International Journal of Computer Applications (0975 – 8887)

Volume 72– No.8, May 2013

Verification and Validation of MapReduce Program model for Parallel K-Means algorithm on Hadoop Cluster

Amresh Kumar
Department of Computer Science & Engineering, Christ University Faculty of Engineering Bangalore, Karnataka, India

Kiran M.
Department of Computer Science & Engineering, Christ University Faculty of Engineering Bangalore, Karnataka, India

Saikat Mukherjee
Department of Computer Science & Engineering, Christ University Faculty of Engineering Bangalore, Karnataka, India

Ravi Prakash G.
Department of Computer Science & Engineering, Christ University Faculty of Engineering Bangalore, Karnataka, India

ABSTRACT
With the development of information technology, a large volume of data is growing and getting stored electronically. Thus, the data volumes processing by many applications will routinely cross the petabyte threshold range, in that case it would increase the computational requirements. Efficient processing algorithms and implementation techniques are the key in meeting the scalability and performance requirements in such scientific data analyses. So for the same here, it has been analyzed with the various MapReduce Programs and a parallel clustering algorithm (PKMeans) on Hadoop cluster, using the Concept of MapReduce.
Here, in this experiment we have verified and validated various MapReduce applications like wordcount, grep, terasort and parallel K-Means Clustering Algorithm. It has been found that as the number of nodes increases the execution time decreases, but also some of the interesting cases has been found during the experiment and recorded the various performance change and drawn different performance graphs. This experiment is basically a research study of above MapReduce applications and also to verify and validate the MapReduce Program model for Parallel K-Means algorithm on Hadoop Cluster having four nodes.

Keywords
Machine learning, Hadoop, MapReduce, k-means, wordcount, grep, terasort.

1. INTRODUCTION
Machine Learning is the study of how to build the systems that adapt and improve with experience. Machine Learning focus on designing of the algorithms that can recognize patterns & take decisions.

Broadly speaking the two main subfields of machine learning are supervised learning and unsupervised learning. In supervised learning the focus is on accurate prediction, whereas in unsupervised learning the aim is to find compact descriptions of the data.

Hadoop [1] was created by Doug Cutting; he is person behind the Apache Lucene creation, Apache Luence is the text search library which is being widely used. Hadoop has origin in Apache Nutch, Apache Nutch is an open source search engine and it is a web search engine, which is a part of the Lucene project. Apache Hadoop [1] is a software framework that supports data-intensive distributed applications. It empowers the applications to work with thousands of computational autonomous and independent computers and petabytes of data. Hadoop is the derivative of Google's MapReduce and Google File System (GFS) [2]. These include reliability achieved by replication, scales well to thousands of nodes, can handle petabytes of data, automatic handling of node failures, and is designed to run well on heterogeneous commodity class hardwares. However, Hadoop is still a fairly new project and limited example code and documentation is available for non-trivial applications.

HDFS [3], the Hadoop Distributed File System, is a file system which is designed to hold very large amounts of data (terabytes or even petabytes), and provide high-throughput access to the information. Files are stored in a redundant fashion across multiple machine s to ensure their durability to failure and high availability to parallel applications. HDFS presents a single view of multiple physical disks or file systems.

MapReduce [4] is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. The name derives from the application of map ( ) and reduce ( ) functions. In its simplest form MapReduce is a two-step process: Map step and Reduce step. In the Map step a master node divides a problem into a number of independent chunks of problems that are allocated to map tasks. Each map task performs its own assigned part of the problem and outputs results as key-value pairs. In the reduce step the node takes the outputs of the maps, where a particular reducer will receive only map outputs with a particular key and will process those. As the result of reduce step, a collection of values is produced.

This report includes/presents the study and series of experiments of various MapReduce Algorithms and Parallel K-Means Clustering algorithm on Hadoop cluster, thereby achieving a collection of sample codes and documentations. The data set used here varies up to 1GB and also there is some of the interesting case for different algorithm implementation using MapReduce.

This Experiment covers Literature review (Research Clarification & Descriptive Study I) it describes the machine learning, MapReduce and its design, MapReduce strategy, Hadoop, HDFS, and different clustering algorithms with brief about K-Means & canopy clustering algorithms, cloudera and lists of problems. Methodology (Prescriptive Study), it tells about Hadoop architecture, MR Programming model and Parallel K-Means algorithm. Work done and result (Descriptive Study II) with the brief of experimental setup and experimental results. Finally; conclusion, future work and references.

