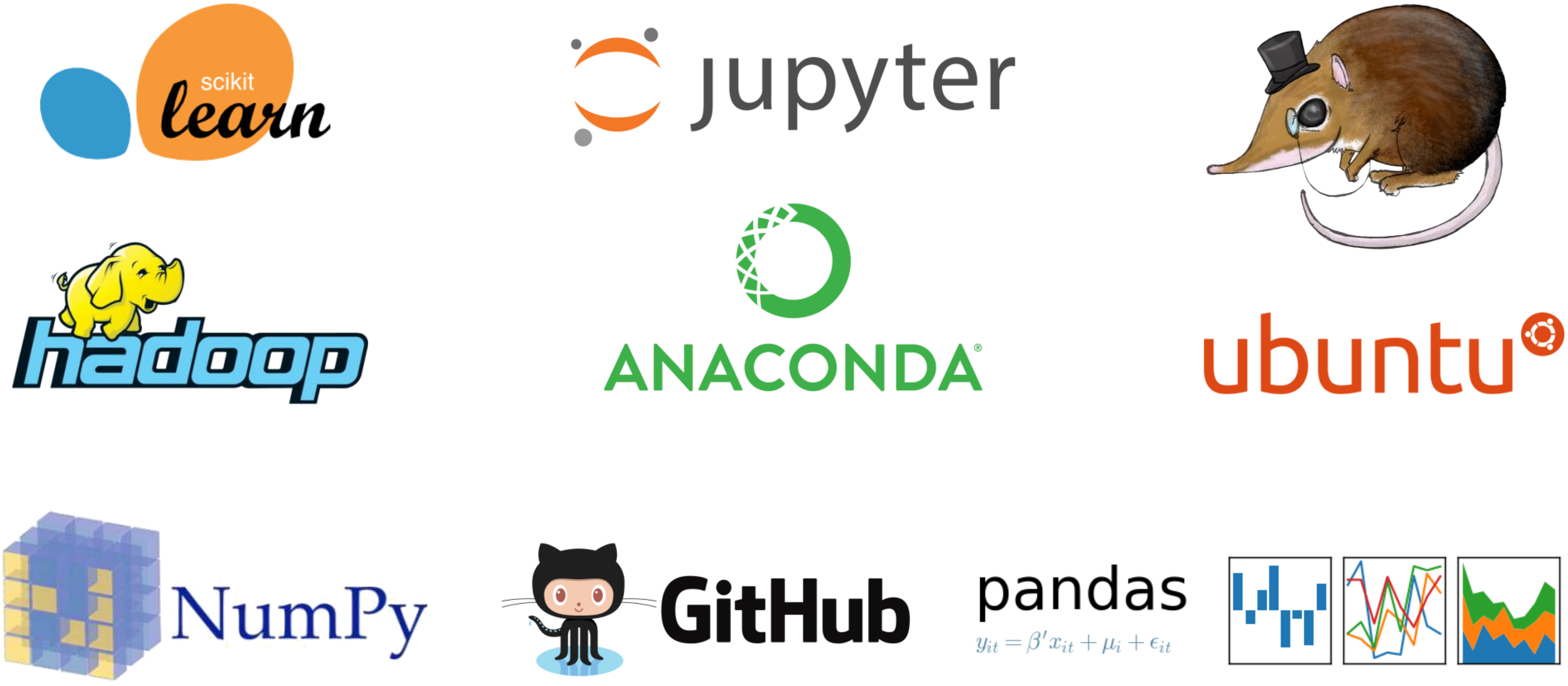
k-means algorithm implementation on Hadoop

Athens University of Economics and Business
Dpt. Of Management Science and Technology
Prof. Damianos Chatziantoniou

Lamprini Koutsokera (8130074) | lkoutsokera@gmail.com
Stratos Gounidellis (8130029) | stratos.gounidellis@gmail.com

BDSMasters

Tools


scikit learn
jupyter
hadoop
ANACONDA
ubuntu
NumPy
GitHub
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Running Hadoop on Ubuntu Linux (Single-Node Cluster) [1]

Prerequisites

Java

- jdk-8uversion-linux-x64.tar.gz (Download)
- Directory modification
- % tar zxvf jdk-8uversion-linux-x64.tar.gz

Hadoop system user

```
$ sudo addgroup hadoop $ sudo adduser --ingroup hadoop
hduser
```

Disabling IPv6

/etc/sysctl.conf

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

References: **[1], [2]**

Configuring SSH

Generate an SSH key for the hduser

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
$ sudo chmod 750 /app/hadoop/tmp

$ ssh-keygen -t rsa -P ""
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
$ ssh localhost
```

Create an RSA key pair with an empty password

```
hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys
```

Running Hadoop on Ubuntu Linux (Single-Node Cluster) [2]

Hadoop configuration

```
$ cd /usr/local
$ sudo tar xzf hadoop-2.7.3.tar.gz
$ sudo mv hadoop-2.7.3 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

hadoop-env.sh

```
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64“
```

Configuration

ownerships and permissions

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
$ sudo chmod 750 /app/hadoop/tmp
```

References: [1], [2]

conf/core-site.xml

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
</property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

conf/mapred-site.xml

```
<configuration>
  <property>
<name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Running Hadoop on Ubuntu Linux (Single-Node Cluster) [3]

yarn-site.xml

```
<configuration>
<property>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>128</value>
</property>
    <property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>2048</value>
</property>
    <property>
        <name>yarn.scheduler.minimum-allocation-vcores</name>
        <value>1</value>
</property>
    <property>
        <name>yarn.scheduler.maximum-allocation-vcores</name>
        <value>2</value>
</property>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>4096</value>
</property>
    <property>
        <name>yarn.nodemanager.resource.cpu-vcores</name>
        <value>4</value>
</property>
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
</property>
</configuration>
```

References: [1], [2]

hdfs-site.xml

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when
the file is created.
  The default is used if replication is not specified in create
time.
  </description>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary
directories.</description>
</property>
</configuration>
```

Running Hadoop on Ubuntu Linux (Single-Node Cluster) [4]

conf/hdfs-site.xml

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
</property>
</configuration>
```

