IS71059B Big Data Applications

Assignment 1 Report

John Downing

Contents

Introduction

This report summarises the results of two machine learning algorithms applied to the data provided with the assignment: KMeans and Expectation Maximisation (EM).

Both are similar in that they attempt to find the best fit between given datapoints and a pre-specified number of clusters, using an iterative process. However, EM allows each cluster to have its own shape, size and orientation – whereas KMeans does not.

Each iterative step is data-parallel for both algorithms, therefore each algorithm is a candidate for a MapReduce implementation. For KMeans, the mapper step calculates the Euclidian distance from a datapoint to each of the cluster centres and assigns it to the one which is closest. The reducer step then recalculates a cluster centre to be the mean coordinates of those datapoints assigned to it. For EM, the mapper (the “E Step”) calculates the likelihood that a datapoint belongs to each of the clusters, based on their probability density functions (PDF) in a Gaussian mixture model. The reducer (the “M Step”) then recalculates the cluster parameters – centroid, weight and covariance matrix – based on the likelihoods of each datapoint belonging to it. In both cases, MapReduce has to be invoked multiple times until the cluster recalculations converge, with the output from one invocation feeding into the next.

KMeans

Files

The following files are used in the KMeans implementation.

- `mapper_kmeans.py`

  This is the completed version of the file provided with the assignment, and determines the nearest cluster for a datapoint based on its Euclidian distance from the cluster centre. It outputs a single key,value pair for each datapoint, with the ID of the nearest cluster as the key and the coordinates of the datapoint as the value.
- `reducer_kmeans.py`

  This is the completed version of the file provided with the assignment, and recalculates cluster centres to be the mean co-ordinates of those datapoints assigned to them by the mapper. It outputs the ID and updated coordinates of each cluster passed to it, in the same format as the original clusters.txt file.
- `plot_kmeans.py`

  This is a supplementary script used to plot the resulting cluster centres and datapoints once they have converged. In theory, it could be run as part of the bash wrapper script (below), however we don't seem to have Python graphics enabled on the DSM cluster.
- `kmeans.mapred.sh`

  This is a bash script which wraps iterations of the MapReduce process in a loop, invoking an Hadoop streaming job at each stage and feeding the output from the previous stage into the next one. The script exists after 10 iterations, or if the output from the reducer does not change from the previous stage.
- `data.txt`

  This is the file of datapoints provided with the assignment.
- `clusters.txt`

  This is the file of initial cluster centres provided with the assignment.

How to run

The command for a single invocation of KMeans MapReduce on the DSM cluster is as follows. This assumes the default Python 2 environment, but should not be version specific. The input (data.txt) file should be copied to HDFS before running the script, but all other files are provided as part of the configuration[1]. Three reducers are specified, one for each cluster[2].

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.0.jar \
-Dmapred.reduce.tasks=3 \
-files=mapper_kmeans.py,reducer_kmeans.py,clusters.txt \
-cmdenv clusters=clusters.txt \
-mapper mapper_kmeans.py \
-reducer reducer_kmeans.py \
-input data.txt \
-output clusters
```

However, rather than repeat this process manually until convergence, invoking the `kmeans.mapred.sh` script does the following:

```
#!/bin/bash

for i in {1..10}
do
        newclusters="clusters.mapred.txt.$i"
        clusters="clusters.mapred.txt.$((i-1))"
        if [ $i == 1 ]; then
                clusters="clusters.txt"
        fi
        echo "iteration $i"
        echo "------------"
        hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.0.jar
-Dmapred.reduce.tasks=3 -files=mapper_kmeans.py,reducer_kmeans.py,$clusters -
cmdenv clusters=$clusters -mapper mapper_kmeans.py -reducer reducer_kmeans.py -
input data.txt -output clusters
        hadoop fs -cat clusters/part-* | sort > $newclusters
        hadoop fs -rm -r clusters
        diff_val=$(diff $clusters $newclusters)
        if  [ -z "$diff_val" ]; then
                echo "Exiting loop as clusters have not changed this iteration"
                break
        fi
done
```

[1] AIUI, the streaming job puts them in the Hadoop Distributed Cache so that they are visible to Hadoop/Python at runtime. This (probably) means that they are temporarily copied to HDFS, but all this is done behind the scenes – there's no need to manually put them there.

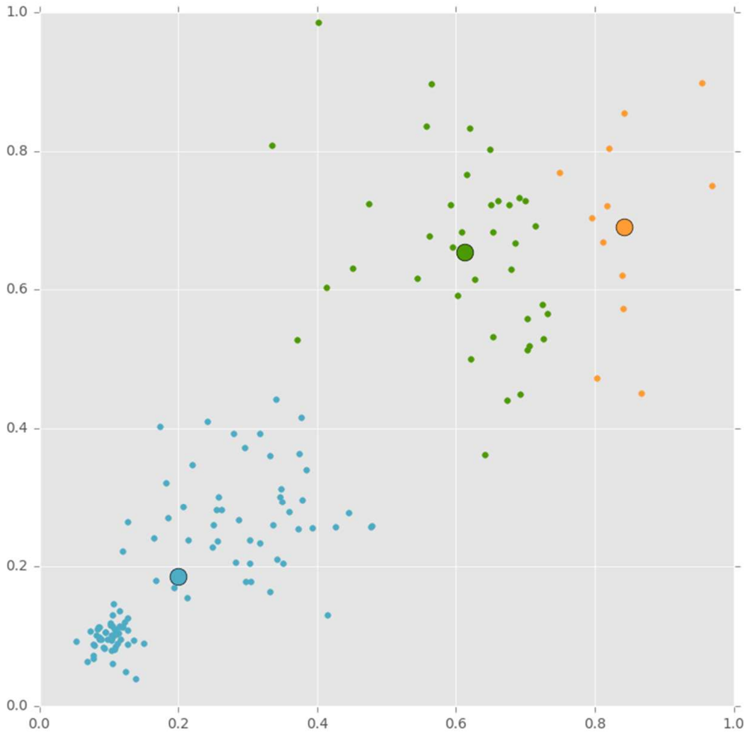
[2] I didn't attempt to tune the number of mappers, as there aren't enough datapoints to get meaningful results – since the natural variance in time taken outside the mapper (e.g. in starting up and invoking jobs), would most likely dominate timings. The same will be for the reducers, but since we have a natural separation of three clusters then it's at least cleaner (in terms of the code path) to have a separate reducer for each one.

Results

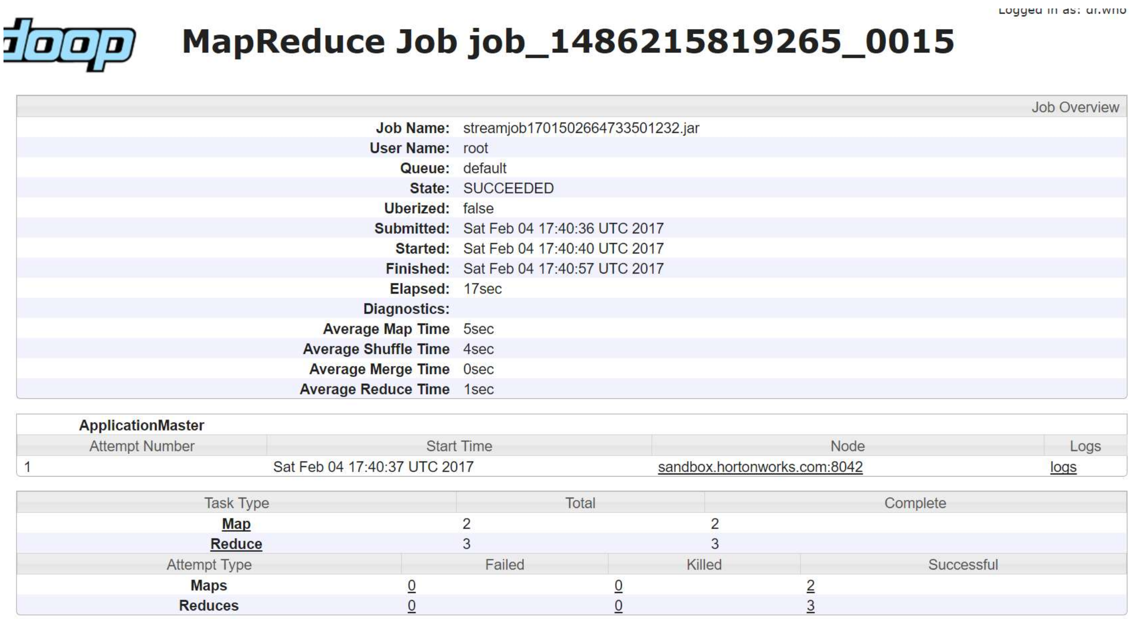
The clusters converged in 7 iterations, with the final co-ordinates given below.

ID	CX	CY
0	0.200302217487	0.186358139854
1	0.842339504006	0.690283942841
2	0.612260390886	0.653529296302

The plot below shows the datapoints and the locations of the final clusters. Datapoints are coloured to show which cluster they are assigned to.



