1.3.2

Sample Reports of Project Work/ Field Work / Internship

Greater Noida Institute of Technology (Engg. Institute)

Plot No. 7, Knowledge Park II, Greater Noida
Uttar Pradesh 201310 India

MAPREDUCE APPLICATION: A STUDY TO IMPROVE HETEROGENEOUS CLUSTER ENVIRONMENT

A thesis submitted in partial fulfilment of the requirements for the degree of

MASTER OF TECHNOLOGY

in

Computer Science Engineering

by

Divya Raj

(Enrollment no. 200132010023009)

Under the Supervision of

Dr. Anuranjan Mishra

Greater Noida Institute of Technology, Greater Noida (U.P)

to the

Faculty of Computer Science Engineering

DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

September, 2022

Greater Noida Institute of Technology (Engg. Inst.) Director * Greater Noida *

[illegible]

DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "MAPREDUCE APPLICATION: A STUDY TO IMPROVE HETEROGENEOUS CLUSTER ENVIRONMENT", in partial fulfilment of the requirements for the award of **Master of Technology** and submitted to **Dr. APJ Abdual Kalam Technical University, Lucknow** is an authentic record of my own work under the supervision of Dr. Anuranjan Mishra. The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institution.

I declare that I have given due credit to the actual authors/sources, where ever required, for all the words, ideas, diagrams, graphics, computer programs, experiment and results that are not my original contribution. I have used quotation marks to identify verbatim sentences and have given credit to the original authors/sources.

I affirm that no portion of my research work is plagiarized, and the experiments and results reported are not manipulated. In the event of any complaint of plagiarism or manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Divya Raj

Roll No.: 2001320105005

Branch: Computer Science Engineering

Divya

(Candidate Signature)

CERTIFICATE

Certified that **DIVYA RAJ** (enrolment no. 200132010023009) has carried out the research work presented in the thesis entitled "MAPREDUCE APPLICATION: A STUDY TO IMPROVE HETEROGENEOUS CLUSTER ENVIRONMENT" for the award of **Master of Technology** from **Dr. APJ Abdual Kalam Technical University, Lucknow** under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature [illegible]

Dr. Anuranjan Mishra

Dean (R&D)

Greater Noida Institute of Technology, Gautam Buddha Nagar, UP.

ABSTRACT

The world's data volume is expanding. rapidly and working as a fuel to run organisations. Data is generating due to internet, smartphones, social network and Electronic Devices. Data is coming from various sources so its variety is diverse & complex and velocity is very high. Big data analytics give organisations and the government new methods to analyse semi structure and unstructured data and pull the meaningful information from this which can help in taking right business decisions.

Advanced data analysing tools can reveal hidden patterns in data that help organisations stay ahead of the competition, save money, reduce risks, and better understand the needs of their consumers.
This tremendous quantity of data cannot be captured, stored, processed, or analysed by conventional database systems. So, Hadoop come in to existence as an open source Frame work. Hadoop can store huge data regardless of its type by using its HDFS file system. HDFS file system store data on various nodes of distributed clusters. This works well with minor restrictions when dealing with homogeneous clusters, but when dealing with underline heterogeneity in a cluster, a few problems such as data locality are seen along with decreased performance to deliver faster results.

Various researches have been already done to overcome the known limitations of map reduce processing of Bigdata. For example – Network aware Scheduling, (LATE) longest-approximate-time to end algorithm, adaptive task scheduling etc. but their focus is limited to only certain aspects to challenges.

The present Thesis highlights important concepts of Hadoop, Map reduce process and challenges involved. In this write up we discuss various aspects of K means features and their limitation in heterogenous network.

To determine whether the processing challenges outlined are handled, a comprehensive two-phased algorithm named parallel K-means has been devised and assessed in this study effort. This approach uses incremental K-means, which increases the number of clusters by one until K clusters are reached.

ACKNOWLEDGEMENT

While bringing out this thesis to life I, there are countless people I met who influenced and contributed directly or indirectly into my field of research and I am thankful to them and I express my sincere gratitude to all.

First of all, I am grateful to my supervisor Dr. Anuranjan Mishra Sir, Dean R&D GNIOT, Greater Noida, UP

I have taken lot of efforts in this dissertation work. I have been very fortunate to get help and support from many people. I feel immense pleasure in expressing my profound gratitude to my Guide during thesis for his invaluable encouragement, constant feedback and guidance. I am highly thankful for his valuable advice, guidance, and push while I am losing momentum and required contribution when it was needed most during the period of research.

Last but not least, I profusely thankful to my Family members and friends for their consistent support as without which I would not be able to carry out this.

Divya Raj

2001320105005

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MR	Map Reduce
FS	File system
HDFS	Hadoop distributed file system
LATE	Longest approximate time to end
HMR	Hierarchical MapReduce
Fed-MR	Federated MapReduce
JVM	Java virtual machine
YARN	Yet another resource negotiator
VN	Virtual node
FIFO	First in and first out
EDF	Earliest Deadline First
ML	Machine Learning
SAMR	Self-Adaptive MapReduce Scheduling Algorithm
KM	K-Means algorithm
PKM	Parallel K-Means algorithm
IKM	Incremental K-Means algorithm
2PPar-KMeans	Two phased parallel K-means algorithm

Chapter 1

INTRODUCTION

Background

Big data is now playing a significant part in data analytics by giving business users insights on data that were previously never thought of as being vital and frequently disregarded or underutilised. This helps industries/organisation taking data driven decision in ensuring an effective marketing, Consumer personalization, new revenue opportunities and reduced operational cost. Thus, it provides them competitive advantages over their rivals. The explosion of data is happening is all types of industries and there are varieties of data being captured now which were not used earlier. Large organisations like Google, Facebook, Amazon etc. have to process large size and variety of data in their daily processing. This data is in-fluxed from web blog posting messages and conversations in the form of logs, pictures posted on web, individual status messages, unstructured audio files and video content messages. Web navigation also produces large sized data by web sites activities such as access files, click stream data, weather prediction reports, e-commerce Browsing and purchasing history etc. Hence analysing such data can give an idea about user's behaviour, their interest and their expending habits.

According to statics New York Stock Exchange (NYSE) generates approximately one terabyte data which belongs to fresh trades on a daily basis. At the beginning of 2020, the total amount of data in the world was predicted to be **44 zettabytes**. At least 1,200 petabytes of information are stored by Amazon, Facebook, Microsoft, and Google. Global data generation is estimated to reach **463 exabytes** per day by 2025. Analysing huge and variety of data is a challenge by using traditional data processing methods and tools using relational databases and data warehouses which are not capable in storing and processing this large amount of data and that too in variety of formats. They became incapable to capture, store, manage and process in an efficient manner.

Scalable log processing is considered as a best way to efficient utilisation of underline infrastructure along with attaining maximum cost benefits. Data analytics enables service providers to drill down logs which are a semi structured data and provide cost effective solutions by analysing access trails of users click pattern to read and conclude user's behavioural patterns which gives an opportunity to know the taste of users and send appropriate marketing messages to identified audience as per their browsing pattern. E-commerce sites and banks could analyse point-of-sales transactions for early fraud detection. Similarly, Infrastructure providers use log information for identifying the possible misconfigurations related to hardware and helps into improvement of load-balancing on huge data clusters and gather relevant utilization statistics and pattern to determine future course of action.

Concept of MapReduce was initially proposed by Google later its open-source implementation was managed and given by Apache Hadoop. MapReduce provides organisations a cost-effective solution to process big data and provide meaningful business information from that unstructured big data to improve their benefits and reduce operational cost.

1.1 Categories of Big Data

Big data can be mainly categorized in two types, as stated below

Structured data

Data having a Certain format and pre-defined organizational properties can be called as structured data. This type of data can be presented in structure and tabular schema and processed by traditional data processing methods. structured data enables quick data collection from numerous database locations. Due to its predefined nature each field is distinct and it can be accessed either independently or in combination with data from other fields. However Significant growth of even structured data is becoming a challenge to exiting traditional approaches to process huge structured data produced in the size of multiple zeta bytes. Using traditional methods and software tools both storage and processing will require lots of reorganisation and will be costly too. This can also impact the processing capabilities of tools. Data reorganisation on such a large scale is itself a challenge.

Unstructured data

Any data which doesn't have any Certain format is known as unstructured or semi structured data which cannot be effectively handled by traditional techniques. Using the transformation and data cleaning techniques however un-structured data can be transformed into semi-structured data. Size of data is not only processing challenges it poses various other processing and storage related challenges to uncover hidden value lies inside data. E.g., hybrid of logs, messages, pictures and videos contents etc., could be example of unstructured data. Now a days "data is king" and is coming from various sources makes organizations difficult to manage and store this growing vast data to utilize and harness benefits out of it.

Big data processing is collection of programming models or techniques that assist companies and organisations to mitigate their large-scale data processing challenge and enable them to perform their analytical processing requirement to extract and bring out meaningful business insights from less utilized data present into its raw format. Bigdata is stored in multiple servers which can be thousands in number so traditional data model such as Message passing interface (MPI) cannot process it efficiently and effectively. Therefore, parallel programming models are used to overcome the limitation. One of the most efficient and well-liked programming models for data processing is MapReduce. which stores data on various nodes of distributed cluster. Map and Reduce function are used to simplify the task.

1.2 Benefits of Big Data Processing

Key advantages of big data processing are stated below: -

Provides decision making insight

Bigdata provides insight which helps organisation in taking better business decision. For example, in Banking and Insurance sector conversation with agents is converted using speech to text model, which is a form of unstructured data and previously it was just a record of no use but using big data processing techniques this data can be handled and analysed for gaining useful information. Collective use of social data allows companies to know their customer better their interest in particular product, demand of certain products on basis of demographic and weather condition. Expanding habits of customers of different age, group and class.

Improved customer experience and services

Feedback system used for storing and analysing customer feedback traditionally were not adequate to manage and analyse data coming from various sources in an unstructured manner. It could be via some application, questionnaires or via some website navigation, storing customer navigation history related data. Post introduction of big data technologies, storage and analysing such data has become quite possible and fast. New enhanced system usages both Big data, artificial languages and natural language processing methods to understand customer likes, dislikes and evaluate and analyse consumer responses in an efficient manner.

Early warning signs detection

Looking at the historical data stored into traditional database system may take long time to retrieve from archive or may not be possible at all, however storing such data into big data could be very handy to perform risk analytics over the historical data. This can provide any trend which may repeat or any early warning sings to prepare about risk and ready with actions and guidelines.

Efficient operational capability

As data processing capability is evolving and provided a key support to churn out and process vast data which was ignored before, thus giving thought to organisation to keep this data for future customer insights or historical pattern recognition in some sources which can stores such a vast and verity of data. Various available big data technologies come very handy to cater such need of staging of data before it archived to warehouse. This makes effective usages of data along with necessary data transformation and analysis before moving to warehouse.

1.3 Hadoop and its components

Hadoop is an open-source framework and wildly used for Bigdata processing. Being a framework Hadoop is made-up of many different modules and served by a large set of technologies. Hadoop provides flexibility to store big data generated from various sources and store them into a file-based storage system for faster accessing. It works on philosophy of sending logic used for accomplishing a task near to node. Hadoop provides its own file system, named as HDFS (Hadoop Distributed File System). When data is uploaded into Hadoop HDFS, Hadoop breaks data and store them into various nodes of cluster for faster processing and minizine the risk of node failure across the cluster (storing multiple copies of data to manage node failures), and then it can send log or processing code to the nodes which contains data to be used. HDFS Stores data into various nodes by fragmenting them into parts and store corresponding keys and values to fetch and access data as per need based on key and value corresponding to each data partition. Each partition of data is allocated a unique key and a value combination to recognize them by HDFS. Relationships among keys are maintained by application not by HDFS.

Components of Hadoop

Hadoop Distributed File System

HDFS is a storage unit which follows design idea of distributed file system to keep vast amount of data by partitioning data and storing them into various nodes of distributed system consisting of 100 to 1000 of high performing nodes having capability to store more than 100+ terabytes data at a time. Hadoop ecosystem provide both type of interface options API method and command-line interface to invoke HDFS related utilities and tasks.

Map Reduce

MapReduce is a heart of Hadoop used to process user assigned task to extract and provide result by breaking jobs into smaller chunks and these subunits are processed on nodes containing data and then data is sorted and combined to get the final outcome. The Mapper unit provides data processing capability, while the Reducer Unit combines various outputs of Map units and sorts them based on same key.

YARN (Yet Another Resource Negotiator)

YARN is made up of three components Application manager, Node manager and Resource manager. This is responsible for scheduling and resource allocation task in Hadoop system. Resource manager allocates resources in system across the clusters, Node manager work on those allocated resource and acknowledge the first one about them. Required Interface between these two managers are provided and handled by Application manager.

Name Node

The Name Node is called the master node in HDFS which controls other nodes in cluster nodes naked as Data Node daemons. This node has meta data information about nodes having data partitions and their key and value association to recognize that data, how to partition data into blocks, which nodes this partition of data resides and also take care over all distributed file system maintenance. Name node plays a key role into Hadoop ecosystem. Every distributed Hadoop cluster has its dedicated Name Node which is making its prone to single-point of failure for processing.

Secondary Name Node

In order to provide node failure safe design Hadoop also has a backup Name Node or alternate name node which keeps on monitoring the health of primary Name Node ,this also captures the data and processing statics of Primary Name Node If Name Node fails then Secondary Name Node will replace it for processing to ensure no downtime for applications. It's doesn't provide auto switch in case of primary node failure but gives assurance of minimum manual intervention by ensuring minimal data lose.

Data Node

Each slave node in Hadoop cluster is called as Data Node. The Data Node are main units who perform data management: They read required data blocks from the HDFS system and then stores them on physical node, and maintains a communication status of data movement to Name Node.

Job Tracker

The Job Tracker component acts an interface between user application and Hadoop. Each Hadoop cluster has a dedicated Job Tracker which is present on name node and maintains various processing status happening on data nodes/slave nodes. When user submits a jobs for execution to Hadoop for execution Job Tracker builds execution plan for processing. This includes deciding nodes which stores data to execute, managing nodes having required data, monitoring running tasks, and restarting task to some other node in case of task is not completed.

Task Tracker

Like data storage is based on master node/slave node methodology, code processing also adopts the master/slave architecture. Every node of cluster keeps a Task Tracker utility which is responsible for execution of assigned tasks assigned to that node and then communicates status back to Job Tracker who has assigned that task to that node.

1.4 Research motivation

Huge volume of data is being generated by various industries and application is becoming a real challenge for organisations to handle this large data and extract some meaningful insight .Hadoop MapReduce was successful in some extent but due to rapidly growing data has made degradation into performance also the underline network heterogeneity has caused speculative execution kind of scenarios .This thesis focus upon on what are primary challenges faced by Hadoop's MapReduce for processing of huge data and what possible solution adopted by various researchers to overcome those challenges. Here evaluation of LATE and K-Means algorithms to improve MapReduce performance into Heterogeneous environment. Later parallelism was introduced into K-Means for improving MapReduce performance which was someway helped into improving performance ,but they didn’t kept their level of performance when size has grown further .Thus amendment into parallel k means known as two phase parallel k-Means was proposed and evaluated to measure improvement into performance, this has been elaborated and test results are shown in chapter 4.However there is still scope of improvement into this algorithms, which may turn to be an interesting topics for future work and are discussed in more detail in Chapter 5.

1.5 Thesis Statement and Approach

Hadoop MapReduce plays an important role for processing and analysing vast sized data across the various organisations and institutions to equip them for driving data driven business decisions and recommendations to either save cost by cost avoidance on inefficient areas or boost sales/revenues using recommendation using past data. With the advent of mobile devices, social networking sites, volume of data has gown multi-fold in last few years which has imposed processing challenges to traditional MapReduce.

This thesis argues the following:

Identify key challenges responsible to downgrade MapReduce performance and how this is further worsened in case of heterogeneous clusters. Identify advancement /new proposal to overcome these challenges and evaluate few of them to measures how much uplift into performance they can provide using suitable amendment into existing framework and infrastructure. To support this research, this thesis outlines amendment or new proposal and how they perform in case of increased data load and configuration variability among nodes of cluster.