Formatting the HDFS filesystem via the NameNode

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

Starting your single-node cluster

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

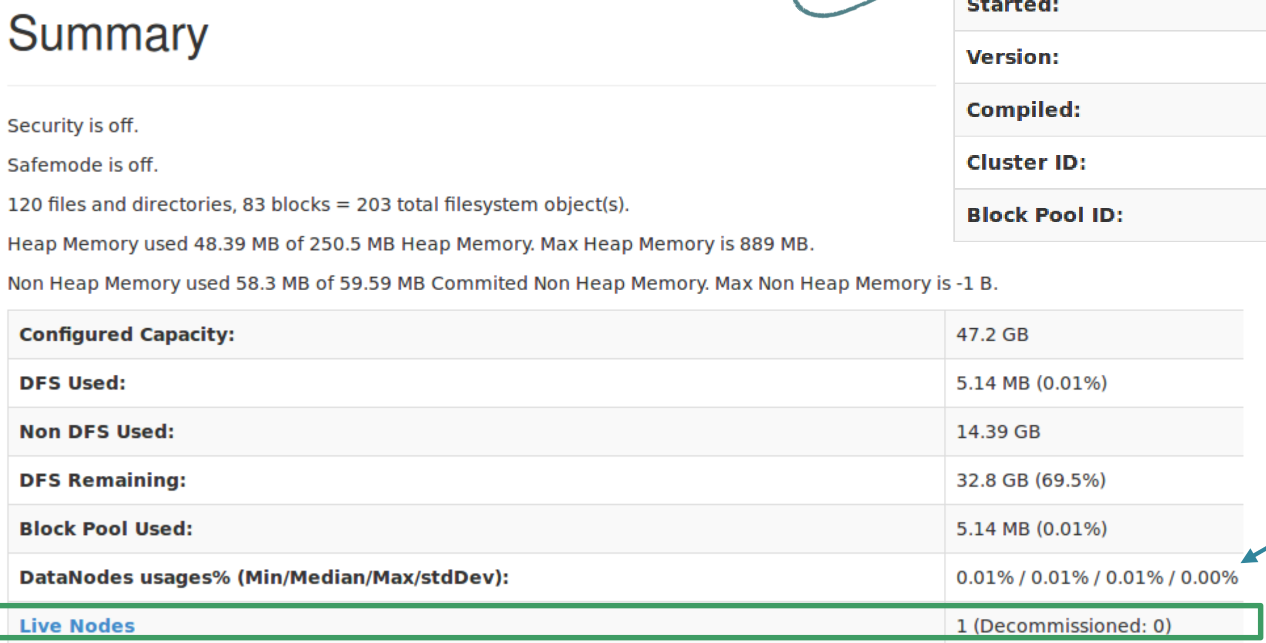
Stopping your single-node cluster

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

References: **[1], [2]**

Setting up a Single Node Cluster - *Hadoop Distributed File System (HDFS)*

*To access Hadoop WEB UI , we need to type http://**localhost:50070**/ though our **core-site.xml** that has as value the http://**localhost:9000**.*



Summary

Security is off.

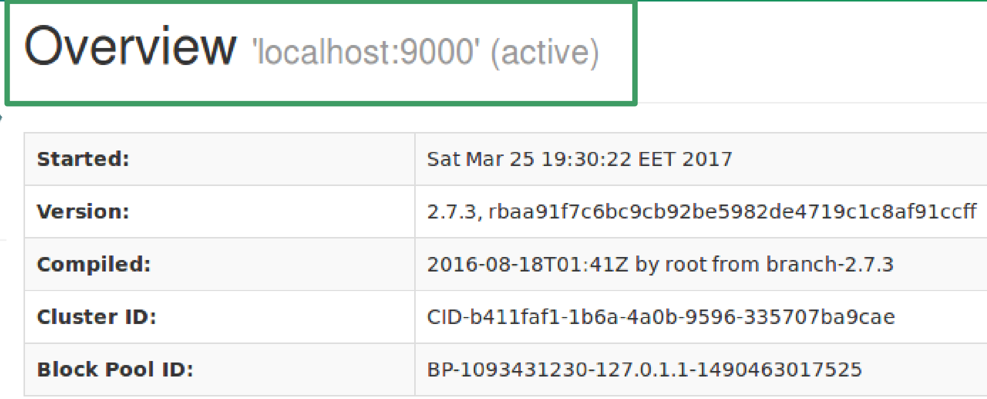
Safemode is off.

120 files and directories, 83 blocks = 203 total filesystem object(s).

Heap Memory used 48.39 MB of 250.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 58.3 MB of 59.59 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

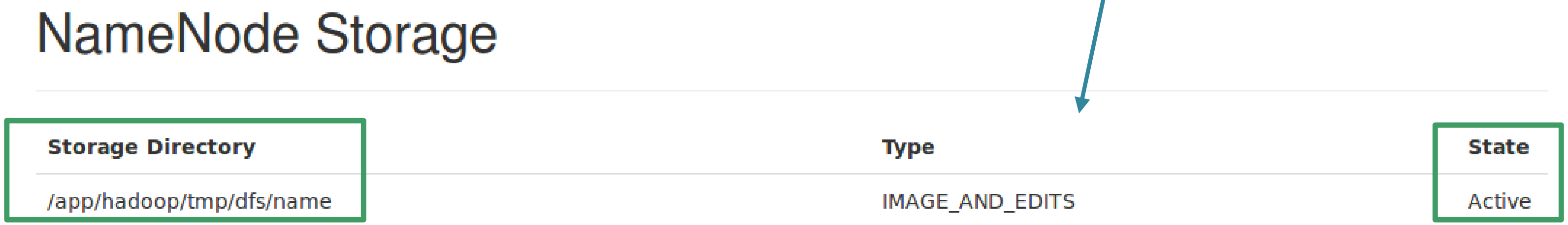
Configured Capacity:	47.2 GB
DFS Used:	5.14 MB (0.01%)
Non DFS Used:	14.39 GB
DFS Remaining:	32.8 GB (69.5%)
Block Pool Used:	5.14 MB (0.01%)
DataNodes usages% (Min/Median/Max/stdDev):	0.01% / 0.01% / 0.01% / 0.00%
Live Nodes	1 (Decommissioned: 0)



Overview 'localhost:9000' (active)

Started:	Sat Mar 25 19:30:22 EET 2017
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-b411faf1-1b6a-4a0b-9596-335707ba9cae
Block Pool ID:	BP-1093431230-127.0.1.1-1490463017525

*Our HDFS cluster consists of a single **NameNode**, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of **DataNodes**, usually one per node in the cluster, which manage storage attached to the nodes that they run on.*

