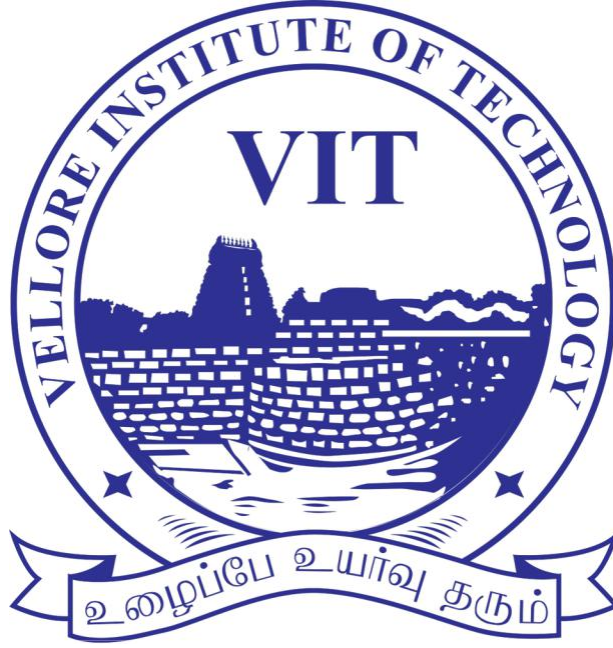


# PARALLEL AND DISTRIBUTED COMPUTING

## J-Component



Registration Number	Name
18BCI0044	Veera Venkata Durga Praveen
18BCE0455	Challangundla Manish Gupta
18BCI0067	Dodda Sumanth

**Submitted to**  
**Professor Dr. Manoov R**

**Application of Map Reduce in Document Clustering**  
**(K-Means Clustering)**  
([Code and Demonstration](#))

## 1. Introduction

1. Why Map-Reduce?

## 2. Literature Review

1. Distributed Document Clustering Analysis Based on a Hybrid Method
2. Data locality in Hadoop Map Reduce
3. Facilitating Understanding of Large Document Collections
4. A Review on Data locality in Hadoop Map Reduce
5. An Evaluation of Hadoop Cluster Efficiency in Document Clustering using Parallel K-means
6. A study and Performance Comparison of Map Reduce and Apache Spark on Twitter Data on Hadoop Cluster
7. A fuzzy approach to clustering of text documents based on Map Reduce
8. Comparison of a Sequential and a Map Reduce Approach to Joining Large Datasets

## 3. Abstract

1. What is clustering?
2. Types Of Clustering
3. What is Map-Reduce?
4. What is K-Means Clustering?

## 4. Motivation and Objective

## 5. Problem Statement

## 6. Pseudo Code

## 7. Methodology and Working

1. Hadoop Installation
  1. What is hadoop ?
  2. Install Java
  3. Create Hadoop User
  4. Login into hadoop user
  5. Copy SSH keys
  6. login ssh into local host
  7. Wget and install hadoop
  8. Configure hadoop variables
  9. Start hadoop Cluster
2. Algorithms for proposed Model
  1. Algorithm for Centroids
  2. Algorithm for Mapper with combiner
  3. Algorithm for Reducer

### 3. Working Model

#### 1. Data set Creation

#### 2. Map Reduce

#### 3. K-Means Clustering Algorithm

##### 1. Introduction to k-mean euclidean 2D

### 8. Conclusion

### 9. Output

### 10. References

### 11. Plot graphs

## **Introduction:**

*The Map Reduce is parallel paradigm which has 2 phases: Map and Reduce which are pipelined phases to give a touch of Parallelism. The input data is processed into independent segments. The segments are passed through functions of Text Mining using Hadoop where Hadoop consists of 2 components say hdfs (storage) and map reduce (processing)*

*The segments are tuned into <key, value> pairs which are parallel processed using Mapper function. The outputs from the Mapper are also: <key, value> pairs known as intermediate pairs which are sorted and grouped based on key and the k-means clustering algorithm will group all the similar object into number of clusters and where it keeps on refining the center point of the cluster as the iteration goes on until the maximum number of intra cluster deviation is reached gives the touch of distributed memory*

### **Why map-reduce?**

*Early in 2004 huge amounts are stored on a single server and every information is stored individually within the server and the the query for the piece of data ? For sure its a nightmare and and the also the threat of data loss and the backup of the data and reduced scalability resulted in the issue of snow bowling and into the crisis of sorts and then in 2014 google introduces a concept called map reduce which maps and reduces the data by that queries could run faster and simultaneously on multiple servers with ease,search results could be logically integrated and data could be analyzed in real time*

## **Literature review:**

### **1.Distributed Document Clustering Analysis Based on a Hybrid Method [vi]**

**Abstract :**As per recently challenging tasks Clustering is one best task since there is an ever-growing amount of data in scientific research and commercial applications. For clustering the computational requirements for bringing such growing amount data to a central site was complex.

The Particle Swarm Optimization (PSO) is the algorithm proposed for optimal centroids for K-Means clustering. The advantage of PSO was global search ability to provide optimal centroids which aids in generating more compact clusters with improved accuracy. For this methodology Hadoop and MapReduce framework which provides distributed storage and analysis to support data intensive distributed applications was utilised.

**Introduction:-** In this era the constant progress was powered by the Information Technology ventures on multitudes of opportunities on data analysis, information retrieval and transaction processing as the size of information repositories keep on expanding each day. This was happens because of enormous growth in the volume of information available on the internet, such as Digital Libraries, Reuters, Social Medias, etc.

The important fields of machine learning was Clustering or unsupervised learning which splits the data into groups of similar objects helping in extraction or summarization of new information. The variety used for fields such as statistics, pattern recognition and data mining. The procedure to discover patterns from large datasets was data mining. The sub-field of data mining was Text mining that analyses a large collection of document datasets. due to these reasons Distributed computing plays a major role in data mining. to solve the issues Distributed Data Mining (DDM) has evolved for hot research area.

**Experimental setup:-** Hadoop cluster environment is The distributed environment to evaluate the performance of the proposed clustering algorithm. Version of Hadoop used is 0.20.2. entire nodes were connected by standard gigabit Ethernet network on a flat network topology. For parallel environment Parallel jobs are submitted like Hadoop (MapReduce). On pseudo distributed mode hadoop will run where each Hadoop daemon runs as a separate process.

**Conclusion :-** For design and implementation of a hybrid PSO KMeans clustering (MR-PKMeans) algorithm using MapReduce framework was proposed. For document clustering PKMeans clustering algorithm is an effective method, however, it takes a long time to process large data sets. To overcome the inefficiency of PKMeans for big data sets MR-PKMeans was proposed. proposed method can efficiently will be parallelized with MapReduce to process very large data sets. The formulated KMeans algorithm utilizes the best centroids generated by PSO. For good accuracy and reduced execution time The global search ability of optimization algorithm was used.

### **2.Data locality in Hadoop Map Reduce[vii]**

**Abstract:-** Map Reduce is model for processing and distributed data for huge datasets and it was strong model. A source of Hadoop implemented of Map Reduce has a prooven by Map Reduce widely.

Hadoop will provide a scheduling with effectiveness to the process of data blocks in efficient way. Data locality is the main issue in efficient processing of Map Reduce which was caused due to overhead of network. In distributed environment Data Locality is the main key that influences the tasks accomplishing time.

**Introduction:-** Complex problem was the problem that can be solved by, the problem can be partitioned into sub problems. Atleast one of the computing code can be tackled by each sub problem. Computing nodes will talk to one another via sending and receiving messages. The Big data focuses on compute and data intensive tasks was the present scenario. The Big Data was termed as a collection of huge data sets which are non-manageable for conventional tool.

The model that supports parallel and distributed processing of a huge datasets of main programme is called Map Reduce. The parallelization of lower level particulars taken care at run time. For Map Reduce platform Hadoop is extensively used.

storing of data Hadoop consists of Hadoop Distributed file system (HDFS). data processing is done via Map Reduce model in HDFS following master slave architecture. The slave nodes are Task Trackers and The Job Tracker master node. Input splits are used as the input of a job in Hadoop is divided into fragments of same size.

Several problems was arsed for Scheduling the reduce tasks in Hadoop which was related to congestion because task scheduling process in Hadoop is based on pulling the data from source node. The thing that was responsible for sending a heartbeat message was Task Tracker and the responsible for ensuring each task to run on the designated Task Tracker which holds the required input splits was Job Tracker.

**Data Locality in Hadoop:**Hadoop which makes considerable impact on system performance was Data Locality. In Data Locality Elements that play vital role was number of replicas, cluster size, and job execution time and stage.Results of creation one or more replicas leads to improved data locality but it employs more storage space.

