National Institute of Technology, Karnataka

MAP REDUCE IN HADOOP

Presentation by-
- Gavali Deshabhakt Naganath
- Mohammad Ahsan
- Mayur Anil Shimpi

Course Instructor: Dr. Pushparaj Shetty
Subject: Big Data Analytics
Specialization: CDS
Date: 14/04/2021

Agenda

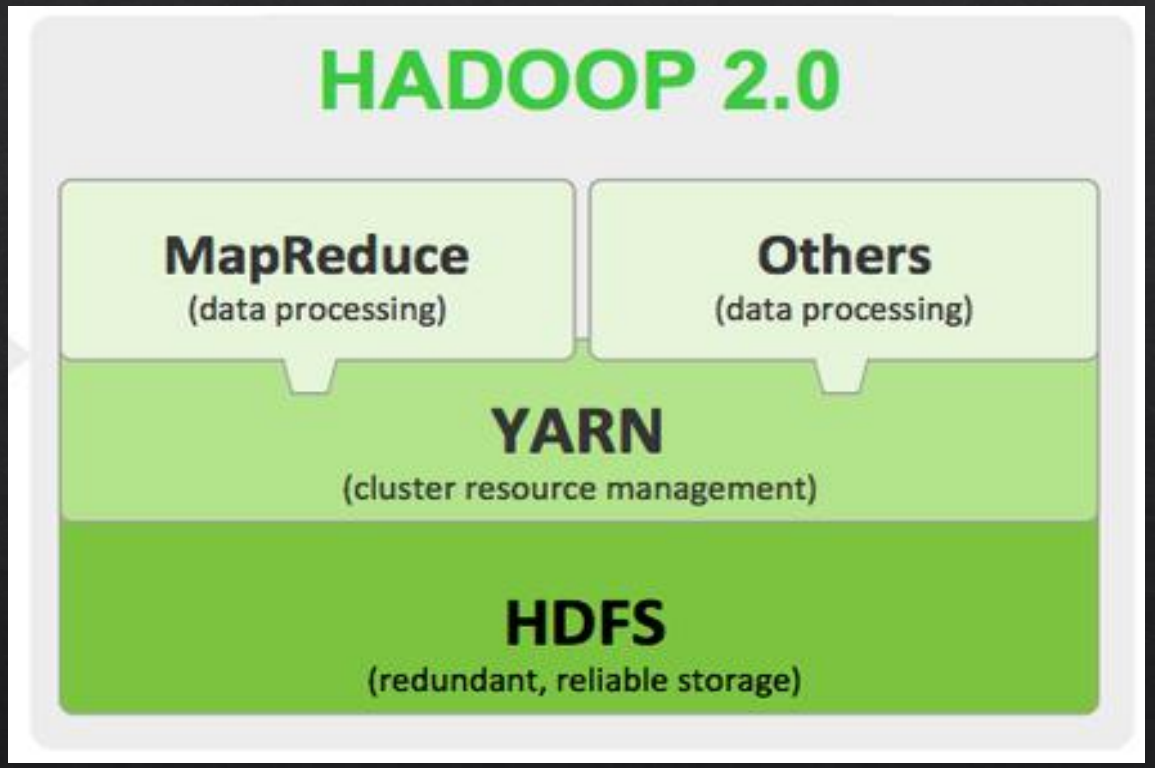
- Introduction to Hadoop
- Introduction to MapReduce
- Different Stages in MapReduce
- MapReduce Use Case: K means clustering Algorithm
- Hadoop Installation
- Hadoop Demonstration
- Covid-19 Database analysis

Key features of Hadoop

- Open Source
- Highly Scalable
- Distributed
- Massive Storage
- High Availability
- Fault Tolerance