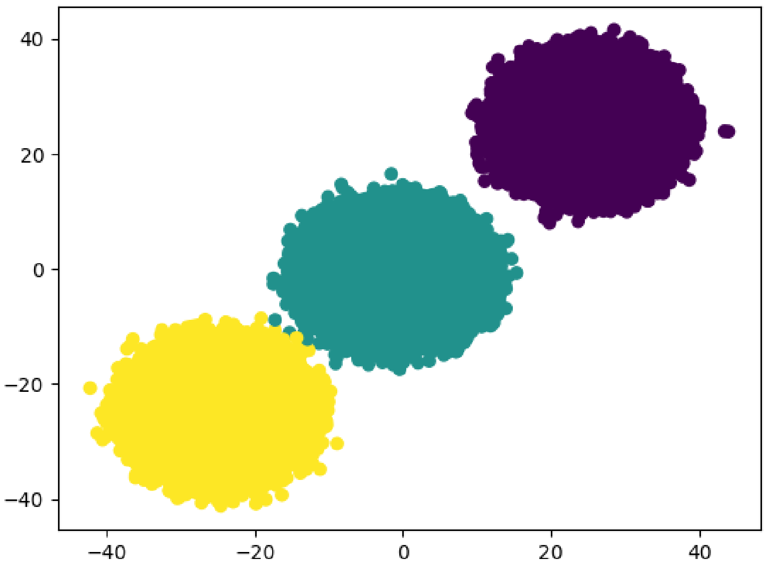


NameNode Storage

Storage Directory	Type	State
/app/hadoop/tmp/dfs/name	IMAGE_AND_EDITS	Active

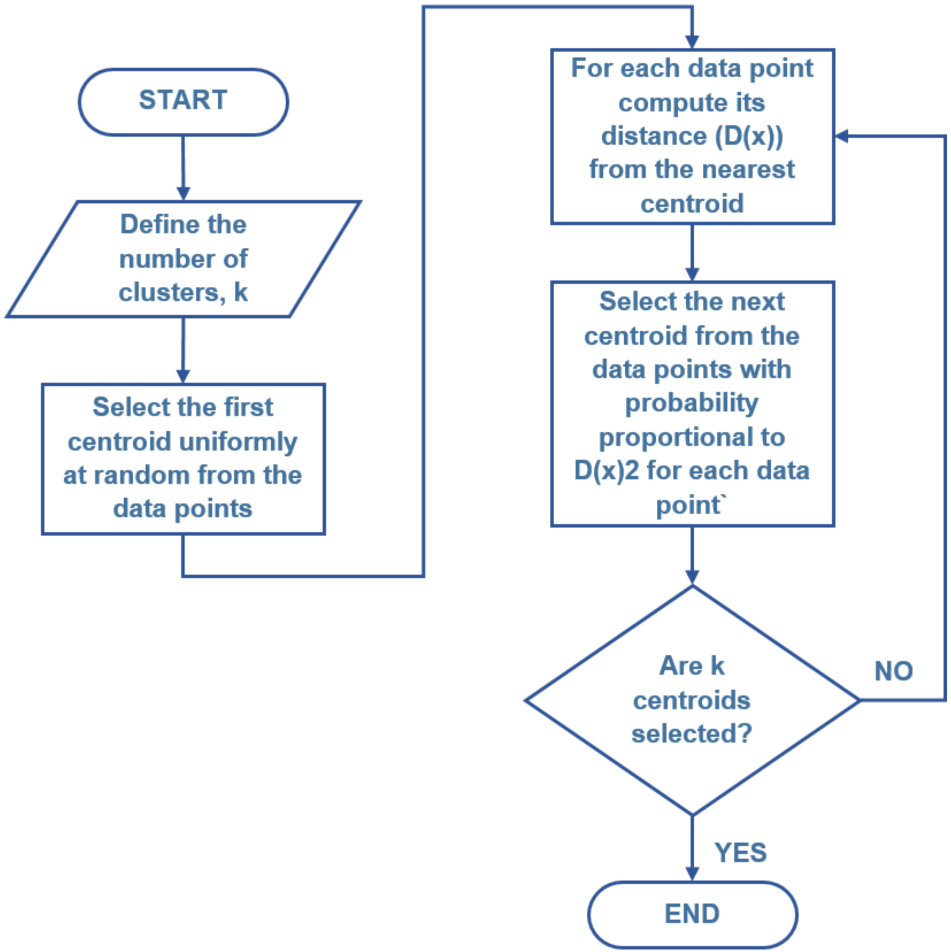
From **hardware/software** to **algorithms**

Data Generation



- **Isotropic Gaussian blobs**
- 2.000.000 **points**
- **centers** = [[25, 25], [-1, -1], [-25, -25]]
- **cluster_std** = 3.5

K-means++ : Calculation of initial centroids



START

Define the number of clusters, k

Select the first centroid uniformly at random from the data points

For each data point compute its distance (D(x)) from the nearest centroid

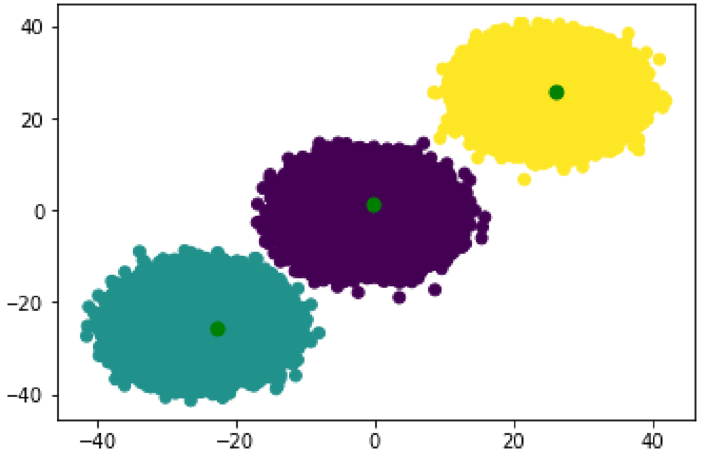
Select the next centroid from the data points with probability proportional to D(x)2 for each data point`

Are k centroids selected?

