Spark Based K-Means Algorithm on KMeans_Data.txt

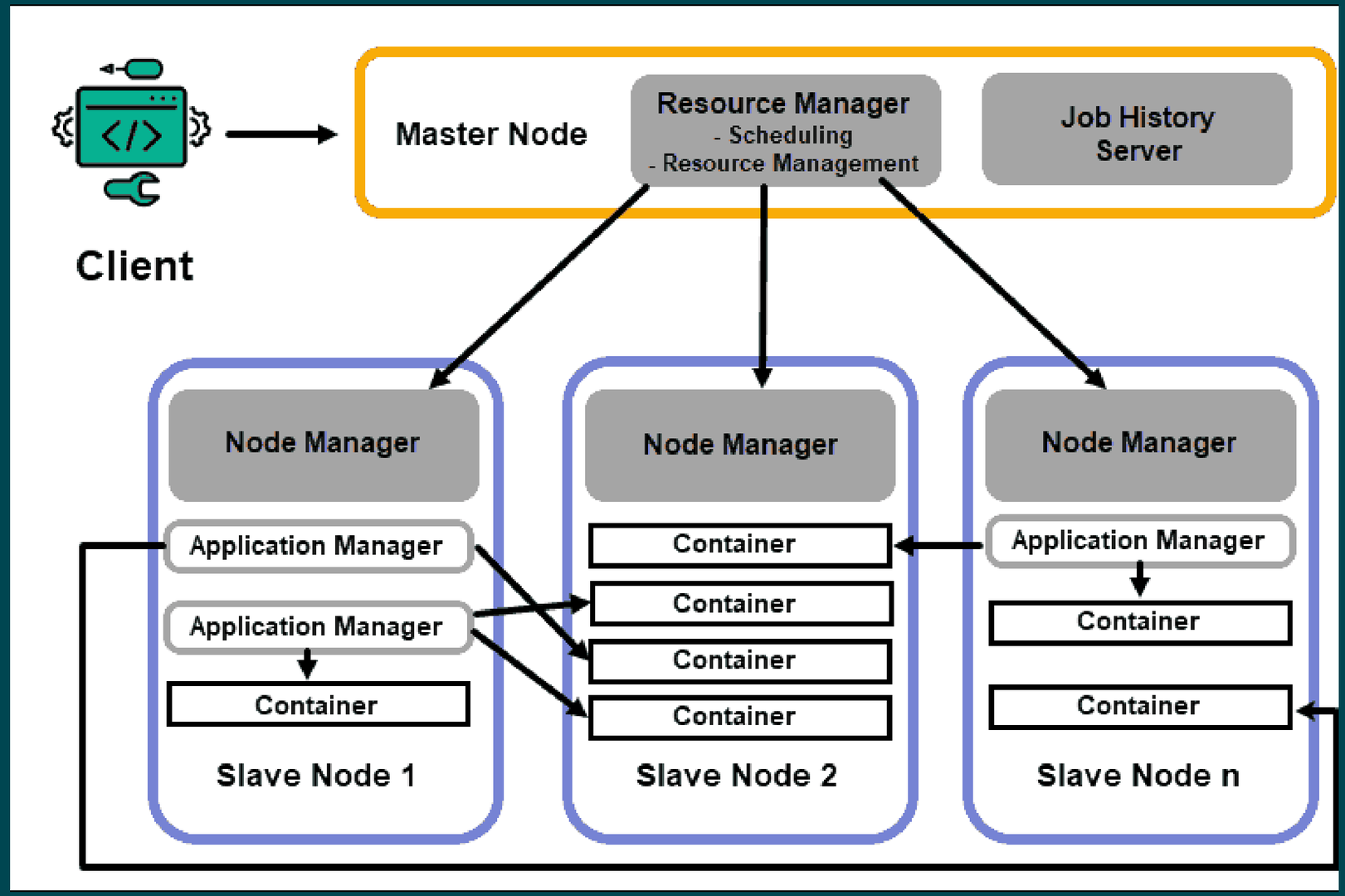
Table Of Contents :