Core Components of Hadoop

- HDFS
  - NameNode
  - DataNode
- YARN
- MapReduce



Map Reduce

Map Reduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

Key Features of Map Reduce

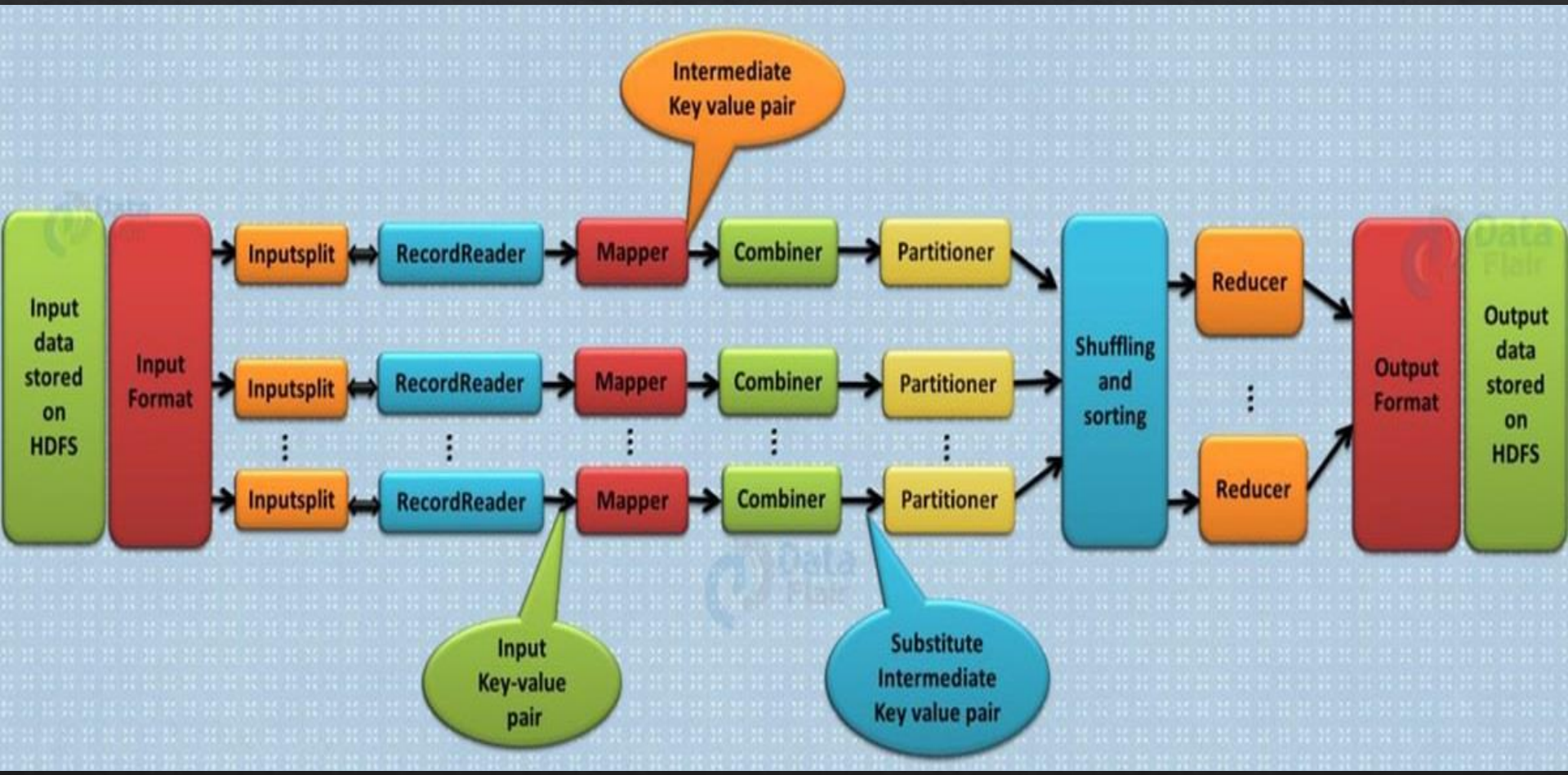
- Take small pieces of data from big data
- Take Key, Value pairs as input dataset at all stage
- All stages of Map Reduce generate key, value pairs as output dataset.
- Work on principle of distributed data and parallel processing
- Data saved within HDFS in certain format.

  eg. bZip2, Snappy, AVOC ,etc.