NO

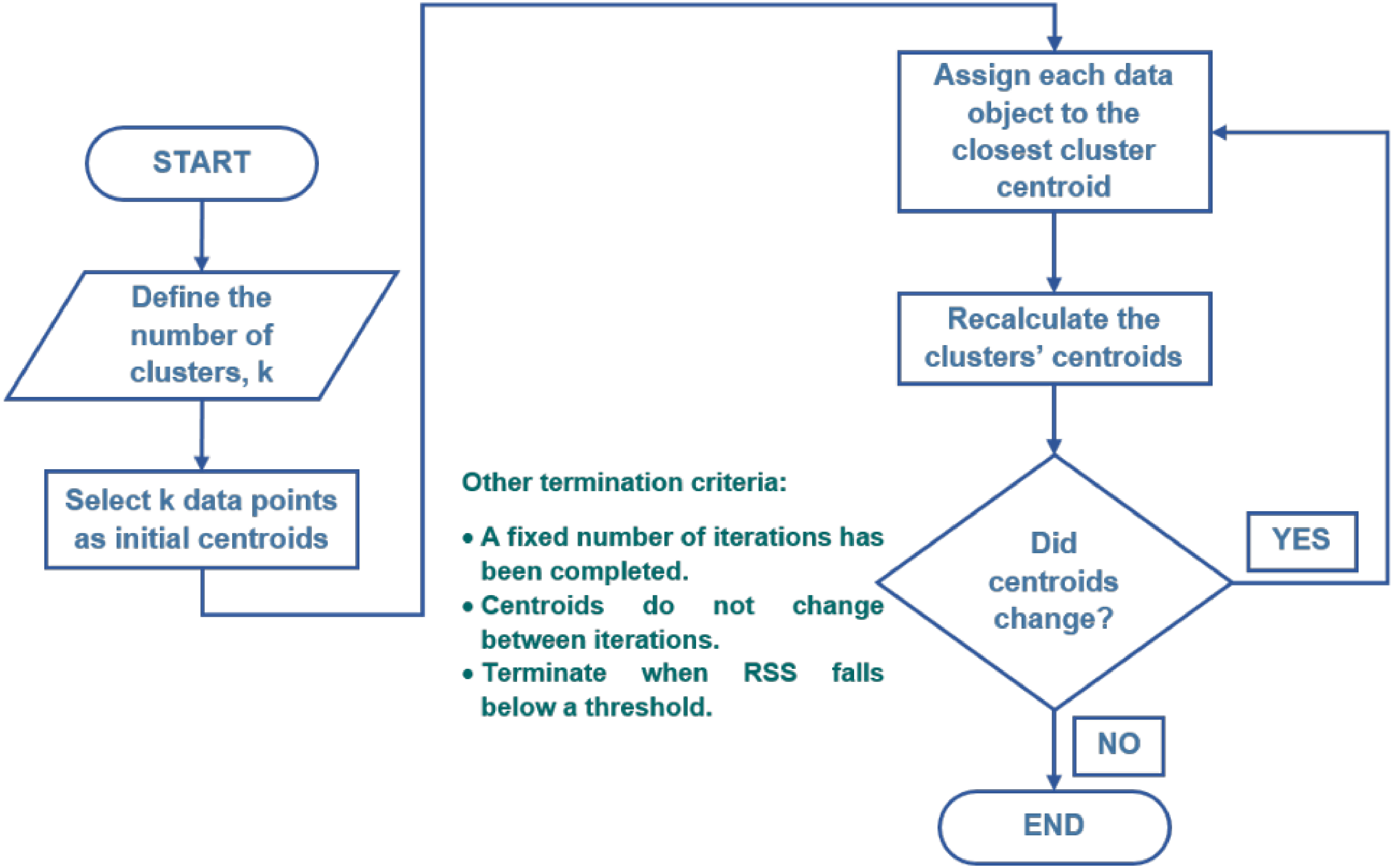
YES

END



Reference: [3]

K-means : Clustering algorithm


START
Define the number of clusters, k
Select k data points as initial centroids
Assign each data object to the closest cluster centroid
Recalculate the clusters' centroids
Did centroids change?
YES
NO
END

Other termination criteria:

- A fixed number of iterations has been completed.
- Centroids do not change between iterations.
- Terminate when RSS falls below a threshold.

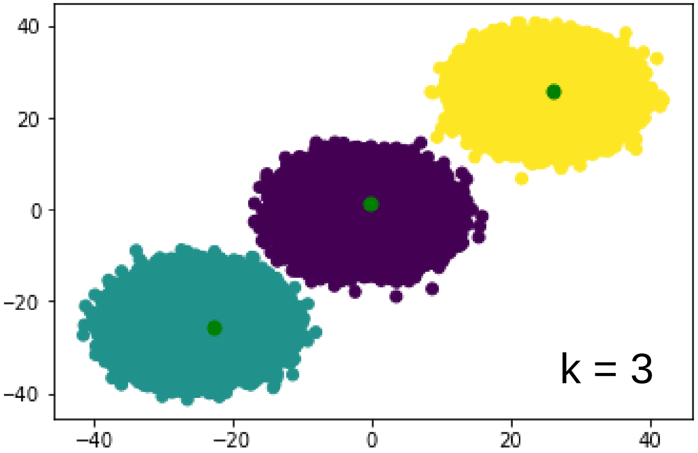
References: [4], [5]

K-means using Map Reduce

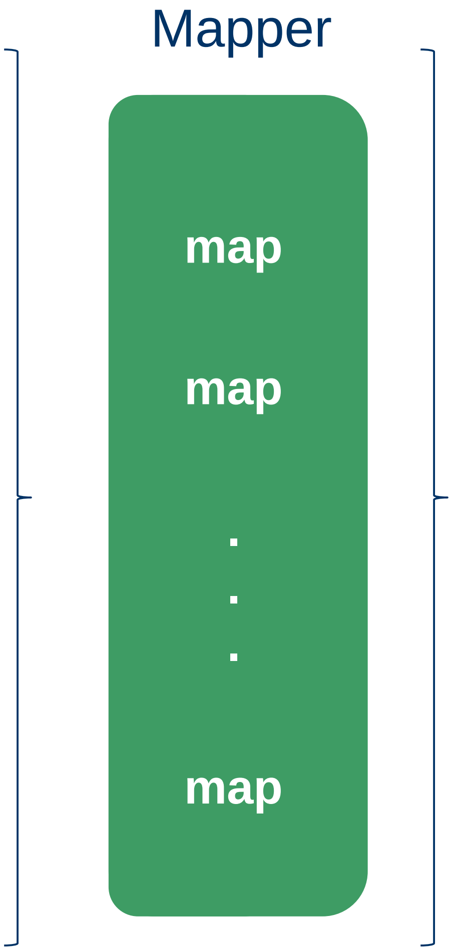
```
Do
    Map
        Input is a data point and k centers are broadcasted
        Finds the closest center among k centers for the input point

    Reduce
        Input is one of k centers and all data points having this center as their closest center.
        Calculates the new center using data points

Until all of new centers are not changed
```

Reference: **[6]**

K-means using Map Reduce [1]



Centroids are broadcasted to every map function



key value

[centroid1, point1]

[centroid2, point3]

[centroid3, point10]