Also provided below are screenshots from the Hadoop Job History app, as confirmation that this particular iteration of the streaming job was split into 2 map tasks and 3 reduce tasks – with each reduce task handling a separate cluster[3].



Logged in as: dr.who

MapReduce Job job_1486215819265_0015

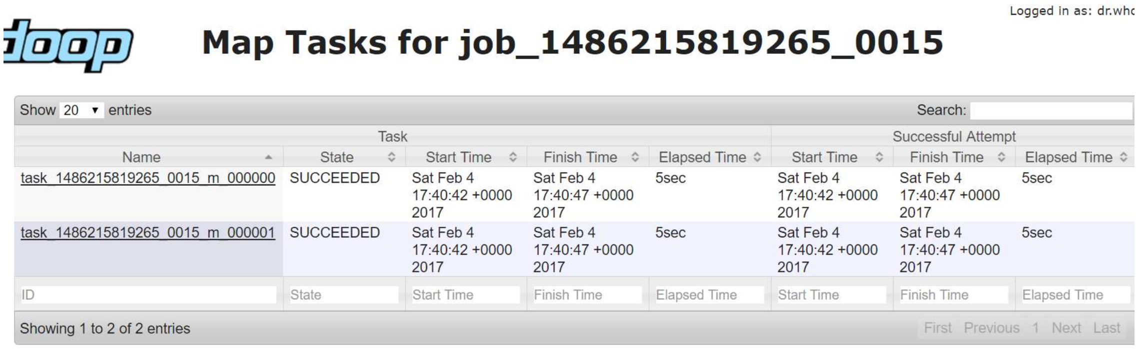
	Job Overview
Job Name:	streamjob1701502664733501232.jar
User Name:	root
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sat Feb 04 17:40:36 UTC 2017
Started:	Sat Feb 04 17:40:40 UTC 2017
Finished:	Sat Feb 04 17:40:57 UTC 2017
Elapsed:	17sec
Diagnostics:	
Average Map Time	5sec
Average Shuffle Time	4sec
Average Merge Time	0sec
Average Reduce Time	1sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Sat Feb 04 17:40:37 UTC 2017	sandbox.hortonworks.com:8042	logs

Task Type	Total	Complete
Map	2	2
Reduce	3	3

Attempt Type	Failed	Killed	Successful
Maps	0	0	2
Reduces	0	0	3



Logged in as: dr.whc

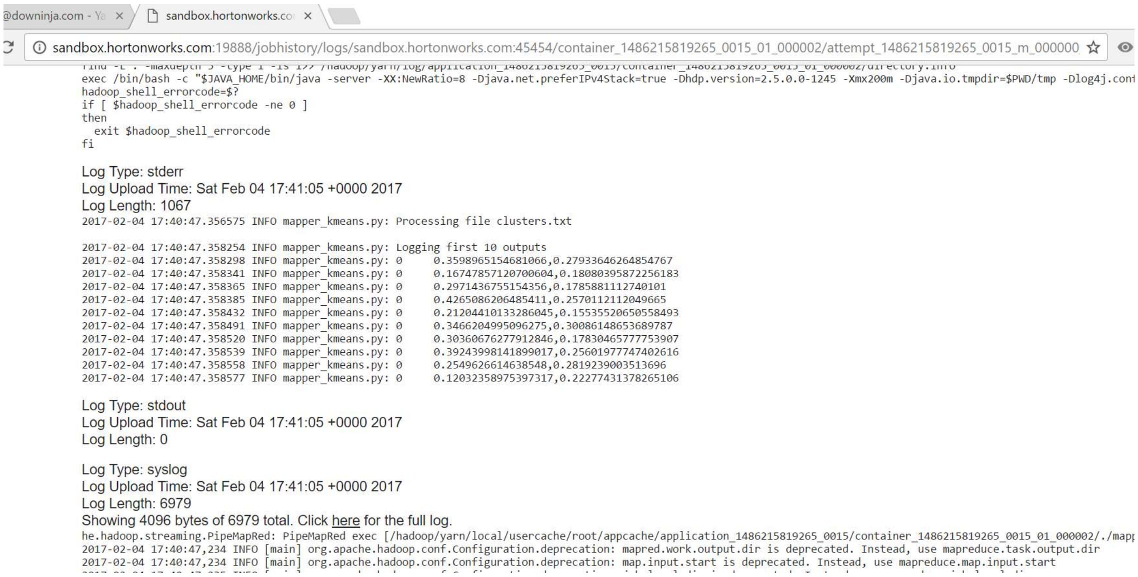
Map Tasks for job_1486215819265_0015

Show 20 entries Search:

Task					Successful Attempt		
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time
task_1486215819265_0015_m_000000	SUCCEEDED	Sat Feb 4 17:40:42 +0000 2017	Sat Feb 4 17:40:47 +0000 2017	5sec	Sat Feb 4 17:40:42 +0000 2017	Sat Feb 4 17:40:47 +0000 2017	5sec
task_1486215819265_0015_m_000001	SUCCEEDED	Sat Feb 4 17:40:42 +0000 2017	Sat Feb 4 17:40:47 +0000 2017	5sec	Sat Feb 4 17:40:42 +0000 2017	Sat Feb 4 17:40:47 +0000 2017	5sec
ID	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time

Showing 1 to 2 of 2 entries

[3] This was actually run on my PC, running Hortonworks Sandbox on Oracle Virtual Box; the timestamps reflect the date I installed the Linux VM (4th Feb) rather than the time of processing (15th Feb). But the same information should now be visible on the DSM cluster now that Eamonn has enabled YARN log aggregation and job history.


@downinja.com - Ya
sandbox.hortonworks.co
sandbox.hortonworks.com:19888/jobhistory/logs/sandbox.hortonworks.com:45454/container_1486215819265_0015_01_000002/attempt_1486215819265_0015_m_000000