Different Stages in Map Reduce

1. Mapper
2. Intermediate Stages
   i) Combiner
   ii) Partitioner
   iii) Sorting
3. Reducer
4. Compression

Flow Chart Of Map Reduce

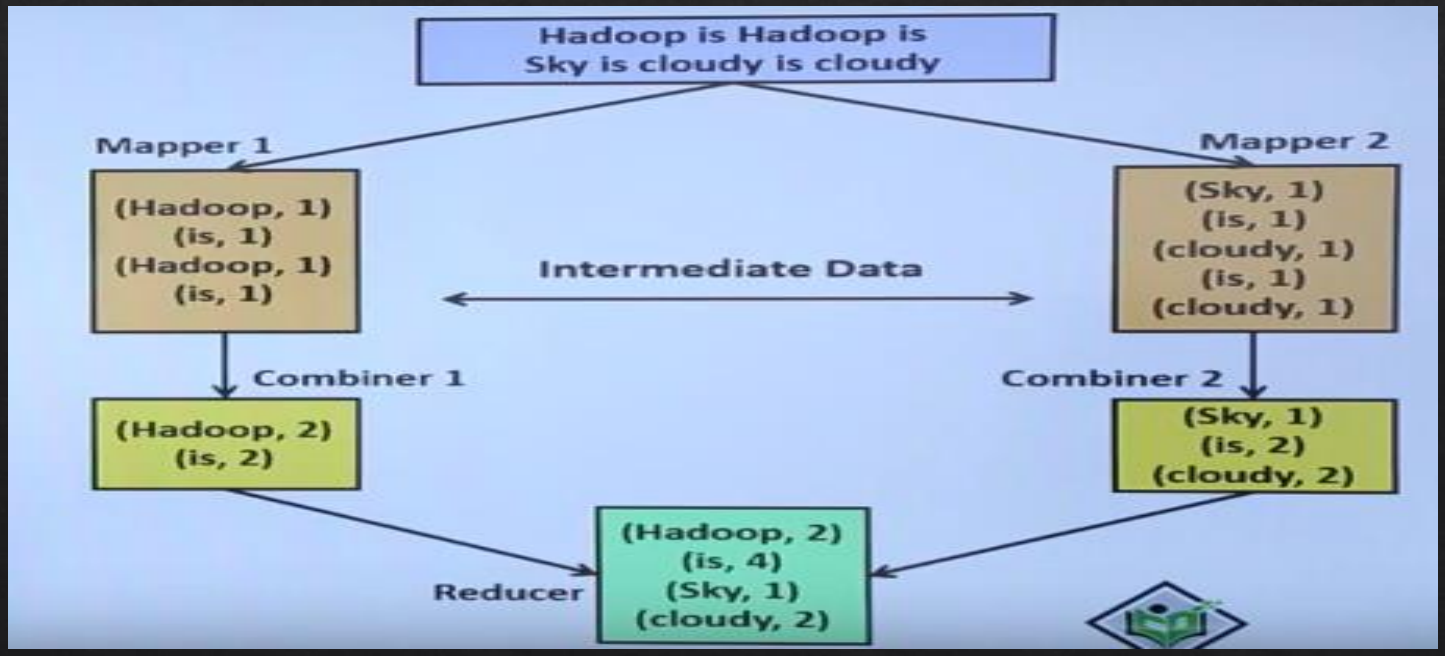


Mapper

- It is where parallel processing take place on datasets.
- Business logical Model is implemented.
- Semi processed Key, Value pair generated as output.
- Crucial stage of Map Reduce where actual processing takes place.

