CHAPTER 1

1. INTRODUCTION

In Present scenario there is demand of quick and accurate result. As data on network is growing faster day by day so it needs to store in proper manner with power of easy retrieval. Volume, velocity and complexity of data generation are increasing day to day that requires modern tools to handle it and more important is within time limit. Traditional database is not sufficient for doing all these calculation under the time limit. Traditional database looks good to handle this situation but its "scale in" property bound user in limitations. Big data on Hadoop solve these problems up to certain limit.

In Big data processing input data is very large so traditionally it needs to split into small files first and then perform operations on it. For Big data there is use of MapReduce programming model that perform operation on single large file so that there is no need to split data before using it. As per records till January 2015, every minute usage of social networking site Facebook the Monthly Active Users has a count of 1.44 Billion, having largest number of users, generating share of 1482,478 pieces of contents, Youtube have a counts of 1 Billion Monthly Users, Google+ have 540 million Monthly Active Users, Instagram users counts to 300 million Monthly Active Users, Twitter have 302 billion Monthly Active Users and Linkedin 187million Monthly Active Users (Demchenko, Laat, & Membrey, 2014). If millions of the users are being active at a same moment of time consider the amount of data they are generating in a pass of seconds that data is out of reach to be maintained with the traditional data mining tools.

Consider the number of tweets being generated in a moment of time. Considering both, space and time for data maintenance it's not an easy task to handle data with the old traditional tools Companies like Facebook, Twitter, Linkedin etc start using Hadoop. Hadoop ecosystem includes MapReduce, Apache Hive, Apache Spark, PigLatin, Sqoop, flume, zookeeper and HBase. MapReduce is different than traditional databases as processing can occur on file system (unstructured data) or in column-oriented database.

1.1 Big Data

Big Data is becoming one of the most talked about technology trends nowadays. The real challenge with the big organization is to get maximum out of the data already available and predict what kind of data to collect in the future. How to take the existing data and make it meaningful that it provides us accurate insight in the past data is one of the key discussion points in many of the executive meetings in organizations. With the explosion of the data the challenge has gone to the next level and now a Big Data is becoming the reality in many organizations.

The term "Big Data" has launched a veritable industry of processes, personnel and technology to support what appears to be an exploding new field. Giant companies like Amazon and Wal-Mart as well as bodies such as the U.S. government and NASA are using Big Data to meet their business and/or strategic objectives. Big Data can also play a role for small or medium-sized companies and organizations that recognize the possibilities (which can be incredibly diverse) to capitalize upon the gains. In a nutshell, Big Data is your data. It's the information owned by your company, obtained and processed through new techniques to produce value in the best way possible. The 3D Model of Big Data is based on the following Vs (Elagib, Najeeb, Hashim, & Olanrewaju, 2014):

1. **Volume:** refers to management of data storage

2. **Velocity:** refers to the speed of data processing

3. **Variety:** refers to grouping data of different, seemingly unrelated data sets

1.2 Clustering

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. A loose definition of clustering could be *"the process of organizing objects into groups whose members are similar in some way"* (Li, Wu, Hu, Zhang, Li, & Wu, 2011). A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

1.3 K-Means Clustering

K-Means clustering algorithm was developed by J. MacQueen in 1967 and then by J.A. Hartigan and M.A. Wang around 1957 (Xu & Franti, 2004). K-Means clustering is one of the simplest unsupervised learning algorithms to classify or group your objects based on attributes, features into k number of group. K is positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

In our work, we have implemented this algorithm on a large dataset consisting of user reviews of mobile phones using MapReduce architecture. These reviews are taken from Amazon and the dataset has been taken from SNAP datasets. We are classifying the mobile phones into three clusters namely – good, average and bad based on the scores given by the user in each review.

Due to the large size of this dataset, such clustering could not be possible using serial implementation of K-Means algorithm. This algorithm has been chosen for clustering because it is simple to implement and requires less parameter. The use of Hadoop Distributed File System (HDFS), splits this large dataset into small chunks and then using MapReduce Architecture clustering is performed on these chunks parallely.

1.4 Thesis Outline

The thesis is divided into 5 chapters:

Chapter 1 is the introduction part. It describes the basic topics of the work, objective of the thesis and also the structure of the thesis.

Chapter 2 is the literature survey. It includes the description of the work and contribution of various people in the same field and their findings.

Chapter 3 is basically the research background. This chapter discusses Hadoop, MapReduce and clustering. Also a brief overview of big data, its characteristics, challenges faced by big data and its benefits. Workflow of MapReduce architecture is also discussed in this chapter.

Chapter 4 presents the implementation or proposed work and the results obtained after running the code on our setup.

Conclusion and future work are presented in **chapter 5**.

CHAPTER 2

2. LITERATURE REVIEW

Today structured or unstructured data is being produced enormously around the globe and is continue to increase. It is possibly impossible to handle or process this data by the organization applying existing Technology especially the unstructured data. Mining becomes computationally expensive to analyze and measure such huge data. Hadoop is becoming popular in organizations and they are using this Technology for storing and processing their large amount of data which is being developed every day. Hadoop works on the big data sets and where the data sets come into the picture, Clustering proposed in some way to organizing objects into groups whose members are similar. K-Means clustering is well known and very popular unsupervised learning algorithm.