Table 1.5 Summary of contributions of each study into thesis

Study	Area covered in thesis
Bigdata processing challenges using Map reduce	▪ MapReduce data storage challenges ▪ Lack of recursive operations ▪ Lack of online processing in MapReduce ▪ Lack of support to Machine learning
Data locality issue	▪ Reasons for data locality ▪ Resolution of data locality using data placement in heterogenous environment. ▪ Application of aware task scheduling to enhance MR performance ▪ A self-adaptive MapReduce scheduling
Speculative execution in map reduce	▪ What is Speculative execution in MapReduce processing.

	▪ How this hampers MapReduce performance. ▪ Ways to improve performance using LATE algorithm
Application of machine learning to fine tune map reduce performance	▪ Types of machine learning ▪ Supervised/unsupervised/semi supervised ▪ How ML can help into improving MapReduce performance into Heterogeneous environment ▪ Clustering of big sized datasets and various clustering techniques
Improving Map reduce performance using K-means and two-phase parallel k-means	▪ study of k-means algorithm, evaluate results. ▪ Challenges of K-Means and how Parallel processing evaluate to ensure improvement into MapReduce performance ▪ Challenges of parallel k-means and ways to improve them. ▪ Proposal for two phase parallel k-means to obtain improved performance and compare results with K-means and parallel k Means

1.6 MapReduce usage and its virtue to process big data

As amount of data generated over the years has grown as the evolution of storage has also increased a lot and this has led to requirement for exploring and drive meaningful business insight. Business and research organisations are realising that “data is king,” but biggest challenge remains to be addressed is the storing this huge and fast generating unstructured data and methods to analyse data in the. Hadoop has developed an open-source framework to facilitate big data challenges. Data processing evolution journey started from storing structured data into flat files and then to structured and normalized data into traditional relational databases and further from relational databases to data bases made appropriately for storing and processing variety of un-structured NoSQL databases. This was due to phenomenal growth of data increase has made traditional data processing methods deemed to insufficient to take care of processing needs.

Traditional database approaches were good and efficient to manage and process structured data and performed well with data of size megabytes and gigabytes, but now that companies realize “data is king,” the size of data is increasing into terabytes and petabytes. Even with NoSQL data stores, the challenge reside is how to analyse that amount of data. Hadoop is designed to handle vast amount of data and run-on huge clusters of nodes. Hadoop is designed to cope with node failure and thus gracefully manages most failures. Besides node failure management, its architecture allows nearly linearly nodes scalability to add more processing capacity as and when demands for more processing is required, however this incurs cost to add more processing nodes in cluster. There are various applications now days which are contributing into massively generation of bulk data Some of the examples of such applications are social media areas from robotics, web search, mobile devices. The data generated from these sources at the speed never before and posing a challenge to store and process such data into efficient manner to drive a meaningful business insight out of the data collected from various medium exponentially. Business decisions are concluded after performing exhaustive analysis on such data and of data which is being generated at high speed. Distributed environment enables users to apply data analysis techniques on the top of refined data which is eventually a result of ETL and some other cleaning process applied on the collected big data generated from various application and platforms. This also provided business acumen by identifying relationship between data and some other hidden characteristic which was not known before.

Various Machine learning algorithms can be applied on this unstructured or semi structured data for unearthing the hidden useful patterns or knowing likes of individuals etc .Processing of such a vast data becomes easy by distributed Computing methodology, which is boon for overcoming vast data execution challenges and provide shared storage across various nodes by alleviating concerns of data storage posed by storing data into single server or machine .Distributed environment provides faster processing due to availability of high CPU processing which can be consider as collection processing capability of network systems. Each node of a distributed environment is known as node and all collection of such nodes or machines is known as cluster or network. Apache Hadoop [3] is a one of the prominent open source framework which is known for its ability to process large data without causing any challenges to its end users .This was developed by Google's MapReduce and GFS (Google File Systems) and later Apache adopted this model for big data processing .This offers users faster processing by its parallel processing ability and self-managing event of any node failure of cluster without even letting its end user known about this by its fault tolerant mechanism. This proves as boon for such an area which demands the higher processing of data which is not structured and generated by fast pace with an assumption of getting back results or business insights as quickly as possible without losing any information.

Organisation apt for decision making and other areas which were not addressed before by leveraging Hadoop platform's MapReduce component which is known processing big data is a parallel manner on distributed environment. It does so by applying Map and Reduce methodology which divides processing into two sub components known as Map and reduce and these components are is saved on nodes of distributed cluster to achieve the objective of parallel processing. Post the competition of subtasks results are aggregated from subcomponent output as a final result. Other than processing vast amount of data Hadoop also provides data storage capability like a file system to store them and organize output data in HDFS System.

In case of any nodes is not responding or not performing as it should be Hadoop's invokes its fault tolerant capability which manages the event of node failure without even end user knowing what has happened and continue the processing by scheduling that task on some other idle or less occupied node .Which gives user to rely and benefit from such framework for processing and storing big data without bothering such complexity and enjoy the tasks they are intended to do Google File Systems [and MapReduce methodology provider greater capability to work upon vast data bothering node failure ,processing `capability and available storage.

MapReduce paradigm provided by Hadoop is a very flexible method to process big data us batch mode ,as the processing is faster due to the distributed nature of network and tasks is divided into sub tasks and given to underline nodes of network for processing in parallel manner and later results are aggregated to get the final outcome .This provides a seamless execution of vast amount of data processing and gives accurate results by harnessing the distributed clusters virtual speed which can be imagined as sum of processing speed of all underline nodes CPU speed and thus provided results in much promptly.

MapReduce methodology provides faster execution using two prominent functions called as map and reduce, they provided end users or developers ability to customize their tasks or jobs as per their need to get faster results. This also provide developers seamless execution.

MapReduce was provided to process big data in parallel manner by dividing the inputs into smaller chunks and then run them independently on the underline nodes of cluster. MapReduce provides great advantage to application users and application developers by concealing system level complex implementation details and technical intricacies from users to concentrate on its development and logical stuffs. MapReduce is basically a two-level methodology, which are known as: Map step and Reduce step. Map step receives input files located on various nodes of distributed cluster. In case Map function is present on same node where data sub-partition is present while processing, in order to get better processing results Hadoop will send data portion closer to node which contains Map function to encourage the Nobel idea of reducing data movement. Once processing of Map function is completed, in next stage all these interim results are aggregated by reduce function which executes all intermediate results having common interim key value and as a result it gives a pair of key and associated value as output. MapReduce processing worked on master and slave design wherein one of the node is designated as master node which owns Job Tracker and rest of the nodes are known as slave nodes .Slaves nodes are responsible for executing the tasks assigned by Master nodes and post completion of processing communicates assigned task status to master node for recording task status and same will be used in case of any node failure issue happens ,internal slave node status is maintained within Task tracker which is present for every node of the cluster.

The Job Tracker works as interface between application developer who has requested for result using the job submission in Hadoop and the job executes on the interconnected hardware. Job Tracker present on master node receives user's developer task and then further it breaks as per map and reduce sub jobs which are later processed by the order they are submitted or following first come and first-served basis.

The Job Tracker ensures all necessary resources needed for map and reduce phases are allocated and then further keeps monitoring underline nodes which executes allocated task via task Tracker present on each slave nodes. The Task Tracker which is present on each nodes of clusters and accepts tasks allocated to them for processing by master node Job Tracker .Map tasks are executed parallelly using fragmented input datasets and intermediate result is further used as an input by reduce phase and which is combined and sorted to produce final result .This is pictorially represented and explained in below fig-1.6 .Job tacker is accountable for managing data movement between the map, and reduce step and collecting and informing end user about completion of assigned task.

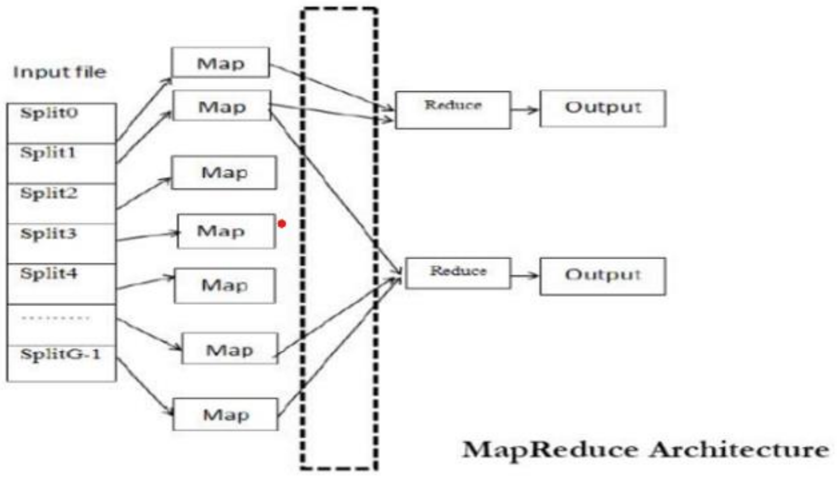


Fig 1.6 Elementary MapReduce processing.

There may be scenario when one of node in cluster is executing a task poorly in terms of average processing time by rest of nodes, this slowness processing phenomena is known as straggler. In event of straggler node issues Map Reduce attempts to re-run a copy straggler task (“backup task”) on another node which is relatively less occupied and have capability to complete task faster and as good as other nodes. Without having speculative execution presence, processing job will be performing very poorly and this will result into longer than average finish time and responsible for performance degradation. Stragglers situation may be result of many causes, this could be due to faulty hardware on node also inappropriate hardware configuration. The presence and availability of speculative execution has been demonstrated by several notable academics to boost task response times by 44%. Due to the heterogeneous environment's fluctuating processing capabilities of nodes, various hardware and software versions, and varied storage availability, this problem degrades processing in speculative events.

1.7 Application of MapReduce for the processing of huge data

MapReduce name came from Lisp Language Map and Reduce primitives. Using the map function to create key-value pairs, which are then sorted to create a collection of interim key-value pairs, and the reduce phase to aggregate all interim values corresponding to an intermediate key, MapReduce carries out processing based on user requirements

MapReduce is best suited for managing parallel processing of large data sets which are scattered across multiple nodes.

Data is first organised in the batch processing paradigm before analysis. The widely used batch processing model is MapReduce. Google recently popularized MapReduce approach of distributed and parallel execution and was implemented on various open-source projects, but Apache Hadoop has most prominent user of this approach to execute queries which runs over large volume data scattered across multiple nodes of cluster.

MapReduce model is not very rigid, it has given liberty to users or application developers to develop and customize their own logics using existing of map and reduce. Which looks very convenient to harness the distributed and parallel processing advantage by hiding the intrinsic complexity from the developer, in event of any occurrence of node failure happens in any nodes of cluster.

1.7.1 MapReduce components and execution stages

MapReduce is a core processing engine provided by Apache Hadoop which is designed and capable to process large volumes of data parallelly in an efficient manner. In order to process the large sized data, MapReduce divides the large sized data into set of smaller independent data splits which executes parallelly so reduce the processing time and provided quick result. MapReduce gives faster result which comes with ability to hide the system related complexities from developer in case of any node failure happens and application user never came to know about node failure has happened and gets result uninterrupted manner. MapReduce is basically a two staged process to execute big data, first step is known as Map stage and second stage is known as reduce stage. Map phase accepts the split of data input, data execution is done on various nodes of distributed file system, post processing of data splits gives key and value pair of kind of output. There may be possibility that data split file can be present on node consisting of Map function or on another node of cluster. If both input file and Map logic doesn’t exist on same node processing becomes challenging and in that case to speedup processing Hadoop will transfer data file closer to the node where Map logic is residing or the node closer to the Map logic exists. MapReduce imbibe the concept of “Moving data closer to compute” for eliminating the data transfer issues and improving the execution time. Next stage is Reduce operation which aggregates key and value output of map logic having common intermediate key and gives again sorted and aggregated pair of key and value. In cluster having multiple nodes one of the nodes is designated as master node which contains Job Tracker for keeping and managing the assigned tasks to rest of nodes in clusters known as slave nodes, which in turns executes the allocated task from master node using task trackers present on each node and reports the status and result back to master node post the processing is completed.

The Job Tracker residing on Master node will accept the users or analytics developer execution requests and sends to underline framework for processing. Users processing request is accepted by Job Tracker and its then passed to map and reduce unit, these map and reduce units which are prioritized as per first come and first serve. Post accepting users processing request JobTracker allocates map and reduce tasks to underline nodes which contains TaskTracker running on any of the salve node in distributed clusters. The TaskTracker executes the instructions received from job tracker and then communicates the status of assigned task and result back to JobTracker for management and also co-ordinates for data transfer among the map, and reduce phases.

Master node of Map Phase which has job tracker breaks the input problem into smaller sub units and these smaller units are individually assigned to slave nodes for further processing by the TaskTracker residing on slave nodes of cluster. A slave node could break the task into further smaller units if task tracker computes that processing may require more time, then it recommends to divide the processing sub units further and assigns to remaining slave nodes, this iteration can be further repeated until best result is obtained. End result of Map phase is kind of takes one set of data with a one type of data, and gives a result which is list of key and value couples in a different domain: Map (P1, s1) → list (P2, s2)

A. User request processing by Map logic: – Map logic operates single time for each value of key P1 key and gives result which is grouped by key value P2.
B. Processing of Map output by Reduce stage – While processing MapReduce one node is dedicatedly allocated to run reduce phase which allocates the P2 key value processing to each underline node and in return its gives node executing reduce gives corresponding data belonging to Map for that key value.
C. Processing of user defined Reduce step – All the results from step3 is aggregated by executing reduce phase only one time for each P2 key value given by Map stage.
D. End result – All the results generated by reduce stage is aggregated by MapReduce and which is further sorted based on P2 key to produce final result.

MapReduce is a programming model and suitable and effective for processing huge amount of data efficiently without bothering about nodes failures by exploiting the virtual processing capability of distributed environment and the virtual processing capability is sum of total nodes capability in distributed cluster. Traditional MapReduce model gives best result when jobs are scheduled within a single cluster.

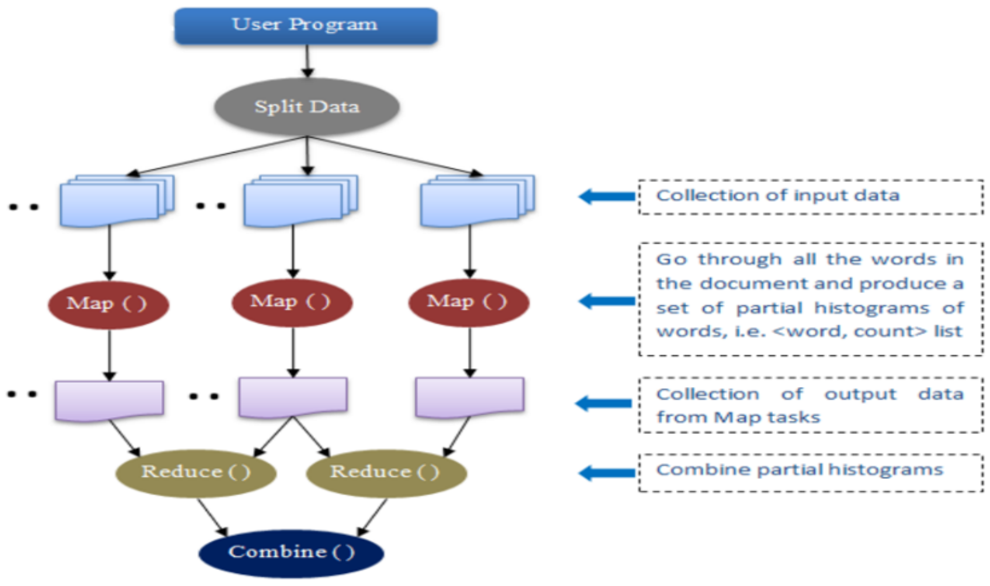

User Program
Split Data
Collection of input data
Map ( )
Map ( )
Map ( )
Go through all the words in the document and produce a set of partial histograms of words, i.e. <word, count> list
Collection of output data from Map tasks
Reduce ( )
Reduce ( )
Combine partial histograms
Combine ( )

Fig 1.6.1 - A Detailed MapReduce stages representation

When job sizes grow beyond a single cluster, single-cluster solutions become inadequate and complex to process big data. This happens because of varying processing capability of a compute nodes of a cluster from another computing node of different cluster in terms of CPU processing, cache size, available memory also depends upon network availability and configured storage limit. However, each cluster pre-defines how much maximum nodes can be provided to a single user at any given time. If these individual clusters work collectively in an integrated manner, they collectively become highly powerful.