Introduction

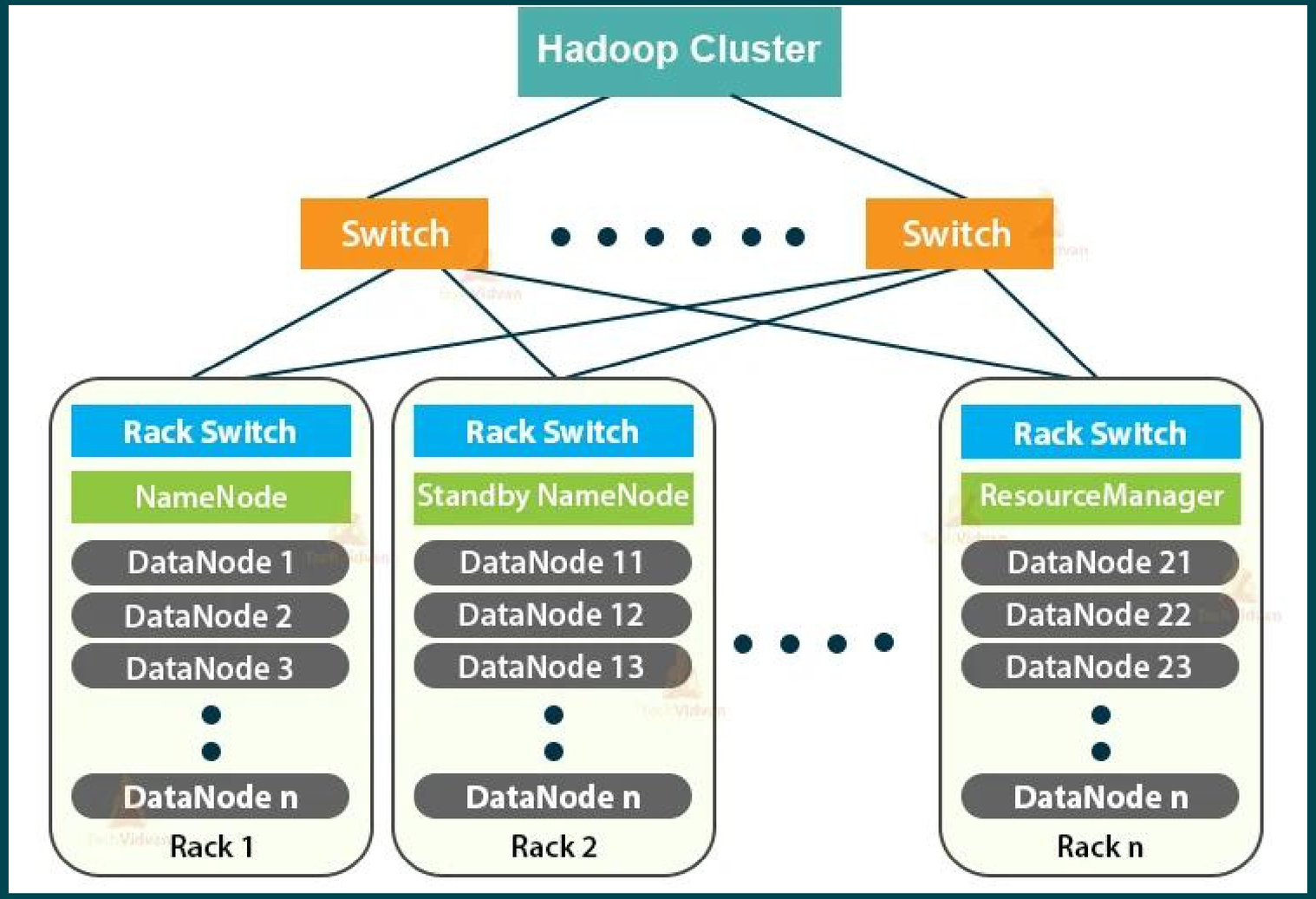
- Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications.
- To implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters, HDFS uses the NameNode and DataNode architecture.
- Manages big data pools and support big data analytics applications.