The work proposed by Shuhua Ren, Alin Fan, they proposed coefficient of variation (CV-k-means) method which overcome the problem of commonly chosen Euclidean distance method.
They introduces variation coefficient weight vector to decrease the effects of irrelevant attributes in order to get more enhances and better results. The Euclidean distance method measures the data set features equally and does not gives the efficient clustering Results. The Proposed method resolves the problems. Introduces variation coefficient weight vector to decrease the effects of irrelevant attributes based on coefficient of variation.

The work proposed by Hai-Guang Li, Gong-Qing Wu, Xue-Gang Hu, Jing Zhang, Lian Li, Xindong Wu (Li, Wu, Hu, Zhang, Li, & Wu, 2011), they proposed K-Means Clustering with Bagging and MapReduce. As the K-Means clustering is most widely used technique to analyse the data and also it is used across all disciplines but it comes with some drawbacks likewise instable and sensitive to outliers and other problem that they encounter in their research work, is the K-means clustering algorithm gives insufficient results applying on large data sets. In this paper, they proposed an approach to encounter these problems: for the instability and sensitivity to outliers and for large data sets.

In order to solve those problems, they proposed a method named MBK means which adopts bagging to improve the stability and accuracy of k-means and uses MapReduce to clustering to solve the problem regarding large data sets.

The work proposed by HaiTao Yu, Xiaoxu Cheng, Meijuan Jia and Qingfeng Jiang (Yu, Cheng, Jia, & Jiang, 2013). They proposed Optimized K-Means Clustering Algorithm based on Artificial Fish Swarm. Their work is based on to optimize the K-Means method using Artificial Fish Swarm, which overcomes the problems of traditional K-Means algorithm and gets the optimal global clustering partition. To improve the precision of clustering algorithm, Artificial Fish algorithm is presented to calculate the inner-class distance and inter-class distance.

The work proposed by Weizhong Zhao, Huifang Ma, and Qing He (Zhao, Ma, & He, 2009). They proposed Parallel K-Means Clustering Based on MapReduce. In their research, they found that Data clustering is drawing attentions of the researcher of many fields such as data mining, document retrieval, image segmentation and pattern classification. As the K-Means clustering algorithm has many drawbacks, they propose a parallel k-means clustering algorithm based on MapReduce, which is a simple yet powerful parallel programming technique. They implemented their proposed approach in MapReduce Big data Programming framework. They evaluate the performance of their proposed algorithm with respect to speedup, scale up and size up.

In the work of Junjun Wang, Dongfeng Yuan and Mingyan Jiang (Wang, Yuan, & Jiang, 2012)**,** the authors have proposed an improved method called parallel K-PSO. As K-Means has limited processing capability because of its time complexity in serial scenario, improving the performance of K-Means has become challenging and significant. The K-Means Clustering Algorithm has following two inherent drawbacks:

i) Excessively depends on the initial cluster centers.

ii) Converges easily to the local optimum.

The idea proposed focuses on enhancing the global search capability of K-Means by taking the improved PSO to optimize its initial clusters, and improving the performance in dealing with massive data by transforming K-Means from serial to parallel using MapReduce.

CHAPTER 3

3. RESEARCH BACKGROUND

3.1 Big Data Overview

Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected.

3.1.1 Big Data Definition

Big data means really a big data; it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data; rather it has become a complete subject, which involves various tools, techniques and frameworks.

The 3D Model of Big Data is based on the following Vs:

1. Volume
2. Velocity
3. Variety

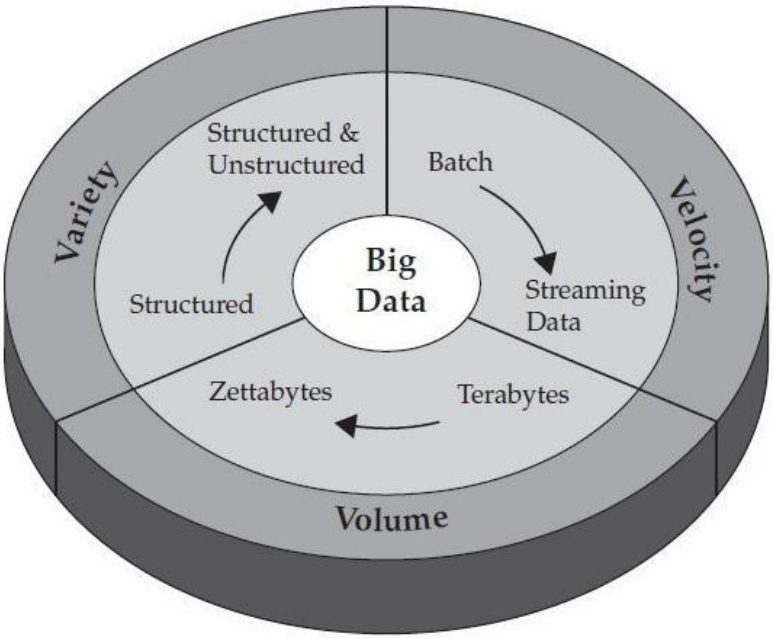


Fig. 3.1: 3-D Model of Big Data

- **Volume**

We currently see the exponential growth in the data storage as the data is now more than text data. We can find data in the format of videos, music and large images on our social media channels. It is very common to have Terabytes and Petabytes of the storage system for enterprises. As the database grows the applications and architecture built to support the data needs to be re-evaluated quite often. Sometimes the same data is re-evaluated with multiple angles and even though the original data is the same the new found intelligence creates explosion of the data. The big volume indeed represents Big Data.