However, in case when source data and computing platform are widely distributed, e.g., data is distributed and saved into various geographically located data centre. Hence traditional single-cluster MapReduce methodology doesn’t give optimum result in scenarios where data and processing nodes of a clusters are distributed widely.

Chapter-2

REVIEW OF LITERATURE

2.1 Overview of Hadoop and MapReduce

MapReduce is an approach for analysing a single record or query in HDFS framework. It then combines results into a suitable solution. The Map unit provides data processing capability, while the Reducer Unit combines various outputs of Map units and sorts them based on same key.

Hadoop Common layer

Common layer in Hadoop is a collection of system defined libraries and utilities provided by Apache Foundation, these libraries and utilities can be further taken as reference in other modules of Hadoop system ecosystem. These utilities can be extended in case of HBase or Hive modules wants to access HDFS functionalities or data, they can do that by using Java archives also known as JAR files which are saved into common library of Hadoop Common layer.

HDFS- Hadoop Distributed File System

HDFS is a primary big data storage layer for Apache Hadoop. Hadoop is an area where users can store big data not only for analytical purpose but also archiving the data which can be used later also and its one of the important components of Hadoop ecosystem. As Hadoop is designed based on distributed system, it creates many copies of data in HDFS before storing in order to mitigate the node failure event happens. This data replication makes Hadoop fault tolerant and also capable of continuing processing if any node failure happens. HDFS is mainly formed by three main components known as Name Node, Data Node and backup Name Node. HDFS follows Master and slave kind of architecture where Name Node which works as master node for managing and maintaining information about of the storage cluster and the Data Node works as a slave node collecting various systems into Hadoop cluster.

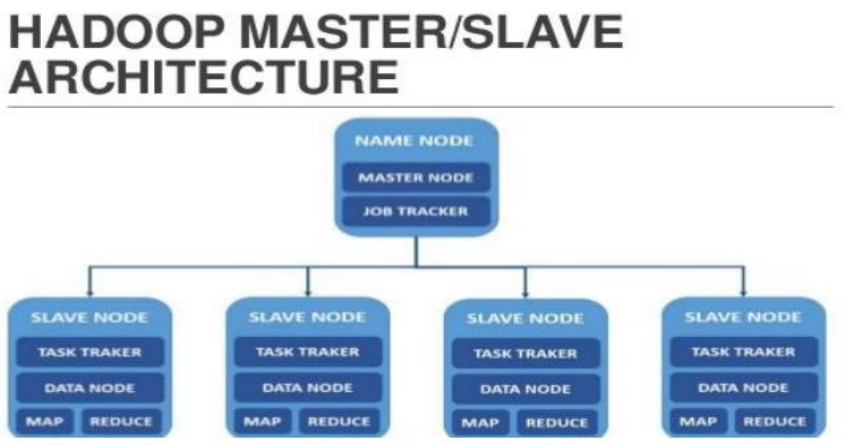
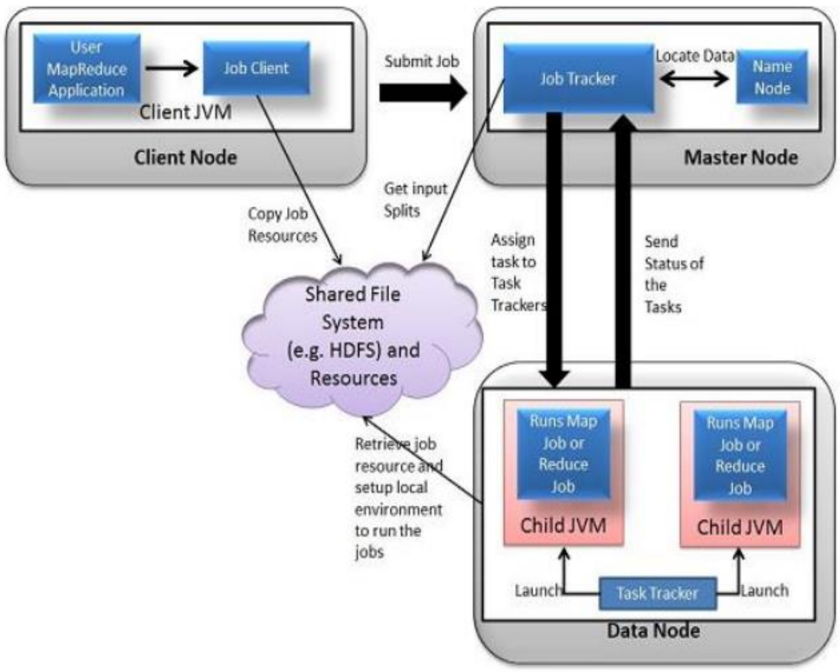


Fig 2.1-Master slave architecture of MapReduce

MapReduce- Distributed Data Processing Framework

MapReduce was developed by Google using Java used to work on big data created to manage applications which are highly data intensive like web crawling and document search which are processed easily using. MapReduce process big data by breaking the data into smaller units of data as these smaller units of data can be processed very quickly in comparison to large data. MapReduce works efficiently by analysing datasets parallelly and then aggregates results to produce final out-come. In the Hadoop ecosystem, Hadoop MapReduce framework adopts the YARN architecture of YARN which provides and assist for processing of large data into parallel manner effectively. This provides convenient and suitable framework for MapReduce to easily develop applications which runs over thousands of nodes for faster and prompt execution by ensuring fault management along with and suitable failure management. Map step of Hadoop takes input data to be processed and breaks that data into independent sub files or blocks and output of Map stage is sent as input feed for Reduce stage. Reduce Stage aggregates all output of mapped data values into smaller. During the processing of MapReduce both input and output of Map and reduce unit are saved into HDFS.

MapReduce of Hadoop ecosystem acts like a processing engine or execution node while the HDFS file system works like data node. The jobs processing and task managements are done by two components of MapReduce which are known as Job Tracker and Task Tracker as per below figure.



User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Locate Data
Name Node
Master Node
Copy Job Resources
Get input Splits
Assign task to Task Trackers
Send Status of the Tasks
Shared File System (e.g. HDFS) and Resources
Retrieve job resource and setup local environment to run the jobs
Runs Map Job or Reduce Job
Child JVM
Runs Map Job or Reduce Job
Child JVM
Launch
Task Tracker
Launch
Data Node

Fig 2.1.1-key components of Hadoop

YARN

Advanced version of Hadoop has main processing engine known as YARN which is an essential part of Hadoop2.0. YARN provides dynamic resource availability over the Hadoop framework without acquiring new nodes for increasing processing for managing increased workload.

Data Access Components of Hadoop Ecosystem- Pig and Hive

Below are the Hadoop components which provides SQL kind of platform to extract data using SQL kind of interactive query processing language.

Pig

Yahoo developed a tool known as Apache Pig which is very helpful into analysing vast data sets quickly and conveniently. It allows usage of language Pig Latin which is very efficient to use, scalable and convenient to use. The most remarkable attribute of Pig programs is that they are open structure and allows considerable parallelization which makes them very convenient to use for processing vast data sets efficiently.

Hive

Hive is one of the main querying platforms which was primarily developed by Facebook. This provides an easy SQL kind of language for querying data and data summarization and data analysis purposes. Usage of indexing makes execution of Hive faster than other query processing platforms.

Data Storage tools of Hadoop – HBase

HBase is mainly a column-oriented database which uses Hadoop distributed file system to store data eventually. HBase assist reading data randomly along with batch read via MapReduce. It’s basically a kind of NoSQL data base, where NoSQL database users or organisations can form large tables having huge number of rows and columns on physical machine. HBase is best suitable for those applications which requires frequent writing or reading access to vast datasets.

2.2 MapReduce introduction and components

Now a days for execution of high volume of data, MapReduce provides faster and convenient processing leveraging the distributed processing and running various tasks parallelly. MapReduce executes large data in parallel manner efficiently on leveraging the processing power of various computing nodes of distributed cluster. MapReduce processing is conducted using two of its primitive components known as map and reduce functions, these are flexible enough to be customized by users as per their need. MapReduce does the faster execution by managing node failure effectively without bothering or revealing the complexity of configuration and node failure management to developer or user.

MapReduce is most suitable for faster execution of vast and unstructured data by utilizing the virtual capacity of parallel processing which is sum of all nodes capacity by splitting input into various small sub units which is distributed to underline slave nodes for processing. One main advantages of MapReduce processing is that it just let developer focus on what they wanted to achieve by hiding other system related complexities.

All nodes in a homogeneous environment are considered of having similar possessing and storage capacity. In vent of any one of the processing node failure, MapReduce find out the best suitable node which is currently underutilized or not having any assigned task. There is also possibility that due to resources limitation of kind of task execution, task assigned is taking longer time to finish and running very slowly these types of tasks are also known as straggler tasks. Post identification of such straggler tasks Map Reduce then starts a fresh copy of straggler task which is also commonly known as speculative copy (Backup copy) which is assigned to a node able to process that task faster. If there is no speculative execution, a job will be execute very slowly and continue to run for a very long time which in terms degraded the performance of MapReduce.

MapReduce processing phases

MapReduce is a programming model developed and advocated by Google for its ability to ingest large volume of data and quick and efficient processing of data by dividing the data into smaller copies and saved across various nodes of cluster and then process them into parallel fashion. It doesn't only provide fast processing but also conceals various system and configuration related complexities which are majorly provides nodes failure and parallel processing features from application developer or its end users to provide them tension free working environment.

MapReduce processes large data set using two functions first one is Map and second one is reduced. There is multiple Map function spanned across various nodes which accepts a partitioned data file as its input which can be located across any nodes in the distributed file system and stores the key and value output There can be possibility that data chunks can exists along with the Map function or not may resides into different node as well. In case input data and the Map function are not available on same node, in such case Hadoop will try to move data file closer to same node where Map log exists.

As explained in the below figure 2.6 the processing of each component of. MapReduce has been displayed with example. This shows in detail processing by taking an input file and then split that into sub units and then how combine and sort is done along with generation of intermediate keys.

These keys are further used to combine data and then create final result using Reduce component using intermediate keys.

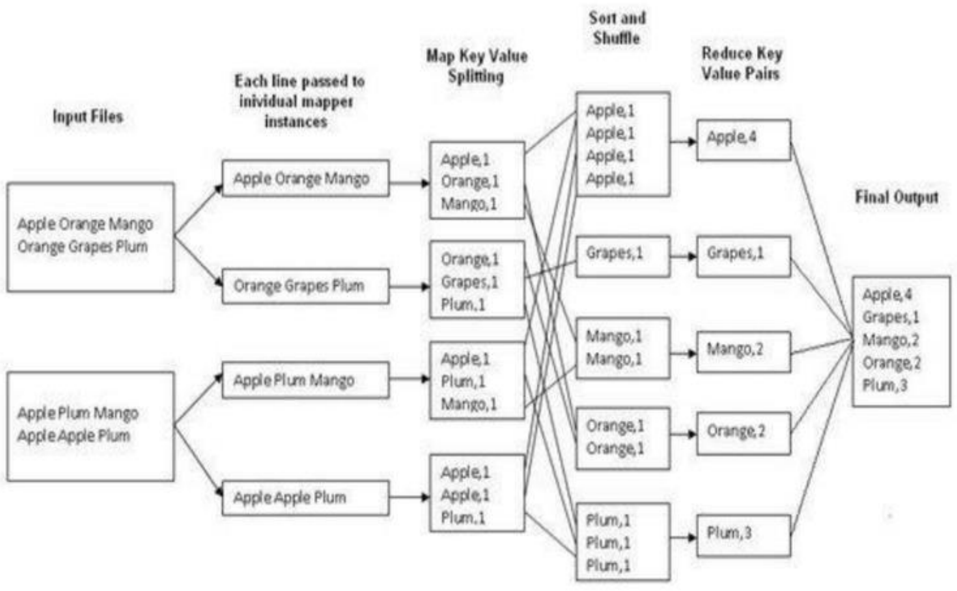


Fig 2.2 -Map and Reduce processing example

The JobTracker is the prime element of this architecture which helps accepting users commands and assign them to underline nodes for processing and it also maintains task status along with ensuring effective node failure management. For processing of large data sets users submit a command which can be form of a map and reduce jobs to the JobTracker for processing, which is accepted and prioritized for processing by using firs come and first-served basis. The JobTracker in MapReduce processing is mainly responsible for assigning task to slave nodes for processing and then monitor the status if tasks and take appropriate measure when node failures happen on any of node processing the assigned task. The TaskTracker running on slave node acknowledge the instruction from Job Tracker and process as per instruction, post completion of task it communicates the job Tracker about the status of task along with management of data transfer among the map and reduce phases.

2.3 Traditional MapReduce data processing

MapReduce name came from Lisp Language Map and Reduce primitives. MapReduce is based on Users defined map function which acts as mapping of a key and value tuple then sort and produce a collection of interim key and value pairs, and a reduce phase which aggregates all interim values corresponding to a intermediate key as pictured in below diagram. This is a highly efficient and popular high-performance computing method to process large data sets.

MapReduce is a best suited for processing and managing processing of huge data sets present on various nodes of distributed clusters parallelly. In the batch-processing system, initially data is organized and then analysed. MapReduce has become the popular batch-processing model. In compared to typical processing methods, MapReduce is a tremendously scalable programming paradigm that can work with massive volumes of data and provide results rapidly by running jobs in parallel over a large number of computing nodes.

Traditional MapReduce data processing limitations

Initially, MapReduce has been running over local clusters and cloud kind of network which is integration of various tightly coupled resources having a single centralized data source. However, this implementation of MapReduce suffers the performance degradation when data source and the computing nodes are widely distributed. There are numerous applications e.g., scientific applications, weather forecasting applications, click-stream analysis and social connect applications have data sources distributed cross other location as well. This is due to high volume of data generated on day-to-day basis and need to process that data makes them to have geographically located data centres or even across the Internet.

There are various degrees of data locality and parallelisation in such multicore clusters environment which affects performance of MapReduce. Additionally, the input data could be very vast and distributed across multiple isolated clusters coupled with wide area network. There may be significant I/O speed differences from local disk storage to connected network topology.

Processing and transferring a huge data sets recursively degraded processing speed and cause the unnecessary overhead into processing. Hence in order to accomplish high data driven tasks JobTracker and its running scheduling methods has to consider execution time also into consideration along with the overhead to data movement among nodes.

A detailed analysis should be carried out conclude the merit and demerit of moving data closer to logic or logic closer to data, depending upon its data movement overhead or processing overhead or is there is possibility that data can be supplied as the need on the fly basis.

It’s not advisable for users to install Hadoop on the multiple clusters having MapReduce in attempt to develop a single largest cluster, because it won’t be possible to access the nodes of any internal cluster from outside due to lack of any software with integrates and acts like JobTracker at the top. In such scenario MapReduce often requires master node of a cluster to assign tasks or communicate directly to any of the underline slave node, which exposes the major limitations of MapReduce frameworks that it is only suitable for processing of large data into a single cluster.

Therefore, it was a need of hour to discover a new framework for MapReduce or method which can collaborate seamlessly with multiple clusters in such way that they appear to be part of single largest cluster wherein one of the nodes is designated as Master node and can access all underline nodes irrespective and devise mechanism to efficient datasets partitioning methodology. This approach will focus to distribute between clusters or among data centres where the data load is similar to or data load is more than required load.

2.4 Hierarchical MapReduce architecture to overcome data locality

In order to mitigate the problem of internal nodes not accessible from outside world, two different approaches are advocated and analysed which will enable internal nodes of clusters accessible from outside world. The first approach emphasises on making synchronization of all physical clusters into one large cluster by introduction of an intermediate layer which acts as infrastructure layer, MapReduce will run over this newly formed virtual cluster. Though this approach effectively addresses the trade-off of inaccessibility of internal nodes from outside world but at same time it suffers from expensive data shuffling across virtual cluster under certain conditions. Second approach is to make some modification into existing ecosystem in a way that MapReduce framework is accessible straightaway to multiple clusters without need to have any additional layer of infrastructure, and this approach is also popularly renowned as Hierarchical MapReduce.