Hadoop Architecture

Hadoop architecture is a distributed computing framework that consists of HDFS for storage and MapReduce/YARN for processing, enabling scalable and fault-tolerant handling of large datasets across a cluster of machines.



Hadoop cluster :



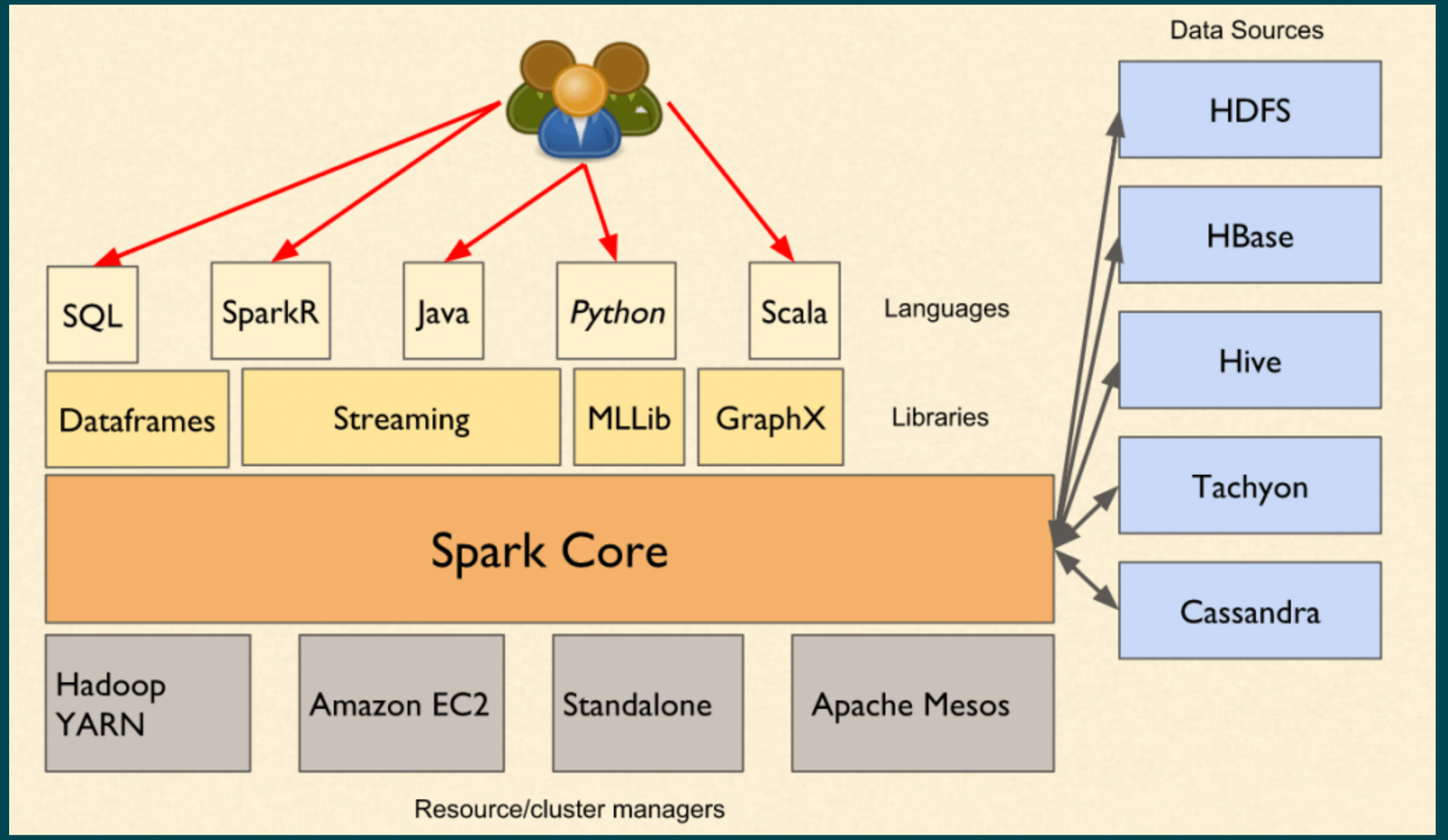
A Hadoop cluster is a group of interconnected computers or servers that work together to store and process large volumes of data. It uses the Hadoop framework, including the Hadoop Distributed File System (HDFS) and MapReduce, to enable distributed storage and parallel processing. The cluster provides scalability, fault tolerance, and high-throughput data access, making it suitable for big data processing and analytics.