- **Velocity**

The data growth and social media explosion have changed how we look at the data. There was a time when we used to believe that data of yesterday is recent. The matter of the fact newspapers is still following that logic. However, news channels and radios have changed how fast we receive the news. Today, people reply on social media to update them with the latest happening. On social media sometimes a few seconds old messages (a tweet, status updates etc.) is not something interests users. They often discard old messages and pay attention to recent updates. The data movement is now almost real time and the update window has reduced to fractions of the seconds. This high velocity data represent Big Data.

- **Variety**

Data can be stored in multiple format. For example database, excel, csv, access or for the matter of the fact, it can be stored in a simple text file. Sometimes the data is not even in the traditional format as we assume, it may be in the form of video, SMS, pdf or something we might have not thought about it. It is the need of the organization to arrange it and make it meaningful. It will be easy to do so if we have data in the same format, however it is not the case most of the time. The real world have data in many different formats and that is the challenge we need to overcome with the Big Data. This variety of the data represent Big Data.

3.1.2 What Comes Under Big Data?

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data (Armour, Kaisler, & Espinosa, 2015).

- **Black Box Data**: It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Social Media Data:** Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- **Stock Exchange Data:** The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.
- **Power Grid Data:** The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data:** Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data:** Search engines retrieve lots of data from different databases.

Fig. 3.2: Big Data Generation from several Areas

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types:

- **Structured data :** Relational data
- **Semi Structured data:** XML data
- **Unstructured data:** Word, PDF, Text, Media Logs

3.1.3 Benefits of Big Data

Big data is really critical to our life and its emerging as one of the most important technologies in modern world. Follow are just few benefits which are very much known to all of us:

- Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
- Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.

- Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

3.1.4 Big Data Technologies

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business.

To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real time and can protect data privacy and security.

There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data (Frank, Kaisler, & Espinosa, 2015).

3.1.5 Operational Big Data

This includes systems like MongoDB that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

NoSQL Big Data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow massive computations to be run inexpensively and efficiently. This makes operational big data workloads much easier to manage, cheaper, and faster to implement.

Some NoSQL systems can provide insights into patterns and trends based on real-time data with minimal coding and without the need for data scientists and additional infrastructure.

3.1.6 Analytical Big Data

This includes systems like Massively Parallel Processing (MPP) database systems and MapReduce that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

MapReduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines.

These two classes of technology are complementary and frequently deployed together.

Table 1: Operational vs. Analytical Systems **(Verma, Patel, & Patel, 2015)**

	Operational	**Analytical**
Latency	1 ms - 100 ms	1 min - 100 min
Concurrency	1000 - 100,000	1 - 10
Access Pattern	Writes and Reads	Reads
Queries	Selective	Unselective
Data Scope	Operational	Retrospective
End User	Customer	Data Scientist
Technology	NoSQL	MapReduce, MPP Database

3.1.7 Big Data Challenges

The major challenges associated with big data are as follows (Armour, Kaisler, & Espinosa, 2015):

a. Capturing data
b. Curation
c. Storage
d. Searching
e. Sharing
f. Transfer
g. Analysis
h. Presentation

To fulfill the above challenges, organizations normally take the help of enterprise servers.

3.1.8 Big Data Solutions

I. Traditional Approach

In this approach, an enterprise will have a computer to store and process big data. Here data will be stored in an RDBMS like Oracle Database, MS SQL Server or DB2 and sophisticated software can be written to interact with the database, process the required data and present it to the users for analysis purpose.

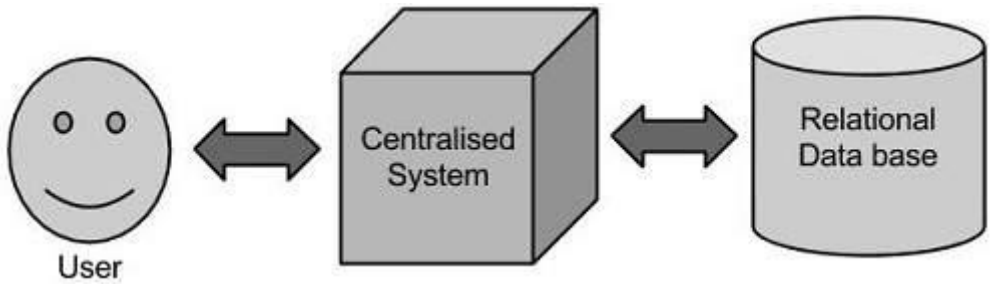


Fig. 3.3: Traditional Approach

❖ **Limitation**

This approach works well where we have less volume of data that can be accommodated by standard database servers, or up to the limit of the processor which is processing the data. But when it comes to dealing with huge amounts of data, it is really a tedious task to process such data through a traditional database server.

II. Google's Solution

Google solved this problem using an algorithm called MapReduce. This algorithm divides the task into small parts and assigns those parts to many computers connected over the network, and collects the results to form the final result dataset.