To match its operating paradigm, it must first consolidate the data into a single, centralised cluster before deploying MapReduce to analyse the dispersed data. This approach is incredibly time-consuming and difficult for large-scale data. "Hierarchical MapReduce" frameworks are an alternate strategy that distributes compute jobs around the data centres. Post implementation of typical MapReduce applications on multi-cluster environment it has based found that current Java Virtual Machine based runtime also known as Task Worker didn't succeed to fully utilize the data locality and parallelisation of tasks on one node level. Specially MapReduce open-source implementation of Hadoop uses JVM to execute MapReduce tasks, which doesn't fully utilize the cache hierarchy and parallelisation of task on multi core clusters.

Hadoop need both key and value for implementing Writable interface of Hadoop to capitalize serialization, this may create additional objects and remove overhead along with memory footprint. In some cases, applications may require processing of same tasks multiple times to get final outcome or recursive processing the same. In case of recursive execution, Hadoop fully utilizes data locality for only one iteration of jobs by transferring logical computation near to node having its data or to the node having data, due to not considering data locality for iterative processing requirement for loading of same data from the underline file systems to a node which is processing that data. Following the above facts, Zhiwei Xiao, Haibo Chen and Binyu Zang proposed Azwraith, a Hierarchical MapReduce model whose objective was to maximize the data locality and parallelisation of task for execution of MapReduce applications under Hadoop framework. In case of hierarchical MapReduce approach of Azwraith, ,when a Map or Reduce task is allocated to same node for processing then they are treated as a different MapReduce job and they are further divided into a Map and a Reduce tasks, which are later executed run time by a MapReduce job which intend to boost upon a single node Azwraith design utilizes the data locality on various node of a cluster by encapsulating in-memory cache methos which saves data in memory which can be further utilized while processing .

Azwraith design

In this design Ostrich which is a MapReduce implementation of Hadoop for shared memory multiprocessor was reused. Name Ostrich follows the "tiling strategy" of MapReduce on heterogeneous multi-cluster environment and optimally exploits the functionality of data locality and parallelisation of task on multicluster environment. In Azwraith focus was mounted to increase effective exploitation of data locality and task parallelisation and less rest of the features were borrowed from Hadoop to minimize the development and hence most of the Hadoop components were used as is e. Master node hosted job tracer, slave node hosted Task trackers and Hadoop distributed file system without changing anything into these components. Existing Hadoop can run using original Java base jobs by adding Azwraith extension just like a normal job.

As shown in Fig 2.8 below a complete architecture of Azwraith design [33] for implementation of hierarchical MapReduce processing which looks as actual Hadoop without integration of Azwraith and have reused existing Hadoop components without any changes into them to reduce architecture build complexity.

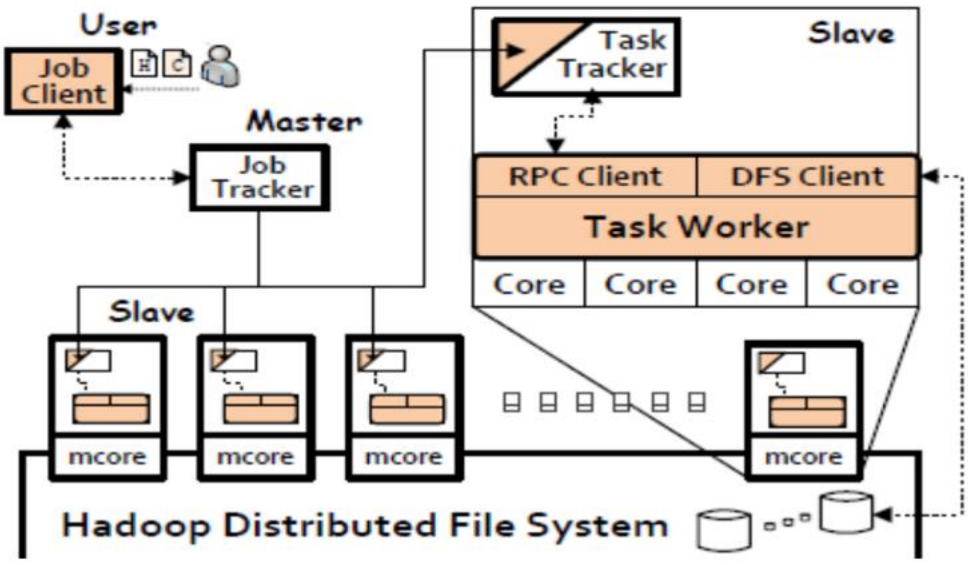


Fig 2.4 - Hierarchical MapReduce Implementation using Azwraith

User submitted job follows same work flow as its in usual Hadoop. User submitted job is executed either Azwraith or normal Hadoop for execution by the selection of processing job type either Azwraith or Hadoop. This selection is made by developer or application users before jobs is submitted for processing. Based on user selection task Tracker residing on slave node will kick-off a process when an Azwraith job is opted by user or trigger and begin a JVM to process a Hadoop job.

In order to make Ostrich compatible with existing Hadoop, there was need to amend the existing Hadoop's processing flow and communication methods which provides critical information's like executing task success or failure status or interim output. SO, in Ostrich new TaskTracker known as TaskWorker which was bind with remote Procedure Call hosted (RPC) on a client node. The remote Procedure Call client acts as a bridge between Azwraith and Hadoop Task trackers services. The Trackworker access the Hadoop distributed files by using a distributed file system also known as DFS client, which establish access with Hadoop.

Networking and disk traffic.

Unfortunately, despite all of its benefits, the Hierarchical MapReduce structure requires the creation of a second Global Reducer function in order to produce results comparable to those of the original MapReduce. The goal of the MapReduce philosophy, which is to simplify parallel data processing and provide transparency for parallelization and scheduling, becomes unintentionally vulnerable as a result. The simplicity of the MapReduce programming approach is also broken, and any existing programmes must be updated in order to run on hierarchical MapReduce frameworks. Therefore, a paradigm that functions like Federated MapReduce (Fed-MR) is needed to conduct MapReduce computation across several clusters transparently. Fed-major MR's goal is to do away with the cumbersome need for the Global Reduce function in hierarchical MapReduce. Though the implementation of hierarchical MapReduce was a challenge and it was less commonly used by organisation.

Improvement into MapReduce processing using scheduling techniques

There are multiple scheduling techniques used to execute jobs efficiently which focuses on faster data processing along with quick data availability to user and fault tolerance. Though these techniques can be categorized into there are three important categories for MapReduce based on their prime focus areas for optimization such as data locality, data synchronization and fairness while distribution of resources. Data locality is a very important factor to impact on performance of a task into shared cluster environment, which is due to availability of limited network bandwidth.

Longest Approximate Time to End (LATE)

In case of heterogeneous environment where all underline nodes of clusters are not if same processing capacity. hence due to varying processing power of nodes makes a scenario wherein some of the tasks may be running very slow due to absence of proper resource sharing or varying processing speed. There are many reasons to bring such difference in processing on cluster nodes e.g., one of the cluster nodes is excessively occupied and CPU intensive jobs are running into that node and this slows down other processing jobs into that node.

In ideal condition and for best results all jobs should finish before the computed average time. Initially this concept was brought into life by Zaharia who provided a solution to reduce the number of tasks talking more than average time by adopting and testing this using LATE scheduler was successful to bring processing time down on significant percentage.

They advised a scheduling algorithm named as longest- approximate-time to end or LATE algorithm which focus on finding the slow jobs to improve Hadoop by identification of real slow tasks by calculating the time left for all the tasks to finish processing.

LATE follows three fundamental rules to reduce the slowness and optimize processing, first one is to give highest priority to slow tasks which are taking longer time to finish in comparison the average time, locate the node which is idle or underutilized and has ability to fish assigned slow performing task faster than average time, deciding the numbers of nodes which can be speculated at a time.

LATE algorithm not only take care of slowness of task but it also optimizes processing by identifying nodes which is underutilized also this exploits node heterogeneity optimally because this will pick up only those tasks which are poorly performing for restarting or re-running. This will maintain the synchronization among map and reduce phases, but consider only poorly performing tasks for processing and appropriate actions.

Data Locality Algorithms

In a homogeneous environment all processing nodes are of same processing capability and have same amount of storage limits of all nodes of clusters. However, in heterogeneous cluster having various nodes with varying processing a good CPU node can finish task quicker than lesser efficient CPU node. An efficient data placement algorithm plays vital role to determine how to exploit the data locality effectively in heterogeneous data cluster and optimize MapReduce execution performance. Data locality supports the idea by moving logic closer to data as this saves time data movement time and resources and increase processing remarkably. This reduces network congestion and enhances the system processing.

Fault tolerant algorithms

Longest Approximate Time to End Algorithm (LATE) utilize idle nodes efficiently by allocating tasks. Hadoop allocates a task using one of three approaches. Any failed task gets highest priority next priority is given to task is waiting status, then map tasks which contains local data on this machine. Last priority is assigned to speculative tasks.

Hadoop computes a turnaround time for speculative processing by averaging each type of tasks progress percentage. A task having less progress percentage in comparison to its category average will be marked as a straggler. LATE identify task which will finish is longest approximate time and will run that speculatively. LATE computes the task's completion time based on the task progress score determined by Hadoop. This further computes the task progress rate of each task along with calculation of task's competition time.

LATE scheduling was one of the options to help reducing the speculation and improve the turnaround time.

2.5 Analysing Scheduling methods to increase MapReduce performance

It's now becoming essential to find out challenges faced by organisations to process and store huge volume of data and techniques to cater the requirement of current heterogeneous infrastructures like cloud to seamless processing of cloud-enabled applications which requires processing of huge data in timely manner along with the assurance of finishing of task in a timely manner. These applications are having strict job completion timelines e.g., to provide critical tasks result into timebound manner), and in event of slippage of deadlines may be unacceptable into organisations along with tremendous reputational damage and impact into revenue.

Linh T.X. Phan Zhuoyao, Zhang Qi Zheng Boon, Thau Loo Insup Lee has conducted a case study to up bring the complexities and challenges pertaining to above issue for online scheduling of MapReduce jobs processed by Hadoop. They have used Amazon EC2 for timelines evaluations and pointed out that traditional Hadoop scheduler is not equipped to manage and process jobs which are supposed to be completed in defined time. However, they have used existing multiprocessor scheduling techniques into processing for cloud environment hosted applications, they have found out significant improvement in performance to minimize challenges involved into processing of a job which require processing under tight timelines. Based on their research and study, they have discussed and elaborated challenges encountered and faced into such kind of processing requirement due to machines virtualization and growth of such applications by utilizing the real-time scheduling techniques which are advanced and suitable for processing application working into cloud environment.

There is lot of development happened into this area and numerous middleware platforms intend to focus on primarily managing and executing jobs which involves vast sized data processing for applications which are hosted into cloud environment, for example, MapReduce, Pig and Dryad LINQ. These intermediate platforms make developer jobs easy for processing into cloud environments by just adopting them which provides convenient and easiest programming, and these middleware's plugins cab be utilized into numerous domains which requires analysis of vas sized data analytics.

Chapter 3

MATERIALS AND PROPOSED METHODOLOGY

Objective of this research is to find-out challenges faced by processing of vast amount of data using MapReduce and also identify challenges of MapReduce processing into heterogeneous environment. Post identification of challenges focus in on to explore the solutions used by other researchers and what is the level of performance they were able to achieve and also any further limitation they had observed into experiment or research.

3.1 Research methodology used for this thesis

Selection of a suitable research methodology is fundamental step for conducting any research. Here blend of two popular research methodology is used to identify potential challenges MapReduce has faced and few of them are tested and evaluated to ensure they provides improved processing time for large sized dataset in an efficient manner. Exploratory research methodology is adopted to uncover potential challenges faced by MapReduce processing and limitations. Understand algorithms used or finetuned in order to alleviate the potential challenges and how much they are able to resolve identified issues.

While experimental research methodology is followed to choose an effective algorithm test it and evaluate results to ensure that they are providing better execution time for processing of large sized data on heterogeneous environment.

Exploratory research methodology comes handy wherein there is less or limited information is available and problem statement is also not very well described. This for better understanding of problem various methods of exploratory can help, however they come with a drawback that they didn’t provide a conclusive result. Thus, we begin with a out statement of problem and then perform various methods suggested for exploratory research which can be helpful for future research as well.

Exploratory research methodology is helpful to know the best research design and selection of subject. There are various methods to available to conduct exploratory research, however keeping objective in centre of research very popular method of exploratory research known as literature is used to identify all prominent challenges faced by MapReduce along with limitations.

Post reviewing various literatures related to challenges faced while processing of large sized data sets using MapReduce various limitation has been highlighted by researchers. Primary identified challenges are explained in details as below.

3.2 MapReduce Performance considerations

It’s not always essential that MapReduce while processing of large data sets always provides fast and efficient results. The primary benefit of MapReduce programming framework is maximum utilization of data shuffling operation, and application users only need to develop logic for processing of Map and reduce phases of processing. Though MapReduce programmer however has to reconsider the sorting and reshuffling stage after the execution. Hence an efficient data partitioning function and volume of data to be generated by Map function could have a significant impact on MapReduce processing. This also adds overhead in form of writing back the interim results from Map stage to cache of node its running. Involving and utilizing an additional feature called as Combiner to minimize the volume of data which is generated and written to disk, and migrated on the connected network. It's very essential for an application developer to think and designing a MapReduce algorithm in such a way that it focuses primarily on processing of job by effective management of data transfer by minimizing the data communication cost. This has been observed by various researchers that while writing and developing MapReduce logic often data movement and communication cost comes more than the cost of data processing, resulting into inappropriate design and management.

While evaluating the MapReduce performance, the challenges of data split mapping, interim results of map stage shuffle, interim results of map phase sorting and then aggregating the results by reducing should be wisely considered as these all plays important roles into performance analysis evaluation. The volume of data generated by the Map phase is primary parameter which shifts the majority of the processing overhead among the map and reduce stages. Reduce phase involves sorting of interim results based on keys which has a relatively nonlinear complexity. Hence, smaller sized partition minimizes time required for sorting of data, but this will become impractical due to having a more numbers of reducers.

3.3 Prominent performance issues in MapReduce

MapReduce is methodology which is designed to process vast data in parallel manner on the very big cluster. These nodes may be of different processing capacity and storage capacity. This was initially developed by Google [3] and later adopted by Apache Hadoop who is responsible for management of this framework. Due to its processing capacity and ability to process large data this has been widely accepted in most he open-source projects by various organisations the most regarded by Apache Hadoop. The popularity of MapReduce is due to its scalability, fault-tolerance and not too much worry of writing lengthy codes for achieving results.

3.3.1 Limitation of data Storage.

Traditional data bases which works on Relational database methodologies are also known as RDBMS were designed and suitable for those applications which generates and stores data which has certain structure and relationship among the data partitions .These data bases were able to accessed structured query kind of language which is popularly known as SQL language .Traditional data bases are facing severe challenge .Technical advancement and various applications are generating verity of data very rapidly which can help organisations to delve into it and extract meaningful business insights out of this. Due to size and variety this kind of data was not able to house into existing RDBMS.

Hence RDBMS were facing severe threat for managing big volume of data along with scalability if new nodes are requiring for quick availability of data and efficient data processing required for Big Data applications. MapReduce provides enhanced processing scalability which is based on available data storage in a distributed file system which may be Google File System or Hadoop Distributed File System. This has given an opportunity to look beyond the RDMBS for storing verity of data and given an option in forms of NoSQL database and NewSQL data bases which provides an alternative to store unstructured and vast big data storage.

NoSQL means Not Only SQL which makes SQL not essential for querying data from NoSQL data stores. Hence, they exploit features like flexibility of defining schema and parallel processing on large number of nodes into cluster without worrying about node failures. These NoSQL based data stores improves read and write operations as an when they are required along with scaling at horizontally level.