Combiner

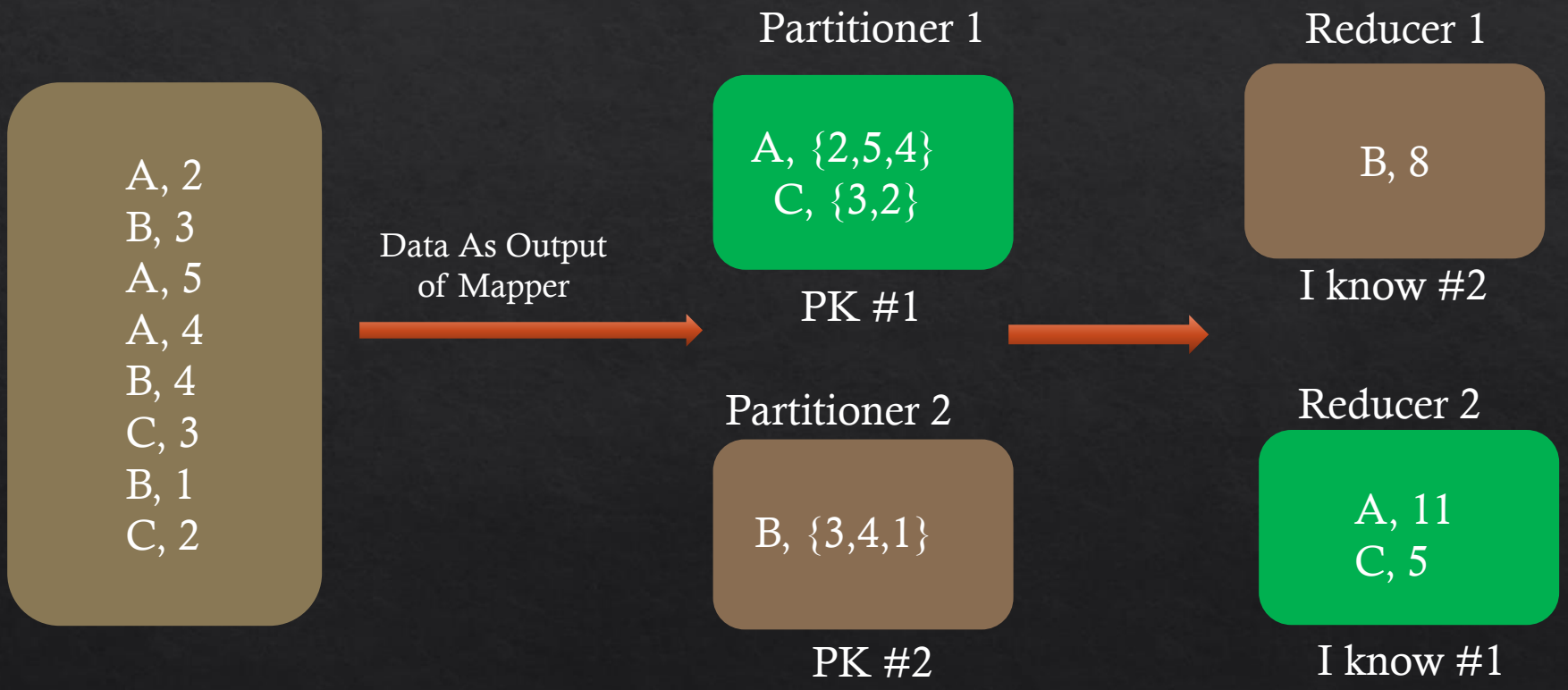
- It's main function is to summarize the mapper output records with the same key.
- Also called Mini Reducer.
- It Aggregate data coming out from mapper.



Partitioner

- It partitions the key-value pairs of intermediate Mapper outputs.
- It partition data using user-defined function (like hash function).
- Different practitioners' assign different partition keys.
- Partition keys are used to map key, value paired data to a particular reducer.
- Number of Practitioners' = Number of Reducers

Flow Chart Of Partitioner



Sorting

- It’s an intermediate stage where in coming data get sorted.
- Sorting take place according to key of paired data.
- Sorting speed up the reducer process.

Reducer

- Reducer is the final stage of Map Reduce
- It take the preprocessed data coming from some intermediate stages and do either aggregation, filtration or summation independently.
- The OutputCollector.collect() method, writes the output of the reduce task to the HDFS file system
- Reducer output is not sorted when stored