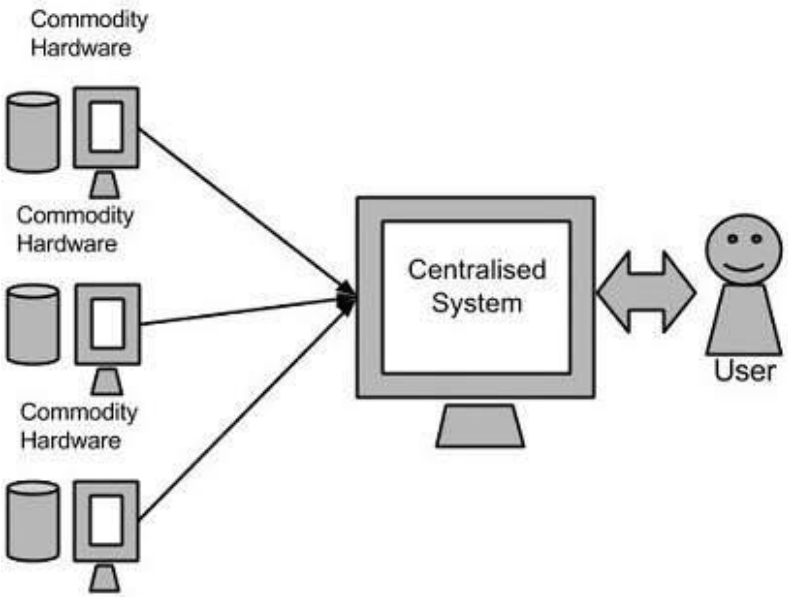


Fig. 3.4: Google's Approach

Above diagram shows various commodity hardware which could be single CPU machine or servers with higher capacity.

3.2 Hadoop

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Daug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

3.2.1 Hadoop Introduction

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

3.2.2 Hadoop Architecture

Hadoop framework includes following four modules (Verma, Patel, & Patel, 2015):

A. **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provide file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

B. **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.

C. **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data.

D. **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.

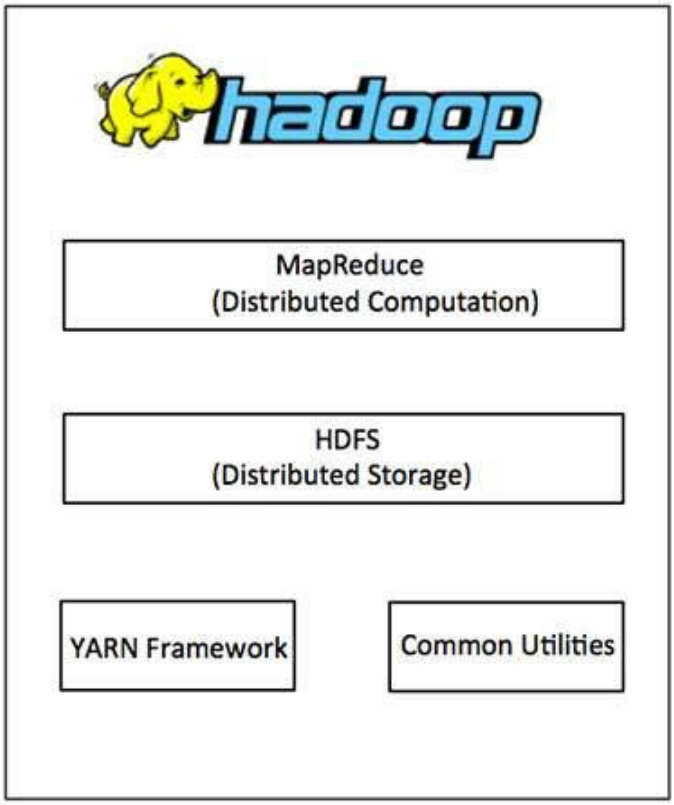


Fig. 3.5: Hadoop Architecture

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

3.2.3 MapReduce

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner (Maitrey & Jha, 2015).

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform (Bing & Chan, 2014):

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave **TaskTracker** per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of

machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

❖ **The Algorithm** (Gohil, Garg, & Panchal, 2014)

- Generally MapReduce paradigm is based on sending the computer to where the data resides!

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

  - **Map stage**: The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

  - **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

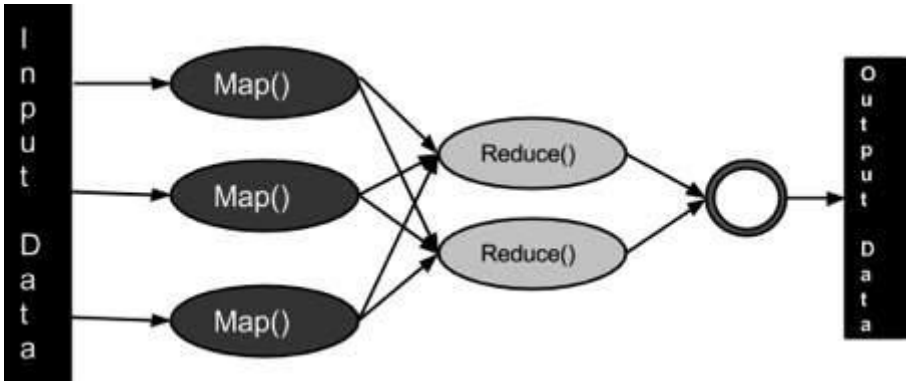


Figure 3.6: Mapper and Reducer Tasks

❖ Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job (Gohil, Garg, & Panchal, 2014):

(Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3> (Output).

Table 2: Inputs & Outputs of MapReduce

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

❖ **Terminology** (Maitrey & Jha, 2015)

i. **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.

ii. **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.

iii. **NameNode** - Node that manages the Hadoop Distributed File System (HDFS).

iv. **DataNode** - Node where data is presented in advance before any processing takes place.

v. **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.

vi. **SlaveNode** - Node where Map and Reduce program runs.

vii. **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.

viii. **Task Tracker** - Tracks the task and reports status to JobTracker.

ix. **Job** - A program is an execution of a Mapper and Reducer across a dataset.

x. **Task** - An execution of a Mapper or a Reducer on a slice of data.

xi. **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

3.2.4 Hadoop Distributed File System

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop is the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single **NameNode** that manages the file system metadata and one or more slave **DataNodes** that store the actual data.