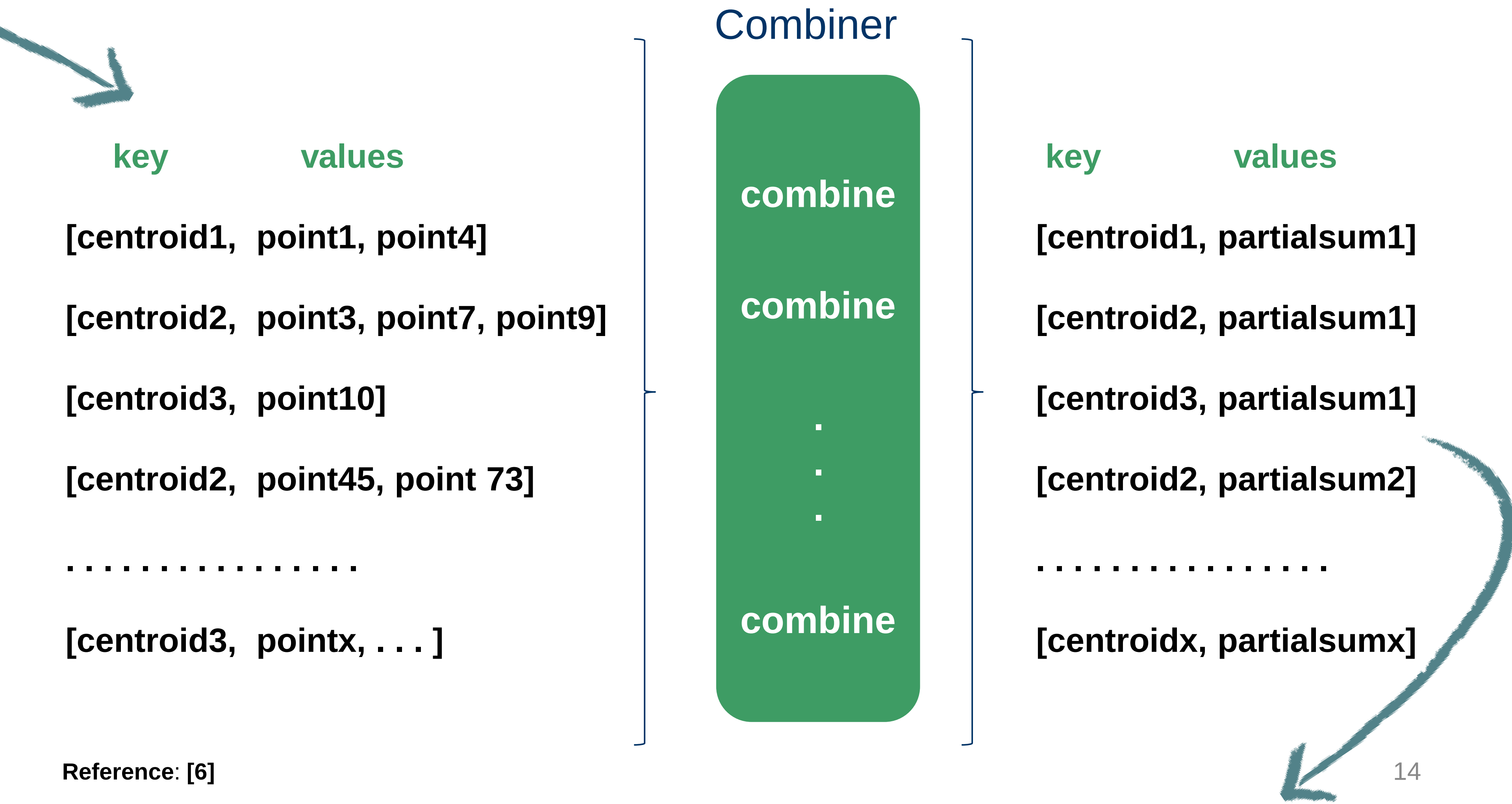
[centroid2, point45]

. . . .

[centroidx, pointx]

Reference: [6]

K-means using Map Reduce [2]



Combiner

key values

[centroid1, point1, point4]

[centroid2, point3, point7, point9]

[centroid3, point10]

[centroid2, point45, point 73]

. . . .

[centroid3, pointx, . . .]

key values

[centroid1, partialsum1]

[centroid2, partialsum1]

[centroid3, partialsum1]

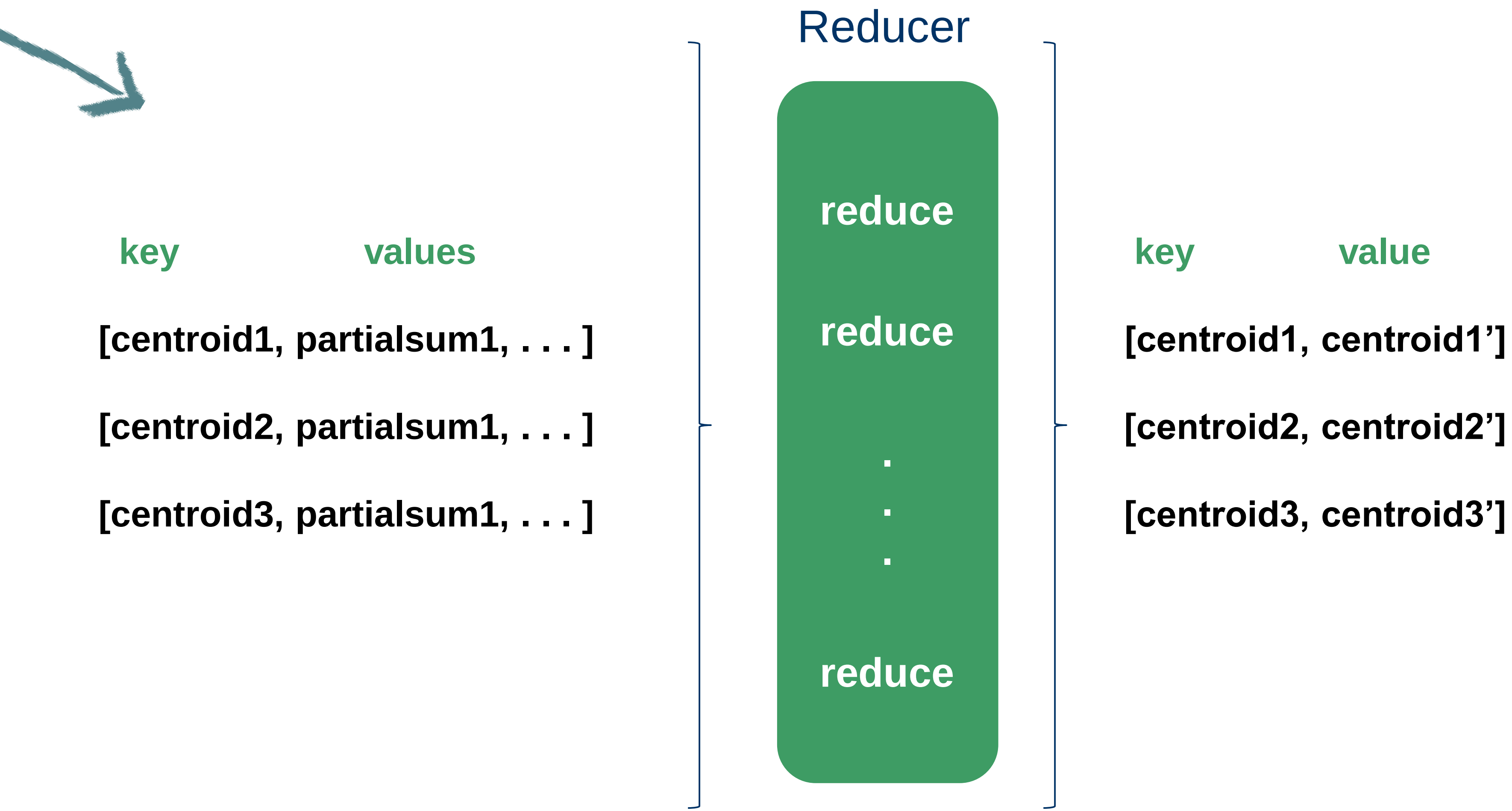
[centroid2, partialsum2]

. . . .

[centroidx, partialsumx]

Reference: [6]

K-means using Map Reduce [3]



Reducer

key values

[centroid1, partialsum1, . . .]

[centroid2, partialsum1, . . .]