When slave node sent message to master of Job Tracker node it attempts to look for a map task in the queue for a job whose input data is with that node. If we get successes while searching the data it results in node level locality and task is launched on particular node, if node level doesnot achieved then JobTracker efforts for rack locality then a task is picked randomly which provides rackoff locality. Then data locality is supported by FIFO but it does have shortcoming of performing scheduling task to task irrespective of its impact on other tasks.

For predicting performance of number of MapReduce jobs and dealing with designation of resources to each job at runtime Dynamic task scheduler is used. Scheduler observes the time taken by previously finished tasks to know average task length for envisaging the job finishing time if both individually submitted and yet not finished jobs.

In High Performance MapReduce Engine has implement two schemes that was Prefetching and Pre-shuffling schemes.it has proved the performance of shared environment of MapReduce computation. Hadoop- On-Demand (HOD) has solution to the problem upsurses the use of resources on the cost of performance when number of users are in race for network and hardware resources which

significantly deteriorates the performance. Prefetching scheme will show data locality using intrablock prefetching and inter-block prefetching.

There are Two issues of intra-block prefetching:-

- 1 To synchronize essential of computation and catch suitable prefetching rate. handled by introducing concept of bi-directional processing bar to synchronize computation with assessing the efficacy of technique and it looks for suitable prefetch rate to maximize performance by reducing I/O overhead and eliminating duplicate read operations.
- 2 Inter-block prefetching, prefetch the required replica of block to the local rack by perused it from the node with minimum loaded replica's so as not to limit the effect of performance as whole.

**Discussion:-**Hadoop of Various scheduling algorithms that attempts to resolve data locality issue has been reviewed. To improve some algorithms lacks in other aspects, example Scheduler needs more energy and lacks in data placement. Prefetching and Pre-shuffling HPMR was used to improve performance of shared environment of MapReduce computation but it fails in reducing map tasks existence.

**Conclusion:**Distribution of data, cluster and network load, complex load, cost, resource sharing, cluster environment (homogeneous or heterogeneous), unplanned clients requests, size of data blocks, number of mappers and reducers are the couple of issues that inconveniences for data locality. For develop/improve techniques to efficiently handle the heavy load for heterogeneous clusters is no need in future.

### **3.Facilitating Understanding of Large Document Collections[viii]**

**Introduction:** Document collection refers to a set of documents that may be from the same domain. This involves that internal documents have an internal relationship. Next, this relationship is most often described among other texts relating to the same specific function, or belonging to the same function. Unlike keyword identification and acquisition models based on keywords, resource acquisition refers to groups of documents that are closely related. These definitions are used as access points to help users find information within small text groups. Traditionally, building access assistance is a manual process that requires reading of documents and making references about their relationship to make explanations

**Methodology:**This work is related to topic acquisition and data acquisition, as well as further use of algorithm integration systems. A lot of research has been done to find titles in document collections using compilation algorithms. A combination of positioning in phased writing is used for browsing and navigation. Latent Dirichlet Allocation (LDA) is a production model used to extract articles over the copy text in an unconventional way. The LDA can combine words related to topics and topics, and texts into a random mix of hidden topics. uses LDA to identify the number of topics in Enron's email collection. While LDA-based approaches may target topics to a larger corpus by reducing size, the



results contain a set of unconventional words that cannot be easily traced back to a set of documents. The results are therefore difficult to use as a point of access to our problem.

They classify long texts into several categories according to the minimum Character Threshold (MNCT). The value of MNCT is determined by the distribution of the number of characters in each category of document collection. Then converts each component into a TFIDF vector into a vector space model after the terminology is removed. They are using the Mahout Library in Hadoop for word removal and vector modification.

**Conclusion:** On Comparing to the results of density-based clustering with segmented emails, K-means, and LDA, our approach improved. Emails are strictly integrated as a segment function and that audio notes are discarded. Using this approach we have found sensible collections that provide relevant information about activities and communication practices. Archives can easily define these collections as access points for access assistance. Next, the combination of various stories and categories of information provides a general definition of the collection.

#### **4.A Review on Data locality in Hadoop MapReduce[ix]**

**Introduction:** MapReduce is one of the main program models that supports the processing and distribution of large datasets. It is an awesome and efficient platform because it allows for the use of a number of computer distribution equipment and also provides program planners with the conceptual design of maps and subtractors. A low level of comparative data is maintained during operation. Key features of MapReduce include measurement, error tolerance and can be easily applied for data mining, machine learning and technical comparisons. Between Hadoop, Sphere, Mars etc. Hadoop is the most widely used and widely used MapReduce platform.

**Methodology:** The Dynamic Work Plan is used to predict the performance of a number of MapReduce jobs and to address the selection of resources for each employee during the operation to achieve the target without wasting resources. This editor sees the expected time to complete each MapReduce task by exploiting the fact that the task at MapReduce has several tasks (including mappers and trimmers) to accomplish, in addition to the number of tasks performed ahead of time during the initial phase of the input phase, and to view task performance during operation. When looking at individual tasks that have not yet been completed, the editor looks at the time taken by the previously completed tasks to determine the length of the work time to consider the completion time. This limitation allows the organizer to modify the required job posts for each task to be allocated during operation. Experiments conducted to study the impact of local data and problems associated with memory usage in the MapReduce phase of Hadoop revealed that local data readings are faster than readings from distant HDFS nodes that reflect data transfer stress on the network, and large input leads to failure of one slave node as well. affecting another slave node resulting in downtime in terms of performance and data transfer to remote TaskTracker.

**Conclusion:** Using time-varying algorithm releases for Map operations in the MapReduce framework, as a result of data skew and enhanced visual equipment interference, can be managed. To separate jobs randomly between subsets, the proposed method leads to a reduction in the cost of virtual equipment and also reduces the operating time by 46.7% compared to other previous methods. The

test is performed in a coherent component assuming that the standard deviation of the function and definition is the same for each node. As a result, different groups were not considered due to the need to evaluate the general distribution of all allocations of individual arrangements independently.

## **5. An Evaluation of Hadoop Cluster Efficiency in Document Clustering using Parallel K-means[x]**

**Introduction:** Traditional integration algorithms do not work in combining large data sets. Although k-means is a fast and easy integration algorithm, the rapid growth of data storage needs to be improved by traditional methods k in order to collect large data sets efficiently. Distributed frameworks provide a computer environment where multiple connected devices share all algorithm processing functions. Each computer device connected to a computer network is known as a node and the entire computer framework is known as a cluster. Apache Hadoop is an open source distributed computer architecture designed to process large data sets efficiently. Use the MapReduce program model

**Methodology:** K means can work in numerical databases. Therefore the document database must be converted before being inserted into kmeans. We modified the database using a vector space model. The Vector space model has multiplied the data depending on the availability of words in text files. This converted database then uploaded to HDFS. Text file contains n. The Vector space model calculates the metals for each word based on its occurrence and represents the file in the numerical data bundle  $w_1, w_2, \dots, w_n$ . The relationship between texts is determined by the distance between vectors. This converted data is stored in HDFS and the collection number k is considered an input from the user. The number of centroids k is then randomly selected from the database.

It is difficult to determine which part of the kmeans should be used using the Map process and which part to use Minimize the MapReduce model. It is evident that the distance calculation in k-means iterative performance and calculation of the new centroid serial performance. Following the above-mentioned process of k methods, the same k methods use the system model MapReduce is designed. The mapper performs the multiplication function of calculating the distance between the vectors of data to centroids and the slider performs the serial function of calculating the new centroids.

**Conclusion:** Large databases are being created as a result of landscaping and digital integration in each field of work. To process such large data sets, traditional data integration techniques are introduced instead of the same implementation. Hadoop is a widely distributed framework that considers algorithms developed in the MapReduce model. The K-means algorithm is simulated using MapReduce and works on top of various Hadoop collections with 2 different data sets. Execution time obtained in parallel k-means compared with traditional methods k.

The proposed k-is traditionally successful k-means even the size of a small 3-node cluster. The larger the Hadoop collection, the better the performance of the available combination. It is also found that Hadoop processing speed is uneven and inconsistent with the acceleration speed in relation to the Hadoop cluster size.

## **6.Distributed Document Clustering Analysis Based on a Hybrid Method[xi]**

**Abstract:-** For analysing the big data, Apache Spark is the newest tool, large data sets perform in memory computation in a fault tolerant manner. For Big Data programming framework Map Reduce is a high-performance which is highly preferred by most big data analysts and is out there for a long time with a very good documentation. On the Hadoop Distributed File System, with the increase of the number of blocks run-time of both the Map Reduce and Spark programs also increases. Compare to Map Reduce Spark performs far more better. This shows that in future replacement of Map Reduce will be Spark.

