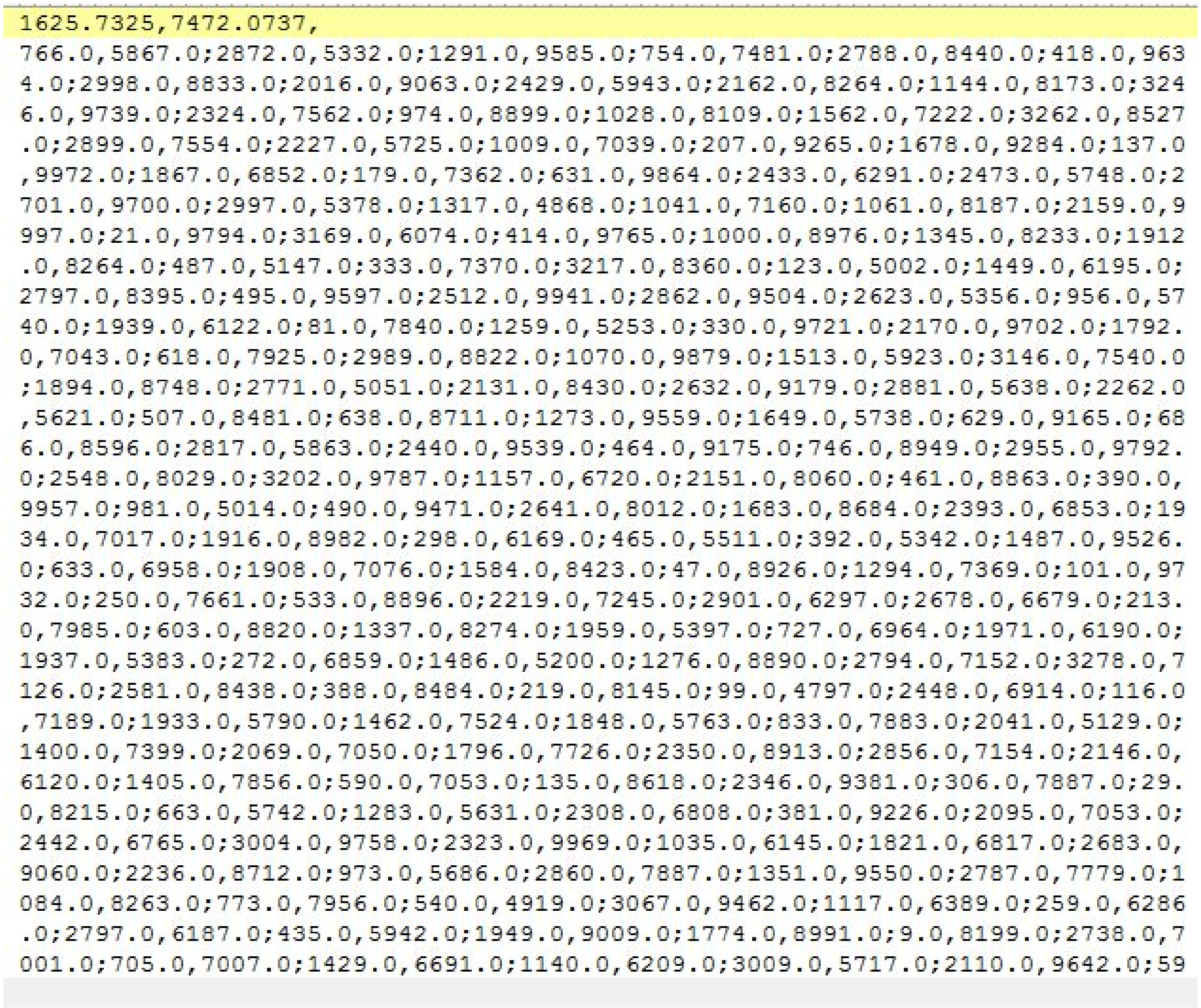
```
1625.7435,7472.129,
2168.8093,2512.1958,
4936.062,7209.6343,
6264.145,1555.0343,
8327.882,3638.3225,
8280.897,7966.4873,
KMeans clustering is not convergent
```

We hypothesize that the reason our algorithm did not converge is because our synthetic dataset was built to simulate over 100 clusters, but we chose a k value that was too small. Or perhaps we need more rounds of iteration.