```
find -L . -maxdepth 5 -type l -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000002/directory.info"
exec /bin/bash -c "$JAVA_HOME/bin/java -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhdp.version=2.5.0.0-1245 -Xmx200m -Djava.io.tmpdir=$PWD/tmp -Dlog4j.con
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

Log Type: stderr
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 1067

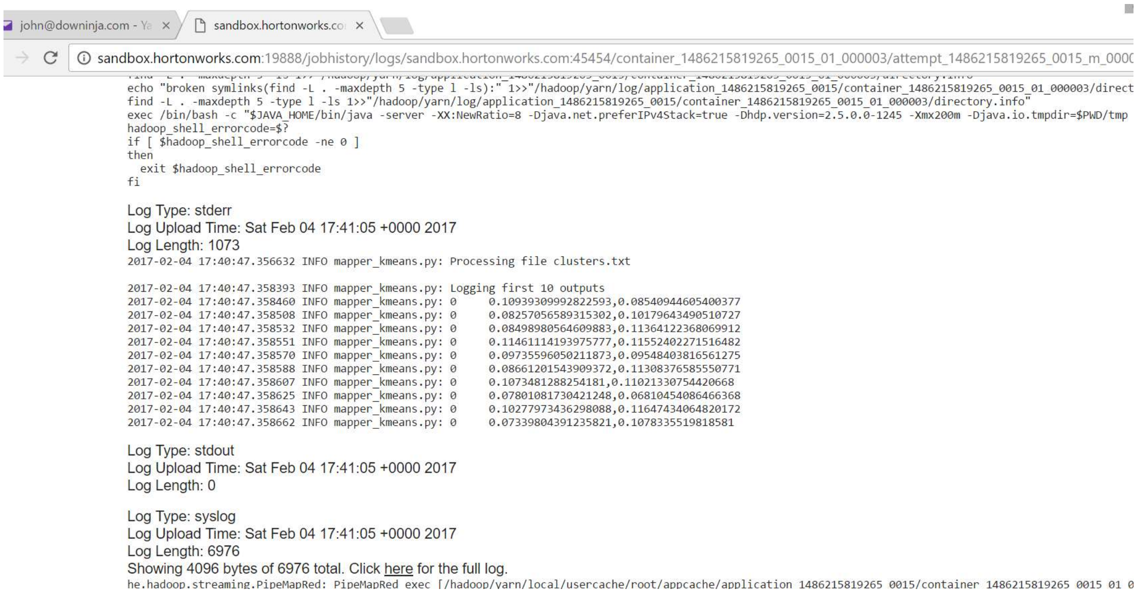
```
2017-02-04 17:40:47.356575 INFO mapper_kmeans.py: Processing file clusters.txt

2017-02-04 17:40:47.358254 INFO mapper_kmeans.py: Logging first 10 outputs
2017-02-04 17:40:47.358298 INFO mapper_kmeans.py: 0     0.3598965154681066,0.27933646264854767
2017-02-04 17:40:47.358341 INFO mapper_kmeans.py: 0     0.16747857120700604,0.18080395872256183
2017-02-04 17:40:47.358365 INFO mapper_kmeans.py: 0     0.2971436755154356,0.1785881112740101
2017-02-04 17:40:47.358385 INFO mapper_kmeans.py: 0     0.4265086206485411,0.2570112112049665
2017-02-04 17:40:47.358432 INFO mapper_kmeans.py: 0     0.21204410133286045,0.15535520650558493
2017-02-04 17:40:47.358491 INFO mapper_kmeans.py: 0     0.3466204995096275,0.30086148653689787
2017-02-04 17:40:47.358520 INFO mapper_kmeans.py: 0     0.30360676277912846,0.17830465777753907
2017-02-04 17:40:47.358539 INFO mapper_kmeans.py: 0     0.39243998141899017,0.25601977747402616
2017-02-04 17:40:47.358558 INFO mapper_kmeans.py: 0     0.2549626614638548,0.2819239003513696
2017-02-04 17:40:47.358577 INFO mapper_kmeans.py: 0     0.12032358975397317,0.22277431378265106
```

Log Type: stdout
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 0

Log Type: syslog
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 6979
Showing 4096 bytes of 6979 total. Click here for the full log.

```
he.hadoop.streaming.PipeMapRed: PipeMapRed exec [/hadoop/yarn/local/usercache/root/appcache/application_1486215819265_0015/container_1486215819265_0015_01_000002/./mapp
2017-02-04 17:40:47,234 INFO [main] org.apache.hadoop.conf.Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
2017-02-04 17:40:47,234 INFO [main] org.apache.hadoop.conf.Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
```


john@downinja.com - Ya
sandbox.hortonworks.co
sandbox.hortonworks.com:19888/jobhistory/logs/sandbox.hortonworks.com:45454/container_1486215819265_0015_01_000003/attempt_1486215819265_0015_m_0000

```
echo "broken symlinks(find -L . -maxdepth 5 -type l -ls):" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000003/direct
find -L . -maxdepth 5 -type l -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000003/directory.info"
exec /bin/bash -c "$JAVA_HOME/bin/java -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhdp.version=2.5.0.0-1245 -Xmx200m -Djava.io.tmpdir=$PWD/tmp
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

Log Type: stderr
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 1073

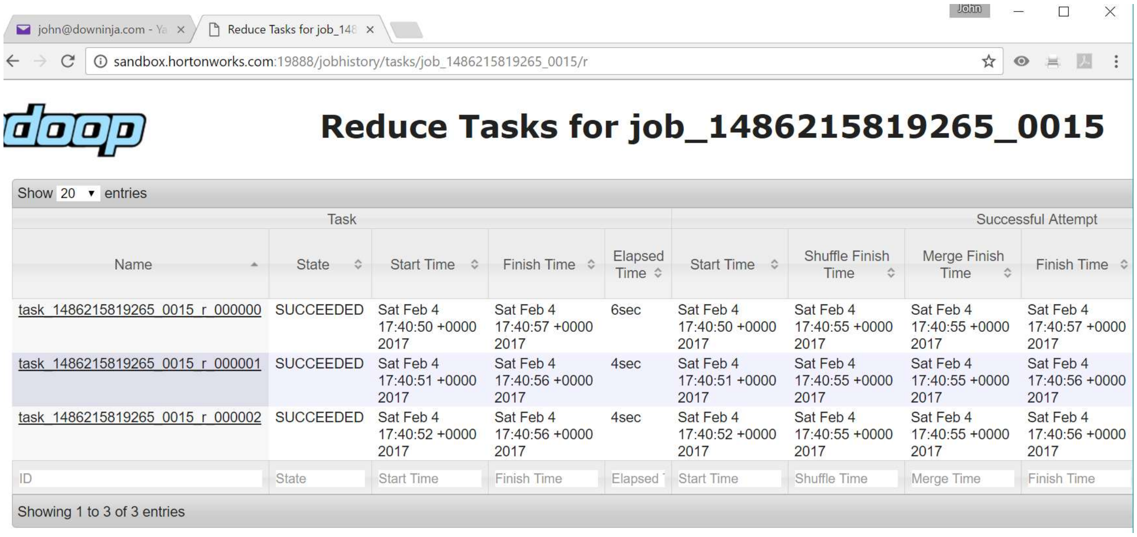
```
2017-02-04 17:40:47.356632 INFO mapper_kmeans.py: Processing file clusters.txt