A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

HDFS provides a shell like any other file system and a list of commands are available to interact with the file system. These shell commands will be covered in a separate chapter along with appropriate examples.

3.2.4.1 HDFS Overview

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

3.2.4.2 Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of NameNode and DataNode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

3.2.4.3 HDFS Architecture

Given below is the architecture of a Hadoop File System.

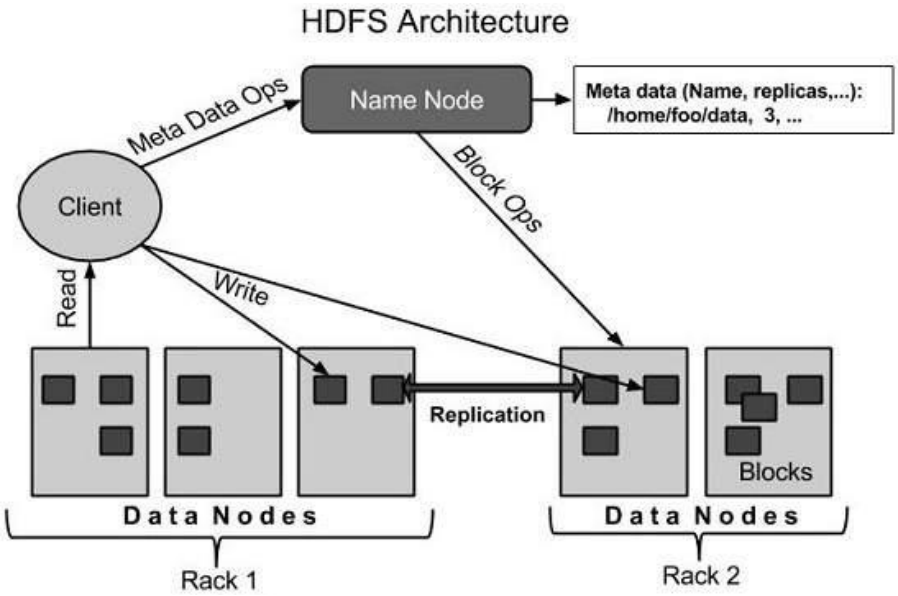


Figure 3.7: HDFS Architecture

HDFS follows the master-slave architecture and it has the following elements:

i. Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

ii. Datanode

The DataNode is a commodity hardware having the GNU/Linux operating system and DataNode software. For every node (Commodity hardware/System) in a cluster, there will be a DataNode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.

- They also perform operations such as block creation, deletion, and replication according to the instructions of the NameNode.

iii. Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

3.2.4.4 Goals of HDFS

I. **Fault detection and recovery**: Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

II. **Huge datasets**: HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

III. **Hardware at data**: A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

3.2.5 Advantages of Hadoop

- ✓ Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.

- ✓ Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.

- ✓ Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

3.3 CLUSTERING

Clustering is one of the important methods in the field of data mining. It is a powerful tool for knowledge acquisition from unlabeled data. It is a typical unsupervised learning technique for grouping a set of physical and abstract objects into classes of similar objects. The objects are clustered or grouped based on the principle of maximizing the intra class similarity and minimizing the inter class similarity. Thus cluster is a collection of data objects that are characterized by high similarity within the cluster and dissimilarity between clusters.

3.3.1 Clustering Methods (Xu & Franti, 2004)

3.3.1.1 Hierarchical Methods:

These methods construct the clusters by recursively partitioning the instances in either a agglomerative (bottom-up) or divisive (top-down) fashion.

i. **Agglomerative:** algorithms start with each object being a separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all objects are in a single group or at any other point the user wants. These methods generally follow a greedy-like bottom-up merging.

ii. **Divisive:** algorithms follow the opposite strategy. They start with one group of all objects and successively split groups into smaller ones, until each object falls in one cluster, or as desired. Divisive approaches divide the data objects in disjoint groups at every step, and follow the same pattern until all objects fall into a separate cluster. This is similar to the approach followed by divide-and-conquer algorithms.

3.3.1.2 Partitional:

Partitional clustering algorithm constructs partitions of the data, where each cluster optimizes a clustering criterion, such as the minimization of the sum of squared distance from the mean within each cluster. Partitional Clustering algorithms try to locally improve a certain criterion. First, they compute the values of the similarity or distance, they order the results, and pick the one that optimizes the criterion. Hence, the majority of them could be considered as greedy-like algorithms.

3.3.1.3 Density-Based Clustering:

These algorithms group objects according to specific density objective functions. Density is usually defined as the number of objects in a particular neighborhood of a data objects. In these approaches a given cluster continues growing as long as the number of objects in the neighborhood exceeds some parameter. This is considered to be different from the idea in partitional algorithms that use iterative relocation of points given a certain number of clusters.

3.3.1.4 Grid-Based Clustering:

The main focus of these algorithms is spatial data, *i.e.*, data that model the geometric structure of objects in space, their relationships, properties and operations. The objective of these algorithms is to quantize the data set into a number of cells and then work with objects belonging to these cells. They do not relocate points but rather build several hierarchical levels of groups of objects. In this sense, they are closer to hierarchical algorithms but the merging of grids, and consequently clusters, does not depend on a distance measure but it is decided by a predefined parameter.