**Introduction :-** As data mining was one of the most reliable sources of one of the largest sources of data. data sets in RDBMSs like MySQL stores actually more or less impossible of huge datasets. It can be either text or image formats as there is no specific formats of the data. For more accurate analysis and which may lead to more concrete decision-making resulting in greater operational efficiencies BIG DATA TECHNOLOGIES are provided. For managing and process huge volumes of structured and unstructured data in real-time we would require an infrastructure and can protect data privacy and security. Apache Hadoop, Apache Flume, and Apache Spark are big technologies have given us the power to capture. For very efficient and less costly ways these analyze this huge amount of data. to process data coming from sources like Twitter and getting the information we want we use these technologies.

- **Relational Database Management System:-** The data base which is used for the storage of relatively small to medium size data and for the access of data using realtime queries is known as Relational Database. fields are represented as columns and records are represented as rows in Relational Database.
- **Big Data Analytics :-** For real-time queries on relatively small and medium data sets MySQL database was designed. For large data sets or Big data analysis it was not supported. Sharing of Parallel DBMSs same capabilities as traditional, to the end user the distribution of data is transparent. high performance and reliability but much more expensive than traditional single-node RDBMS has offered in Parallel DBMS
- **Apache Hadoop :-** the cluster of low-end systems containing only hardware providing a cost-effective solution for Big data analysis therefore Hadoop can be deployed on a cluster. For scale up from single servers to thousands of machines Hadoop is designed. It contains different components it is not single entity, Apache Hadoop ha two main components which was HDFS and Map Reduce.

**Conclusion :-** Apache Hadoop, Apache Flume and Apache Spark are big data technologies to explore various fields. HDFS is type of Hadoop that can store very large amount of data in its file system in a fault tolerant manner, and can scale up to any size as needed. Map Reduce is another component of Hadoop that processing large data sets in a parallel fashion. cluster and their working methodology was monitored using Horton works Ambari Server for these frameworks were installed. With the help of data sets on HDFS Map Reduce and Spark programs were written.

## **7.A fuzzy approach to clustering of text documents based on Map Reduce[xii]**

**Introduction:**At times of rapid growth of information, the volume of data is directly proportional to the gradual evolution of the Internet technology, so extraction of information from huge amount of data is undoubtedly the key of the complete analysis. Clustering, involving the grouping of data points and manner for statistical data analysis used in many sectors like bio-medical, geo-spatial, data mining, marketing, machine learning, pattern recognition. In the analysis of clustering, the objects are set in such a way that objects in the same group (called as cluster) are more similar to each other than to those in other clusters. It itself is not one specific algorithm, but the general task to be solved. Here the basic issues being not efficient to convince the actual necessity of applications, speed of clustering not well built and not working productively. Luckily, MapReduce is now the approved distributed computing framework supplying a strong tool for handling for mass data. It basically, on the platform of Hadoop, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples. Analysis of logs, data analysis, genetic algorithms, user behaviour analysis etc. are the applications that use MapReduce.

**Methodology:**In the activity of pre-processing of text documents, there are the stop words that are to be ceased out from the document. If data is inconsistent, then there is possibility, leading to inaccurate results. The objective is to enhance the quality of data and at the same time reduces the difficulty of mining process. The weight of words is calculated by using TFIDF (Term Frequency, Inverse Document Frequency), computing formula. The theory of Information entropy states that the information entropy of words is inversely proportional to the frequency of word appearing in all documents. The amount of information entropy will be more if the word centralizes in some of the documents, considering the word of characteristic. The implementation of program is suitable for a fuzzy equivalent matrix by reducing operation of normalization to some extent, as it doesn't kill the time much.

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm in a cluster. This model is divided into two operations, Map and Reduce. A Map Reduce program is composed of a map procedure, which performs filtering and sorting, and a reduced method, which performs a summary operation. The characteristic of dividing the input data, organizing the execution of program, handling machine failures etc. are always the responsibility of run-time system of Map Reduce. The problem of clustering of text documents is overcome with the help of mapper and reducer by sort shuffling and the input and output as files in HDFS. The operation of coupling hence been the major for conquering the drawbacks in clustering of text documents.

**Conclusion:**In brief, the process of clustering of text documents heads to the inclusion of extraction, integration, analysis and interpretation of data. Although there are numerous models for clustering, the desired level value will strengthen the momentum of good clustering. Moreover, some parallel algorithms fail to implement the MapReduce framework but the nice blend of algorithm and platform made it to some extent. Most important being the redefining of key and value for the input format that managed to develop the efficiency of programs. Also, it's the text clustering that help you figure out the useful information from a large collection of documents.

## **8.Comparison of a Sequential and a MapReduce Approach to Joining Large Datasets[xiii]**

**Introduction:** MapReduce works on a number of processes that are responsible for processing very large data. Use MapReduce to perform such tasks involves using two functions: map and minimize. The map function converts the input into a collection (key, value) pairs. After that, the reduction function finds a set of all values by the same key and produces the final result based on combining these values. More recently, the explosion of information has become a well-known phenomenon causing data sets where applications work to grow to an extremely high level. Examples of such databases can be found in social networks where millions of users produce terabytes of data each day, e.g. Facebook stores at least 60 TB of new data every day. This type of system requires multiple join operations to match users with their content: comments, status updates, etc. In this paper, for the purpose of comparing a sequential solution to a distributed integration problem based on MapReduce, a large set of data was performed.

**Methodology:** Sequential joining function uses a well-known hash algorithm. In the simplest way, it begins to learn a little of the two tables it works on in an amazing hash table, where the hash value is calculated based on the merit attribute. After this, move through a larger table calculate the hash value of the joining attribute for each row, and then, using it to find rows in the hash table, thus achieving the join. always small enough to be stored in the in-memory hash table, so there is no need to use a more sophisticated scheme of partitioning the input set.

The use of MapReduce to join is using a method called repartition join. The map function can find and run a queue from one of two tables. It produces a pear (key, value) where the key is the value of the join and the value is the record itself, in addition to being marked with an identifier belonging to the table. The reduction function will find all records in both tables with the same value of joining qualification, thanks to the MapReduce framework. It then divides the records into two sets based on the value of the tag and makes a cartesian product into two sets, thus leading to a new set of integrated records.

**Conclusion:** This paper demonstrates the benefits of using parallelism, acquired by Amazon Elastic MapReduce, with successive implementation of large data sets. After this work was done sequentially, it was also done with four sets of numbers 1, 3, 5 and eight Amazon EC2. As the number of nodes increased, it took less time to complete the task. However, moving from one location to three points in the cluster provided a greater increase in performance than moving from five to eight nodes, because performance time also depends on the speed of access to data and the header of the input set. This leads to the conclusion that further node expansion does not improve performance significantly. Also, performance in one single location has shown better performance than sequential algorithm performance, because multiple processes can be run simultaneously on a single node in a cluster.

# Abstract:

## What is clustering?

*Clustering is the process of dividing the datasets into groups consisting of similar data points*

- *Points in the same group are similar as possible*
- *Points in the same group are dissimilar as possible*

*Clustering is also called as unsupervised learning technique to better understanding lets consider a group of dinners can be considered as a cluster or may be items arranged in a mall say eatables (fruits,vegetables,meat,etc) lets connect it with our clustering lets consider a table t1 and t2 where the people sitting on t1 are entirely related to each other and the people sitting on t2 are related on each other when related with people from t1 and t2 they are completely different from each. And to be more specific amazon show you recommendations based on your past searches or purchase history and even Netflix recommends you some movies or series based on your watch list and where as business field say bank sector where the clustering can be used for customer segmentation,fraud detection,risk factor and many more!*

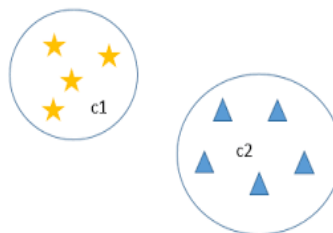
## Types Of Clustering?