SPARK

- Spark is an open-source, distributed computing framework that provides a unified analytics engine for large-scale processing.
- Spark provides high-level APIs for various data processing tasks, including batch processing, stream processing, machine learning, and graph processing.
- Spark is known for its speed and scalability, and it has been used by many companies, including Netflix, Uber, Airbnb, and IBM.

Spark Architecture

Spark architecture consists of a cluster manager, a distributed storage system, and a computational engine that operates on data stored in memory, enabling fast and efficient data processing.



Spark Cluster

- A Spark cluster is a group of machines that work together to perform distributed data processing.
- Spark cluster consists of a single driver node and multiple worker nodes, with each node running its own Spark processes.
- Spark cluster uses a master-worker architecture, where the driver node acts as the master node and the worker nodes act as the slave nodes.
- Spark cluster provides fault-tolerance through its RDD (Resilient Distributed Datasets) model, which allows for data to be automatically replicated across worker nodes in case of node failures.

K-Means Algorithm

K-means is an unsupervised machine learning algorithm used for clustering and pattern recognition tasks.

It is a centroid-based algorithm, which means that it assigns data points to clusters based on their proximity to the centroids.

The algorithm works by first randomly selecting K centroids, where K is the number of clusters desired.

K-means is a computationally efficient algorithm, making it suitable for large datasets.