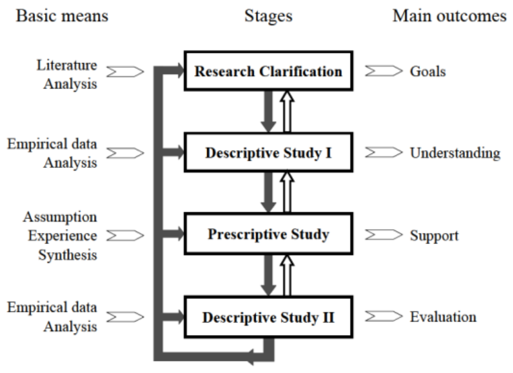


Fig 1: Research Plan: Basic Means, Stages, Main Outcomes

2. LITERATURE REVIEW (RESEARCH CLARIFICATION & DESCRIPTIVE STUDY I)

2.1 Machine Learning

Machine learning [9] is a branch of artificial intelligence, and it is a scientific approach which care of the design and expansion of an algorithms which take input as empirical data, such as data from the sensors or some databases, and yield patterns or predictions, that thought to be features of the original mechanism which generated the data. A major emphasis of the machine learning exploration is the design of algorithms that recognize complex patterns and make intelligent decisions based on input.

Machine learning is the science of making computers that can take decisions by its own. In the past days, machine learning has given us automated cars, auto speech recognition, effective web searching, and an immensely improved understanding of the human genome (the complete set of human genetic information, stored as DNA sequences within the 23chromosome pairs of the cell nucleus and in a small DNA molecule within the mitochondrion). Machine learning is so pervasive today that it has been undoubtedly use it many times a day without knowing about it. Many researchers also think it is the best way to make progress towards human-level Artificial Intelligence (AI). In this way, it will be learned about the most effective machine learning techniques, and practice by implementing them and getting them to work for our self.

There are two major categories of types of learning: Supervised learning and Un-supervised learning. In Supervised learning [9] [15], it is known (sometimes only approximately) that the values of the m samples in the training set T. It has been assumed that if a hypothesis h that closely agrees with f is been found for the members of T. Then this hypothesis will be a good guess for f especially if T is large. For example, in case of the classification problem, the learner estimates a function mapping a vector into classes by looking at the input-output examples of the function. Whereas, In Un-supervised learning [9] [15], there is a training set of vectors without function values for them. The problem in this case, usually, is to divide the training set into subsets, t1. . . tr, in some appropriate way. Unsupervised learning methods have application in taxonomic problems in which it is desired to invent ways to classify data into meaningful categories. Approaches to unsupervised learning include Clustering (e.g., k-means, mixture models, hierarchical clustering).

It has been also describe methods that are intermediate between supervised and unsupervised learning i.e. Semi-supervised learning. Semi-supervised learning [15] is a class of machine learning techniques that make use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of the unlabeled data. Semi-supervised learning lies between unsupervised learning and supervise learning.

2.2 MapReduce

MapReduce is a programming model for expressing distributed computations on massive amounts of data, and also an execution framework for large-scale data processing on clusters of commodity servers. In other word it can tell that MapReduce represents to a framework that runs on a computational cluster to extract the Knowledge from a large datasets. The name MapReduce is derived from two functions map ( ) and reduce ( ) functions. The Map ( ) function usually applies to all the members of the dataset and then returns a list of results. And “Reduce ( ) function” collates and resolves the results from one or more mapping operations executed in parallel.

MapReduce Model splits the input dataset into independent chunks called as subsets, which are processed by map ( ) and reduce ( ). Generally, compute nodes & storage nodes are the same. That is the entire computation involving map ( ) and reduce ( ) functions will be happening on DataNodes and result of computation is going to be stored locally.

In the MapReduce Model, programs written in the functional style are automatically parallelized and executed on the large cluster of commodity hardware. The run-time system takes care of the details of broken input data, and schedules the program's execution across the number of machines; it handles the machine failures, and manages inter-machine communication. In this way without any experience with parallel and distributed systems this allows programmers, to easily utilize the resources of a large distributed system.

2.2.1 *MapReduce Design*

In MapReduce, records are treated in isolation by tasks called as Mappers. The output from the Mappers is then brought together into a second set of tasks called as Reducers; here results from many different mappers are being merged together. Problems suitable for processing with MapReduce must usually be easily split into independent subtasks that can be processed in parallel. The map and reduce functions are both specified in terms of data is structured in key-value pairs.

$$\text{map: } (k_1, v_1) \rightarrow [(k_2, v_2)]$$
$$\text{reduce: } (k_2, [v_2]) \rightarrow [(k_3, v_3)]$$

The power of MapReduce is from the execution of many map tasks which run in parallel on a data set and gives output of the processed data in form of intermediate key-value pairs. Each reduce task receives and processes data for one particular key at a time and outputs the data which processes as key-value pairs.