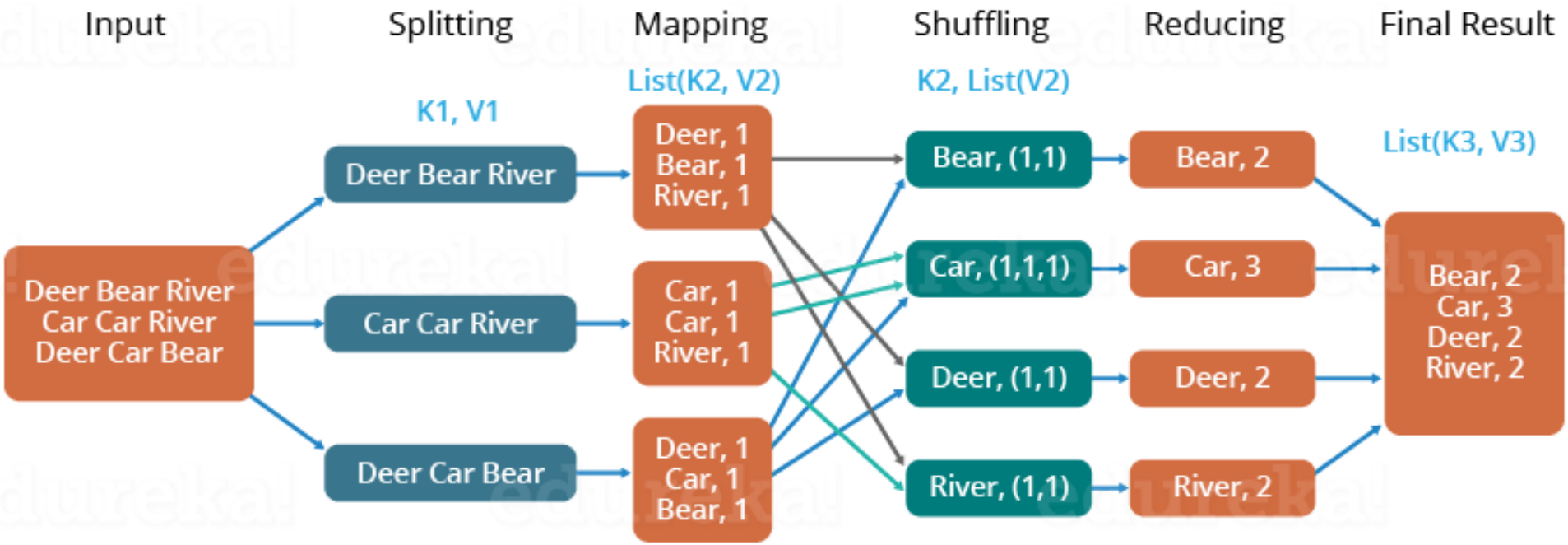
Compression

- Compression saves storage, network flow, processing cost significantly.
- Without compression hadoop would become impractical as it would be using enormous resources.
- General format used to store files in hadoop are Snappy, bZip2, AVOC, Parquet etc.

Features to care before selecting the right format

- Compressed file should be optimal in storage.
- Faster read and write can be done after compression.
- Compressed file should split if needed. (Distributive Nature)
- Should follow the dynamic schema even after compression.