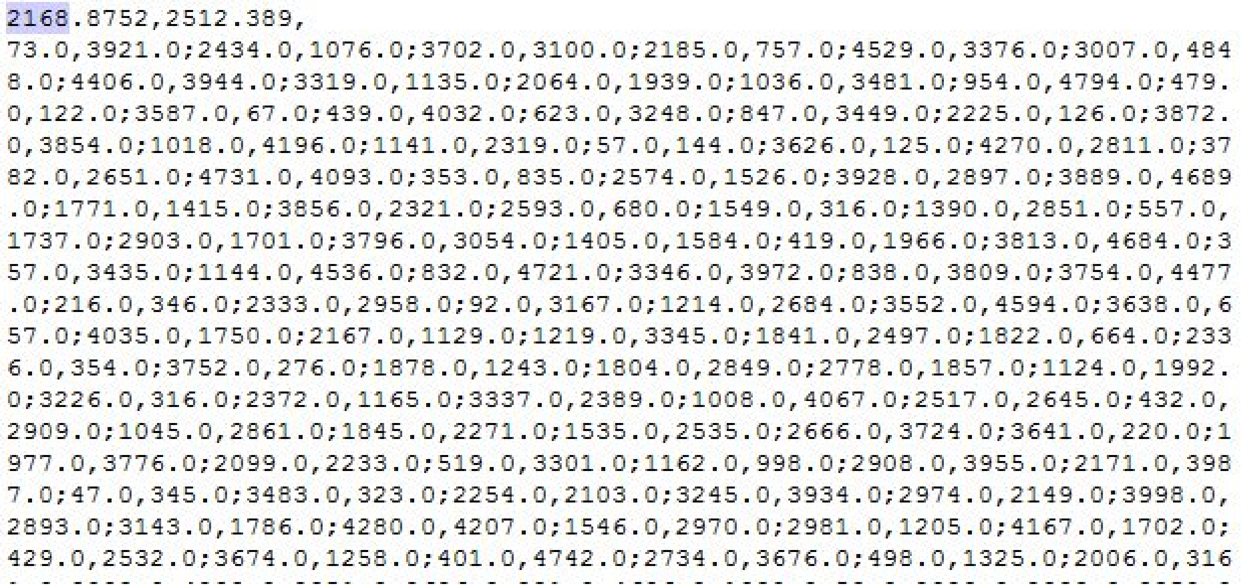
5.2(b):

For a singlereducer task, based on 4.1, in the Reducer class, we use a StringBuilder object to store all the clustered points for a certain center, returning Key (centers)-Pair (a set of points belongs to a center)

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.optallpoint 6
/home/yifan/hw2/data.txt /home/yifan/hw2/5_4
```



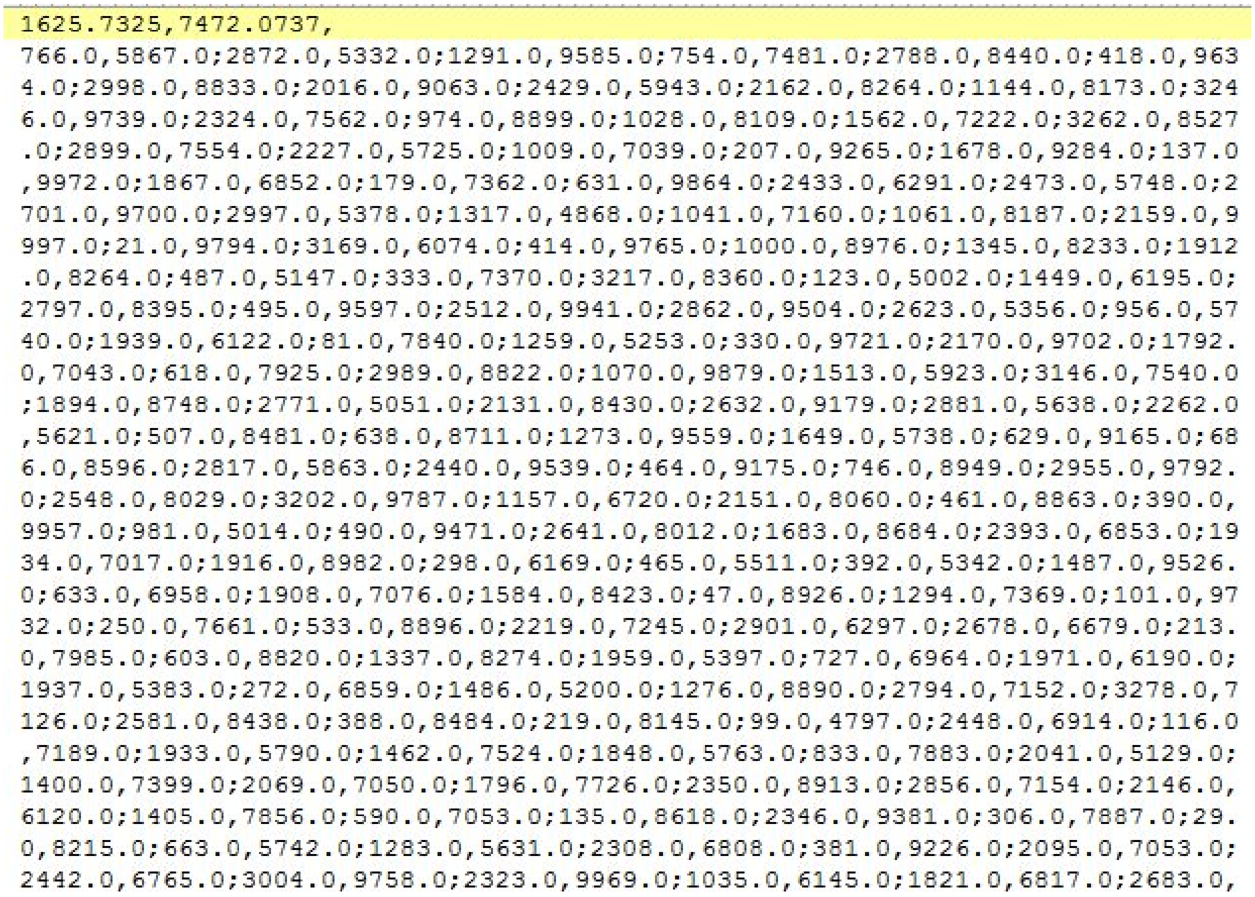
```
1625.7325,7472.0737,
766.0,5867.0;2872.0,5332.0;1291.0,9585.0;754.0,7481.0;2788.0,8440.0;418.0,963
4.0;2998.0,8833.0;2016.0,9063.0;2429.0,5943.0;2162.0,8264.0;1144.0,8173.0;324
6.0,9739.0;2324.0,7562.0;974.0,8899.0;1028.0,8109.0;1562.0,7222.0;3262.0,8527
.0;2899.0,7554.0;2227.0,5725.0;1009.0,7039.0;207.0,9265.0;1678.0,9284.0;137.0
,9972.0;1867.0,6852.0;179.0,7362.0;631.0,9864.0;2433.0,6291.0;2473.0,5748.0;2
701.0,9700.0;2997.0,5378.0;1317.0,4868.0;1041.0,7160.0;1061.0,8187.0;2159.0,9
997.0;21.0,9794.0;3169.0,6074.0;414.0,9765.0;1000.0,8976.0;1345.0,8233.0;1912
.0,8264.0;487.0,5147.0;333.0,7370.0;3217.0,8360.0;123.0,5002.0;1449.0,6195.0;
2797.0,8395.0;495.0,9597.0;2512.0,9941.0;2862.0,9504.0;2623.0,5356.0;956.0,57
40.0;1939.0,6122.0;81.0,7840.0;1259.0,5253.0;330.0,9721.0;2170.0,9702.0;1792.