*In General there are 3 types of clustering they are:*

- *Exclusive Clustering*
- *Overlapping Clustering*
- *Hierarchical Clustering*

*Exclusive Clustering:*

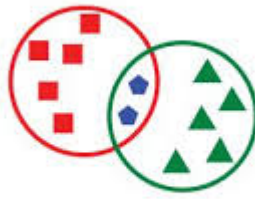
- *Hard Clustering*
- *Items belongs exclusively to one cluster*
- *Ex: K-Means Clustering*



Here all the stars(points) belong to cluster c1 only and the the triangles(points) belong to cluster c2 only

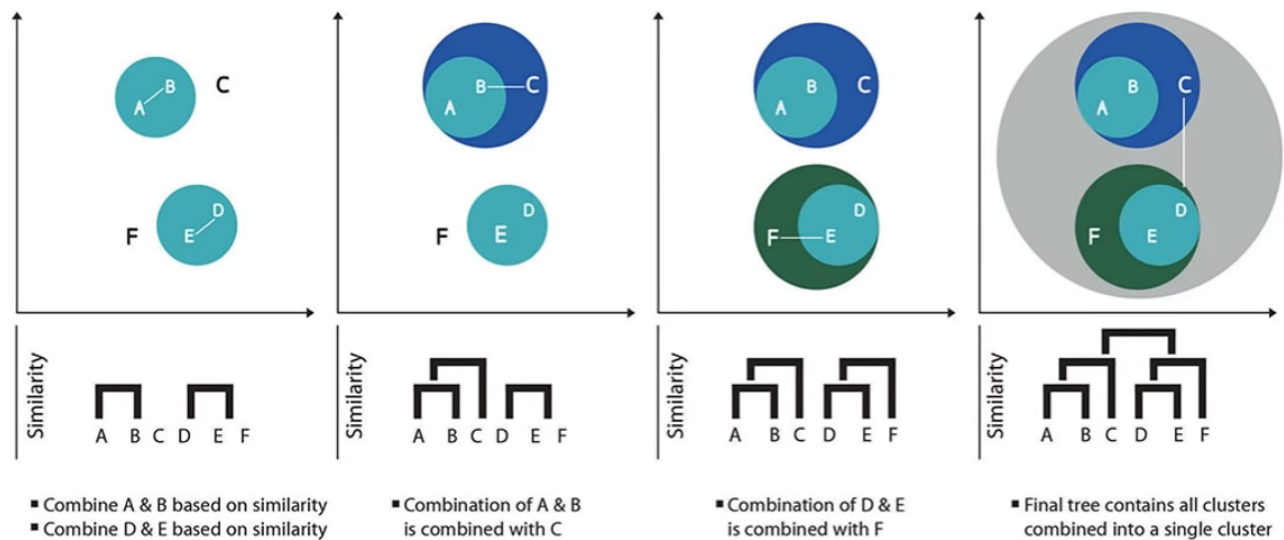
*Overlapping Clustering:*

- *Soft Clustering*
- *Items belong to multiple clusters*



### *Hierarchical Clustering:*

Its a combination and includes the above 2 types of clustering



### **What is Map-reduce?**

Map reduce is a programming model that simultaneously processes and analyzes huge data sets logically into separate clusters while map sorts the data, reduce segregates it into logical clusters thus removing the bad data and retaining the necessary information

#### **Step 1:**

#### **Mapper**

**Input:** <key 1 , value 1 >: <id i , sentence i > pairs

each id:1

for

dict i => Empty Dictionary

for each word in sentence:

```

        append 1 to list: dct i [word]
    return dict I

```

**Output:** dict i for each sentence.

**Step 2:**

## Shuffle and Sort

**Input :** dict i for each sentence

collect => Empty Dictionary

for each dict i :

for each word in dict i :

Add list items from: dict i [word] to the list: collect[word]

sort collect with respect to keys

**Output:** collect => dictionary having keys as each token and value as list of ones equal to number of occurrences in entire document.

**Step 3:**

## Reducer

**Input:** collect dictionary of intermediate key-value pairs

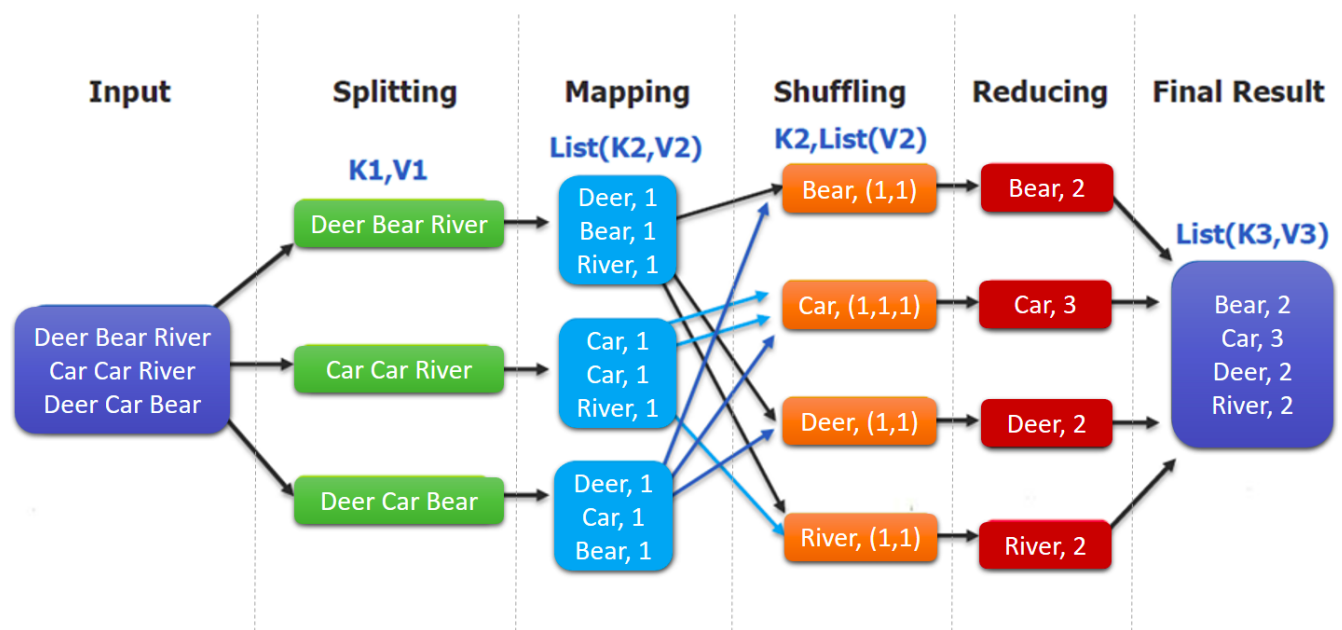
output = Empty dictionary

for each word in collect:

output[word] => sum of the list: collect[word]

return output

**Output =** dictionary having keys as tokens in the document and frequency





## What is K-Means Clustering

K-means is a clustering algorithm whose main goal is to group similar data points into a cluster where  $k$  represents the number of clusters the algorithm runs iteratively to assign each data point a cluster. We will be understanding it using an example: say I have a pile of different types of clothes (data points) and I need to wash them and there are different types of clothes, one needs to be washed in cold water and some need to be washed with some specifications and for them what I will do is say I have to make 3 clusters (types) and I will be picking them up 3 randomly from the pile of clothes and they will become my representation of my cluster and technically they are called centroids and the whole cluster relies on the centroid and the variance calculation takes place at every iteration and the best variance will be considered and we can fix the iterations for it and the centroid calculation will be on the mean distances from the initial point to the final point.