Grolinger [8] had earlier analysed features for efficient data replication, data consistency, and concurrency control. NoSQL systems eventually adopts the MapReduce paradigm and pushes processing to the nodes having data is located to efficiently improve read operations. On these NoSQL data stores it's not essential to have SQL kind of query language, however MapReduce jobs can be effectively utilized for data analytics purpose.

MapReduce which is a schema free and index free which give enormous liberty and makes MapReduce to work very effectively with unstructured data which can be semi structured and completely unstructured. Moreover, MapReduce can begin processing when data is available but it doesn't support features supported in relational databases such as unavailability of indexes on MapReduce which makes index related processing performing very poorly. This may be overcome by MapReduce ability of parallelization and scalability.

Map and Reduce functions can be written by application users or developer in any of the languages from R, Python or Perl and these logics are passed to the database or data stores for further execution. NoSQL datastores can be of type column-based category or document-based categories which process user queries as per MapReduce paradigm by ensuring the support for executing application which need various indexing methods. This approach makes MapReduce jobs accessing data using the index which results into significant improvement of query performance. For example, Cassandra which has ability to work based on primary indexes and secondary indexes , Couch DB the also uses Map reduce framework of java script as main language for query and data extraction for achieving the desired task.

A view is consisted of Map function and reduces function which can be optional. Processing output of Map logic is used to create an index and which is further used for faster execution using defined view. Another challenge with MapReduce in terms of data storage is the absence of a any query language which is as convenient or data extraction as SQL. Therefore, various researchers have focused on developing interface or method by which SQL kind of query language can be used over ab above to MapReduce. Apache Hive provide such functionality which provides features to use and run SQL-like language on Hadoop. Another Apache product known as Mahout, constructs scalable ML libraries which can be utilized long with existing MapReduce provided by Hadoop.

Though all these products focus on providing and ensuring efficient data exploration and execution features but they appear insufficient into providing features like data management or advanced indexing.

3.3.2 Lack of Big Data Analytical features

Few of the important features essential for quicker processing are missing into MapReduce which are elaborated as below in detail.

Machine Learning

Big Data applications were developed with an intent, sought to build advanced decision-making systems. This is founded on the idea that having more data allows you to train algorithms in a better way to get the most accurate results for critical decision-making.

Essential element to extract meaningful business insights from Big Data by effective application of Machine Learning (ML) approaches and methodologies. However, the size of big data itself comes with problem to execute the ML algorithms for analysis and training purpose. The analytical and processing challenge thrown out due to size of data makes conventional machine learning algorithms underutilized or less convenient under the conventional development environments. This violated the assumption of ML algorithms design that complete data can be stored in memory and they will work on small size of dataset. With the growing size of Big Data, this assumption was violated and poses a huge challenge and degrades these algorithms performance.

MapReduce which is a distributed processing algorithms was used to cure such problems. Although some machine learning algorithms are parallel in nature and therefore MapReduce paradigm can adopt such machine learning algorithms, however for others kind of algorithms such transition seems to be quite complex. Most of the machine learning algorithms depends upon the effective utilization of in-memory data and therefore this assumption is suffered and this impact on all such algorithms and they appear to be inadequate into adjusting into MapReduce paradigm.

The distributed and parallel characteristic of MapReduce framework is cause of not using them for efficient processing of big data. In case of iterative Graph algorithms for reaching the convergence, multiple iterations are needed and each of the iteration denotes a job in MapReduce. These jobs require more time for commencing and thus they are not suitable due to poor starting time. Moreover, if the data is very skewed then it may increase the probability of stragglers specially during reduce phase, which starts copy task and degraded processing by adding the processing load. Similarly in Gradient Descent algorithms which process in sequential manner and need very big jobs to be grouped together. It also needs each parameter to be modified as per the progress status of jobs after each successful or unsuccessful iteration, which will increase over head related to data movement and overhead pertaining to communication into the processing of job. Both of these algorithms don’t provide big success in terms of minimizing execution. Researchers and industry experts have started thing an alternative solution to address and overcome the shortcomings of MapReduce which will be completely new and independent from existing Map reduce or will be working together with an existing implementation of MapReduce. Pregel and Giraph [17], provides alternative methods which usages the concept of Bulk Synchronous parallel paradigm. They do by enabling all status of processing task to be saved into memory this helps the processing of iterative logics.

Spark [22] is another alternative of proving iterative MapReduce processing. Spark is dependent on abstraction of resilient distributed datasets, and saves shred status of iterative task status into memory to facilitate implementations of iterative processing this is similar to gradient descent. Another alternative known as HaLoop [26] and Twister [23] are designed to adopted by Hadoop as extension to facilitate the processing and support the features required for executing iterative algorithms in conventional MapReduce implementation .Above described extension or add-on interfaces comes certain benefits along with few limitations or challenges while integrating with existing MapReduce applications also some underline incompatibilities between the proposed interfaces or addon software and existing frameworks gives an opportunity to come up with a new interface or tool which will seamlessly provide solution for uniform machine learning solutions for application which requires such capabilities support in existing MapReduce.

For handling of rapidly increasing unstructured or semi structured data has push for need to have more computation availability as per need and additional storage .This has forced various organisation who works on such kind of data and various research labs to invent solutions which takes data growth pace element and do required modifications into either existing methodologies or have new one to ensure required resources are conveniently available to process data .These solutions should not be specific to few algorithms used for development ,hence it’s very important to identify the pre-processing of data, which determine the selected supervised machine learning success. Hence, it’s very important to compute the steps and techniques going to be used for pre-processing of data as this is a very crucial factor for getting reliable for accurate end result from machine learning algorithm selected for processing. To determine appropriate training date sets all unwanted data should be removed by choosing correct data cleaning mythology, establishing appropriate relationship among data by selecting right level of data normalization and selecting appropriate features for data extraction. Hence conducting such data cleaning and normalization imposes tremendous threat to algorithms for pre-processing of big data which involves massive amounts of data tuples which is often very cumbersome and complex as well.

Also processing of various types of data it could be logs, video, audio or pictures types of big data introduces complexity for processing such data. This brings the challenges in form of data heterogeneity and data of having multi-dimensionality. This data heterogeneity brings certain challenges which needs to be addressed for better performance .There may be situations in case of audio data when noise can be accumulated into data and this has more impact on quality of data and thus resulting into false data correlation wherein no such relationship or data exists .Also anomaly arises due to regressors which is known as regression error .This regression error can bring data anomaly or inconsistencies and provide incorrect results or misleading discoveries. This noise has posed big challenge for processing an underlying algebra which is used by machine learning algorithms.

Dalessandro has presented that advantage of taking noise as it comes into data, and then using more capable but relatively less precise and accurate learning models. He advocated that by adopting machine learning algorithms which requires less computations, who will also produce incorrect or relatively less correct interim steps can provide such a model which believes to be processing as good while forecasting new results when they are trained on variety of Big Data. Due to incorporation of noise these algorithms could need more processing iterations in comparison to algorithms which requires less computation. However, these processing iterations doesn't slow down overall processing and produce faster results. Due to this hypothesis, it is concluded that algorithms which requires less computations tend to run much faster and provides similar level of accuracy. Stochastic gradient descent [27] is an example of algorithm which requires less computations tend to run much faster and provides similar level of accuracy.

Besides challenges elaborated above existence of other data sources which may generate unstructured data or data of different characteristics quite frequently which can impact the execution. Hence requirement for data pre-processing or data cleaning can help to clean data and minimize challenge arising due to huge variety of data. In order to mitigate such data risk, solution has been proposed to developed and implement algorithms which assist into data pre-processing and data cleaning by incorporating the features of MapReduce.

Discussion till now was to highlight a need for developing and incorporating data cleaning and data pre-processing techniques but important question is to explore the techniques to integrate the data analysis along with pre-processing, which throws challenge and opportunity for organisations to come-up with new research and alleviate such issues for providing accurate results for machine learning algorithms.

Another characteristic of big data known as velocity brings the challenges known as concept drift in to a machine learning model. This concept drift become a threat into MapReduce as this requires data cleaning and data pre-processing which unnecessarily causes additional delays. Hence precise and correct computation of concept drift identification is very necessary and important for researchers to remove the anomalies and gives machine learning approaches accuracy with application using Big Data. Predictive modelling is an important component of machine learning algorithms. Machine learning algorithms used for predictive modelling takes set of inputs and outputs and on the basis of them it predicts an unknown output having certain probability. Adopting machine learning predictive algorithms many organisations can gain lots of early warnings signs specially related to fraud detection of credit cards, choice recommendation systems, harmful URL identification, and there are more areas where predictive modelling can be helpful. Online movie provider companies e.g., Netflix can know the list of movies based on previous selection of a client based on data collected from clients browsing history and choices they have made historically to assists clients to develop an accurate recommending system.

Various researchers and industry experts have categorized predictive modelling parallelisation techniques into below three areas.

Execute predictive modelling algorithm on smaller set of data and then collect the result out of them. Produce interim results from smaller set of data and then user interim output to generate final output. Parallelize the logical computation for better and faster result.

In big data most importantly preferred and used categories are two and three. Category 2 which is actually defining a MapReduce job which accepts smaller partitions as inputs and gives interim output using Map phase and this output is further aggregated based on interim output's commonalities to give final result using Reduce phase processing. Category 3 also have correlation with a MapReduce job, in case an inherent linear algebra can be broken down into units of Map and Reduce sub tasks.

MapReduce combined with predictive modelling phenomena have a major challenge which constraints its applicability in case of predicting data which is highly corelated. MapReduce performs best in case of observations which are processed individually. Means data can be divided, computed and then combine together. However, in case of correlated observations processing, MapReduce proves handy on working with non-distributed architectures. This is due to fact that observations that are correlated and present within disparate clusters, this adds up great performance overheads while data communication among clusters.

An appropriate and wisely adoption of Big Data open avenues for giving efficient decision-making systems along with other business intelligence related business insights which could be beneficial for organisations to offer superior decision making. This is due to the accessibility of variety of much more data which can help predictive algorithms to give accurate and cost-effective solutions to their business problems. Application of Machine Learning can be very helpful to explore great business insights related information from Big Data. However, vast and huge size of dataset poses challenge into processing and execution of machine learning algorithms for analysis and data training purpose.

3.3.3 Lack of online data processing capability

Managing and working on big data where results or information needs to be available on real time makes applications adopt the features which can provide and support online processing capabilities. This can be categorized as real time execution of data which is generated continuously. This continuous data stream is also known as data streams. Organisations wants to get analytical insights from these data streams to make real time data trends or devise some strategies for swift reaction to them. This fast action can help organisations to save cost and stay competitive along with providing right services and add values into services for their customers. Now a days fraud detection algorithms in financial institutions or banks widely monitors real time data of customers transaction patterns and invokes fraud alerts in case of algorithm detects some abnormal behaviour.

3.3.4 Processing variation due to cluster heterogeneity

In MapReduce computation methodology, map and reduce sub jobs are allocated to map and reduce slots to the slave nodes of distributed cluster. Usually there is due diligence is taken while determining the how many maps and reduce sub jobs are required for achieving optimal resource utilization. We have observed that resource utilization become poor in situation having not enough tasks as comparison to resources reserved for idle slots deemed to be wasted. Resource sharing can be one of the areas to improve resources utilization by enabling executing tasks to claim unclaimed or free resources and relinquish control back so that resources can be reused whilst nay new job is assigned for processing and required similar resources for processing. It exploits the available resources which are currently in free status or not been used by any jobs else these resources will be wasted and will reduce the efficiency, thus making optimum utilization of free or unclaimed resources will improve overall resource utilization and gives optimum job execution time. To capitalize the above underutilization an algorithm named as benefit Aware Speculative Execution was conceptualized and developed by experts which predicts the probability of starting new copy tasks against the task which is performing poorly and helps eliminating redundant spawning of duplicate tasks also known as speculative tasks. Primarily, MapReduce performs well in distributed cluster wherein all nodes are of same processing capability or homogeneous in nature, however this processing suffers a lot into distributed network wherein nodes are of varying processing capability or heterogeneous network. There is another scheduling methodology popularly known as network heterogeneity aware scheduling which is a predictive scheduling technique which takes network variability into consideration for scheduling which allocates both map and reduce tasks for resource optimization. Overall objective is to boost Hadoop performance for efficient processing in heterogeneous environment by optimally exploiting available resources. These all systems adopt native data locality aware scheduling which drastically minimized the movement of data across various nodes of Hadoop while executing any job.

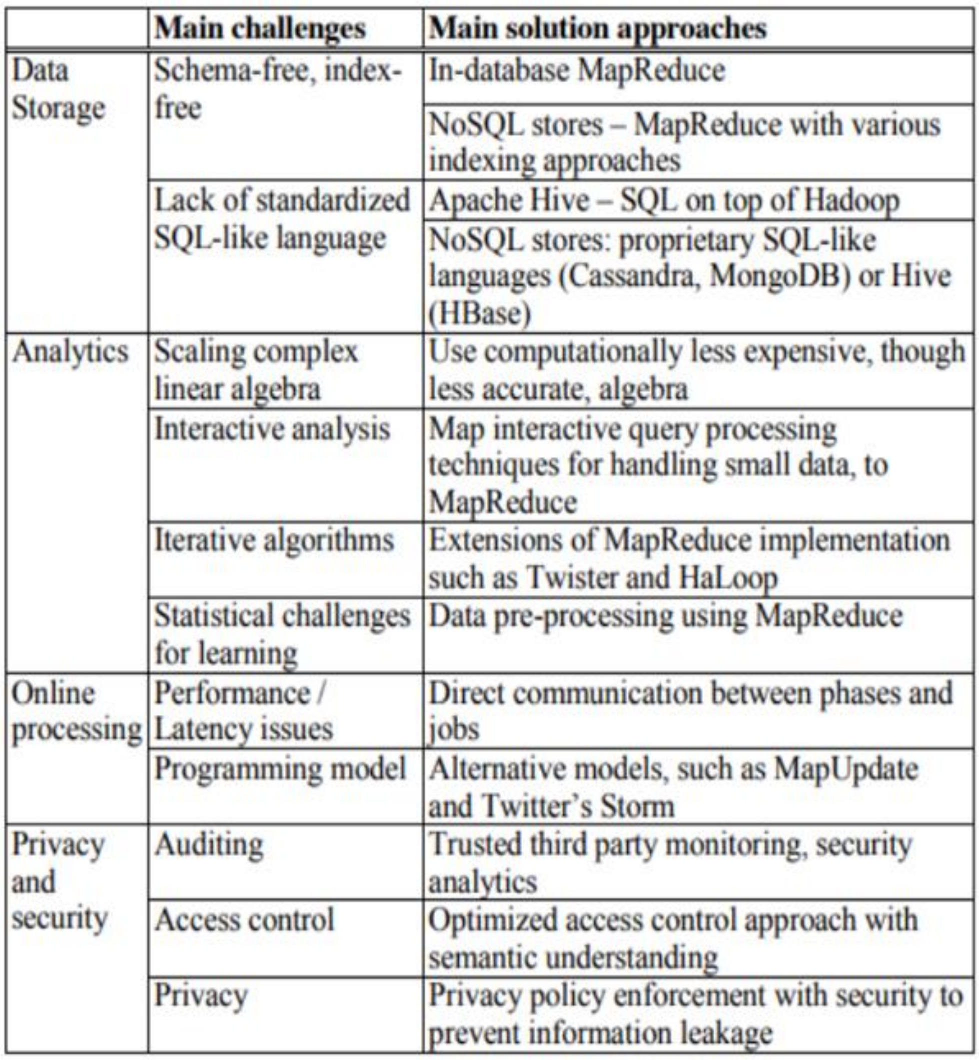
In today’s high-tech world most of the servers comes with processing supported by multicore processors to facilitate multiple computational tasks running concurrently on single node without arising situations related to contention of resources while executing the job in parallel manner. However, Hadoop should ensure that concurrency of parallel processing task is maintained so that resources and processing is not overshoot and they are exploited within limit for achieving optimal resources utilization.