Flow Chart Of Map Reduce With Example



Solving Word Count Problem With Map Reduce

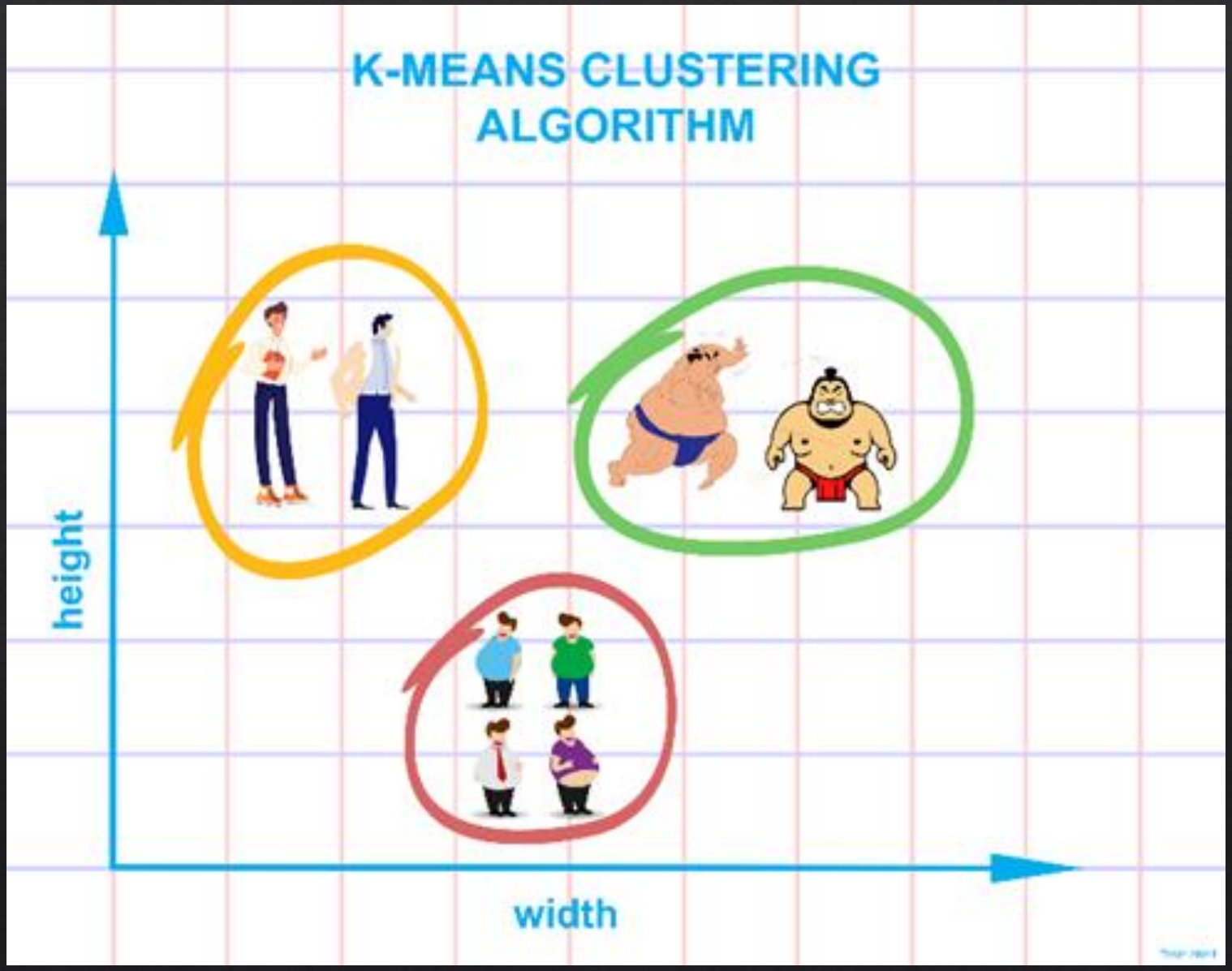
Clustering

- Find homogeneous subgroups within the data
- Data points in each cluster are as similar as possible
- Similarity measures are such as Euclidean-based distance or correlation-based distance

K-Means Clustering

- K-Means algorithm is an iterative algorithm
- It partitions the dataset into K pre-defined distinct clusters where each data point belongs to only one group
- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid is at the minimum

K-MEANS CLUSTERING ALGORITHM



K-Means Algorithm

- Choose k initial means μ1, . . . , μk uniformly at random from the set X.
- For each point x ∈ X, find the closest mean μi and add x to a set Si
- For i = 1, . . . , k, set μ_i to be the centroid of the points in Si
- Repeat steps 2 and 3 until the means have converged.

K-Means via MapReduce

- Two Phases of K-Means
  1. Map will Assign Observations to Closest cluster Center
  2. Reduce will revise cluster centers as mean of assigned observations

K-Means via MapReduce Algorithm

- Choose k initial means μ1, . . ., μk uniformly at random from the set X.
- Apply the MapReduce given by k-meansMap and k-meansReduce to X.
- Compute the new means μ1, . . ., μk from the results of the MapReduce.
- Broadcast the new means to each machine on the cluster.
- Repeat steps 2 through 4 until the means have converged.

Classification step as Map

- Classify: Assign observations to closest cluster center

  1. map($[\mu_1, \mu_2 \ldots\ldots\ldots \mu_k], X_i$]
  2. $Z_i \leftarrow \underset{\text{‘i’}}{\text{argmin}} ||\mu_i - X_i||_2^2$
  3. emit(Zi,Xi)

Re-center Step as Reduce

- Recenter: Revise cluster centers as Mean of Assigned observations

1]reduce (Zi, X in cluster i:[X1,X3…….])

sum=0

count=0

2]for x in cluster i

sum +=x

count+=1

3]emit(Zi, sum/count)

Installation Of Hadoop

- Standalone Mode - link
- Pseudo Distributed mode
- Distributed mode - link

Creating Hadoop User

```
$ su
   password:
# useradd <username>
# passwd <username>
   New passwd:
   Retype new passwd
```

```
[deshabhakt@deshabhakt-pc ~]$ sudo -u hadoop zsh
[sudo] password for deshabhakt:
 home/deshabhakt  cd ~                                    ✔
 ~  whoami                                                ✔
hadoop
 ~  pwd                                                   ✔
/home/hadoop
 ~                                                        ✔
```