Benefit of HDFS cluster On kmeans algorithm

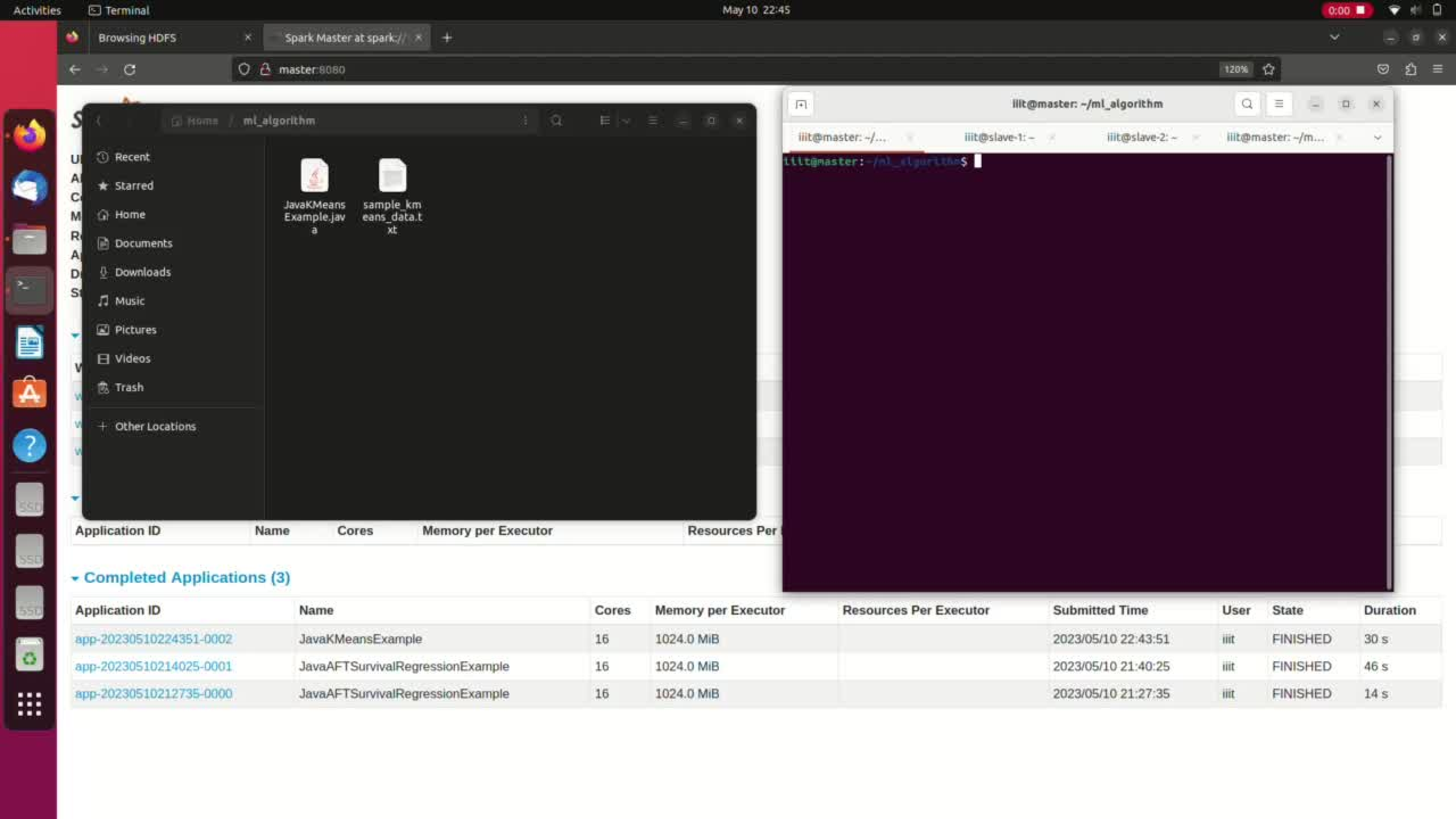
- **Scalability**: HDFS is designed to handle large-scale data storage and processing.
- **Fault Tolerance**: HDFS provides built-in fault tolerance mechanisms.
- **Data Locality**: HDFS's data locality principle is beneficial for iterative algorithms like K-means.
- **Cost-Effective Storage**: HDFS is designed to run on commodity hardware, making it cost-effective for storing large datasets.