0,7043.0;618.0,7925.0;2989.0,8822.0;1070.0,9879.0;1513.0,5923.0;3146.0,7540.0
;1894.0,8748.0;2771.0,5051.0;2131.0,8430.0;2632.0,9179.0;2881.0,5638.0;2262.0
,5621.0;507.0,8481.0;638.0,8711.0;1273.0,9559.0;1649.0,5738.0;629.0,9165.0;68
6.0,8596.0;2817.0,5863.0;2440.0,9539.0;464.0,9175.0;746.0,8949.0;2955.0,9792.
0;2548.0,8029.0;3202.0,9787.0;1157.0,6720.0;2151.0,8060.0;461.0,8863.0;390.0,
9957.0;981.0,5014.0;490.0,9471.0;2641.0,8012.0;1683.0,8684.0;2393.0,6853.0;19
34.0,7017.0;1916.0,8982.0;298.0,6169.0;465.0,5511.0;392.0,5342.0;1487.0,9526.
0;633.0,6958.0;1908.0,7076.0;1584.0,8423.0;47.0,8926.0;1294.0,7369.0;101.0,97
32.0;250.0,7661.0;533.0,8896.0;2219.0,7245.0;2901.0,6297.0;2678.0,6679.0;213.
0,7985.0;603.0,8820.0;1337.0,8274.0;1959.0,5397.0;727.0,6964.0;1971.0,6190.0;
1937.0,5383.0;272.0,6859.0;1486.0,5200.0;1276.0,8890.0;2794.0,7152.0;3278.0,7
126.0;2581.0,8438.0;388.0,8484.0;219.0,8145.0;99.0,4797.0;2448.0,6914.0;116.0
,7189.0;1933.0,5790.0;1462.0,7524.0;1848.0,5763.0;833.0,7883.0;2041.0,5129.0;
1400.0,7399.0;2069.0,7050.0;1796.0,7726.0;2350.0,8913.0;2856.0,7154.0;2146.0,
6120.0;1405.0,7856.0;590.0,7053.0;135.0,8618.0;2346.0,9381.0;306.0,7887.0;29.
0,8215.0;663.0,5742.0;1283.0,5631.0;2308.0,6808.0;381.0,9226.0;2095.0,7053.0;
2442.0,6765.0;3004.0,9758.0;2323.0,9969.0;1035.0,6145.0;1821.0,6817.0;2683.0,
9060.0;2236.0,8712.0;973.0,5686.0;2860.0,7887.0;1351.0,9550.0;2787.0,7779.0;1
084.0,8263.0;773.0,7956.0;540.0,4919.0;3067.0,9462.0;1117.0,6389.0;259.0,6286
.0;2797.0,6187.0;435.0,5942.0;1949.0,9009.0;1774.0,8991.0;9.0,8199.0;2738.0,7
001.0;705.0,7007.0;1429.0,6691.0;1140.0,6209.0;3009.0,5717.0;2110.0,9642.0;59
```



```
2168.8752,2512.389,
73.0,3921.0;2434.0,1076.0;3702.0,3100.0;2185.0,757.0;4529.0,3376.0;3007.0,484
8.0;4406.0,3944.0;3319.0,1135.0;2064.0,1939.0;1036.0,3481.0;954.0,4794.0;479.
0,122.0;3587.0,67.0;439.0,4032.0;623.0,3248.0;847.0,3449.0;2225.0,126.0;3872.