[centroid3, partialsum1, . . .]

reduce

reduce

.
.
.

reduce

key value

[centroid1, centroid1’]

[centroid2, centroid2’]

[centroid3, centroid3’]

Reference: [6]

From **algorithms** to **coding**

Python Coding [1]

createDataPoints

The initial task of the project is to generate **a set of more than one million data points** to be used later as an input for the k-means clustering algorithm. Using this python script three **isotropic Gaussian blobs** for clustering are generated. More specifically, the centers are the following data points **[25, 25], [-1, -1], [-25, -25]**.

plotSilhouetteScore

The silhouette score constitutes a useful criterion for determining **the proper number of clusters**. A silhouette close to 1 implies the datum is in an appropriate cluster, while a silhouette close to −1 implies the datum is in the wrong cluster. The specific python script calculates the silhouette score for different numbers of clusters ranging from 2 to 6.

Python Coding [2]

kmeans

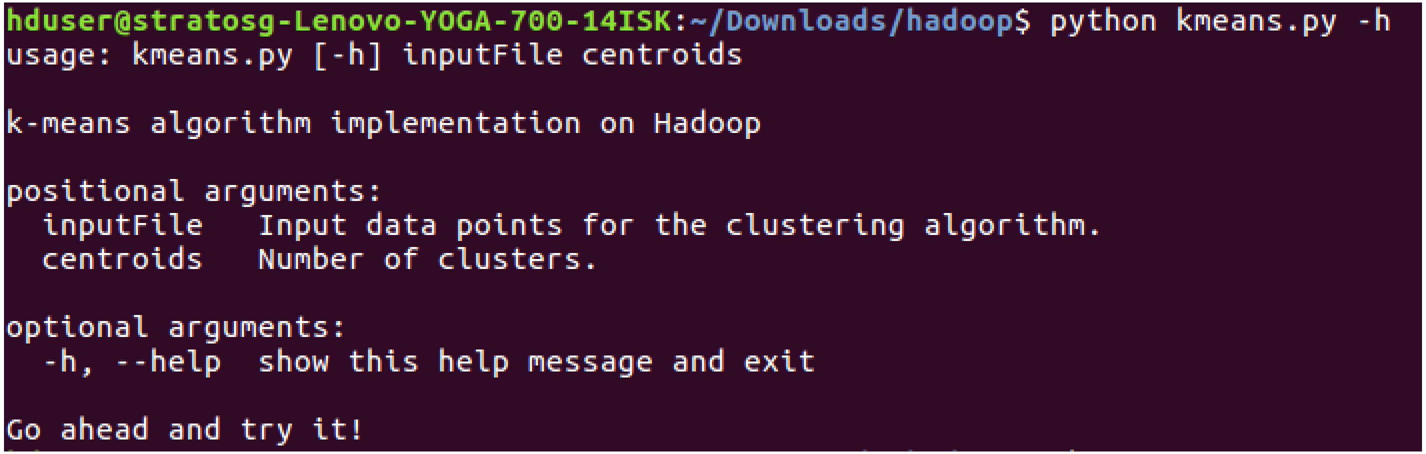
This python script calls the **k-means algorithm** implemented on hadoop. However, before implementing k-means the initial centroids are computed using the **k-means++ algorithm** proposed in 2007 by Arthur and Vassilvitskii.

kmeansAlgorithm

In order to implement k-means algorithm on hadoop **mrjob** is used. Mrjob is a python package, which allows to write multi-step MapReduce jobs in pure Python and run them on a hadoop cluster. In our case mrjob run on a single-node cluster.

- The mapper function returns each data point and the cluster, to which it belongs.
- The combiner function returns partial sums of batches of data points belonging to the same cluster.
- The reducer returns the new centroids of each cluster.
- If the centroids remain unchanged the algorithm terminates. Otherwise, the steps are repeated from the beginning.

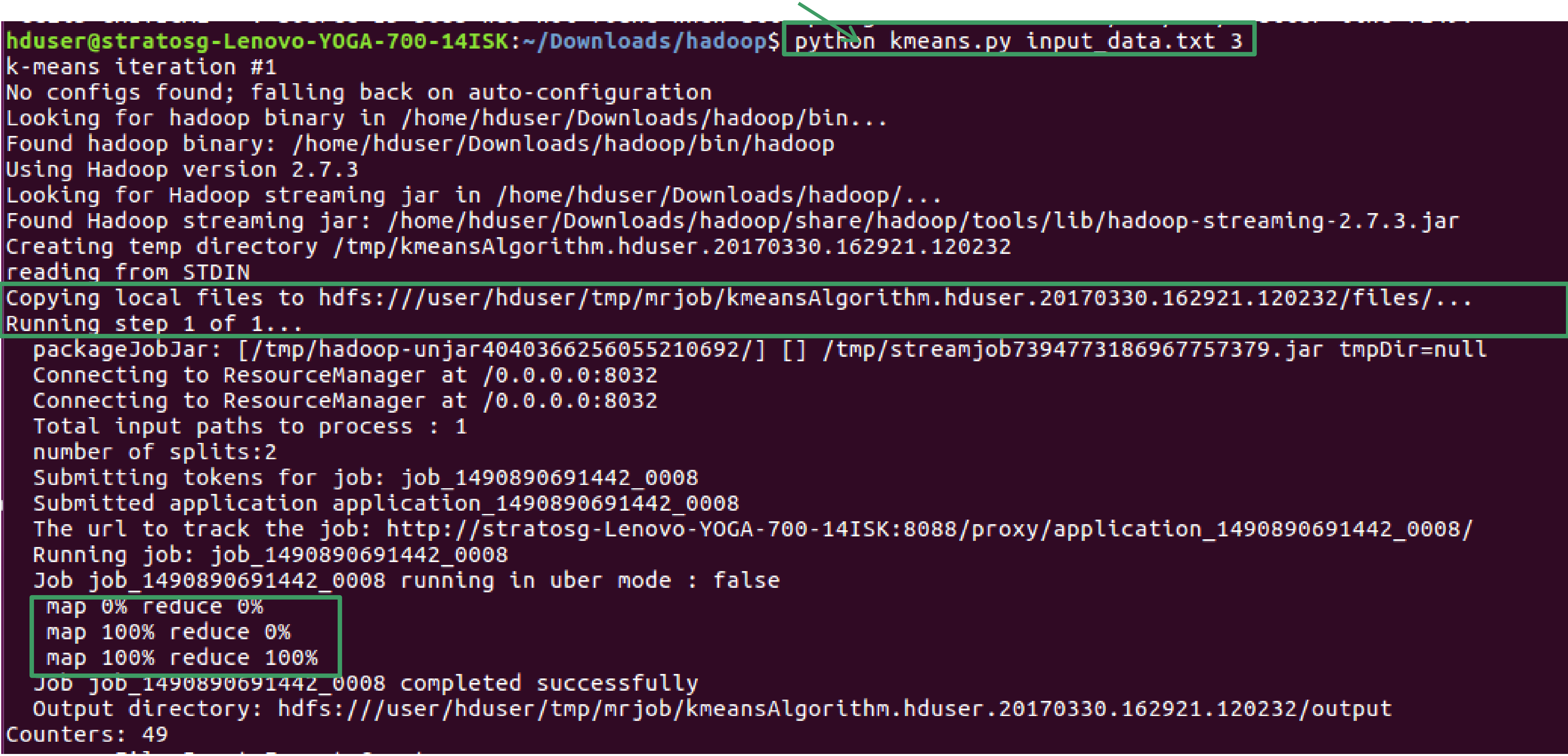
Coding Running [1]



```
hduser@stratosg-Lenovo-YOGA-700-14ISK:~/Downloads/hadoop$ python kmeans.py -h
usage: kmeans.py [-h] inputFile centroids

k-means algorithm implementation on Hadoop

positional arguments:
  inputFile   Input data points for the clustering algorithm.
  centroids   Number of clusters.

optional arguments:
  -h, --help  show this help message and exit

Go ahead and try it!
```