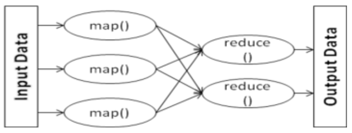


Fig 2: MapReduce key-value pair's generation

2.2.2 *MapReduce Strategy (Execution)*

The Map invocations are distributed across multiple machines by automatically partitioning the input data into a set of M chunks. The input chunks can be processed in parallel by different machines. Reduce requests are being distributed by partitioning the intermediate key space into R pieces using a partitioning function (e.g., hash (key) mod R). The number of partitions (R) and the partitioning function are specified by the user. Figure give below shows the overall flow of a MapReduce operation. As soon as the MapReduce function is called by the user program, the following sequence starts

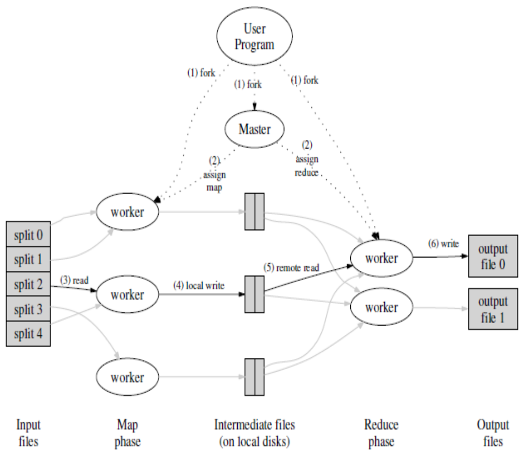


Fig 3: MapReduce Execution overview

2.3 Hadoop

Apache Hadoop [1] is an open source it is built on Java framework and it is built for implementing the reliable and scalable computational networks which supports data intensive distributed applications, and it is licensed under Apache v2 license. It enables many applications to work with thousands and thousands of computational independent computers and petabytes of the data. Hadoop was derived from Google's MapReduce and Google File System (GFS).

Hadoop is a top-level Apache project which is built and used by a global communal of contributors, it has been written in the Java programming language. Hadoop includes several subprojects: [1] [3]

Table 1. Hadoop Project Components (Sub-Projects)

HDFS	Distribute File System; One of the Subject of this Experiment
MapReduce	Computational Framework for distributed environment
HBase	Column-oriented table service
Pig	Dataflow Language and it is a Parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	It is for distributed coordination service
Chukwa	Collecting management data: System
Avro	Data serialization system

2.4 HDFS

The Hadoop Distributed File System (HDFS) is designed to store large data sets with high reliability, and to stream those data sets with high bandwidth. In a large cluster, more that thousands of servers both host and Client are directly attached to storage and execute user application tasks. Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm.

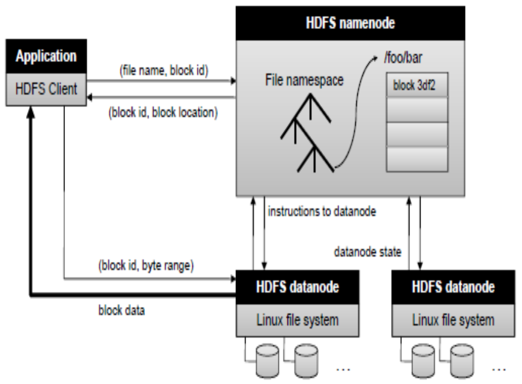


Fig 4: The architecture of HDFS

HDFS is the file system constituent the Hadoop. While the interface to HDFS is being formed after the UNIX file system, the truthfulness to standards was sacrificed in favor of enhanced performance for the applications at hand. HDFS stores the file system, metadata and the application data independently. Alike to distributed file systems, like PVFS, Lustre and GFS. HDFS stores metadata on a dedicated server, known as NameNode. Application data are stored on other servers known as DataNodes. All servers are connected and communicate with each other using TCP-based protocols [1].

2.4.1 HDFS Client
Using the HDFS client [17], user applications access the file system. Similar to most conventional file systems, HDFS is having provisions of reading, writing and deleting files, and provides operations for creating and deleting directories.

2.4.2 HDFS Name Node (Master)
It manages the file system name space ,keeps track of job execution, manages the cluster, replicates data blocks and keeps them evenly distributed, Manages lists of files, list of blocks in each file, list of blocks per node, file attributes and other meta-data and also it Tracks HDFS file creation and deletion operations in an activity log. Depending on system load, the NameNode and JobTracker daemons may run on separate computers. JobTracker dispatches jobs and assigns splits (splits) to mappers or reducers as each stage completes

2.4.3 Data Nodes (Slaves)
It stores blocks of data in their local file system, stores meta-data for each block, serves data and meta-data to the job they execute, sends periodic status reports to the Name Node, and sends data blocks to other nodes required by the Name Node. Data nodes execute the DataNode and TaskTracker daemons. TaskTracker executes tasks sent by the JobTracker and reports status.

2.5 Clustering Algorithms
Data clustering [5] is the partitioning of a data set or sets of data into similar subsets; this can be accomplished by using some of the clustering algorithms.

2.5.1 K-Means Algorithm
K-MEANS [5] [6] [11] is the simplest algorithm used for clustering also it an unsupervised clustering algorithm. The K-Means algorithm is used to partitions the data set into k clusters using the cluster mean value so that in the resulting clusters is having high intra cluster similarity and low inter cluster similarity. K-Means clustering algorithm is iterative in nature.