0,3854.0;1018.0,4196.0;1141.0,2319.0;57.0,144.0;3626.0,125.0;4270.0,2811.0;37
82.0,2651.0;4731.0,4093.0;353.0,835.0;2574.0,1526.0;3928.0,2897.0;3889.0,4689
.0;1771.0,1415.0;3856.0,2321.0;2593.0,680.0;1549.0,316.0;1390.0,2851.0;557.0,
1737.0;2903.0,1701.0;3796.0,3054.0;1405.0,1584.0;419.0,1966.0;3813.0,4684.0;3
57.0,3435.0;1144.0,4536.0;832.0,4721.0;3346.0,3972.0;838.0,3809.0;3754.0,4477
.0;216.0,346.0;2333.0,2958.0;92.0,3167.0;1214.0,2684.0;3552.0,4594.0;3638.0,6
57.0;4035.0,1750.0;2167.0,1129.0;1219.0,3345.0;1841.0,2497.0;1822.0,664.0;233
6.0,354.0;3752.0,276.0;1878.0,1243.0;1804.0,2849.0;2778.0,1857.0;1124.0,1992.
0;3226.0,316.0;2372.0,1165.0;3337.0,2389.0;1008.0,4067.0;2517.0,2645.0;432.0,
2909.0;1045.0,2861.0;1845.0,2271.0;1535.0,2535.0;2666.0,3724.0;3641.0,220.0;1
977.0,3776.0;2099.0,2233.0;519.0,3301.0;1162.0,998.0;2908.0,3955.0;2171.0,398
7.0;47.0,345.0;3483.0,323.0;2254.0,2103.0;3245.0,3934.0;2974.0,2149.0;3998.0,
2893.0;3143.0,1786.0;4280.0,4207.0;1546.0,2970.0;2981.0,1205.0;4167.0,1702.0;
429.0,2532.0;3674.0,1258.0;401.0,4742.0;2734.0,3676.0;498.0,1325.0;2006.0,316
```

Same as above, except we have multiple reducers.

```
hadoop jar /home/yifan/hw2/hw2.jar hw2.problem3opt4 6
/home/yifan/hw2/datas.txt /home/yifan/hw2/5_2
```



```
1625.7325,7472.0737,
766.0,5867.0;2872.0,5332.0;1291.0,9585.0;754.0,7481.0;2788.0,8440.0;418.0,963
4.0;2998.0,8833.0;2016.0,9063.0;2429.0,5943.0;2162.0,8264.0;1144.0,8173.0;324
6.0,9739.0;2324.0,7562.0;974.0,8899.0;1028.0,8109.0;1562.0,7222.0;3262.0,8527
.0;2899.0,7554.0;2227.0,5725.0;1009.0,7039.0;207.0,9265.0;1678.0,9284.0;137.0
,9972.0;1867.0,6852.0;179.0,7362.0;631.0,9864.0;2433.0,6291.0;2473.0,5748.0;2
701.0,9700.0;2997.0,5378.0;1317.0,4868.0;1041.0,7160.0;1061.0,8187.0;2159.0,9
997.0;21.0,9794.0;3169.0,6074.0;414.0,9765.0;1000.0,8976.0;1345.0,8233.0;1912
.0,8264.0;487.0,5147.0;333.0,7370.0;3217.0,8360.0;123.0,5002.0;1449.0,6195.0;
2797.0,8395.0;495.0,9597.0;2512.0,9941.0;2862.0,9504.0;2623.0,5356.0;956.0,57
40.0;1939.0,6122.0;81.0,7840.0;1259.0,5253.0;330.0,9721.0;2170.0,9702.0;1792.
0,7043.0;618.0,7925.0;2989.0,8822.0;1070.0,9879.0;1513.0,5923.0;3146.0,7540.0
;1894.0,8748.0;2771.0,5051.0;2131.0,8430.0;2632.0,9179.0;2881.0,5638.0;2262.0
,5621.0;507.0,8481.0;638.0,8711.0;1273.0,9559.0;1649.0,5738.0;629.0,9165.0;68
6.0,8596.0;2817.0,5863.0;2440.0,9539.0;464.0,9175.0;746.0,8949.0;2955.0,9792.