2017-02-04 17:40:47.358393 INFO mapper_kmeans.py: Logging first 10 outputs
2017-02-04 17:40:47.358460 INFO mapper_kmeans.py: 0     0.10939309992822593,0.08540944605400377
2017-02-04 17:40:47.358508 INFO mapper_kmeans.py: 0     0.08257056589315302,0.10179643490510727
2017-02-04 17:40:47.358532 INFO mapper_kmeans.py: 0     0.08498980564609883,0.11364122368069912
2017-02-04 17:40:47.358551 INFO mapper_kmeans.py: 0     0.11461114193975777,0.11552402271516482
2017-02-04 17:40:47.358570 INFO mapper_kmeans.py: 0     0.09735596050211873,0.09548403816561275
2017-02-04 17:40:47.358588 INFO mapper_kmeans.py: 0     0.08661201543909372,0.11308376585550771
2017-02-04 17:40:47.358607 INFO mapper_kmeans.py: 0     0.1073481288254181,0.11021330754420668
2017-02-04 17:40:47.358625 INFO mapper_kmeans.py: 0     0.07801081730421248,0.06810454086466368
2017-02-04 17:40:47.358643 INFO mapper_kmeans.py: 0     0.10277973436298088,0.11647434064820172
2017-02-04 17:40:47.358662 INFO mapper_kmeans.py: 0     0.07339804391235821,0.1078335519818581
```

Log Type: stdout
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 0

Log Type: syslog
Log Upload Time: Sat Feb 04 17:41:05 +0000 2017
Log Length: 6976
Showing 4096 bytes of 6976 total. Click here for the full log.

```
he.hadoop.streaming.PipeMapRed: PipeMapRed exec [/hadoop/yarn/local/usercache/root/appcache/application_1486215819265_0015/container_1486215819265_0015_01_0
```



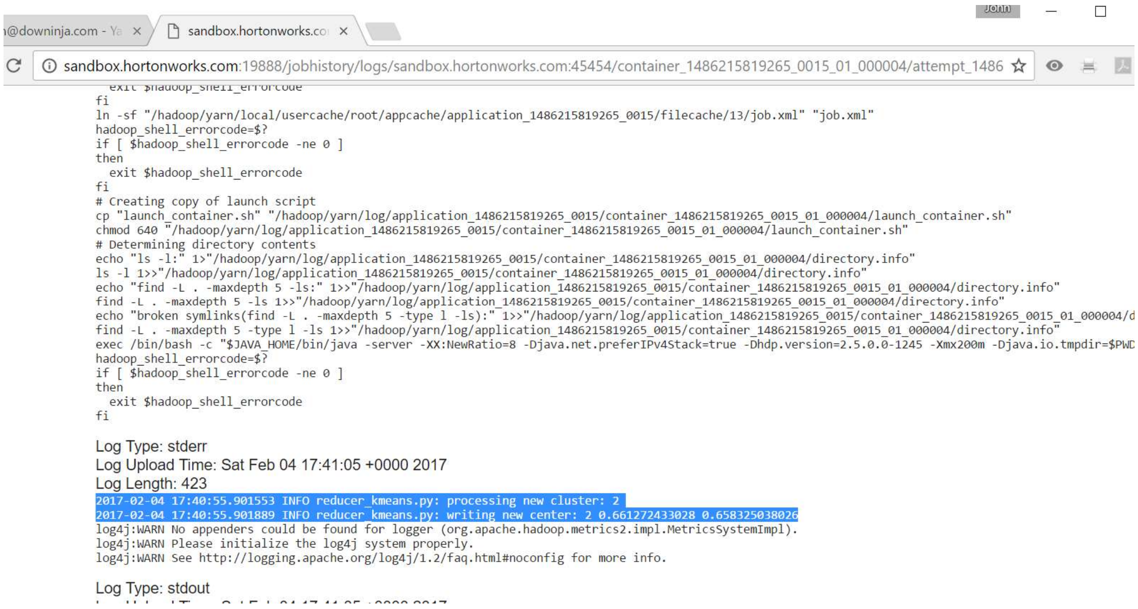
john@downinja.com - Ya
Reduce Tasks for job_148
John
sandbox.hortonworks.com:19888/jobhistory/tasks/job_1486215819265_0015/r
doop

Reduce Tasks for job_1486215819265_0015

Show 20 entries

Task					Successful Attempt			
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Shuffle Finish Time	Merge Finish Time	Finish Time
task_1486215819265_0015_r_000000	SUCCEEDED	Sat Feb 4 17:40:50 +0000 2017	Sat Feb 4 17:40:57 +0000 2017	6sec	Sat Feb 4 17:40:50 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:57 +0000 2017
task_1486215819265_0015_r_000001	SUCCEEDED	Sat Feb 4 17:40:51 +0000 2017	Sat Feb 4 17:40:56 +0000 2017	4sec	Sat Feb 4 17:40:51 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:56 +0000 2017
task_1486215819265_0015_r_000002	SUCCEEDED	Sat Feb 4 17:40:52 +0000 2017	Sat Feb 4 17:40:56 +0000 2017	4sec	Sat Feb 4 17:40:52 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:55 +0000 2017	Sat Feb 4 17:40:56 +0000 2017
ID	State	Start Time	Finish Time	Elapsed	Start Time	Shuffle Time	Merge Time	Finish Time

Showing 1 to 3 of 3 entries



n@downinja.com - Ya
sandbox.hortonworks.co
John
sandbox.hortonworks.com:19888/jobhistory/logs/sandbox.hortonworks.com:45454/container_1486215819265_0015_01_000004/attempt_1486