Data Set

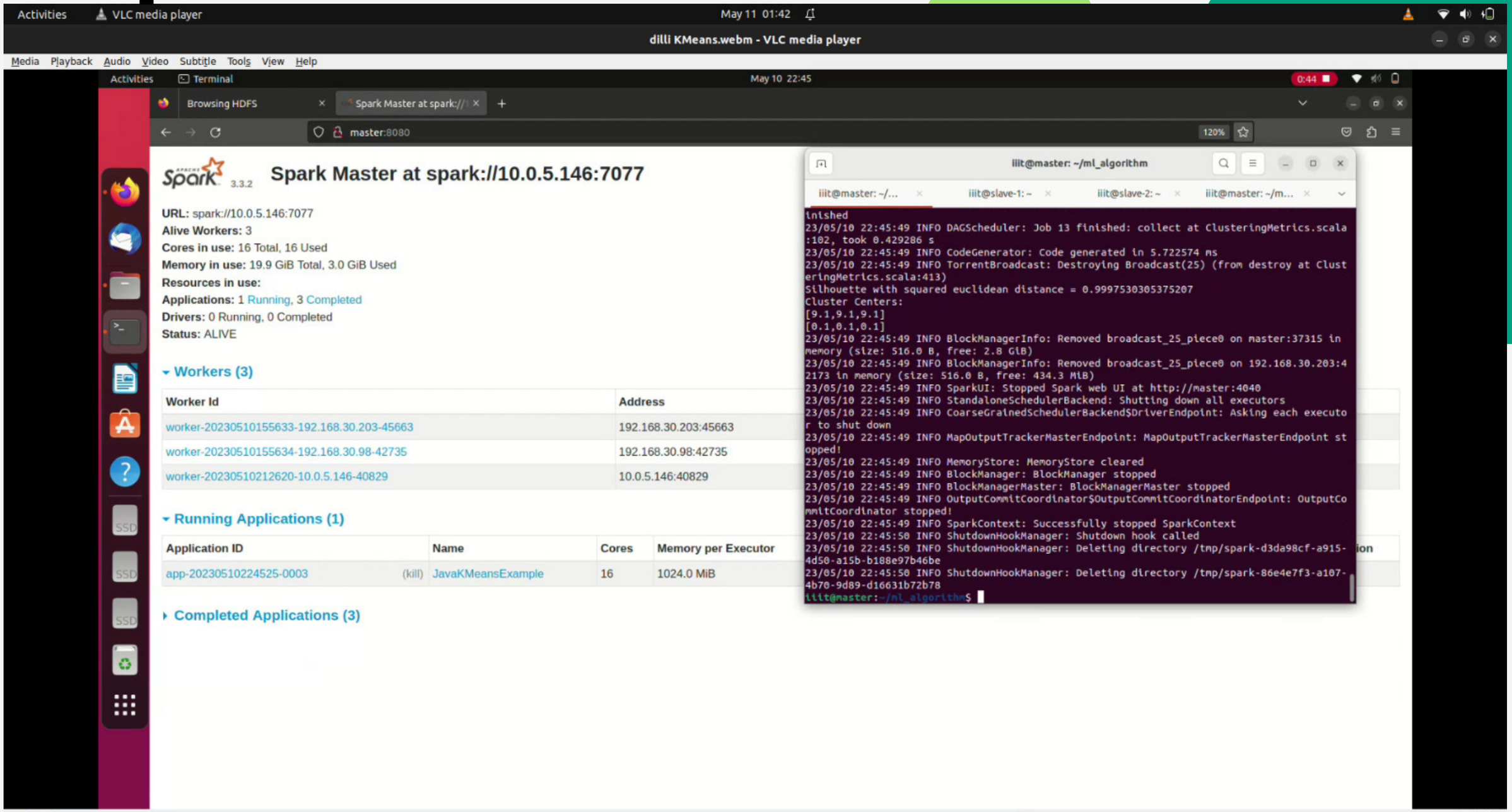
The given dataset represents a three-dimensional data set with six data points. Each data point consists of three values separated by spaces.

Each line represents a data point with three values. Let's denote these values as X, Y, and Z for simplicity. The dataset can be interpreted as follows:

Data Point 1: X=0.0, Y=0.0, Z=0.0
Data Point 2: X=0.1, Y=0.1, Z=0.1
Data Point 3: X=0.2, Y=0.2, Z=0.2
Data Point 4: X=9.0, Y=9.0, Z=9.0
Data Point 5: X=9.1, Y=9.1, Z=9.1
Data Point 6: X=9.2, Y=9.2, Z=9.2