0;2548.0,8029.0;3202.0,9787.0;1157.0,6720.0;2151.0,8060.0;461.0,8863.0;390.0,
9957.0;981.0,5014.0;490.0,9471.0;2641.0,8012.0;1683.0,8684.0;2393.0,6853.0;19
34.0,7017.0;1916.0,8982.0;298.0,6169.0;465.0,5511.0;392.0,5342.0;1487.0,9526.
0;633.0,6958.0;1908.0,7076.0;1584.0,8423.0;47.0,8926.0;1294.0,7369.0;101.0,97
32.0;250.0,7661.0;533.0,8896.0;2219.0,7245.0;2901.0,6297.0;2678.0,6679.0;213.
0,7985.0;603.0,8820.0;1337.0,8274.0;1959.0,5397.0;727.0,6964.0;1971.0,6190.0;
1937.0,5383.0;272.0,6859.0;1486.0,5200.0;1276.0,8890.0;2794.0,7152.0;3278.0,7
126.0;2581.0,8438.0;388.0,8484.0;219.0,8145.0;99.0,4797.0;2448.0,6914.0;116.0
,7189.0;1933.0,5790.0;1462.0,7524.0;1848.0,5763.0;833.0,7883.0;2041.0,5129.0;
1400.0,7399.0;2069.0,7050.0;1796.0,7726.0;2350.0,8913.0;2856.0,7154.0;2146.0,
6120.0;1405.0,7856.0;590.0,7053.0;135.0,8618.0;2346.0,9381.0;306.0,7887.0;29.
0,8215.0;663.0,5742.0;1283.0,5631.0;2308.0,6808.0;381.0,9226.0;2095.0,7053.0;
2442.0,6765.0;3004.0,9758.0;2323.0,9969.0;1035.0,6145.0;1821.0,6817.0;2683.0,
```

Performance:

Job	Running Time
One iteration	60s
Multiple iteration	337s
Not convergent (iteration: 6)	345s
convergent (iteration: 20)	1258s
Single Reducer	344s
With Combiner	215s
Whether convergent (Single Reducer)	330s

Write out final clustered points (Single Reducer)	475s
Write out final clustered points (Multiple Reducer)	500s

Findings: we need more running time if we set more iterations. Single Reducer is a little faster than multiple reducer because we have less output and input. We can calculate difference in the reducer node for a single reducer. Using the combiner will reduce running time because combiner provides local combining, decreasing tasks for shuffling. Writing out all final points needs more time than just indicating whether it is convergent because there is more write out and read in cost.

Team Member Background and Contributions

Although our group is quite diverse in terms of demographics and program of study, we were surprised to find that we are quite similar in terms of background skills. Although this class was our first time using Hadoop, all three of us were comfortable coding in Java. Thus, our general strategy for this project was that each of us selected 1-2 of the problems that were most interesting to us and solved them. We shared our solutions via GitHub, read each other's solutions, and provided feedback until we were satisfied with three final solutions.

Problem 1: Allison and Abhinav both completed this problem independently. Abhinav chose to solve the problem with a small rectangles dataset and large points dataset, while Allison also explored other solutions. Xiaoting reviewed their codes.

Problem 2: Allison and Abhinav both completed this problem independently. Their solutions were very similar, and they discussed the pros and cons of the slight differences in their code before submitting a final solution.

Problem 3: Xiaoting completed this problem in its entirety. Allison decided to generate synthetic data that had "clusters," i.e. not uniformly distributed, to test the code on a more meaningful dataset. Abhinav re-ran her code to verify performance. The team reviewed Xiaoting's code and provided comments, which she applied herself.

Report: Allison wrote the documentation for Problem 1, Abhinav wrote Problem 2, and Xiaoting wrote Problem 3. Everyone read and edited the document to prepare it for final submission.

The skills we all acquired while working on this project include implementation of a broadcast join, handling different input format types, and more familiarity with the Hadoop framework. Our team worked together effectively to complete all problems and organize our results.