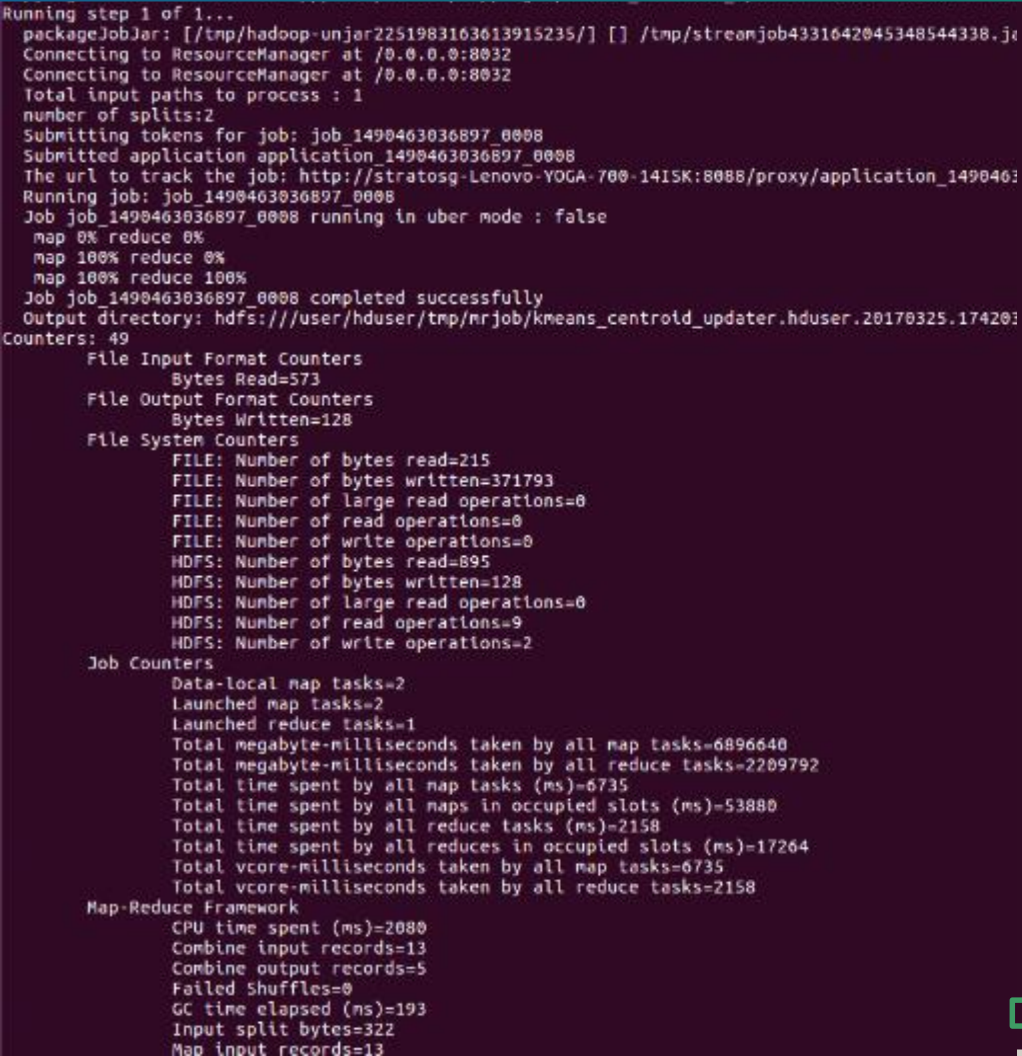
Coding Running [2]