Activities
Terminal
May 10 22:45
Browsing HDFS
Spark Master at spark://
master:8080
120%
Home / ml_algorithm
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
JavaKMeansExample.java
sample_kmeans_data.txt
iiit@master: ~/ml_algorithm
iiit@master: ~/...
iiit@slave-1: ~
iiit@slave-2: ~
iiit@master: ~/m...
iiit@master:~/ml_algorithm$
Application ID
Name
Cores
Memory per Executor
Resources Per
Completed Applications (3)
Application ID
Name
Cores
Memory per Executor
Resources Per Executor
Submitted Time
User
State
Duration
app-20230510224351-0002
JavaKMeansExample
16
1024.0 MiB
2023/05/10 22:43:51
iiit
FINISHED
30 s
app-20230510214025-0001
JavaAFTSurvivalRegressionExample
16
1024.0 MiB
2023/05/10 21:40:25
iiit
FINISHED
46 s
app-20230510212735-0000
JavaAFTSurvivalRegressionExample
16
1024.0 MiB
2023/05/10 21:27:35
iiit
FINISHED
14 s

Output


Activities
VLC media player
May 11 01:42
dilli KMeans.webm - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Activities
Terminal
May 10 22:45
Browsing HDFS
Spark Master at spark://
master:8080
120%
Spark 3.3.2
Spark Master at spark://10.0.5.146:7077
URL: spark://10.0.5.146:7077
Alive Workers: 3
Cores in use: 16 Total, 16 Used
Memory in use: 19.9 GiB Total, 3.0 GiB Used
Resources in use:
Applications: 1 Running, 3 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
Workers (3)
Worker Id
Address
worker-20230510155633-192.168.30.203-45663
192.168.30.203:45663
worker-20230510155634-192.168.30.98-42735
192.168.30.98:42735
worker-20230510212620-10.0.5.146-40829
10.0.5.146:40829
Running Applications (1)
Application ID
Name
Cores
Memory per Executor
app-20230510224525-0003 (kill)
JavaKMeansExample
16
1024.0 MiB
Completed Applications (3)
iiit@master: ~/ml_algorithm
iiit@master: ~/...
iiit@slave-1: ~
iiit@slave-2: ~
iiit@master: ~/m...
inished
23/05/10 22:45:49 INFO DAGScheduler: Job 13 finished: collect at ClusteringMetrics.scala:102, took 0.429286 s
23/05/10 22:45:49 INFO CodeGenerator: Code generated in 5.722574 ms
23/05/10 22:45:49 INFO TorrentBroadcast: Destroying Broadcast(25) (from destroy at ClusteringMetrics.scala:413)
Silhouette with squared euclidean distance = 0.9997530305375207
Cluster Centers:
[9.1,9.1,9.1]
[0.1,0.1,0.1]
23/05/10 22:45:49 INFO BlockManagerInfo: Removed broadcast_25_piece0 on master:37315 in memory (size: 516.0 B, free: 2.8 GiB)
23/05/10 22:45:49 INFO BlockManagerInfo: Removed broadcast_25_piece0 on 192.168.30.203:42173 in memory (size: 516.0 B, free: 434.3 MiB)
23/05/10 22:45:49 INFO SparkUI: Stopped Spark web UI at http://master:4040
23/05/10 22:45:49 INFO StandaloneSchedulerBackend: Shutting down all executors
23/05/10 22:45:49 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
23/05/10 22:45:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
23/05/10 22:45:49 INFO MemoryStore: MemoryStore cleared
23/05/10 22:45:49 INFO BlockManager: BlockManager stopped
23/05/10 22:45:49 INFO BlockManagerMaster: BlockManagerMaster stopped
23/05/10 22:45:49 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
23/05/10 22:45:49 INFO SparkContext: Successfully stopped SparkContext
23/05/10 22:45:50 INFO ShutdownHookManager: Shutdown hook called
23/05/10 22:45:50 INFO ShutdownHookManager: Deleting directory /tmp/spark-d3da98cf-a915-4d50-a15b-b188e97b46be
23/05/10 22:45:50 INFO ShutdownHookManager: Deleting directory /tmp/spark-86e4e7f3-a107-4b70-9d89-d16631b72b78
iiit@master:~/ml_algorithm$
00:44
01:04