Java Installation

```
$ cd Downloads/
$ ls
jdk-8u281-linux-x64.tar.gz
$ tar xvf jdk-8u281-linux-x64.tar.gz
$ ls
jdk-8u281-linux-x64   jdk-7u71-linux-x64.gz
```

```
~ java -version
openjdk version "15.0.2" 2021-01-19
OpenJDK Runtime Environment (build 15.0.2+7)
OpenJDK 64-Bit Server VM (build 15.0.2+7, mixed mode)
~
```

Installing/Configuring Hadoop

1. **Setting up environment variables for Hadoop**

```
export HADOOP_HOME=/home/hadoop/hadoop-3.2.2
export HADOOP_CONF_DIR=/home/hadoop/hadoop-
3.2.2/etc/hadoop
export HADOOP_MAPRED_HOME=/home/hadoop/hadoop-3.2.2
export HADOOP_COMMON_HOME=/home/hadoop/hadoop-3.2.2
export HADOOP_HDFS_HOME=/home/hadoop/hadoop-3.2.2
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=/home/hadoop/hadoop-
3.2.2/lib/native
export HADOOP_OPTS="-
Djava.library.path=/home/hadoop/hadoop-3.2.2/lib"
export PATH=$PATH:/home/hadoop/hadoop-3.2.2/bin
export HADOOP_PID_DIR=/home/hadoop/hadoop-
3.2.2/hadoop_data/hdfs/pid
```

2. Hadoop Configuration

- core-site.xml
- hdf-site.xml
- yarn-site.xml
- mapred-site.xml
- hadoop-env.sh

```
~/hadoop-3.2.2/etc/hadoop ls                                                    ✓
capacity-scheduler.xml            httpfs-log4j.properties     mapred-site.xml
configuration.xsl                 httpfs-signature.secret     shellprofile.d
container-executor.cfg            httpfs-site.xml             ssl-client.xml.example
core-site.xml                     kms-acls.xml                ssl-server.xml.example
hadoop-env.cmd                    kms-env.sh                  user_ec_policies.xml.template
hadoop-env.sh                     kms-log4j.properties        workers
hadoop-metrics2.properties        kms-site.xml                yarn-env.cmd
hadoop-policy.xml                 log4j.properties            yarn-env.sh
hadoop-user-functions.sh.example  mapred-env.cmd              yarnservice-log4j.properties
hdfs-site.xml                     mapred-env.sh               yarn-site.xml
httpfs-env.sh                     mapred-queues.xml.template
~/hadoop-3.2.2/etc/hadoop                                                       ✓
```