```
  exit $hadoop_shell_errorcode
fi
ln -sf "/hadoop/yarn/local/usercache/root/appcache/application_1486215819265_0015/filecache/13/job.xml" "job.xml"
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
# Creating copy of launch script
cp "launch_container.sh" "/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/launch_container.sh"
chmod 640 "/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/launch_container.sh"
# Determining directory contents
echo "ls -l:" 1>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/directory.info"
ls -l 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/directory.info"
echo "find -L . -maxdepth 5 -ls:" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/directory.info"
find -L . -maxdepth 5 -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/directory.info"
echo "broken symlinks(find -L . -maxdepth 5 -type l -ls):" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/d
find -L . -maxdepth 5 -type l -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000004/directory.info"
exec /bin/bash -c "$JAVA_HOME/bin/java -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhdp.version=2.5.0.0-1245 -Xmx200m -Djava.io.tmpdir=$PWD
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

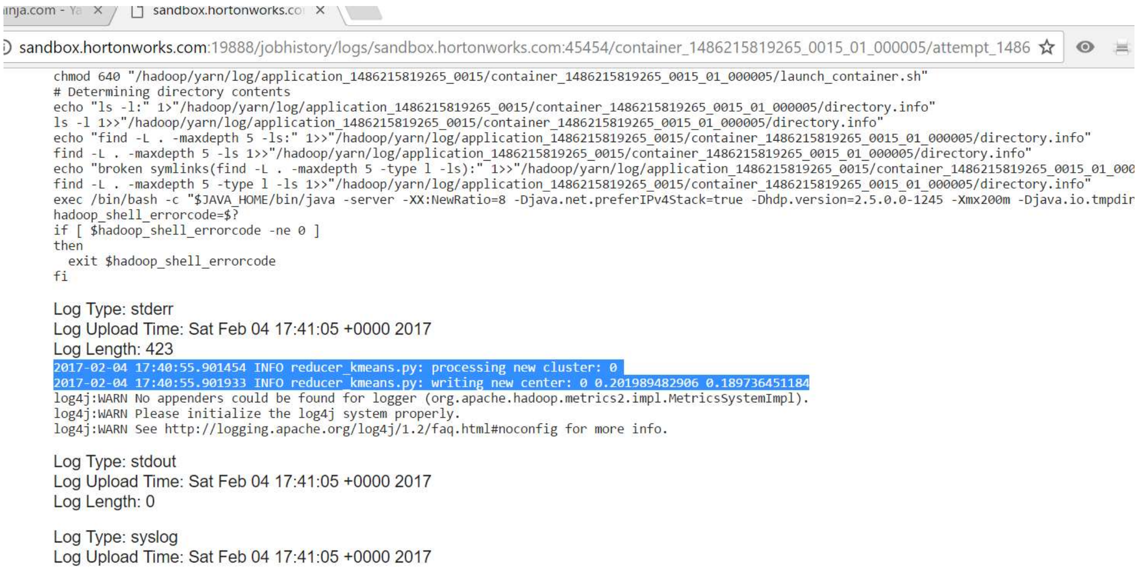
Log Type: stderr

Log Upload Time: Sat Feb 04 17:41:05 +0000 2017

Log Length: 423

```
2017-02-04 17:40:55.901553 INFO reducer_kmeans.py: processing new cluster: 2
2017-02-04 17:40:55.901889 INFO reducer_kmeans.py: writing new center: 2 0.661272433028 0.658325038026
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.impl.MetricsSystemImpl).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Log Type: stdout



sandbox.hortonworks.com:19888/jobhistory/logs/sandbox.hortonworks.com:45454/container_1486215819265_0015_01_000005/attempt_1486

```
chmod 640 "/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/launch_container.sh"
# Determining directory contents
echo "ls -l:" 1>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/directory.info"
ls -l 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/directory.info"
echo "find -L . -maxdepth 5 -ls:" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/directory.info"
find -L . -maxdepth 5 -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/directory.info"
echo "broken symlinks(find -L . -maxdepth 5 -type l -ls):" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000
find -L . -maxdepth 5 -type l -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000005/directory.info"
exec /bin/bash -c "$JAVA_HOME/bin/java -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhdp.version=2.5.0.0-1245 -Xmx200m -Djava.io.tmpdir
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