**Input :**  $S, k$  where  $S$ =Set of classified instances,  $k$ =integer

**Start**

**Initialize  $K$  random centroids**

**repeat**

**for all instances in  $S$  do**

$shortest \leftarrow 0$

$membership \leftarrow null$

**for all centroid  $c$  do**

$dist \leftarrow Distance(c)$

**if  $dist < shortest$  then**

$shortest \leftarrow distance$

$membership \leftarrow c$

**end if**

**end for**

**end for**

**Recalculate Centroids( $c$ )**

**until convergence**

**End**

**Output:**  $K$  Clusters

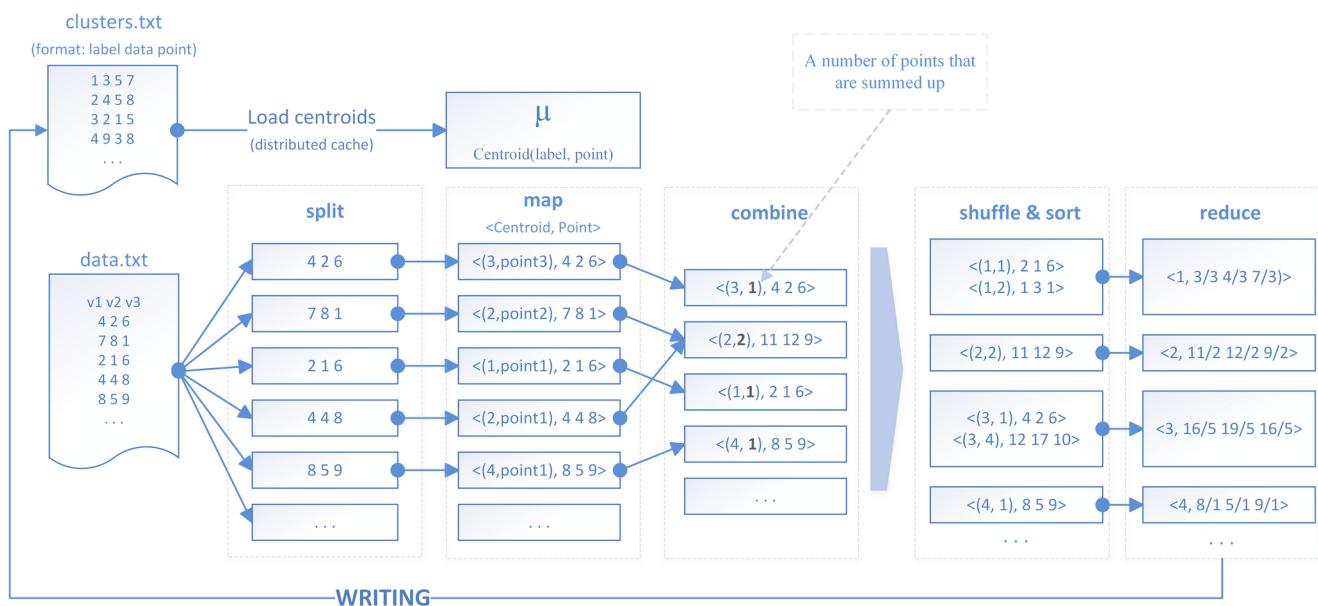
## Motivation and Objective:

Big Data is the current talk. Document Clustering is another trend in the fields of Machine Learning. Before applying Document Clustering, it is necessary to collect all distinct terms from each document and process them as tokens. Tokens are then grouped as Term-Frequency table using fast algorithms. One such algorithm is Map Reduce – an application of parallel and distributed computing in building the frequencies of each document faster than before. The master node distributes data among worker nodes which process the data in a parallel fashion for each document. Thus, our objective is to build Map Reduce paradigm

## Problem Statement :

Where big data and handling big data is always a big problem in the present generation where map reduce is on one of the way and to increase efficiency we will be combining map reduce with k-means and combiner clustering which reduces overheads

## Pseudoscope : ([G drive URL](#))



## **Methodology:**

### **Hadoop Installation:**

What is hadoop map reduce?

*Hadoop map-reduce is a processing component of apache hadoop and it processes data parallely in distributed environment and using this we will be processing information that present in the other component of hadoop say hadoop hdfs and the data will be stored in chunks and the processing will go there and processes the data*

### **Install Java**

*Apache Hadoop is a Java-based application. So you will need to install Java in your system. You can install it with the following command : sudo apt install default-openjdk -y*

### **Create Hadoop User**

*Create user Hadoop and we have to add this user to the sudoers group:*

```
sudo useradd hadoop  
sudo usermod -aG sudo hadoop
```

### **Login into the user hadoop:**

```
su - hadoop  
service ssh start : Initiates the ssh  
ssh-keygen -t rsa : generates the rsa finger print
```

### **Login into the SSH**

```
ssh localhost
```

### **Install Hadoop**

*First, log in with hadoop user and download the latest version of Hadoop with the following command:*

```
su - hadoop  
download Hadoop from the Hadoop server:
```

*Once the download is completed, extract the downloaded file with the following command:*

*Extract it and move them to the usr/local/*

*Next, move the extracted directory to the /usr/local/:*

**`sudo mv hadoop-* /usr/local/hadoop`**

*Change the hadoop users to sudoers group*

`chmod -aG hadoop sudo`

### **Configure Hadoop Variables:**

*You can do it by editing ~/.bashrc file:*

`nano ~/.bashrc`

*Add the following lines:*

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

*Save and close the file when you are finished.*

*Then, activate the environment variables with the following command:*

`source ~/.bashrc`

### **Configure Hadoop:**

***In this section, we will learn how to setup Hadoop on a single node.***

#### **Configure Java Environment Variables**

***Next, you will need to define Java environment variables in `hadoop-env.sh` to configure YARN, HDFS, MapReduce, and Hadoop-related project settings.***

*First, locate the correct Java path using the following command:*

#### **Configuration files:**

`sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh:`

```
GNU nano 5.3 /usr/local/hadoop/etc/hadoop/hadoop-env.sh
# Specify the JVM options to be used when starting the HDFS Balancer.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_BALANCER_OPTS=""

###
# HDFS Mover specific parameters
###
# Specify the JVM options to be used when starting the HDFS Mover.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_MOVER_OPTS=""

###
# Router-based HDFS Federation specific parameters
# Specify the JVM options to be used when starting the RBF Routers.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_DFSROUTER_OPTS=""

###
# HDFS StorageContainerManager specific parameters
###
# Specify the JVM options to be used when starting the HDFS Storage Container Manager.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HDFS_STORAGECONTAINERMANAGER_OPTS=""

###
# Advanced Users Only!
###

#
# When building Hadoop, one can add the class paths to the commands
# via this special env var:
# export HADOOP_ENABLE_BUILD_PATHS="true"

#
# To prevent accidents, shell commands be (superficially) locked
# to only allow certain users to execute certain subcommands.
# It uses the format of (command).(subcommand)_USER.
#
# For example, to limit who can execute the namenode command,
# export HDFS_NAMENODE_USER=hdfs
export JAVA_HOME=/usr/lib/jvm/java-14-openjdk-amd64
export HADOOP_CLASSPATH="$HADOOP_HOME/lib/*.jar"
```

```
hadoop@kali:~$ hadoop version
Hadoop 3.2.1
Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r b3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled by rohithsharmaks on 2019-09-10T15:56Z
Compiled with protoc 2.5.0
From source with checksum 776eaf9eee9c0ffc370bcbcb1888737
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.2.1.jar
hadoop@kali:~$
```

## Configure core-site.xml File

Next, you will need to specify the URL for your NameNode. You can do it by editing core-site.xml file:

`sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml`

```
GNU nano 5.3 /usr/local/hadoop/etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://0.0.0.0:9000</value>
    <description>The default file system URI</description>
  </property>
</configuration>
```

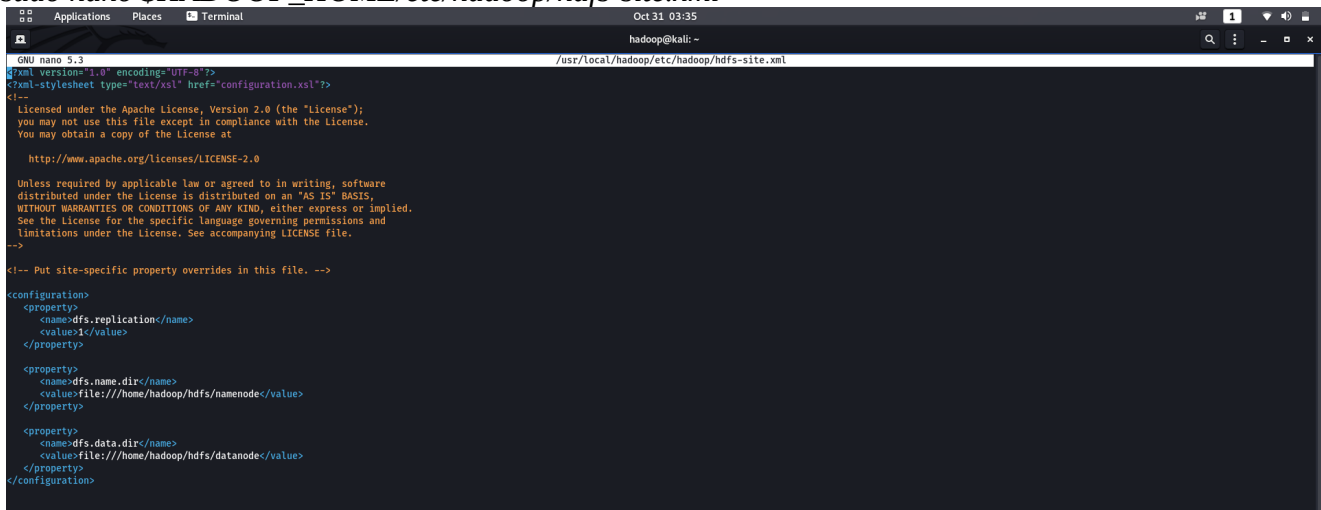
## Configure hdfs-site.xml File

Next, you will need to define the location for storing node metadata, fsimage file, and edit log file. You can do it by editing hdfs-site.xml file. First, create a directory for storing node metadata:

```
sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
sudo chown -R hadoop:hadoop /home/hadoop/hdfs
```

Next, edit the hdfs-site.xml file and define the location of the directory:

```
sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```