The K-means clustering algorithm is known to be efficient in clustering large data sets. This clustering algorithm was originally developed by MacQueen , and is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering problem. The K-Means algorithm targets to partition a set of given objects into k clusters based on their features, where k is a user-defined constant. The core idea is to define k centroids, one centroid for each cluster. The centroid for a cluster is calculated and formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster.

2.5.2 Canopy Clustering Algorithm
Canopy Clustering [5] [6] an unsupervised pre-clustering algorithm related to the K-means algorithm, which can process huge data sets efficiently, but the resulting "clusters" are merely a rough pre-partitioning of the data set to then analyze the partitions with existing slower methods such as k-means clustering.

The basic steps of the canopy clustering are described below. Given two threshold distance T1 and T2; T1>T2 and a collection of points. Now, to determine the Canopy Centers: there is iteration through the set of points, if the point is at distance decide the canopy membership - for each point in the input set if the point is at a distance < T1 from any of points in the list of canopy centers (generated in step) then point is member of the corresponding cluster.

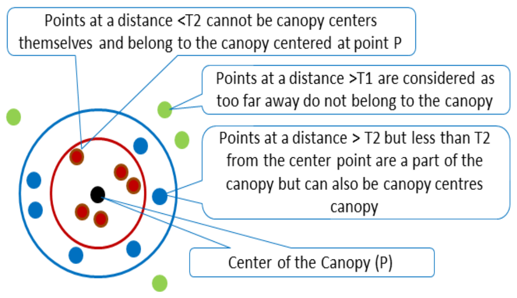

Points at a distance <T2 cannot be canopy centers themselves and belong to the canopy centered at point P
Points at a distance >T1 are considered as too far away do not belong to the canopy
Points at a distance > T2 but less than T2 from the center point are a part of the canopy but can also be canopy centres canopy
Center of the Canopy (P)

Fig 5: Canopy clustering description

The Map Reduce implementation of K-Means Algorithm with Canopy Clustering has the following steps.

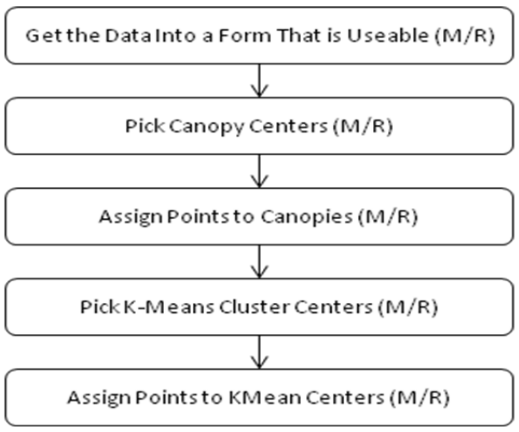

Get the Data Into a Form That is Useable (M/R)
Pick Canopy Centers (M/R)
Assign Points to Canopies (M/R)
Pick K-Means Cluster Centers (M/R)
Assign Points to KMean Centers (M/R)

Fig 6: Map Reduce Steps

2.6 Cloudera
Cloudera [7] [8] Inc. is a software company that provides Apache Hadoop-based software, support and services called CDH. CDH has version of Apache Hadoop patches and updates.

2.7 List of Problems
During the literature review and observation some problems like scalability issues, high computational costs, reliability, compatibility problems (with the Softwares & Hardwares) has been listed out.

3. METHODOLOGY (PRESCRIPTIVE STUDY)

3.1 Hadoop Architecture
The architecture of a complete Hadoop cluster is in the form of the Master-Slave architecture. Here, in the Hadoop architecture, Master is NameNode and Slaves are DataNodes.

The HDFS NameNode runs the NameNode daemon. The job submission node runs the JobTracker, which is the single point of contact for a client wishing to execute a MapReduce job. The JobTracker monitors the progress of running MapReduce jobs and is responsible for coordinating the execution of the mappers and reducers.

Typically, all these services runs on two distinct machines, although in a small cluster they are often co-located. The bulk of a Hadoop cluster consists of slave nodes (only three of which are shown in the figure) that run both a TaskTracker, which is responsible running the user code, and a DataNode daemon, for serving HDFS data.