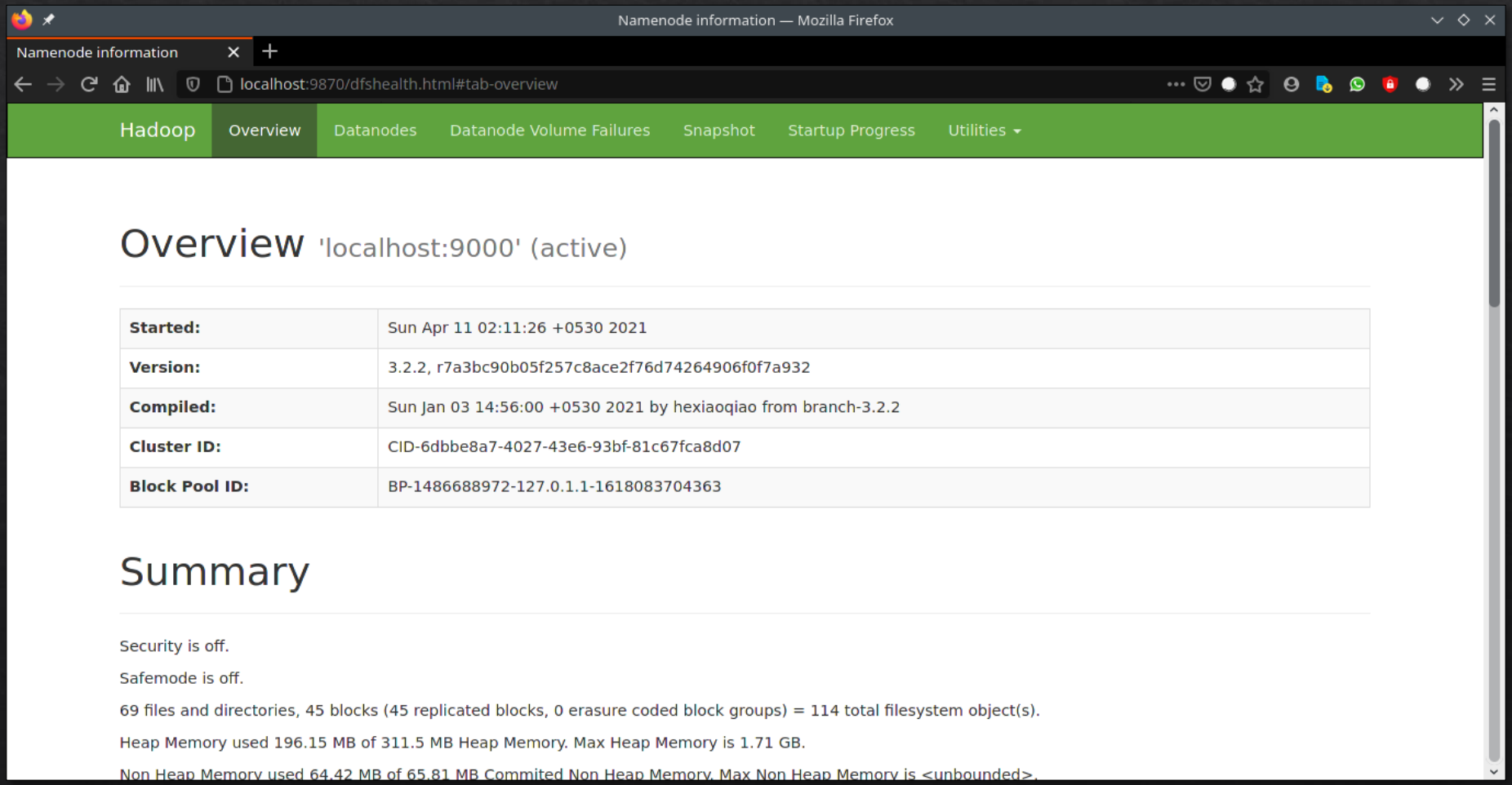
3. Verifying Hadoop Installation

- ‘hadoop version’
- ‘hdfs namenode –format
- ‘./start-all.sh

```
[hadoop@deshabhakt-pc hadoop-3.2.2]$ sbin/start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [deshabhakt-pc]
2021-04-11 14:50:03,999 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting resourcemanager
Starting nodemanagers
[hadoop@deshabhakt-pc hadoop-3.2.2]$ jps
18930 Jps
18050 NameNode
18132 DataNode
18313 SecondaryNameNode
18667 NodeManager
18574 ResourceManager
[hadoop@deshabhakt-pc hadoop-3.2.2]$
```

Accessing Hadoop Web UI

- ‘localhost:9870’



Summary of Hadoop Cluster

Configured Capacity:	210.05 GB
Configured Remote Capacity:	0 B
DFS Used:	3.91 MB (0%)
Non DFS Used:	68.11 GB
DFS Remaining:	131.19 GB (62.46%)
Block Pool Used:	3.91 MB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	16
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Mon Apr 12 14:05:52 +0530 2021
Last Checkpoint Time	Mon Apr 12 14:05:54 +0530 2021
Enabled Erasure Coding Policies	RS-6-3-1024k

References

- Acharya, Seema; Subhashini Chellappan. Big Data and Analytics, 2ed (p. 253). Kindle Edition.
- https://hadoop.apache.org/docs/stable/index.html
- Tutorialspoint.com
- Javatpoint.com
- GeeksForGeeks
- https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/bodoia.pdf
- https://www.coursera.org/lecture/ml-clustering-and-retrieval/mapreduce-for-k-means-EhCYk
- https://medium.com/@tarlanahad/a-friendly-introduction-to-k-means-clustering-algorithm-b31ff7df7ef1
- https://www.baeldung.com/java-k-means-clustering-algorithm