3.3.1.5 Model-Based Clustering:

These algorithms find good approximations of model parameters that best fit the data. They can be either partitional or hierarchical, depending on the structure or model they hypothesize about the data set and the way they refine this model to identify partitionings. They are closer to density-based algorithms, in that they grow particular clusters so that the preconceived model is improved. However, they sometimes start with a fixed number of clusters and they do not use the same concept of density.

3.3.1.6 Categorical Data Clustering:

These algorithms are specifically developed for data where Euclidean, or other numerical-oriented, distance measures cannot be applied. In the literature, we find approaches close to both partitional and hierarchical methods.

3.3.2 Characteristics of a good Clustering technique

i. ***Scalability*:** The ability of the algorithm to perform well with large number of data objects (tuples).

ii. ***Analyze mixture of attribute types***: The ability to analyze single as well as mixtures of attribute types.

iii. ***Find arbitrary-shaped clusters***: The shape usually corresponds to the *kinds* of clusters an algorithm can find and we should consider this as a very important thing when choosing a method, since we want to be as general as possible. Different types of algorithms will be biased towards finding different types of cluster structures/shapes and it is not always an easy task to determine the shape or the corresponding bias. Especially when categorical attributes are present we may not be able to talk about cluster structures.

iv. ***Minimum requirements for input parameters*:** Many clustering algorithms require some user-defined parameters, such as the number of clusters, in order to analyze the data. However, with large data sets and higher dimensionalities, it is desirable that a method require only limited guidance from the user, in order to avoid bias over the result.

v. ***Handling of noise***: Clustering algorithms should be able to handle deviations, in order to improve cluster quality. Deviations are defined as data objects that depart from generally accepted norms of behavior and are also referred to as outliers. Deviation detection is considered as a separate problem.

vi. ***Sensitivity to the order of input records*:** The same data set, when presented to certain algorithms in different orders, may produce dramatically different results. The order of input mostly affects algorithms that require a single scan over the data set, leading to locally optimal solutions at every step. Thus, it is crucial that algorithms be insensitive to the order of input.

vii. ***High dimensionality of data*:** The number of attributes/dimensions in many data sets is large, and many clustering algorithms cannot handle more than a small number (eight to ten) of dimensions. It is a challenge to cluster high dimensional data sets, such as the U.S. census data set which contains 138 attributes.

viii. ***Interpretability and usability***: Most of the times, it is expected that clustering algorithms produce usable and interpretable results. But when it comes to comparing the results with preconceived ideas or constraints, some techniques fail to be satisfactory. Therefore, easy to understand results are highly desirable.

Having the above characteristics in mind, we present some of the most important algorithms that have influenced the clustering community. We will attempt to criticize them and report which of the requirements they meet or fail to meet. We shall treat clustering algorithms for categorical data in a separate section.

3.4 The K-MEANS Clustering Algorithm

K-means is a typical clustering algorithm in data mining and which is widely used for clustering large set of data. In 1967, MacQueen firstly proposed the k-means algorithm (Yu, Cheng, Jia, & Jiang, 2013); it was one of the most simple, non-supervised learning algorithms, which was applied to solve the problem of the well-known cluster. It is a partitioning clustering algorithm, this method is to classify the given date objects into k different clusters through the iterative, converging to a local minimum. So the results of generated clusters are compact and independent.

The algorithm consists of two separate phases (Dashti, Simas, Ribeiro, Assadi, & Moitinho, 2010). The first phase selects k centers randomly, where the value k is fixed in advance. The next phase is to take each data object to the nearest center. Euclidean distance is generally considered to determine the distance between each data object and the cluster centers. When all the data objects are included in some clusters, the first step is completed and an early grouping is done. Recalculating the average of the early formed clusters. This iterative process continues repeatedly until the criterion function becomes the minimum. Supposing

that the target object is x, xi indicates the average of cluster Ci, criterion function is defined as follows (Liao, Yang, & Zhao, 2013):

$$E=\sum_{i=1}^{k}\sum_{x\in C_i}\left|x-x_i\right|^2$$

E is the sum of the squared error of all objects in database. The distance of criterion function is Euclidean distance, which is used for determining the nearest distance between each data object and cluster center. The Euclidean distance between one vector x=(x1, x2…xn) and another vector y=(y1 ,y2 ,…yn), The Euclidean distance d(xi, yi) can be obtained as follows:

$$d(x_i,\ y_i)=\left[\sum_{i=1}^{n}\left(x_i-y_i\right)^2\right]^{1/2}$$

The process of k-means algorithm as follow (Ren & Fan, 2011):

Input:

Number of desired clusters, k, and a database D = {d1, d2,…dn} containing n data objects.

Output:

A set of k clusters

Steps:

1) Randomly select k data objects from dataset D as initial cluster centers.

2) Repeat;

3) Calculate the distance between each data object di (1 <=i<=n) and all k cluster centers cj(1<=j<=k) and assign data object di to the nearest cluster.

4) For each cluster j (1<=j<=k), recalculate the cluster center.

5) Until no changing in the center of clusters.