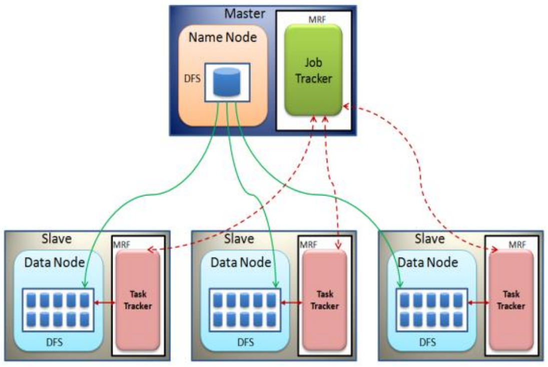


Fig 7: Hadoop Cluster Architecture

3.2 MR Programming Model

MAP-REDUCE programming model [17] is defined by Dean et al. MAP-REDUCE computing model comprises of two functions, Map ( ) and Reduce ( ) functions. The Map and Reduce functions are both defined with data structure of (key1; value1) pairs. Map function is applied to each item in the input dataset according to the format of the (key1; value1) pairs; each call produces a list (key2; value2).

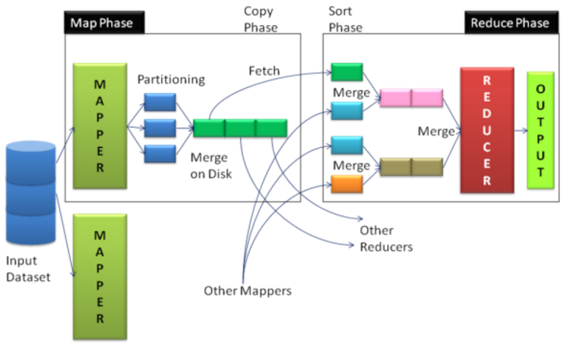


Fig 8: Process of MAP and REDUCE is illustrated

All the pairs which is having the same key in the output lists is kept to reduce function which generates one (value3) or an empty return. The results of all calls from a list, list (value3). This process of MAP and REDUCE is illustrated in figure below.

3.3 PKMeans Algorithm (Parallel K-means Algorithm) [11]

Here is the presentation of a design for Parallel K-Means based on MapReduce. First, it has been already seen with the overview of the k-means algorithm and then analyzed parallel parts and serial parts of the algorithms. Now, it has been seen that how the computations can be formalized as map and reduce operations in detail. So, there got the details of PK-Means Algorithm [11] [16] [17] [18] using MapReduce with the flow diagram shown below.

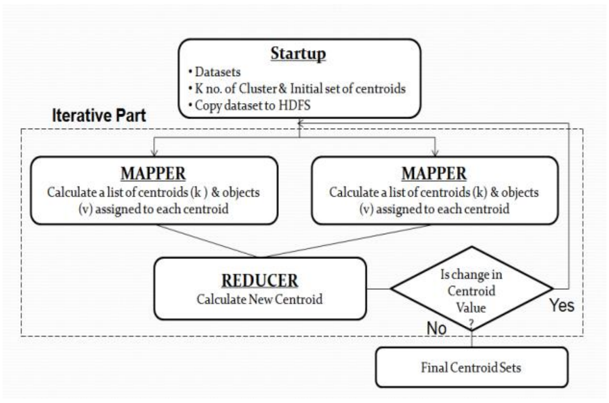

Startup
• Datasets
• K no. of Cluster & Initial set of centroids
• Copy dataset to HDFS
Iterative Part
MAPPER
Calculate a list of centroids (k) & objects (v) assigned to each centroid
MAPPER
Calculate a list of centroids (k) & objects (v) assigned to each centroid
REDUCER
Calculate New Centroid
Is change in Centroid Value ?
Yes
No
Final Centroid Sets

Fig 9: Flow diagram of PKMeans Algorithm using MapReduce

4. WORK DONE AND RESULT (DESCRIPTIVE STUDY II)

4.1 Experimental Setup

The experiments were carried out on the Hadoop cluster. The Hadoop infrastructure consists of one cluster having four nodes, distributed in one single lab. For the series of experiments, I have used the nodes in the Hadoop cluster, with Intel Core 2 Duo CPU@ 2.53 GHZ, 2 CPUs and 2GB of RAM for each node. With a measured bandwidth for end-to-end TCP sockets of 100 MB/s, Operating System: CentOS 6.2 (Final) and jdk 1.6.0_33 (SUN JAVA).

4.2 Experimental Result and its Analysis

In this Experiment, performance has been shown with respect to execution time and number of nodes. Here, there are different cases as given in table below.

Table 2. Various cases for verification and analysis.