Log Type: stderr

Log Upload Time: Sat Feb 04 17:41:05 +0000 2017

Log Length: 423

```
2017-02-04 17:40:55.901454 INFO reducer_kmeans.py: processing new cluster: 0
2017-02-04 17:40:55.901933 INFO reducer_kmeans.py: writing new center: 0 0.201989482906 0.189736451184
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.impl.MetricsSystemImpl).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

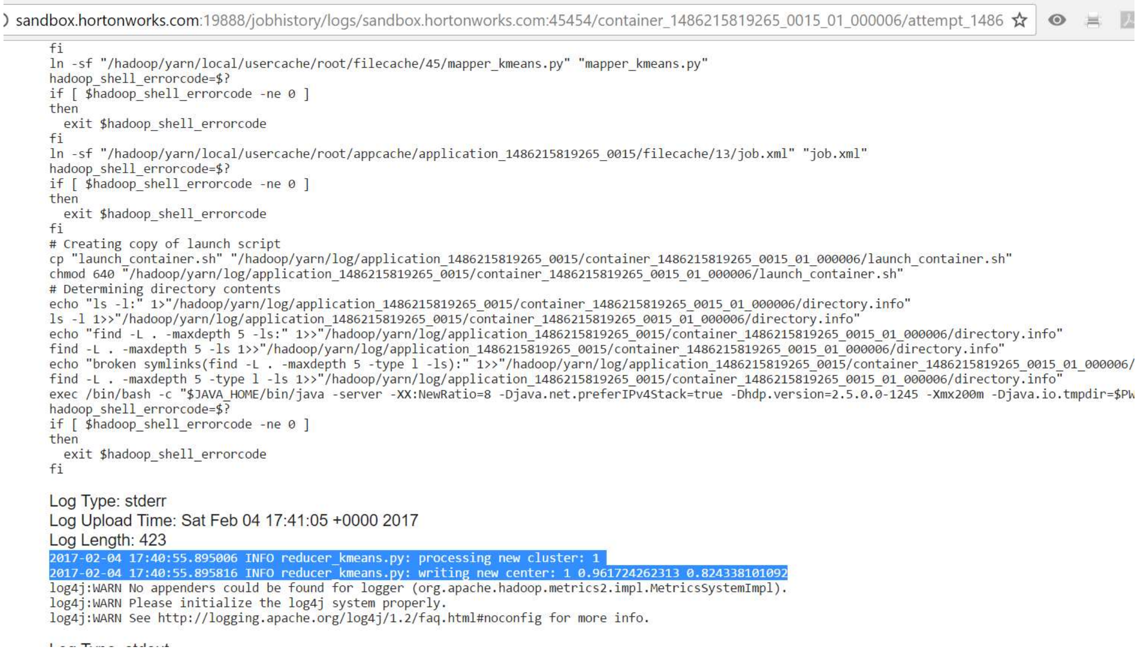
Log Type: stdout

Log Upload Time: Sat Feb 04 17:41:05 +0000 2017

Log Length: 0

Log Type: syslog

Log Upload Time: Sat Feb 04 17:41:05 +0000 2017



sandbox.hortonworks.com:19888/jobhistory/logs/sandbox.hortonworks.com:45454/container_1486215819265_0015_01_000006/attempt_1486

```
fi
ln -sf "/hadoop/yarn/local/usercache/root/filecache/45/mapper_kmeans.py" "mapper_kmeans.py"
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
ln -sf "/hadoop/yarn/local/usercache/root/appcache/application_1486215819265_0015/filecache/13/job.xml" "job.xml"
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
# Creating copy of launch script
cp "launch_container.sh" "/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/launch_container.sh"
chmod 640 "/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/launch_container.sh"
# Determining directory contents
echo "ls -l:" 1>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/directory.info"
ls -l 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/directory.info"
echo "find -L . -maxdepth 5 -ls:" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/directory.info"
find -L . -maxdepth 5 -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/directory.info"
echo "broken symlinks(find -L . -maxdepth 5 -type l -ls):" 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/
find -L . -maxdepth 5 -type l -ls 1>>"/hadoop/yarn/log/application_1486215819265_0015/container_1486215819265_0015_01_000006/directory.info"
exec /bin/bash -c "$JAVA_HOME/bin/java -server -XX:NewRatio=8 -Djava.net.preferIPv4Stack=true -Dhdp.version=2.5.0.0-1245 -Xmx200m -Djava.io.tmpdir=$PW
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

Log Type: stderr

Log Upload Time: Sat Feb 04 17:41:05 +0000 2017

Log Length: 423

```
2017-02-04 17:40:55.895006 INFO reducer_kmeans.py: processing new cluster: 1
2017-02-04 17:40:55.895816 INFO reducer_kmeans.py: writing new center: 1 0.961724262313 0.824338101092
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.impl.MetricsSystemImpl).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Log Type: stdout

EM

Files

The following files are used in the EM implementation.

- `mapper_em.py`

  This uses scipy.stats.multivariate_normal to calculate the likelihood that a datapoint belongs to each of the three clusters, based on the PDF of each. It outputs three key/value pairs for each datapoint, corresponding to the likelihood of that datapoint belonging to each of the three clusters.
- `reducer_em.py`

  This takes the likelihoods that each datapoint belongs to a given cluster, and uses them to recalculate the cluster parameters. These parameters are the cluster centre (coordinates), its size and orientation (covariance matrix) and its weighting (e.g. what proportion of the total datapoints likely belong to it).
- `em_util.py`

  This contains functions which are shared by the mapper and reducer files, e.g. for logging to stderr and for loading and parsing data.
- `plot_em.py`

  This is a supplementary script used to plot the resulting cluster centres and datapoints once they have converged. In theory, it could be run as part of the bash wrapper script (below), however we don't seem to have Python graphics enabled on the DSM cluster.
- `em.mapred.sh`

  This is a bash script which wraps iterations of the MapReduce process in a loop, invoking an Hadoop streaming job at each stage and feeding the output from the previous stage into the next one. The script exists after 100 iterations, or if the output from the reducer does not change from the previous stage.
- `data.txt`

  This is the file of datapoints provided with the assignment.
- `clusters_em.txt`

  This is the file of initial cluster centres provided with the assignment – only with added cluster weights and covariance matrices. Initial weights and matrices are 1/3 and [[3 0; 0 3]][4] in each case.

[4] Following the example at https://www.coursera.org/learn/ml-clustering-and-retrieval/supplement/s9OBQ/optional-a-worked-out-example-for-em

How to run

The command for a single invocation of EM MapReduce on the DSM cluster is as follows. This requires the Python 3 environment[5] as the scipy library is not installed under Python 2. The input (data.txt) file should be copied to HDFS before running the script, but all other files are provided as part of the configuration. Three reducers are specified, one for each cluster.

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.0.jar \
-Dmapred.reduce.tasks=3 \
-files=em_util.py,mapper_em.py,reducer_em.py,clusters_em.txt \
-cmdenv clusters=clusters_em.txt \
-mapper  'scl enable rh-python34 ./mapper_em.py' \
-reducer 'scl enable rh-python34 ./reducer_em.py' \
-input data.txt \
-output clusters_em
```