kmeans

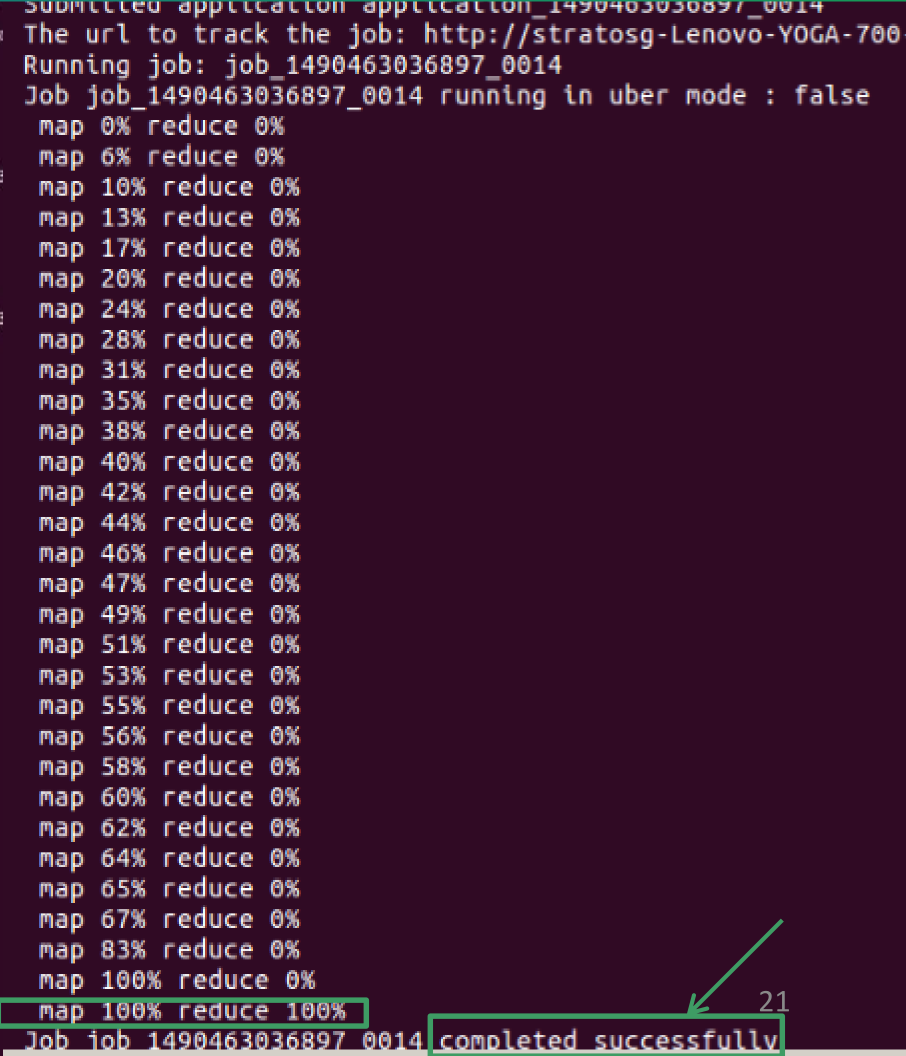


```
hduser@stratosg-Lenovo-YOGA-700-14ISK:~/Downloads/hadoop$ python kmeans.py input_data.txt 3
k-means iteration #1
No configs found; falling back on auto-configuration
Looking for hadoop binary in /home/hduser/Downloads/hadoop/bin...
Found hadoop binary: /home/hduser/Downloads/hadoop/bin/hadoop
Using Hadoop version 2.7.3
Looking for Hadoop streaming jar in /home/hduser/Downloads/hadoop/...
Found Hadoop streaming jar: /home/hduser/Downloads/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar
Creating temp directory /tmp/kmeansAlgorithm.hduser.20170330.162921.120232
reading from STDIN
Copying local files to hdfs:///user/hduser/tmp/mrjob/kmeansAlgorithm.hduser.20170330.162921.120232/files/...
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar4040366256055210692/] [] /tmp/streamjob7394773186967757379.jar tmpDir=null
  Connecting to ResourceManager at /0.0.0.0:8032
  Connecting to ResourceManager at /0.0.0.0:8032
  Total input paths to process : 1
  number of splits:2
  Submitting tokens for job: job_1490890691442_0008
  Submitted application application_1490890691442_0008
  The url to track the job: http://stratosg-Lenovo-YOGA-700-14ISK:8088/proxy/application_1490890691442_0008/
  Running job: job_1490890691442_0008
  Job job_1490890691442_0008 running in uber mode : false
   map 0% reduce 0%
   map 100% reduce 0%
   map 100% reduce 100%
  Job job_1490890691442_0008 completed successfully
  Output directory: hdfs:///user/hduser/tmp/mrjob/kmeansAlgorithm.hduser.20170330.162921.120232/output
Counters: 49
```

Coding Running [3]



```
Running step 1 of 1...
  packageJobJar: [/tmp/hadoop-unjar2251983163613915235/] [] /tmp/streamjob4331642045348544338.ja
  Connecting to ResourceManager at /0.0.0.0:8032
  Connecting to ResourceManager at /0.0.0.0:8032
  Total input paths to process : 1
  number of splits:2
  Submitting tokens for job: job_1490463036897_0008
  Submitted application application_1490463036897_0008
  The url to track the job: http://stratosg-Lenovo-YOGA-700-14ISK:8088/proxy/application_149046
  Running job: job_1490463036897_0008
  Job job_1490463036897_0008 running in uber mode : false
   map 0% reduce 0%
   map 100% reduce 0%
   map 100% reduce 100%
  Job job_1490463036897_0008 completed successfully
  Output directory: hdfs:///user/hduser/tmp/mrjob/kmeans_centroid_updater.hduser.20170325.17420
Counters: 49
        File Input Format Counters
                Bytes Read=573
        File Output Format Counters
                Bytes Written=128
        File System Counters
                FILE: Number of bytes read=215
                FILE: Number of bytes written=371793
                FILE: Number of large read operations=0
                FILE: Number of read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=895
                HDFS: Number of bytes written=128
                HDFS: Number of large read operations=0
                HDFS: Number of read operations=9
                HDFS: Number of write operations=2
        Job Counters
                Data-local map tasks=2
                Launched map tasks=2
                Launched reduce tasks=1
                Total megabyte-milliseconds taken by all map tasks=6896640
                Total megabyte-milliseconds taken by all reduce tasks=2209792
                Total time spent by all map tasks (ms)=6735
                Total time spent by all maps in occupied slots (ms)=53880
                Total time spent by all reduce tasks (ms)=2158
                Total time spent by all reduces in occupied slots (ms)=17264
                Total vcore-milliseconds taken by all map tasks=6735
                Total vcore-milliseconds taken by all reduce tasks=2158
        Map-Reduce Framework
                CPU time spent (ms)=2080
                Combine input records=13
                Combine output records=5
                Failed Shuffles=0
                GC time elapsed (ms)=193
                Input split bytes=322
                Map input records=13
```



```
Submitted application application_1490463036897_0014
The url to track the job: http://stratosg-Lenovo-YOGA-700-
Running job: job_1490463036897_0014
Job job_1490463036897_0014 running in uber mode : false
 map 0% reduce 0%
 map 6% reduce 0%
 map 10% reduce 0%
 map 13% reduce 0%
 map 17% reduce 0%
 map 20% reduce 0%
 map 24% reduce 0%
 map 28% reduce 0%
 map 31% reduce 0%
 map 35% reduce 0%
 map 38% reduce 0%
 map 40% reduce 0%
 map 42% reduce 0%
 map 44% reduce 0%
 map 46% reduce 0%
 map 47% reduce 0%
 map 49% reduce 0%
 map 51% reduce 0%
 map 53% reduce 0%
 map 55% reduce 0%
 map 56% reduce 0%
 map 58% reduce 0%
 map 60% reduce 0%
 map 62% reduce 0%
 map 64% reduce 0%
 map 65% reduce 0%
 map 67% reduce 0%
 map 83% reduce 0%
 map 100% reduce 0%
 map 100% reduce 100%
Job job_1490463036897_0014 completed successfully
```

References

[1]: Noll, M. *Running Hadoop On Ubuntu Linux (Single-Node Cluster) - Michael G. Noll*. [online] Available at: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/ [Accessed 29 Mar. 2017].

[2]: Stackoverflow.com. (2017). *Error launching job using mrjob on Hadoop*. [online] Available at: http://stackoverflow.com/questions/25358793/error-launching-job-using-mrjob-on-hadoop [Accessed 29 Mar. 2017].

[3]: David Arthur, and Sergei Vassilvitskii, (2007). *k-means++: the advantages of careful seeding* - Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, January 7-9, 2007. 1st ed. New York: ACM, pp.1027–1035.

[4]: Nlp.stanford.edu. *K-means.* Available at: http://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html [Accessed 15 Mar. 2017]

[5]: Home.deib.polimi.it. (n.d.). *Clustering - K-means.* [online] Available at: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html [Accessed 8 Mar. 2017].

[6]: Kyuseok Shim, "*MapReduce Algorithms for Big Data Analysis*", VLDB Conference, 2012