Graphical Analysis of Various Cases using Dataset		
Sr. No.	**Data Size**	**No. of Nodes**
Case 1	Increasing	Constant
Case 2	Constant	Increasing
Case 3	Increasing	Increasing

4.2.1 Experiment 01: Sequential Algorithm vs. MapReduce Algorithm

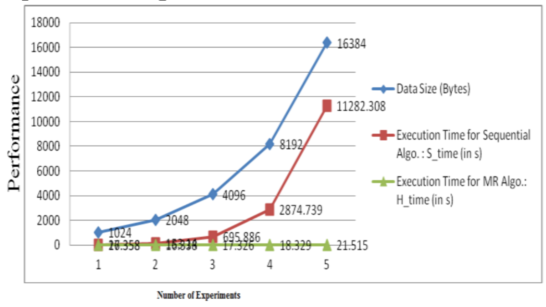
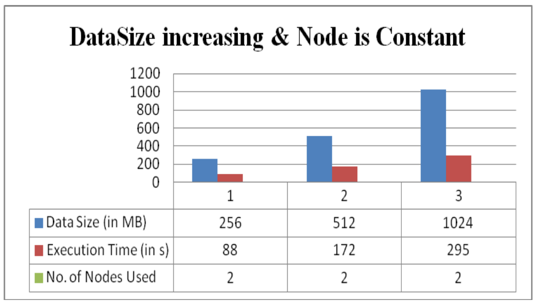


Fig 10: Speed of Sequential Algorithm vs. MapReduce Algorithm

4.2.2 *Experiment 02: Wordcount*

Case 1: When Data size is increasing and Number of nodes is Constant



	1	2	3
🟦 Data Size (in MB)	256	512	1024
🟥 Execution Time (in s)	88	172	295
🟩 No. of Nodes Used	2	2	2

Fig 11: Results of wordcount MapReduce Application

Case 2: When Data size is Constant and Number of nodes are increasing

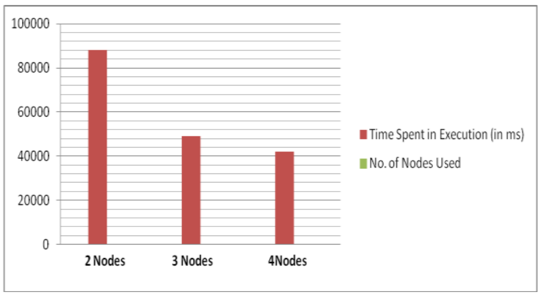


Fig 12: Results of wordcount MapReduce Application

Case 3: When Data size is increasing and Number of nodes are increasing

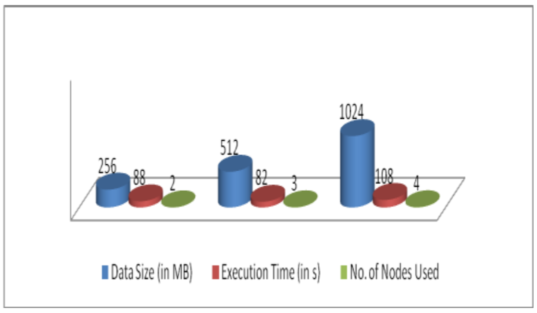
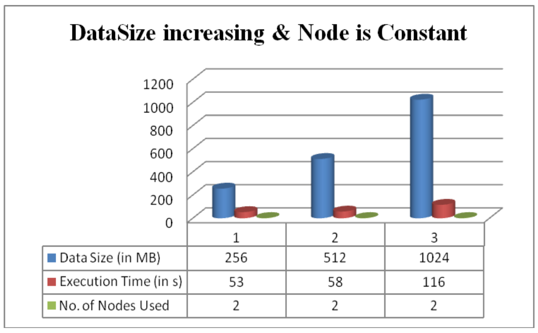


Fig 13: Results of wordcount MapReduce Application

In case of Wordcount ,it has been observed with an interesting case (Case 3) and also analyzed and verified that the execution time is first decreasing and then increasing , as data size as well as number of nodes are increasing (from 1 to 4) because the load on nodes are increasing and the communication between nodes are also increasing.

4.2.3 *Experiment 02: Grep*

Case 1: When Data size is increasing and Number of nodes is Constant



	1	2	3
🟦 Data Size (in MB)	256	512	1024
🟥 Execution Time (in s)	53	58	116
🟩 No. of Nodes Used	2	2	2