The optimal configuration can’t be alone achieved just by tuning the correct configuration of underline hardware but it also depends upon the workload generated by applications. Hadoop provides liberty to users for maintaining and ensuring concurrency control by themselves instead of doing the best setting configuration by itself. In a distributed environment each slave node has many slots which can be configured by developers or users to run map and reduce jobs. This has advantages as nodes having heterogeneous configuration of hardware’s will be configured differently in comparison to node of homogeneous configurations of hardware. At any given time, only one task per node can run and vice versa one slot can accommodate just a single task to be allocated.

However this mechanism has one pitfall, in event of all slots of nodes are not utilizes then full utilization of node can't be achieved, due to wastage of resources belonging to not utilized slots .In order to achieve and improve resource utilization its essential to find a solution which acquires resources dynamically from available resources and relinquish then once processing is complete for each of task executing in system at that time .This will make sure efficient utilization of slots and resources .Hadoop manages fault-tolerance by starting a copy of task which is performing poorly also known as speculative task .When a copy task finished its processing ,all tasks which are duplicate the one which is speculated are instantly aborted to ensure concurrency and reducing the possibility of re-work. Hence this allows the execution of job, in case when task has failed as this failed task can be started in some other node for competing the assigned job. Hadoop uses task progress score to compute and know the tasks which are taking longer than average time and can be speculated while processing of an assigned job. This is to avoid a situation when speculative become inefficient when slowly performing tasks is very close to finish and has been speculated unnecessarily. Hence to achieve Benefit Aware Speculative Execution is an algorithm which ensures that speculative tasks are started only when are believed to finish before the actual running task. Heterogeneity can't be ruled out because any due to procurements of hardware's and machines over different time frames has made the cluster as collection of nodes having varying processing and Hardwar ware configurations. Advancement and excessive usages of cloud also contributes into heterogeneity and this has made organizations and researchers add and grow their infrastructure dynamically by just integrating either organisations dedicated resources or rented out resources from third-party vendors e.g., Amazon.

3.3.5 Challenges of MapReduce in Cloud based heterogeneous environment

Cloud heterogeneous environment is a new paradigm for processing of job in distributed and parallel manner. Cloud is consisted by physical and virtualized computers which are interconnected and can be incrementally added into network for meeting the demand of additional processing. As existing Hadoop MapReduce was not able to fully capable to perform and execute into cloud based framework ,this has led to modify the existing MapReduce process in order to operate into cloud environment by considering the heterogeneity awareness .This will ensure that MapReduce will not only processing into Hadoop framework but it can also be compatible running into private or public clouds, like Amazon EC2 new versions of framework were developed to take exploits features and functionalities supported by cloud computing operating systems. AzureMR and the CloudMR were developed on the top of cloud of Azure which is a Microsoft product Amazon EC2 respectively. The main advantage of this development is a new architecture, without a server node.



	Main challenges	Main solution approaches
Data Storage	Schema-free, index-free	In-database MapReduce
		NoSQL stores – MapReduce with various indexing approaches
	Lack of standardized SQL-like language	Apache Hive – SQL on top of Hadoop
		NoSQL stores: proprietary SQL-like languages (Cassandra, MongoDB) or Hive (HBase)
Analytics	Scaling complex linear algebra	Use computationally less expensive, though less accurate, algebra
	Interactive analysis	Map interactive query processing techniques for handling small data, to MapReduce
	Iterative algorithms	Extensions of MapReduce implementation such as Twister and HaLoop
	Statistical challenges for learning	Data pre-processing using MapReduce
Online processing	Performance / Latency issues	Direct communication between phases and jobs
	Programming model	Alternative models, such as MapUpdate and Twitter's Storm
Privacy and security	Auditing	Trusted third party monitoring, security analytics
	Access control	Optimized access control approach with semantic understanding
	Privacy	Privacy policy enforcement with security to prevent information leakage

Fig 3.3.5 Challenges of MapReduce

3.4 Selection of research methodology to evaluate test results

Experimental research methodology is used to set up test environment and conduct experiment for comparing results related to how data locality and speculative execution has downgraded performance of MapReduce processing into heterogeneous environment.

Experimental setup has been done to evaluate how introduction of LATE (Longest-approximate-time to end) algorithms has helped to boost performance of MapReduce, this has been discussed in Chapter4 in detail. Also, MapReduce application to process K-Means clustering has been evaluated to see how MapReduce based k-Means implementation has booted processing and see results if that proves the experiment right also highlight key limitation of this algorithm. Further advancement into this algorithm has been done by introduction of parallelism to overcome slowness induced while processing of large data set.

Post identification of key challenges of MapReduce processing of Big data, couple of them are chosen for eliminating or working on solution to overcome those challenges and find an appropriate solution. These solutions are tested and evaluated to measure if they have mitigated those identified risks and able to boost the performance.

3.5 Overcome data locality issues of MapReduce to enhance performance

MapReduce name came from Lisp Language Map and Reduce primitives. MapReduce is based on two key components which works together for processing of vast amount of data by breaking task into smaller pieces and executing in parallel manner on various nodes of cluster. These two functions are “Map” and “Reduce” function ,and they can be customized as per the requirement of application user or developer .Map function receives the input file to process which is generally stored in file or directory format in Hadoop HDFS .Map takes input file line by line and then to achieve faster processing it splits that input file into N number of parts which are sent to various nodes of cluster to execute on parallel manner and produce key/value pair for further steps which are sorted and gives a set of intermediate pair consisting of key and values and reduce based on same key value .In Next phase of processing reduce function aggregates all interim values belongs to same intermediate key to provide end result This is a highly efficient and popular high-performance computing method to process large data sets .

MapReduce follows a very convenient execution methodology having a primary layer consisted of two main logical modules map and reduce whose implementations can be changes or redesigned as per the need and request by users. Main advantage of Map Reduce is its processing capability to take care of a node failure by not letting the users about the intricates of node repairing from programmer and let them focus on what they intended to achieve.

The split data file may or may not be present at in node along with the Map function in same node. In a scenario when data required for processing is not present into same node containing Map function, then in order to maximize the processing time system will transfer the data block closer to node having Map logic or on the node having Map Logic. Basically, MapReduce follows the Master and slave kind of architecture commonly followed in distributed systems. Wherein one of the node acts as primary node called as master node containing JobTracker which is issues processing commands to other nodes called as slave nodes and collects their processing status confirmation back from task Tracker running on each slave nodes as shown in Fig 3.5.

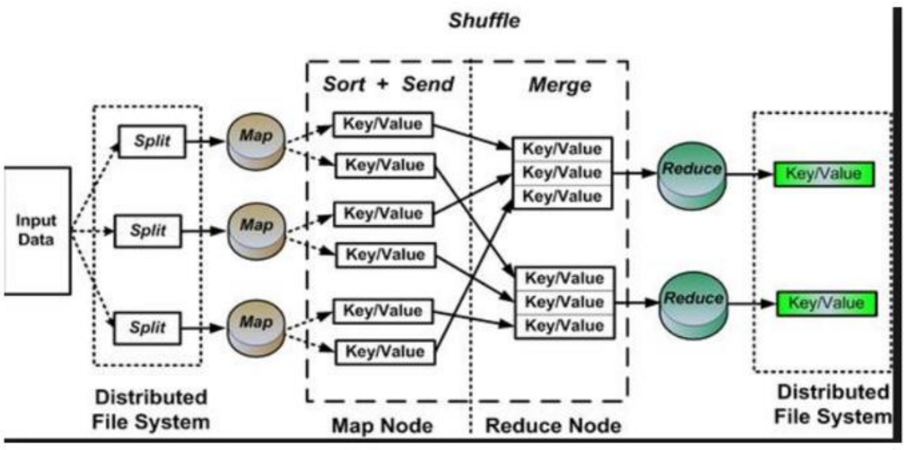


Fig 3.5 Detailed execution of MapReduce

Jobs submitted by user are accepted by Job Tracker. these are allocated to other nodded for processing basis on the order they were submitted. The JobTracker manages the allocation of map and reduce tasks to the slave nodes running Task trackers. The Task trackers gets the tasks and running instruction e.g., required resources or data chunks by Master node JobTracker and maintains transferring of data between the map, and reduce phases.

We can say that JobTracker works as a layer between users or analyst who submits the job for processing and the other nodes responsible for processing without letting user know system complexity or on which node its task in running and later results are aggregated by JobTracker and given back to developer or user.

Below are the detailed steps of processing of MapReduce.

a. Map – The master node accepts the large input file and break into smaller logical processing units, which are further assigned to underline various slave nodes available for processing. A slave node may further assign this processing to other slave node by breaking logics again to smaller units and this can form a multilevel tree structure. Map accepts one tuple of input data and gives back list of tuples of a different domain: Map (p1, s1) → list (p2, s2).
b. Execute user provided logic for map stage – Map phase will run one time for every value of P1 key and it gives an outcome P2 which is a consolidated new set of key values as shown is above figure.
c. Send Map phase outcome to Reduce stage – One of the systems is identified and designated to execute the Reduce phase in the MapReduce system corresponding to key value P2 to each node, and gives that machine all corresponding output yielded by Map processing for associated key value.
d. Processing of user defined Reduce logic – Reduce is run exactly once for each P2 key value produced by the Map step.

e. End result generation– the MapReduce system consolidates all the outputs received from Reduce logic and then these are further sorted using key P2 to give final result.

Below are the popular examples of usages of Map/Reduce for processing huge data related applications:

a. Patterns identification present into a number of files using Grep.
b. Counting frequency of URL access on web.
c. Identifying all the URLs that are connected to a given target using Reverse WebLink Graph.
d. Distributed Sort.

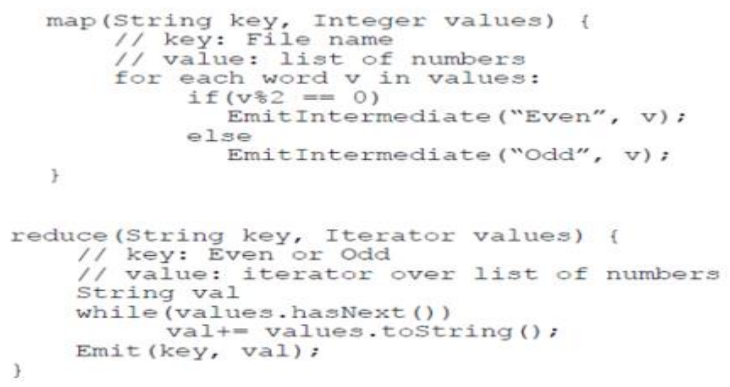
The map function accepts a key and a value tuple and gives another key and value as a result of processing. There are numbers of such Map functions which runs in parallel on the partitioned data, distributed across the various nodes of a cluster and generated a set of interim key and value tuple. Next stage is reduced function which takes all associated results corresponding to a key, generated from Map function. Both Map and Reduce functions are defined as below along with intermediate results used in both stages.

map (p1, s1) → p2, s2

reduce (p2, list (s2)) → s3

Map/Reduce Example

Map/Reduce functionally can be elaborated as per below. This elaborates how a map and reduce steps works into dividing set of numbers in odd and even category number by accepting input string.



```
  map(String key, Integer values) {
      // key: File name
      // value: list of numbers
      for each word v in values:
           if(v%2 == 0)
              EmitIntermediate("Even", v);
           else
              EmitIntermediate("Odd", v);
  }

reduce(String key, Iterator values) {
     // key: Even or Odd
     // value: iterator over list of numbers
     String val
     while(values.hasNext())
           val+= values.toString();
     Emit(key, val);
}
```

Fig 3.5.1 -Algorithm for Odd and even number using MapReduce

For given input numbers 2,5,9,12,13,11,19,45,66,65,72,32,89,91 the output from MapReduce will be as below:

Even: 2, 12,66,72,32

Odd: 5, 9,13,11,45,65,89,91

MapReduce is immensely becoming popular .as this is suitable for processing of large-scale data driven applications such as Web data mining, decision making customer experience, network traffic analysis, machine learning and scientific data analysis distributed across various clusters in parallel fashion. Hadoop regarded to be most common and widely used open-source system which uses MapReduce programming model. In Hadoop, input files are divided into many mall sized data blocks and these data blocks are distributed over many nodes in underline clusters. In order to efficient processing of data blocks, Hadoop must use an efficient scheduling mechanism for improving the performance in distributed and shared cluster environment. Primary concern in Hadoop scheduling is mainly caused due to data locality issues which are caused by limited network bandwidth.

3.6 Detailed processing of Hadoop

Below Fig 4.1 shows key components of Hadoop which are backbone of processing using MapReduce.

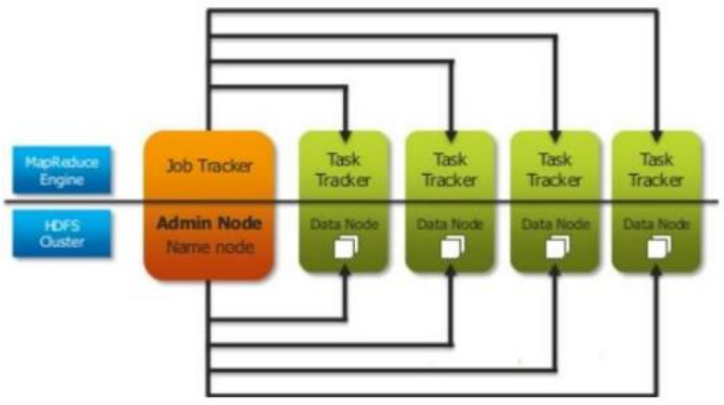


Fig 3.6 Representation Job tracker processing

Job tracker

Job Tracker resides on master node of Hadoop distributed environment and it is responsible for controlling and managing nodes-oriented task trackers, resource availability. The Job Tracker is mainly a node which manages and controls the job processing of Job tracker which executes a MapReduce task on specific nodes of cluster. In event of task is finished Job tracker communicates status to client and provide desired result. Client applications could request for Job tracker logs for information for root cause analysis.

Task tracker

Task Tracker receives instructions or commands from job tracker and executes the assigned task and communicates result to job tracker on regular basis. In event of any failure also Task tracker informs to Job tracker so that job tracker can restart a failed task at different node in order to complete assigned processing. Every Task tracker is pre-configured with a number of slots, and in each slot can accommodate one task to run. Means task tracker slots denotes how may task it can run a given time.

Name node

The name node is components of Hadoop which maintains information about data nodes and data blocks which are partitioned. This maintains information like on a particular a map to, what block locations and which blocks are stored on which data node. In event of corruption of data node for a particular block, table containing the information will be updated and when a data node with problem is found due to node failure of network failure both the tables are updated. The recording in tables is done to capture status of task of data node to make sure it is started correctly from failed status or failed node. It doesn't rely on any nearby blocks or the placement of any other blocks to go to its final destination. Every block, along with the job nodes and related processes, is isolated.

Data node

In Hadoop a node which saves is called as data node. In order to ensure thing are working correctly as per process, to confirm that they are operational, each data node sends a heartbeat message to the name node every three seconds., if any node doesn’t respond then it will be assumed as failed node or out of service and appropriate action will be taken to allocate the failed node running task to some other less utilized nodes in cluster.

In such case any other node of cluster will begin processing on some other data node. The data nodes update the name node with the block information periodically.

Map-Reduce is a logical way of representing and anticipating effort needed for executing and producing appropriately sized datasets which can be used by jobs. Users decides the cost of computation using the processing involved while implementing map and a reduce. Programs written using the MapReduce will be automatically processing in parallel manner.

3.7 Popular Scheduling techniques in Hadoop

Hadoop follows and many scheduling algorithms as per the need. Default type of scheduling it follows is FIFO where jobs were executed as per the order they arrive for execution. But in such case a job which arrives first and wants to take resources hold for execution. But in such case a job which arrives first and wants to take resources hold for longer will push the other jobs into waiting for longer time. Hence few other scheduling jobs were adopted to give best result and faster execution for e.g., priority in jobs was added.

Default FIFO Scheduling

A FIFO queue is the method of operation used by the default Hadoop scheduler. When a job is broken up into separate tasks, those tasks are added to the queue and given free slots when they become available on Tasktrackers nodes. Even so, choosing job priorities is necessary because FIFO scheduling does not automatically take care of them.While processing a job which requires longer processing and acquired resources, causes waiting time increment for rest of jobs which are looking for resources to be released.