```
GNU nano 5.3 /usr/local/hadoop/etc/hadoop/hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

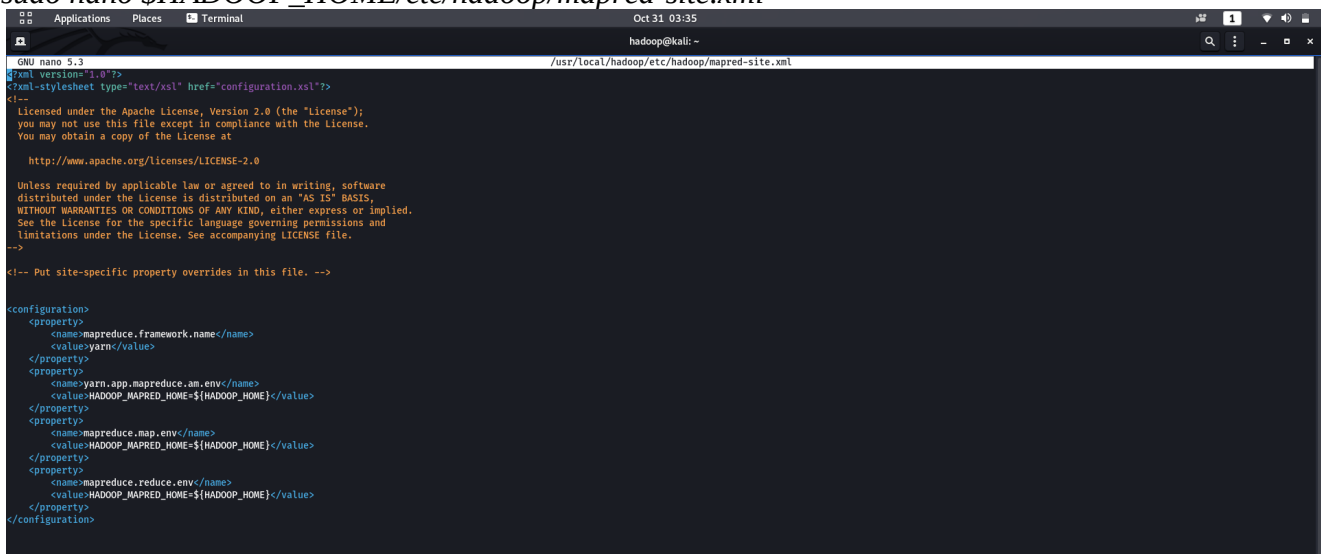
  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hdfs/datanode</value>
  </property>
</configuration>
```

## Configure mapred-site.xml File

Next, you will need to define MapReduce values. You can define it by editing mapred-site.xml file:

```
sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```



```
GNU nano 5.3 /usr/local/hadoop/etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

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

  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>

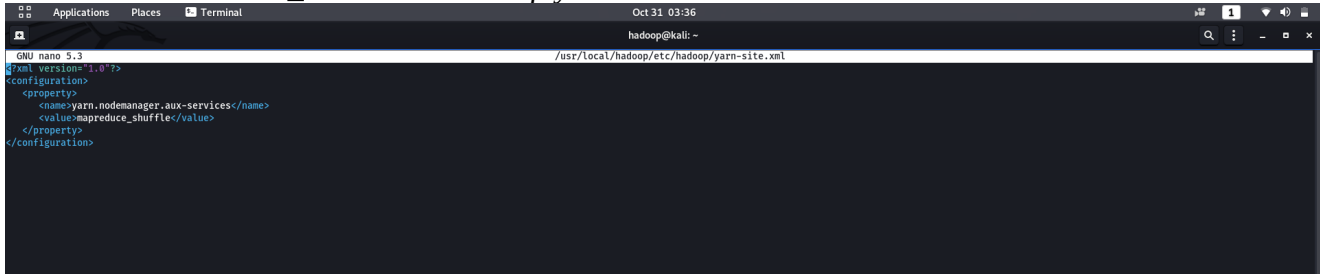
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>

  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
  </property>
</configuration>
```

## Configure yarn-site.xml File

Next, you will need to edit the yarn-site.xml file and define YARN related settings:

`sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml`



```
GNU nano 5.3 /usr/local/hadoop/etc/hadoop/yarn-site.xml
#xml version="1.0"?
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

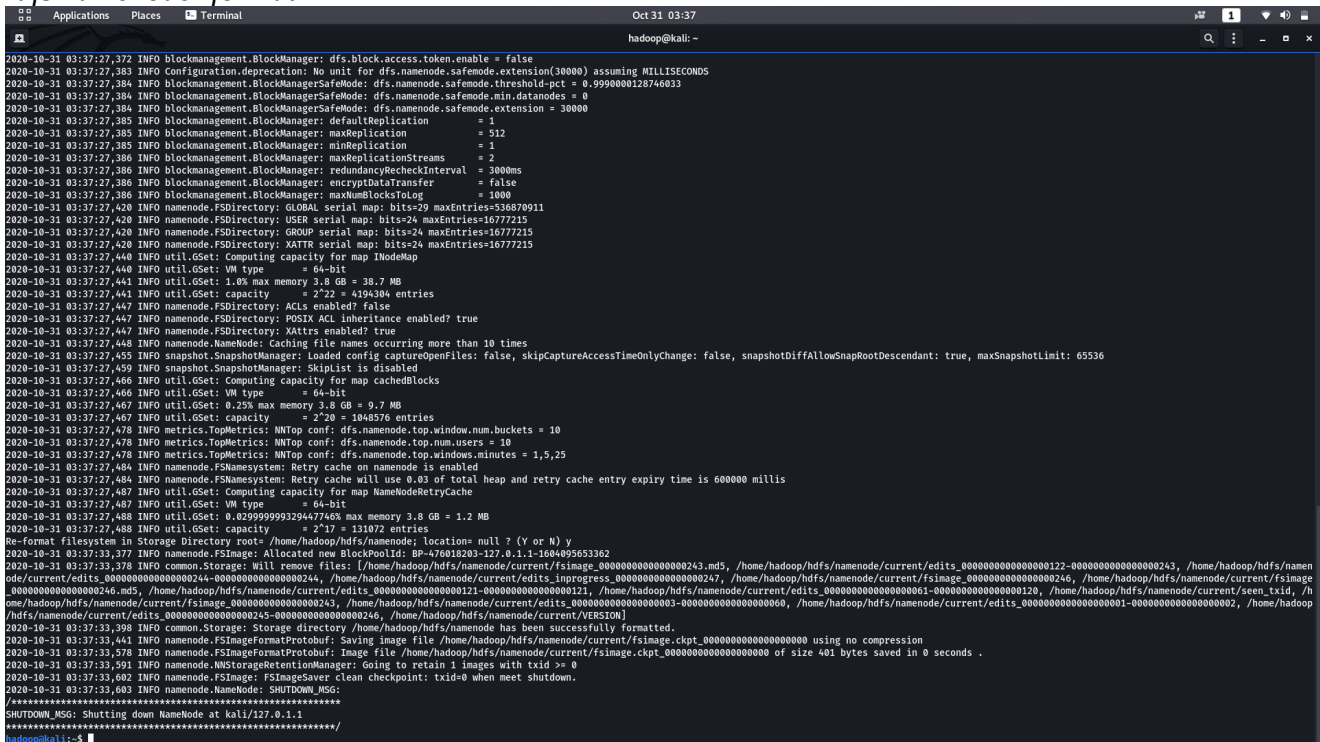
## Format HDFS NameNode

Next, you will need to validate the Hadoop configuration and format the HDFS NameNode.

First, log in with Hadoop user and format the HDFS NameNode with the following command:

`su – hadoop`

`hdfs namenode -format`