Fig 14: Results of grep MapReduce Application

Case 2: When Data size is Constant and Number of nodes are increasing

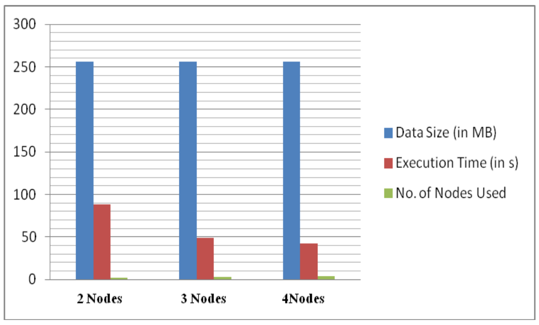


Fig 15: Results of grep MapReduce Application

Case 3: When Data size is increasing and Number of nodes are increasing

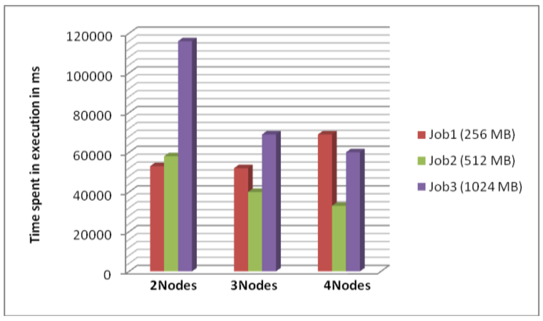
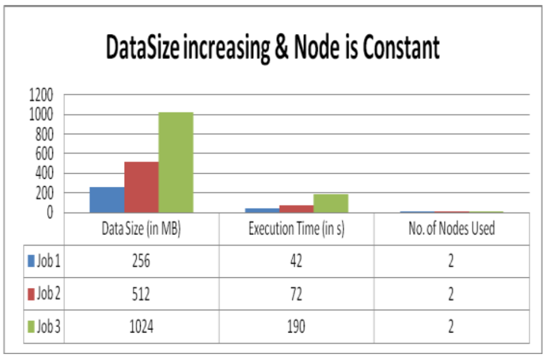


Fig 16: Results of grep MapReduce Application

In case of Grep experiment, it has been observed with an interesting case (case 3). It has been analyzed and verified that the execution time in case of Job1 (256MB of Data Size) is showing first decreasing and then increasing pattern. This is happening because the load (Data set Size) is increasing on nodes and as well as communication between nodes is also increasing (From 1 to 4).

4.2.4 *Experiment 03: Terasort*

Case 1: When Data size is increasing and Number of nodes is Constant



	Data Size (in MB)	Execution Time (in s)	No. of Nodes Used
🟦 Job1	256	42	2
🟥 Job2	512	72	2
🟩 Job3	1024	190	2

Fig 17: Results of terasort MapReduce Application

Case 2: When Data size is Constant and Number of nodes are increasing

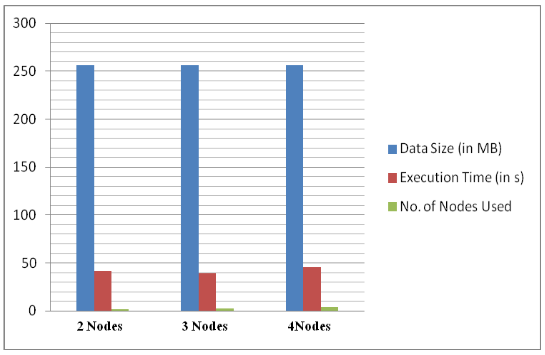


Fig 18: Results of terasort MapReduce Application

Case 3: When Data size is increasing and Number of nodes are increasing

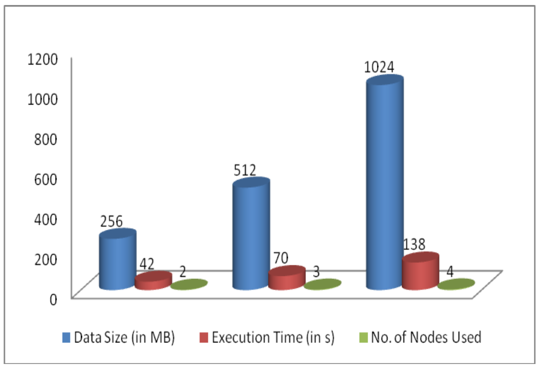
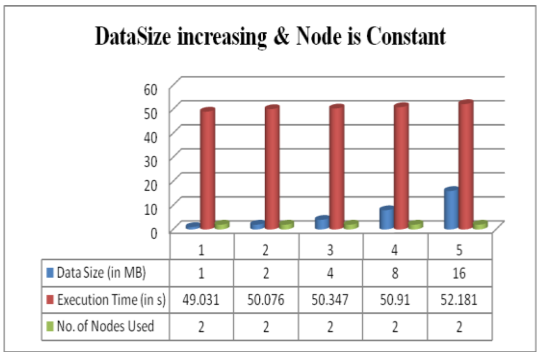


Fig 19: Results of terasort MapReduce Application

In case of Terasort experiment, it has been observed with an interesting case (Case 2). It has been analyzed and verified that the execution time is first decreases and then increases, when Data size is kept constant and increasing number of nodes from 1 to 4. This is happening because the load (Data set size) is constant but the communication between nodes is increasing since number of Node is increasing from 1 to 4.