Fair Scheduling

The Fair Scheduler was invented by Facebook for managing and controlling Hadoop resources in their distributed environment and later post successful usages this was subsequently given to Hadoop community for further management and maintenance. Every user will eventually receive a fair part of the cluster capacity thanks to the Fair Scheduler. Users can assign jobs to pools, and each pool will receive a minimum amount of guaranteed Map and Reduce slots. While excessiveness capacity in a resource pool is shared among jobs, vacant slots in underutilised or idle pools of resources may be provided to new pools. The Fair Scheduler follows the concept of pre-emption, so if no fair deal was received by resource pool for defined period of time, in that case Fair scheduler module veto pools tasks flowing more than defined capacity in dictate to afford the pools slots which is functional under capacity.

Also, Hadoop administrators may impose predefined priorities configurations on doomed pools. Hence Fair schedular schedules tasks in interleaved manner as per their pools assigned priority, and the constellate capacity and activity of pool. While processing of jobs, their allocated tasks processes by Task Tracker slots, The scheduler keeps note of the difference between the user-submitted job's actual measurement error and the recommended fair percentage. Hence over a period of time this ensures that almost equal number of resources are received by all tasks/jobs. Smallest jobs are also assigned sufficient resources to complete as per timelines quickly. This also guarantees that jobs require longer time for execution are not kept away from resources for long time.

Capacity Scheduling

Capacity Scheduler was initially invented and used in Yahoo. This scheduler rather than assigning the fair resources and time to complete the task focus on the utilization of considering large numbers of users, and there is a requirement to ensure that each users has fair capacity for processing of a job by sharing the resources fairly among the users. Each organization has its own resources available who have sufficient processing ability to meet the organization's requirement to process jobs under the desired timelines during the peak conditions or closer to peak conditions. Managing such clusters in an organisation becomes tedious and overhead and results into inefficient resources utilization in each or the organisations. Every organisation has to think about their own mechanism to optimize their processing and spent time and cost to make that suitable for processing during peak hours of processing demand. An opportunity to share resources of organizations can be a cost-effective attempt to meet the demand of processing during peak hours and lending capacity from other organisation cluster at a time of requirement to process a large Hadoop job because this will permit them to harness the advantage of getting more resources with-out incurring any additional cost of procuring and embedding private clusters. However, organizations are apprehensive while sharing of clusters that during peak demand hours their timescales to finish business critical tasks might be hampered and they might be in a situation to miss necessary and vital SLAs for tasks.

This has led Yahoo to come with an idea of resource sharing and making necessary provisions to permit resources sharing among large clusters or organisations and ensuring each of the organization gets a minimum guaranteed processing capacity and didn't land into starvation kind of situations. This is popularly known as capacity schedular. The noble thought behind this holistic approach is to allow partitioning of available resources in the Hadoop Map-Reduce cluster among those organizations who agreed in this noble concept and has contributed to bear the cost of forming such large cluster to meet all organisations computing needs. This resource sharing allows other organisations to access available unused capacity nor currently utilized by others organisations at that time. This provides scalability for an organisation in a cost-effective way.

Sharing clusters between various organizations requires a provision for multi-tenancy because every organization should feel that they have guaranteed processing capacity and safety provisions while sharing of capacity among other organisations. Hence Capacity Scheduler make strong provision to discourage disproportionate usage of resources of cluster by a single job or user or queue. To overcome such situations JobTracker of the cluster and the Capacity Scheduler sets an appropriate limit on initialized or awaiting tasks and jobs submitted by a single user or queue to guarantee fairness to make sure no disproportionate usage of resource sharing happens and rest of clusters users also gets due capacity to process jobs.

Due to amount of data generated by various application now a days has paved way to think for necessity for continuous efforts in order to improve the available scheduling algorithms and research and find option to fine tune existing scheduling algorithms in accordance to data processing need by various industry and research organisations.

Latest advancement was done to propose Delay Scheduler [29] and Dynamic Proportional Scheduler, these schedulers were suitable to process data processing need specific to certain kind of processing need. These schedulers facilitate Hadoop by providing various service to manage jobs processing based on priorities levels assigned to their jobs by application but this suffers from absence of assurance for job completion time. Another attempt was made in form of Deadline Constraint Scheduler [31] which manages deadline challenge by ensuring optimized resource utilization in order to finish jobs under assigned timelines. All researchers have focused on improving and ensuring capacity equally among executing jobs and application users, least focus is given on consideration of resource availability on granular system resources level. This has given new direction to think from making efficient utilization of system resources along with ensuring the timely completion of job. Effort into this direction was turned into development of new scheduling algorithm known as Resource Aware Scheduler [32], this scheduling technique considers the availability of system resource at the point a request was submitted to Hadoop for processing a job instead of overall resources of Hadoop. In the following sections merit and demerits of various Hadoop scheduling jobs will be conducted which is based on the work already progressed by various researchers on different Schedulers.

as we all are aware In heterogeneous situations, MapReduce's ability degrades, and some scholars and industry experts have suggested methods to increase its processing power. Each method just addresses one particular area of the MapReduce characteristics that seems to be a bottleneck when MapReduce is used in a heterogeneous cluster.

3.8 Fine tuning of data Locality algorithms to improve MapReduce processing

Data locality is a decisive factor to access the MapReduce performance. Algorithms that succeeded to optimize data locality utilization into a heterogeneous environment in a Hadoop cluster. These are elaborated in detail as below with approach followed and issues or limitation while implementing such algorithms.

Data Placement in Heterogeneous Hadoop Clusters

Data placement is a methodology to distribute data among various nodes of cluster based on various parameters e.g., storage availability of nodes, distance etc. Large data is distributed by Hadoop among various nodes according to disk storage availability of nodes. This data placement strategy works perfect in case of cluster where all nodes possess the same computing and disk storage capacity which is also known as homogeneous environment. In heterogeneous Hadoop cluster where all nodes are different in terms of processing capability, high-performance node can finish processing of tasks faster when data is locally available in comparison to a node which is lower in performance. After faster node finishes its processing of data located on same node's local disk it makes it self to free and ready to accept unprocessed data present from slow node remotely. This data transfer of unfinished data from poorly performing node to identified faster processing node can add high overhead into cost and degrade processing by reducing processing time. Hence this was need for an effective data transfer/placement strategy which focuses on minimizing the data movement in heterogeneous environments among nodes to exploit the benefit of data processing by faster node capable to finish jobs quickly. J. Xie et developed a data placement algorithm in HDFS which distributes and store a large volume of data in various nodes of heterogeneous nodes as per each node's computing capacity. This ensures that data transfer in form of file splits is done according to disk storage and its processing capability to get optimized results and faster processing.

This data placement algorithm in HDFS follows two level implementations. In first level of data placement algorithm distribution of the file segments among the nodes of heterogeneous cluster is done in accordance to each nodes processing capabilities. In second level data redistribution is done to mitigate the data skewness problem caused due to reorganizing the file segments.

It very crucial to evaluate the level of Hadoop cluster's nature of heterogeneity of a Hadoop with respect to its variability of data processing speed while developing the data placement algorithms. Level of heterogeneity in the cluster may play vital role while the processing of different MapReduce applications because of different processing speed which is essential for data-intensive applications. This has led a concept to compute the "computing ratio" to determine each nodes speed of execution of a task on heterogeneous environment.

The first phase of data placement algorithm begins by breaking huge input data file into a sub segment of equal sized smaller portions. This is responsibility of Centralized data distribution server to distribute equal sized data chunks across all nodes of clusters. There is no defined rule is followed while distribution of data sets in nodes, usually they are distributed across various nodes in round-robin manner to allocate input file segments to nodes of clusters depending upon on nodes computing ratios computed for that specific heterogeneous cluster. This gives a notion that node having smaller computing ratio are capable to process large data with high speed and it can be considered as a fast node. Hence this fast-computing node concluded basis of less computing ratio is deemed to be suitable for task required processing of a large number of data segments quickly. However, node with high ratio of computing ratio implies their low processing speed and they are good to execute a smaller number of file segment.

Seamless distribution of segments of input file allocated by the initial data placement algorithm can be violated due to below situations or scenarios:

• When new data is added into existing input file.

• Data segments are removed from the current input file.

• In case of addition of new computing nodes into an existing cluster. In such case data redistribution algorithm is implemented to reorganize the file segments in accordance to computing ratio of new node added into cluster.

The performance of this data placement algorithms was evaluated into a heterogeneous Hadoop cluster as basis on below mentioned cluster hardware configuration and this was tested and evaluated for two data-intensive applications which are very common and known as Grep and Word Count.

Below tables mentions all critical factors of a cluster including CPU processing speed and number of nodes and evaluated computing ratio of each node for testing of data placement algorithm.

Table 3.8.1- Hadoop Heterogeneous Cluster configuration.

Node	**CPU Model**	**CPU Processor**	**Catch Memory Size in kb**
Node-1	Intel core 2 Duo	2*1=2 GHZ	204
Node-2	Intel Celeron	2.8GHZ	256
Node-3	Intel Pentium3	1.2 GHZ	256
Node-4	Intel Pentium3	1.2 GHZ	256
Node-5	Intel Pentium3	1.2 GHZ	256

Table 3.8.1 shows the computed processing ratios in heterogeneous Hadoop cluster for processing of two major tasks of Grep search and Word Count. Six data placement decisions were used by then to verify their work.

Table 3.8.2 Test scenarios

Notation	**Method of data placement**
T1-2-3.3	Distributing files as per processing ratios of the grep.
T11-2-5	Distributing files as per computing ratios of the wordcount.
480 in each	Average size of segment in each node.
All in Node-1	All segments are allocated to 1.
All in Node-2	All segments are allocated to 2.
All in node-3	All segments are allocated to 3.

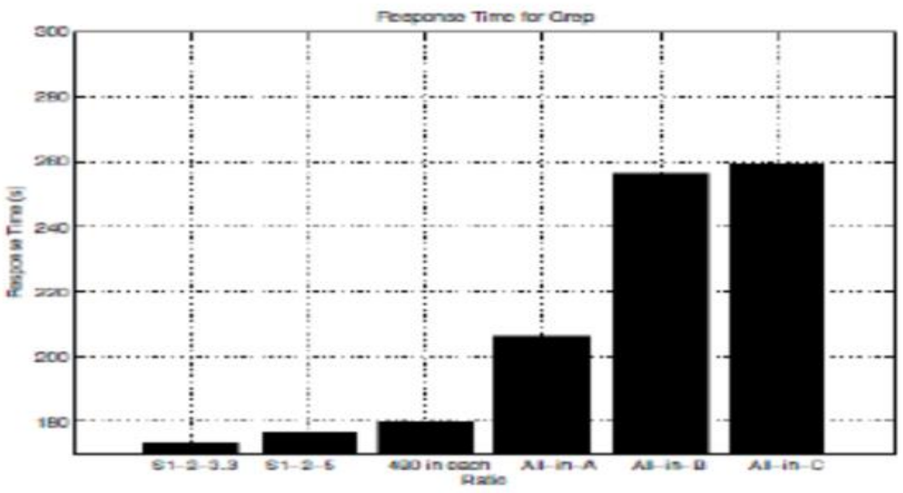


Fig 3.8.1.1 Response time for Grep search

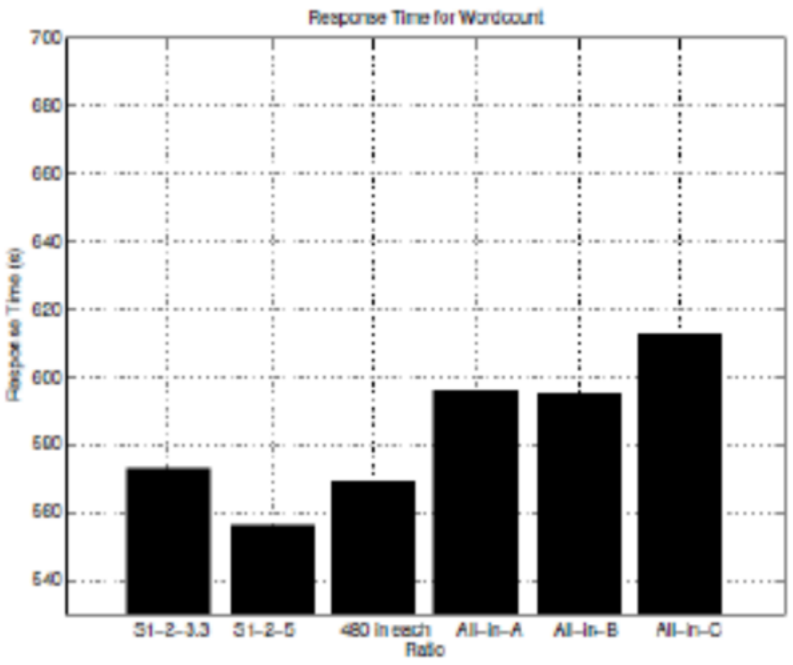


Fig 3.8.1.2 Response time for wordcount search

3.8.2 Data Locality Aware Task Scheduling for Heterogeneous Environments

Most of the MapReduce computing environments it is very common to assume that cluster is homogeneous in nature and all underline nodes are having identical processing speed and storage capacity. Due to increasing need from cloud related clusters which are geographically separated they intend to pay attention for improving the data locality of MapReduce in a heterogeneous computing environment to for giving optimal execution results. In case of homogeneous cluster all nodes have similar speed when they are assigned for process user request. In case of situation where input data is stored in node of homogeneous cluster, that node is reserved for task which requires that input for data processing.

This assumption is not valid in case of heterogeneous cluster or cloud computing wherein all nodes have varying processing capability and have no defined method to ensure data locality for faster job processing. There are numerous factors which can impact on processing speed of the task. Hence there is requirement for robust and dynamic work load schedular for effective computations in the heterogeneous cluster or cloud kind of computing environment This has made researches to give conscious effort to develop an effective scheduling algorithm which is appropriate and best feasible for processing in heterogeneous environment to improve the data locality management for MapReduce model.

X.Zhang et introduced a data locality aware scheduling method for heterogeneous Hadoop environment. They have found that there are two factors which can potentially influence the processing of map task as which is critical for processing. They said first factor is waiting time which is a minimum time required for a job or task before they are assigned to any available node for processing. Second factor which can influence performance is called off by them is transmission time which is suggested by then is minimum time needed to transfer input data from the node data is currently present to the transfer it to node which is requesting that data for processing.

Prime objective of this algorithm was to establish a balance among two critical factor which can impact the processing which are minimizing task waiting time they need before task is scheduled and minimize transmission time in case a replica of data is needed at the time of execution in order to attain the optimal execution time for overall task.

This approach first finds schedules the job whose input data is saved on the requesting node after recognising and receiving a request from a requesting node. If Schedular was unable to locate such a job, the function tries to locate and select a task whose input data is stored in the node that is the closest to the requesting node. The waiting time and the transmission time for that task are assessed when the node is chosen for processing. The approach keeps the task given to the node with the input data if it is discovered that the waiting time is less than the transmission time. The task will be scheduled to the requesting node if the transmission time exceeds the waiting time. In case of peak load hours there always scenario when a node will be executing more than one task together in a same node. This such cases where multiple tasks are assigned to node for processing, node can issue a processing request whenever it completes assigned task and by that time next task have to wait until the node completes the processing of tasks it has taken up by schedular which is currently in execution by the node. This means wait time of tasks whose input data are stored locally on the node have lowest remaining time between all other tasks currently being executed in the same node.

Tasks are scheduled on the requesting node according to their probability to finish the task on that node based on data locality factor of node.

To calculate remaining time of a task, following formula is used.

$$f(X,Y) = \begin{cases} \dfrac{n(1 - x_{n-1}) \prod_{i=0}^{n-1} y_i}{\sum_{i=0}^{n-1} x_i \prod_{j=0\ and\ j\neq i}^{n-1} y_j} & \forall x_i \in X, 0 < x_i < 1 \\ 0 & \exists x_i \in X, x_i = 0 \end{cases}$$

The cluster architecture is shown as below Fig.3. 8.2 System performance is evaluated based on three criteria which are critical factor to determine the success in terms of processing optimization of algorithm which are mentioned as below:

1) Map task number which are not scheduled to those nodes containing required the input segment.
2) The normal time required to finish task.
3) The response time of jobs. It is expected to have values less than the values of the default method.