However, rather than repeat this process manually until convergence, invoking the `em.mapred.sh` script does the following:

```
#!/bin/bash

for i in {1..100}
do
       newclusters="clusters_em.mapred.txt.$i"
       clusters="clusters_em.mapred.txt.$((i-1))"
       if [ $i == 1 ]; then
              clusters="clusters_em.txt"
       fi
       echo "iteration $i"
       echo "------------"

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.0.jar \
-Dmapred.reduce.tasks=3 \
-files=em_util.py,mapper_em.py,reducer_em.py,$clusters \
-cmdenv clusters=$clusters \
-mapper  'scl enable rh-python34 ./mapper_em.py' \
-reducer  'scl enable rh-python34 ./reducer_em.py' \
-input data.txt \
-output clusters_em

       hadoop fs -cat clusters_em/part-* | sort > $newclusters
       hadoop fs -rm -r clusters_em
       diff_val=$(diff $clusters $newclusters)
       if  [ -z "$diff_val" ]; then
              echo "Exiting loop as clusters have not changed this iteration"
              break
       fi
done
```

[5] Turns out we can use the guts of enable-python3 to invoke it within MapReduce – specifying e.g. `-mapper 'scl enable rh-python34 ./mapper_em.py'` as the executable. We can't use enable-python3 directly, as that forks a new bash shell.

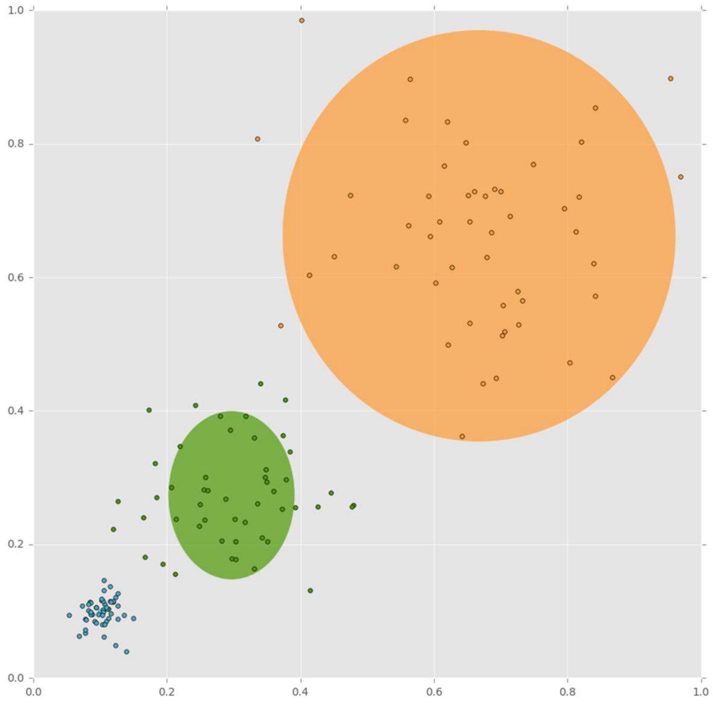
Results

The clusters converged in 65 iterations, with the final parameters given below.

ID	CX	CY	COVAR	WEIGHT
0	0.1028	0.0988	0.00036 3.25527e-05 3.25527e-05 0.00042	16.5938
1	0.6676	0.6621	0.01851 -0.00145 -0.00145 0.01768	16.6545
2	0.2970	0.2736	0.00754 0.00059 0.00059 0.00566	16.7517

The plot below shows both the datapoints and also the shape & locations of the final clusters[6].

Datapoints are coloured by the cluster which they most likely belong to.



[6] I've based the size and orientation of each cluster (ellipse) on its covariance matrix, using the diagonal values for width/height and the off-diagonals for angle. Seems to work, although I'm not sure if this is correct (?)

Discussion

The EM clusters seem – at least visually – to be a better fit to the data. EM has picked out the small, densely populated, cloud of datapoints that lie between the 0.0 and 0.2 on both axes – whereas KMeans has grouped these together with the more diffuse cloud between 0.2 and 0.4. KMeans has also split the datapoints towards the top-right of the space into two clusters, when there doesn't seem to be any natural division there. This is most likely due to the implicit constraint on KMeans that all clusters should be the same shape and size[7].

However, EM is more resource intensive; for every datapoint, multiple key/value pairs are output, rather than a single output for KMeans. Also, the algorithm takes longer to converge. In addition, the reduce step has to loop over datapoints twice; once to calculate the mean coordinates, and once more to calculate the covariance matrix based on the difference of each point from this mean. More critically, this last step requires all datapoints for a cluster to be held in memory – which is unlikely to scale well to big data. KMeans, being a simpler model, can just keep a running total of x and y coordinates, and divide through to get the mean in a single pass.

Ultimately, it could be argued that neither EM nor KMeans are a particularly natural fit for MapReduce. Although they can be split into data-parallel steps, this benefit is only fully realised – at least on the reduce side – if there is a reasonably even distribution of datapoints between clusters (and by extension, a reasonably even distribution of load between reducers). Also, their iterative natures might be better supported by something like Apache Spark, which should allow the clusters data to be cached in memory - rather than having to output it to HDFS and then read it back in again as part of a manually invoked streaming job[8].

[7] Well ok, in KMeans clusters are just points e.g. the centre coordinates. But because cluster assignment is based solely on Euclidean distance, their regions of influence are in each case the same sized sphere (ref: https://www.coursera.org/learn/ml-clustering-and-retrieval/lecture/I6FYH/motiving-probabilistic-clustering-models).

[8] e.g. http://research.ijcaonline.org/volume113/number1/pxc3900531.pdf