```
Oct 31 03:37
hadoop@kali: ~
2020-10-31 03:37:27,372 INFO blockmanagement.BlockManager: dfs.block.access.token.enable = false
2020-10-31 03:37:27,383 INFO Configuration.deprecation: No unit for dfs.namenode.safemode.extension(30000) assuming MILLISECONDS
2020-10-31 03:37:27,384 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
2020-10-31 03:37:27,384 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.min.datanodes = 0
2020-10-31 03:37:27,384 INFO blockmanagement.BlockManagerSafeMode: dfs.namenode.safemode.extension = 30000
2020-10-31 03:37:27,385 INFO blockmanagement.BlockManager: defaultReplication = 1
2020-10-31 03:37:27,385 INFO blockmanagement.BlockManager: maxReplication = 512
2020-10-31 03:37:27,385 INFO blockmanagement.BlockManager: minReplication = 1
2020-10-31 03:37:27,386 INFO blockmanagement.BlockManager: maxReplicationStreams = 2
2020-10-31 03:37:27,386 INFO blockmanagement.BlockManager: redundancyCheckInterval = 300ms
2020-10-31 03:37:27,386 INFO blockmanagement.BlockManager: encryptDataTransfer = false
2020-10-31 03:37:27,386 INFO blockmanagement.BlockManager: maxNumBlocksToLog = 1000
2020-10-31 03:37:27,420 INFO namenode.FSDirectory: GLOBAL serial map: bits=29 maxEntries=536870911
2020-10-31 03:37:27,420 INFO namenode.FSDirectory: USER serial map: bits=24 maxEntries=16777215
2020-10-31 03:37:27,420 INFO namenode.FSDirectory: GROUP serial map: bits=24 maxEntries=16777215
2020-10-31 03:37:27,420 INFO namenode.FSDirectory: XATTR serial map: bits=24 maxEntries=16777215
2020-10-31 03:37:27,440 INFO util.GSet: Computing capacity for map InodeMap
2020-10-31 03:37:27,440 INFO util.GSet: VM type = 64-bit
2020-10-31 03:37:27,441 INFO util.GSet: 1.0% max memory 3.8 GB = 38.7 MB
2020-10-31 03:37:27,441 INFO util.GSet: capacity = 2^22 = 4194304 entries
2020-10-31 03:37:27,442 INFO namenode.FSDirectory: ACLs enabled? false
2020-10-31 03:37:27,447 INFO namenode.FSDirectory: POSIX ACL inheritance enabled? true
2020-10-31 03:37:27,448 INFO namenode.NameNode: Caching file names occurring more than 10 times
2020-10-31 03:37:27,455 INFO snapshot.SnapshotManager: Loaded config captureOpenFiles: false, skipCaptureAccessTimeOnlyChange: false, snapshotDiffAllowSnapRootDescendant: true, maxSnapshotLimit: 65536
2020-10-31 03:37:27,459 INFO snapshot.SnapshotManager: Skiplist is disabled
2020-10-31 03:37:27,466 INFO util.GSet: Computing capacity for map cachedBlocks
2020-10-31 03:37:27,466 INFO util.GSet: VM type = 64-bit
2020-10-31 03:37:27,467 INFO util.GSet: 0.25% max memory 3.8 GB = 9.7 MB
2020-10-31 03:37:27,467 INFO util.GSet: capacity = 2^20 = 1048576 entries
2020-10-31 03:37:27,478 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.window.num.buckets = 10
2020-10-31 03:37:27,478 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
2020-10-31 03:37:27,478 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
2020-10-31 03:37:27,484 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
2020-10-31 03:37:27,484 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
2020-10-31 03:37:27,487 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2020-10-31 03:37:27,487 INFO util.GSet: VM type = 64-bit
2020-10-31 03:37:27,488 INFO util.GSet: 0.029999999329447746% max memory 3.8 GB = 1.2 MB
2020-10-31 03:37:27,488 INFO util.GSet: capacity = 2^17 = 131072 entries
2020-10-31 03:37:27,488 INFO namenode.FSImage: Allocated new BlockPoolId: BP-476018203-127.0.1.1-1604095653362
2020-10-31 03:37:33,378 INFO common.Storage: Will remove files: [/home/hadoop/hdfs/namenode/current/edits_00000000000000000243.md5, /home/hadoop/hdfs/namenode/current/edits_00000000000000000122-0000000000000000000243, /home/hadoop/hdfs/namenode/current/edits_0000000000000000000246.md5, /home/hadoop/hdfs/namenode/current/edits_0000000000000000000121-0000000000000000000121, /home/hadoop/hdfs/namenode/current/edits_0000000000000000000001-0000000000000000000120, /home/hadoop/hdfs/namenode/current/seen_txid, /home/hadoop/hdfs/namenode/current/edits_0000000000000000000245-0000000000000000000246, /home/hadoop/hdfs/namenode/current/VERSION]
2020-10-31 03:37:33,398 INFO common.Storage: Storage directory /home/hadoop/hdfs/namenode has been successfully formatted.
2020-10-31 03:37:33,441 INFO namenode.FSImageFormatProtobuf: Saving image file /home/hadoop/hdfs/namenode/current/fsimage.ckpt_00000000000000000000 using no compression
2020-10-31 03:37:33,578 INFO namenode.FSImageFormatProtobuf: Image file /home/hadoop/hdfs/namenode/current/fsimage.ckpt_00000000000000000000 of size 401 bytes saved in 0 seconds .
2020-10-31 03:37:33,591 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2020-10-31 03:37:33,602 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2020-10-31 03:37:33,603 INFO namenode.NameNode: SHUTDOWN_MSG:
#####
SHUTDOWN_MSG: Shutting down NameNode at kali/127.0.1.1
#####
hadoop@kali: ~
```

## Start the Hadoop Cluster:

`start-dfs.sh :`

You should get the following output:

```
hadoop@kali:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
Starting datanodes
Starting secondary namenodes [kali]
hadoop@kali:~$
```

*start-yarn.sh*

```
hadoop@kali:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
Starting datanodes
Starting secondary namenodes [kali]
hadoop@kali:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@kali:~$
```

*jps*

You should get the following output:

```
hadoop@kali:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
Starting datanodes
Starting secondary namenodes [kali]
hadoop@kali:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@kali:~$ jps
20897 Jps
20197 ResourceManager
19718 SecondaryNameNode
19381 NodeManager
19438 NameNode
hadoop@kali:~$
```

The code that does all this is shown below. If you are familiar with Java and the concept of Object Oriented Programming, this may look understandable to you. Otherwise, do not worry, we will simply use this as a tool to see what we can do with Map Reduce

## Algorithm for Proposed Model:

*Algorithm for Main.java:*

*X=set of d-dimensional objects*

*k=Cluster Numbers*

*d=convergence*

*C=Intial Set of Centroids*



1. Load  $X, C$
2.  $current\_centroids \leftarrow C$
3. initialise num Iter, finalClusters
4.  $C' \leftarrow$  perform Mapreduce
5.  $new\_centroids \leftarrow C'$
6.  $numIter \leftarrow 1$
7. while change ( $new\_centroids, current\_centroids$ )  $> d$  do
  1.  $current\_centroids \leftarrow new\_centroids$
  2.  $C' \leftarrow$  perform MapReduce
  3.  $new\_centroids \leftarrow C'$
  4.  $numIter \leftarrow numIter + 1$
8. end while
9. finalClusters  $\leftarrow$  perform finalClustering
10. return  $current\_centroids, final\ Clusters$

Output: A new set of Centroids, Final Clusters

Algorithm for Mapper with combiner:

Input: A subset of  $d$ -dimensional objects in each mapper initial set of centroids

1.  $M_1 \leftarrow \{x_1, x_2, x_3, \dots, x_m\}$
2.  $centroid\_centroids \leftarrow C$
3.  $distance \leftarrow \sqrt{p^2 - q^2}$  where  $p$  and  $q$  are coordinates
4. outputlist  $\leftarrow$  null
5.  $v = \{\}$
6. for all  $x_i$  belongs to  $M_1$  such that  $1 \leq i \leq m$  do
  1. bestCentroid  $\leftarrow$  null
  2. minDist  $\leftarrow \infty$
  3. for all in  $current\_centroids$  do
    1.  $dist \leftarrow distance(x_i, c)$
    2. if bestCentroid = null ||  $dist < minDist$  then
      1. bestCentroid  $\leftarrow c$
      2. minDist  $\leftarrow dist$
    3. end if
  4. end for
  5.  $v[bestCentroid] \leftarrow (x_i)$
  6.  $i += 1$
7. end for
8. for all centroid belongs to  $v$  do :
  1. localCentroid, sumofLocalobjets, numofLocalObjects  $\leftarrow$  null
  2. for all objects belong to  $v[centroid]$  do
    1. sumofLocalObjets  $+=$  object
    2. numofLocalObjects  $+= 1$
  3. end for
  4. localCentroid  $\leftarrow (sumofLocalObjects / numofLocalObjects)$
  5. outputlist  $\ll$  centroid, localCentroid
9. end for
10. return outputlist

### Algorithm for Reducer:

**Input:**(key,value) *key=best centroid and value=objects assigned to be centroids by the mapper*

1. *output*  $\leftarrow$  *outputlist from mappers*
2. *v*={}
3. *newCentroidList*  $\leftarrow$  *null*
4. *for all y belongs to outputlist do*
  1. *centroid*  $\leftarrow$  *y.key*
  2. *object*  $\leftarrow$  *y.value*
  3. *v[centroid]*  $\leftarrow$  *objective*
5. *end for*
6. *for all centroid belongs to v*
  1. *newCentroid,SumofObjects,numofObjects*  $\leftarrow$  *null*
  2. *for all object belongs to v do*
    1. *sumofObjects*+*object*
    2. *numofObjects*+*1*
  3. *end for*
  4. *newCentroid*  $\leftarrow$  (*sumOfObjects / numOfObjects*)
  5. *newCentroidList*  $\ll$  (*newCentriod*)
7. *end for*
8. *return newCentroidList*

### Conclusion:

In this project we used k means and map reduce and Implemented and applied map-reduce to the the k-means clustering in which we took images and generated points based on those images and we successfully implemented it made it to run on the hadoop cluster and we conclude that in map-reduce overheads occur due to the in between read and writes of the data and adding combiner in between helps and improves the performance a lot by decreasing the nu. Of in between reads and writes

## Output:

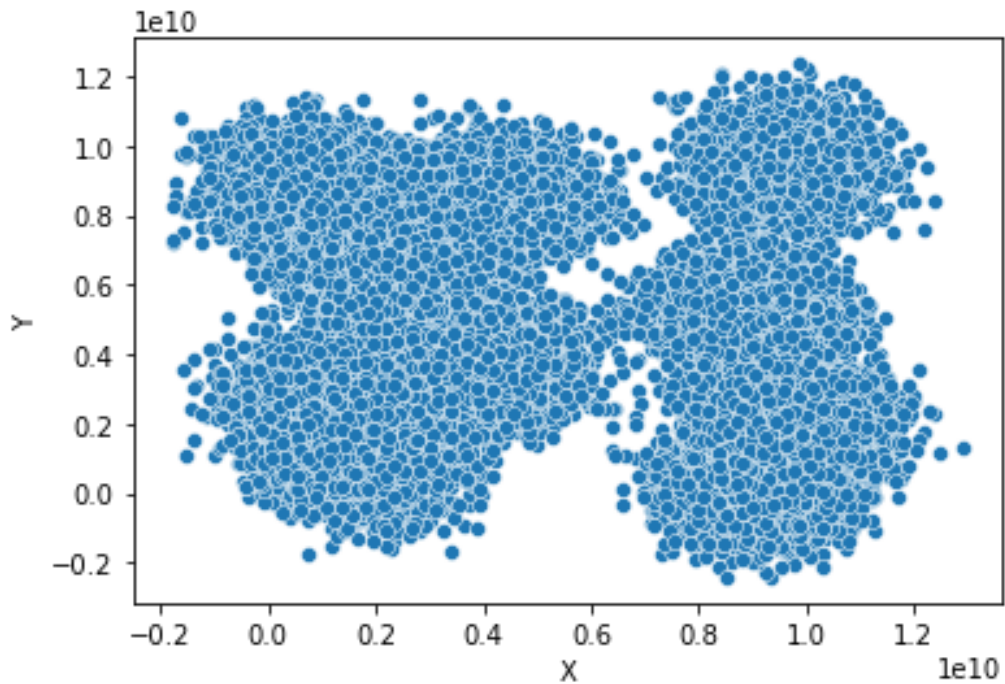
```
00 Applications Places Terminal Nov 3 23:57
hduser@kali: ~/PDC

FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=8663688
HDFS: Number of bytes written=2558
HDFS: Number of read operations=354
HDFS: Number of large read operations=0
HDFS: Number of write operations=56
HDFS: Number of bytes read erasure-coded=0

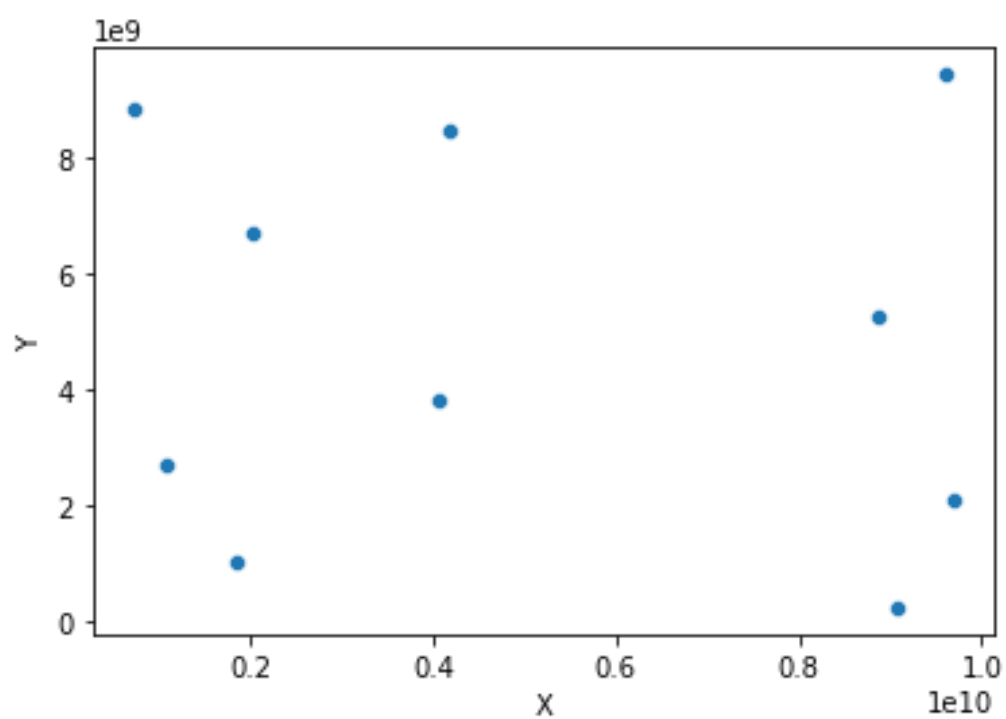
Map-Reduce Framework
  Map input records=33920
  Map output records=33920
  Map output bytes=2242073
  Map output materialized bytes=519
  Input split bytes=120
  Combine input records=33920
  Combine output records=10
  Reduce input groups=10
  Reduce shuffle bytes=519
  Reduce input records=10
  Reduce output records=10
  Spilled Records=20
  Shuffled Maps =3
  Failed Shuffles=0
  Merged Map outputs=3
  GC time elapsed (ms)=0
  Total committed heap usage (bytes)=2778726400

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
  enums.KMeansCounter
    ADJUSTED=0
  File Input Format Counters
    Bytes Read=1082185
  File Output Format Counters
    Bytes Written=425
  Counter value = 0
  Algorithm converged!
2020-11-03 23:44:34.174 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-11-03 23:44:35.980 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-11-03 23:44:37.086 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
1 2.019739154017452E9 6.69540936456472E9
2 6.880151288357885E8 8.893451043269579E9
3 8.8552153246214E9 5.25025892836759E9
4 1.830723114951087E9 2.7916486561886477E9
5 4.1687231168539834E9 8.501227521919083E9
6 9.721651721447945E9 2.1754124837393374E9
7 4.0732944278027844E9 3.811950489179925E9
8 9.580175414494256E9 9.44625802197994E9
9 1.8653308606892638E9 9.384418561792127E8
10 9.053275932688644E9 1.4540331195500597E8
hduser@kali:~/PDC$
```

## Dataset Plot[33919]:



Clusters Plot[10]



## References:

- i. <https://ieeexplore.ieee.org/document/6579448>
- ii. [https://link.springer.com/chapter/10.1007/978-3-642-10665-1\\_71](https://link.springer.com/chapter/10.1007/978-3-642-10665-1_71)
- iii. <https://github.com/shaypal5/awesome-twitter-data>
- iv. [https://www.gutenberg.org/ebooks/search/%3Fsort\\_order%3Ddownloads](https://www.gutenberg.org/ebooks/search/%3Fsort_order%3Ddownloads)
- v. <https://ieeexplore.ieee.org/abstract/document/7046097>
- vi. <https://ieeexplore.ieee.org/document/7868161>
- vii. <https://ieeexplore.ieee.org/document/8745928>
- viii. <https://ieeexplore.ieee.org/document/6065527>
- ix. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8325954>
- x. [https://www.researchgate.net/publication/326397176\\_A\\_study\\_and\\_Performance\\_Comparison\\_of\\_MapReduce\\_and\\_Apache\\_Spark\\_on\\_Twitter\\_Data\\_on\\_Hadoop\\_Cluster](https://www.researchgate.net/publication/326397176_A_study_and_Performance_Comparison_of_MapReduce_and_Apache_Spark_on_Twitter_Data_on_Hadoop_Cluster)
- xi. <https://ieeexplore.ieee.org/document/6643097>
- xii. <https://www.youtube.com/watch?v=SqvAaB3vK8U&t=1854s>
- xiii. <https://www.youtube.com/watch?v=JwnUJ42-JSE>
- xiv.