4.2.5 Experiment 04: PKMeans Clustering Algorithm

Case 1: When Data size is increasing and Number of nodes is Constant



	1	2	3	4	5
🟦 Data Size (in MB)	1	2	4	8	16
🟥 Execution Time (in s)	49.031	50.076	50.347	50.91	52.181
🟩 No. of Nodes Used	2	2	2	2	2

Fig 20: Results of PKMeans MapReduce Application

Case 2: When Data size is Constant and Number of nodes are increasing

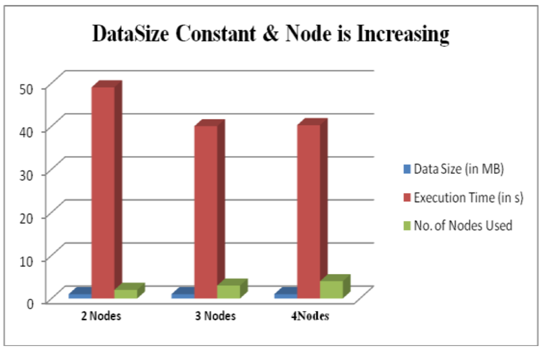
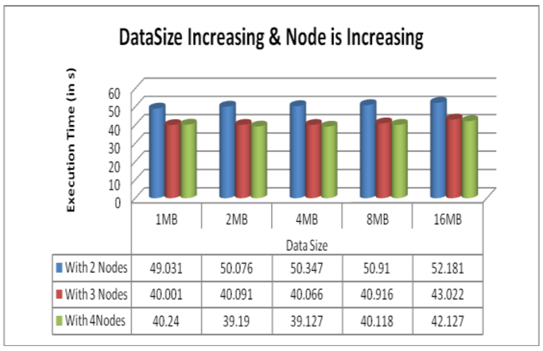


Fig 21: Results of PKMeans MapReduce Application

Case 3: When Data size is increasing and Number of nodes are increasing



	1MB	2MB	4MB	8MB	16MB
🟦 With 2 Nodes	49.031	50.076	50.347	50.91	52.181
🟥 With 3 Nodes	40.001	40.091	40.066	40.916	43.022
🟩 With 4Nodes	40.24	39.19	39.127	40.118	42.127

Fig 22: Results of PKMeans MapReduce Application

In case of PKMeans experiment, it has been observed with an interesting case (Case 2). It has been analyzed and verified that the execution time is first decreases and then increases. The reason behind this is that the load (Data set size) is constant but the communication between nodes is increasing, since number of Node is increasing from 1 to 4.

5. CONCLUSION AND FUTURE WORK

In this experiment, the performance of MapReduce application has been shown with respect to execution time and number of nodes. Also, verified and validated that in MapReduce Program model as the number of nodes increases the execution time decreases. In this way, it has been shown that PKMeans performs well and efficiently and the results totally depend on the size of Hadoop cluster. The performance of above application has been shown with respect to execution time, dataset size and number of nodes. The Experiment involves the Hadoop cluster, which contains total of four nodes i.e. one master (NameNode) and three slaves (DataNodes).

The future research includes developing mechanisms for Hadoop quality of service on the different size of the datasets using MapReduce. Performance evaluation of MapReduce with different version of software and hardware configurations. Security Issues with respect to Hadoop and MapReduce. The performance of PK-Means Algorithm can be enhanced by introducing different preprocessing steps such as Canopy Clustering algorithm. MapReduce concept can be look forward, for the algorithms such as different set of optimization algorithms, different set of Kernel algorithms.

6. REFERENCES

[1] Apache Hadoop. http://hadoop.apache.org/

[2] Sanjay Ghemawat, Howard Gobioff, and Shun-TakLeung “The Google File System”, Google,Sosp’03, October 19–22, 2003, Bolton Landing,New York, USA. Copyright 2003 ACM 1-58113-757.

[3] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, “The Hadoop Distributed File System”. Yahoo! Sunnyvale, California USA, IEEE, 2010

[4] Jeffrey Dean and Sanjay Ghemawat “MapReduce: Simplified Data Processing On Large Clusters” 2009,

[5] ,Google, Inc., Usenix Association OSDI ’04: 6thSymposium on Operating Systems Design andImplementation.

[6] http://www.ise.bgu.ac.il/faculty/liorr/hbchap15.pdf

[7] http://en.wikipedia.org/wiki/Cluster_analysis#Newer_developments

[8] http://www.cloudera.com

[9] https://wiki.cloudera.com/display/DOC/CDH+Installation+Guide.

[10] http://en.wikipedia.org/wiki/Machine_learning

[11] Mahesh Maurya and Sunita Mahajan. “Performance analysis of MapReduce Programs on Hadoop cluster”, World Congress on Information and Communication Technologies 2012.

[12] Weizhong Zhao, Huifang Ma, and Qing He, “Parallel K-Means Clustering Based on MapReduce”, 2009.

[13] http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

[14] http://ieeexplore.ieee.org/

[15] http://www.springerlink.com/content/c621194607866223

[16] David Barber, “Bayesian Reasoning and Machine Learning”, Cambridge 2011; New York: Cambridge University Press.

[17] Ron Bekkerman, Mikhail Bilenko, John Langford, “Scalable Machine Learning” Cambridge University Press, 2012.

[18] Jimmy Lin and Chris Dyer, “Data-Intensive Text Processing with MapReduce”, April 2010, University of Maryland, College Park.

[19] Tom White, “Hadoop: The Definitive Guide”, 2009 Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.