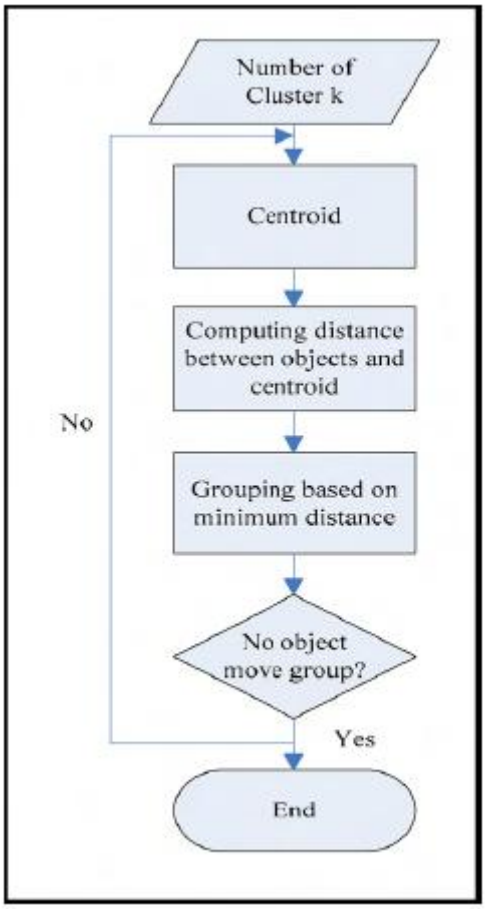


Figure 3.8: Flowchart of K-Means Clustering Algorithm

The k-means clustering algorithm always converges to local minimum. Before the k-means algorithm converges, calculations of distance and cluster centers are done while loops are executed a number of times, where the positive integer t is known as the number of k-means iterations. The precise value of t varies depending on the initial starting cluster centers. The distribution of data points has a relationship with the new clustering center, so the computational time complexity of the K-means algorithm is O(nkt). n is the number of all data objects, k is the number of clusters, t is the iterations of algorithm.

Usually requiring k <<n and t <<n (Wang S. , 2013).

CHAPTER 4

IMPLEMENTATION

In this work we have implemented K-Means clustering algorithm on a large data set of mobile reviews. The data set contains millions of user reviews of various mobile phones extracted from Amazon. The data available is in unstructured form containing following fields:

- **product/productId**: id of the product
- **product/title**: title of the product
- **product/price**: price of the product
- **review/userId**: id of the user
- **review/profileName**: name of the user
- **review/helpfulness**: fraction of users who found the review helpful
- **review/score**: rating of the product
- **review/time**: time of the review (unix time)
- **review/summary**: review summary
- **review/text**: text of the review

K-mean clustering algorithm is implemented to create three clusters of mobiles to represent the class to which a particular mobile belongs to namely **GOOD**, **AVERAGE** and **BAD.** The clusters are formed on the basis of user reviews using the information of score provided in each reviews. Since there are several reviews corresponding to a single mobile phone we have taken average of all the scores for respective mobile phones.

I. Structuring of data

Firstly this unstructured data is transformed into structured data using a MapReduce code. In map function the data is split into tuples and various available fields are extracted from each review and are stored in respective array list. Next the reducer gets executed in which all the fields are written to a file in tabular form converting list it to a structured data.

II. Calculating average score:

There are reviews of many users corresponding to a single mobile phone in the dataset. Thus, for classifying the mobiles into 3 classes on the basis of score, we need the average of score given by all the users for a mobile phone.

This task is accomplished in the reducer itself while structuring the data.

❖ **Pseudo code:**

Map (Key, Value, Context)

1. Split tuples into various fields
2. Store the fields into array – list

Cleanup (Context)

1. Extract fields from array – list
2. Write the fields into an intermediate file in tabular form

Reducer (Key, Value, Context)

1. Extract the score field from each tuple iteratively
2. Compute the sum of score of each mobile phone respectively
3. Count the number of reviews of a particular prod id
4. Compute the average score
5. Write the prod id average score in tabular form

III. Clustering of mobile phones on the basis of average score:

❖ **Algorithm:**

Step1: Initially randomly centroid is selected based on data. In our implementation we used 3 centroids.

Step2: The Input file contains the data points.

Step3: Mapper read the data file and emits the nearest centroid with the point to the reducer.

Step4: Reducer collects all this data and calculates the new corresponding centroids and emit.

Step5: In the job configuration, we are reading both files and checking

if difference between old and new centroid is less than 0.1 then

convergence is reached

else

repeat step 3 with new centroids.

RESULTS

Following results were obtained on implementing K-Means clustering algorithm in a parallel fashion on big data using MapReduce architecture.

I. Cluster formation

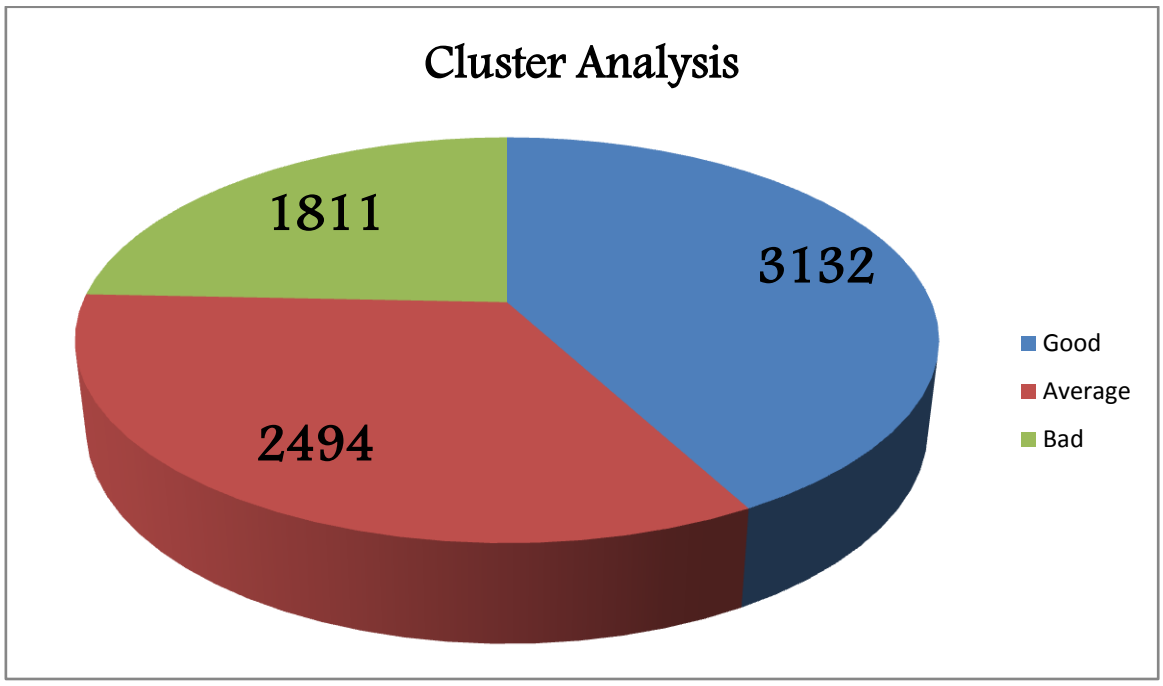
We took the value of 'K' as 3 since we have to classify the mobile phones into three classes namely good, average and bad. We considered following rule to categorize the mobile phones into these classes:

Good if average score >= 4

Average if 2.5 < average score < 4

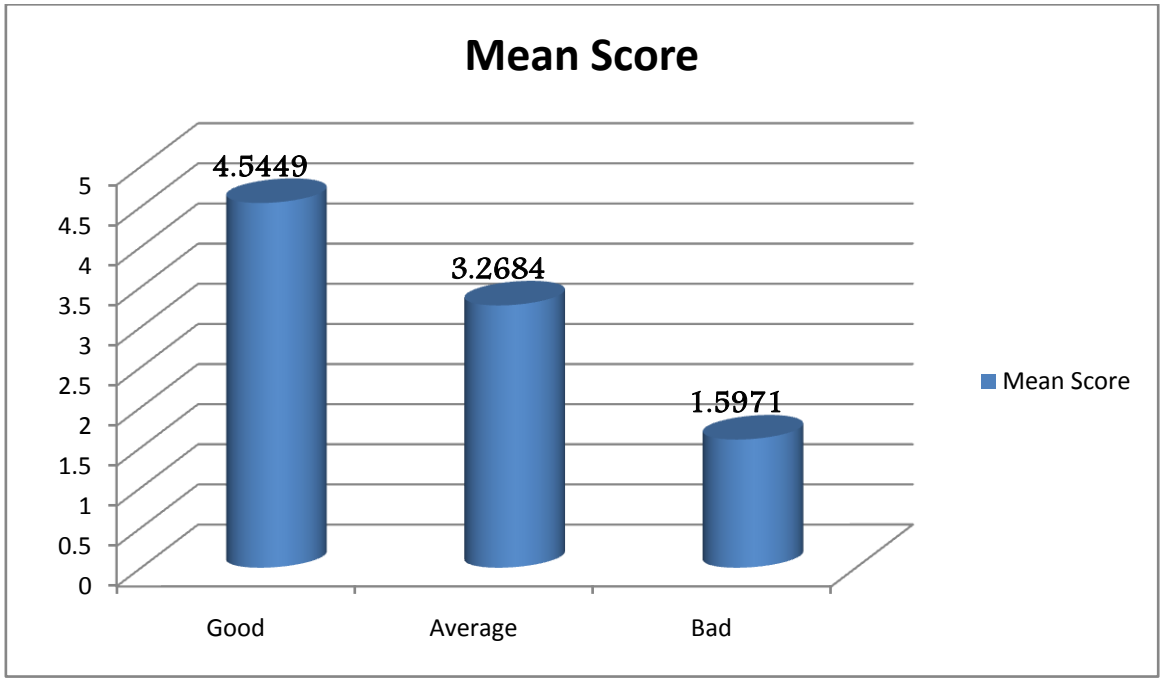
Bad if average score <=2.5.

There were total 7437 mobile phones which were used as data points for clustering. Following clusters were obtained:



II. Mean Value of Each Cluster

After cluster formation, we computed the mean value of each cluster to know the average score of each cluster. The result obtained is as follows:



CHAPTER 5

CONCLUSION AND FUTURE WORK

As data clustering has attracted a significant amount of research attention. Many clustering algorithm have been proposed in the past decades, however the enlarging data in applications makes clustering of very large scale of data a challenging task. We have chosen K-Means clustering algorithm since it is easy to implement and requires less parameters to work (Guoli, LangFang, Tingting, Yanping, Limei, & Jinqiao, 2013). Also past result shows that this algorithm provides good quality results.

The K-Means clustering algorithm can be easily parallelized using MapReduce architecture and thus can be used for clustering of large dataset. The problem of storing large dataset is easily handled by Hadoop Distributed File System (HDFS). Hadoop itself splits the whole dataset into small chunks and distributes it to all the datanodes. The NameNode keeps the metadata of this distribution. The experimental results shows that the clustering can be achieved more efficiently in less number of iterations using this model than its serial implementation. The result shows that clustering of large dataset can be achieved using MapReduce architecture using commodity hardware effectively.

In future work the proposed model should be modified such that it could work upon a big dataset involving large number of dimensions. Also some work could be done in order to improve the execution time of the algorithm examining other features of MapReduce architecture like partitioner, combiner etc. which may reduce the processing.