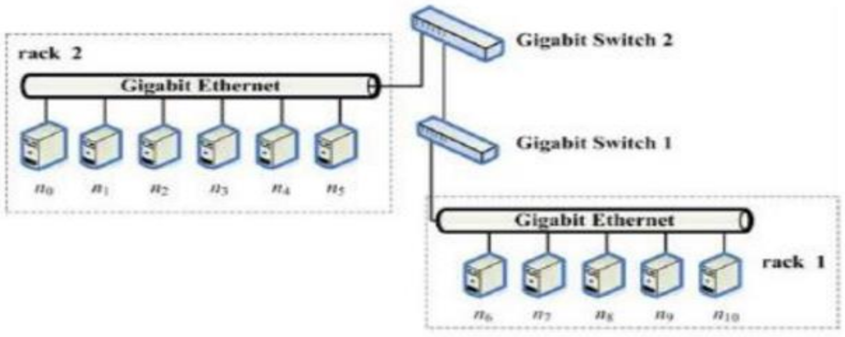


Fig. 3.8.2 Testing cluster topology

Table 4 shows the execution information of the jobs. Jobs results are collected and analysed based on two scenarios as illustrated below. In scenario maximum task are running on node 2 parallelly and require resources to finish task In scenario second scenario maximum tasks are running on node3. However, results are gathered only for scenario one as both results look similar.

Table 3.8.2.1 - Data Locality Aware Task Scheduling test parameters

Job name	HDFS Block size (MB)	Input data size (GB)	Map Task	No of Reduce task
I	64	2.5	41	2
II	128	5	41	2
III	256	10	41	2

Results are displayed as below default Hadoop configurations the execution mean time for number of jobs I and II as shown in below diagram is decreased by 13% and 4 % respectively in (shown in Fig.4). In the best scenario, the normal execution time is also decreased by 12%. Also, Hadoop with the proposed method achieved shorter response time.

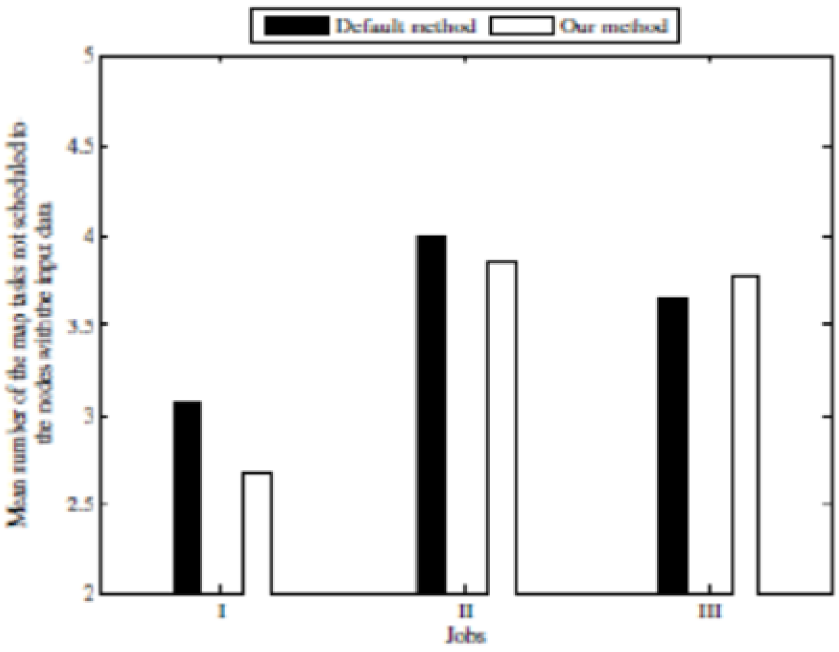


Fig 3.8.2.1 Result as per scenario 1

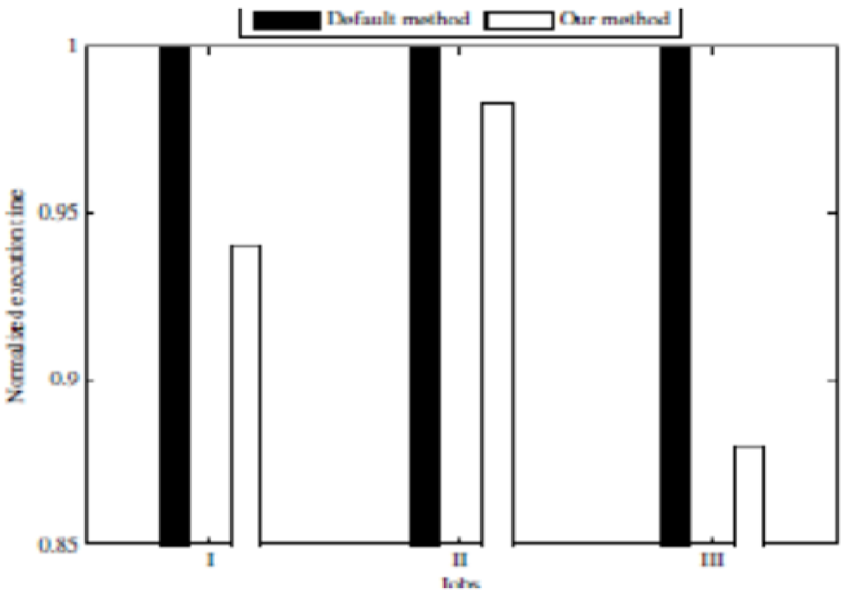


Fig 3.8.2.2 The normalized execution times of jobs in scenario 1

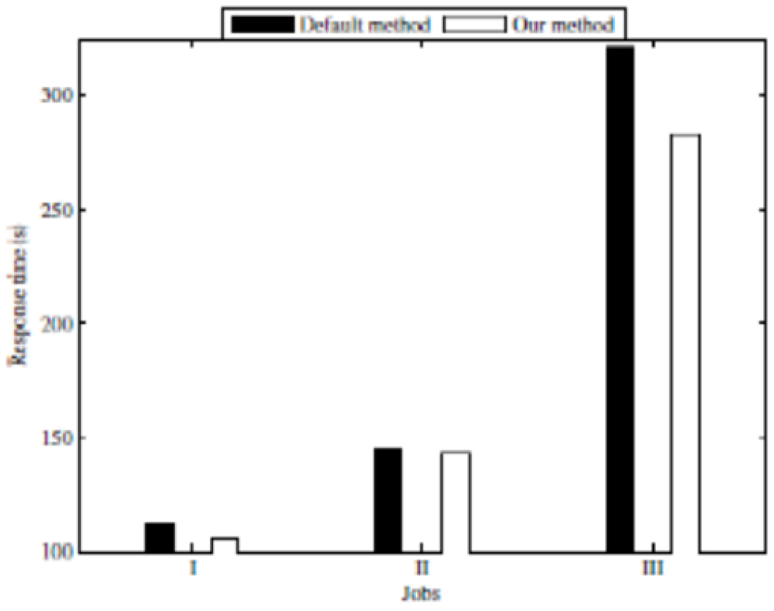


Fig. 3.8.2.3 The response times of jobs in scenario 1

3.9 Longest Approximate Time to End Scheduler

Longest Approximate Time to End Algorithm (LATE) utilize idle nodes efficiently by allocating tasks. Hadoop allocates a task using one of three approaches. Any failed task gets highest priority next priority is given to task is waiting status, then map tasks who have local data on this machine. Last priority is assigned to speculative tasks.

Hadoop computes a turnaround time for speculative processing by averaging each type of tasks (maps and reduces) progress percentage. A task having less progress percentage in comparison to its category average will be marked as a straggler. LATE identify task which will finish is longest approximate time and will run that speculatively. LATE computes the task’s completion time as per progress score provided by Hadoop. Which further computes the task progress rate of each task along with calculation of task’s competition time.

Different methods for calculating time left can be added into LATE. First method known as simple heuristic method which computes the progress rate of each task as Progress Score/T, T is the time since task is in execution, and then computes the time to finish as (1 − Progress Score)/Progress Rate. This has assumption that tasks execution rate is approximately same. However, in some cases, heuristic approach may fail to give desired result, but it is successful in typical Hadoop jobs. Speculative task should be launched in faster computing nodes in order to finish them quickly using a simple method to not launch speculative tasks especially on slow machines.

Suggested heuristic approach gives improved results in comparison to a speculative task to the first available machine on data cluster. Alternate solution is to create speculative task more than one replica, but this consumes many unnecessary resources. Finally, two heuristics are followed to take care of cost overhead as below:

Numbers of speculative tasks allowed executing at a time, which is known as Speculative Cap.Correct computation of slow running task to decide node is “enough slow” to be speculated to avid redundant launching of speculative tasks.

3.10 Advantage of LATE algorithms into MapReduce processing

LATE has improved MapReduce performance significantly. It has improved MapReduce performance in heterogeneous environment where each node has varying processing capability by re-launching slowest tasks only less in number. Considering impact on response time, LATE selects slowest tasks for next execution.

To avoid resource contention LATE caps on number of speculative. However, Hadoop's scheduler comes with a fixed number, which launch all slow tasks and have an equal probability of being launched. This causes exuberant number of speculated tasks.

LATE speculatively picks and performs jobs having the potential to maximise job response time rather than taking up any sluggish tasks by estimating expected remaining time rather than computing progress rate.

3.11 Data Locality issue on Federated Clusters

In high performance systems processing and file system are separated, while in case of MapReduce file system consisting of data nodes and Tasktrackers responsible for computation exist together on a node. HDFS replicates data to data nodes when data is placed into HDFS to check the availability of data before Tasktrackers before start executing task on that node.

Data locality in Hadoop MapReduce is currently attained by copying the data different three phases of cache. In 1st step a data fragment is get stored on a node, while in 2nd step data block out from data segment is saved on a node which is present into same rack and in last phase, A block of data is kept off-rack. Map jobs are attempted to be scheduled on nodes that hold the data blocks known as local nodes in order to take use of the benefit of data proximity. In event of no such node is found then task will be scheduled on nodes in same rack and can get data from any neighbouring node, this phenomenon is known as rack local. In case of both conditions are not succeeded then at last a task to execute a map is scheduled on any Task tracker that is accessible.

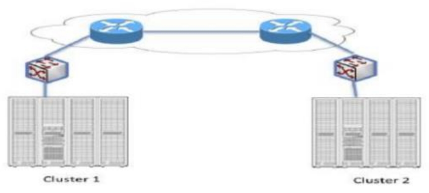


Fig 3.11 -Architecture of Federated clusters

Figure 3.11.6 as shown above depicts the basis architecture used to connect the federated clusters by internet or dedicated link. Data locality suffers in case of virtualization is considered on these federated clusters and provides virtual computing units connected via aggregate switch. this cannot be used to effectively harness local data.

To get optimum performance of Hadoop, it is very crucial to be focused while designing and configuring Hadoop, in a manner that it is aware of network topology across entire federated network.

3.12 Optimize processing using Network aware scheduling

The primary objective is to improve data processing by task schedular in Hadoop by moving computation nearer to node where data is present. If this criterion is not met then task scheduler will look out for a node which has requested for a task which has triggered data to be moved for processing nearest to compute node. When resources are identified and lined-up by distributed clusters, transferring data across the network results into Hadoop's performance degradation.

In case of distributed clusters, one of the Tasktrackers belongs to any underlying cluster may ask for a map task. Network aware scheduling is a scheduling methodology to make Hadoop scheduler aware bout network cluster and its network topologies and enable commination about hardware(rack) aware feature to Hadoop scheduler by providing one additional caching level. Hadoop Administrator will manage script which holds the information of Tasktracker is associated to which cluster.

In order to implement the network topology script for the global MapReduce architecture keeps track of information about virtual machines and the actual cluster that each node is a part of. The dispersed clusters' virtual resources were maintained and made available by Eucalyptus cloud computing software with Neuca support. The locations of the virtual machines are maintained as /clusterN/rackN/vmN. Whereas rackN indicates the rack number, vmN indicates the hostname of the virtual machine, and clusterN indicates the actual location of which cluster.

To maximize usage of data locality, delay scheduling [13] was used. When no compute node is found containing data then scheduling of the task is deferred for some specified amount of time until it finds node for processing. If case of any of the compute nodes having data becomes available for processing, task scheduler allocates a map task to requesting nodes Tasktracker. Duration for delay depends upon the average length of the map tasks for a job hence requires careful tuning of task.

3.12.1 Experimental setup for conducting network aware scheduling

In this set of experiments Hadoop was run on distributed cluster without ability to know network awareness. One of the nodes was designated as a master node which executes JobTracker and NameNode services. Remaining nodes from both clusters run acts as slave nodes and runs tasks services using TaskTracker and Data nodes.

Application of word count was used for result evaluation. Input data size for word count application is varied with an objective so that average execution time for map task is different for each set of input size Multiple jobs consisting of tasks with different execution times as shown in the table 3.12.1 are submitted.

Table 3.12.1 - Experimental setup for conducting network aware scheduling

Nodes	Quantity	Hardware and configuration of Hadoop
Master Node	1	2 GB RAM,10Gbps NIC Job Tracker and Namenode,2 CPU Core
Slave Node	**2**	2 CPU core, 2 GB RAM 10Gbps NIC Task tracker and Data Node Hadoop-0.20.203.0 2 Map and 2 Reduce Tasks

To evaluate the results obtained by an application of network awareness on Hadoop over the federated cluster. One dedicated JobTracker is set up to monitor and manage scheduling requests on all dedicated and managing the scheduling ask by users/application developers However task Tracker of other clusters wants to process map tasks sooner any free slot is available for execution. Cluster awareness technique is adopted by JobTracker which assigns suitable tasks on free or available nodes task Tracker by enhancing the data locality. Experiments with changing inter-cloud capability and delay to compute how performance of Hadoop is impacted using network awareness element by JobTracker.

This has been observed that network aware Hadoop has shown significant performance improvement for map tasks ranging from 500 to 1000. When map tasks number further grows, associated data blocks within with the map tasks will be spanned on further nodes of distributed clusters. While scheduling a Map task, Tracker has to ensure if the Tasktracker is hosted in the same cluster along with data nodes which contains data needed for processing. In event of foresaid condition is successfully meet in that case task will be assigned to the Tasktracker otherwise it will be delayed to execute for specified amount of time usually in seconds for the delay scheduling.

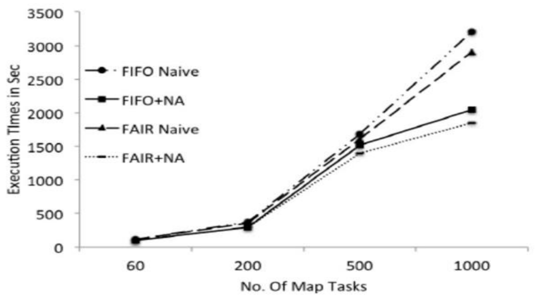


Fig 3.12.1 Execution times for Different schedulers

Comparisons of processing times of Hadoop native scheduling along with network-aware Hadoop for different delay value in seconds. Above figure represent the processing time for changing number of MapReduce tasks, FIFO+NA (FIFO scheduler coupled with network awareness) implementation reduces the processing time by approximately 12% on average and optimum reduction achieved approximately 15% while number of map tasks is higher in comparison to Hadoop Naive scheduler known as FIFO Naive. FAIR+NA (FAIR scheduler with network awareness) shown above produces the nearly same results, FAIR scheduler in general takes less execution time compared to FIFO.

Bandwidth is an important element in case of a federated heterogeneous clusters due to data movement in case of data locality between nodes. Large quantity of data is moved between various nodes of cluster for an execution of map task in scenario wherein tasks didn't get allocated to neither node local or rack local. In case of 100Mbps of bandwidth there is possibility of less availability of bandwidth between nodes.

Figure 3.12.2 shows below depicts improvement in processing in the local map tasks in terms of percentage. Hadoop scheduling imbibes the concept of moving the logic closer to data.

First priority is given to those tasks having data present on the same node as the one requesting for the tasks known as "node local". If scheduler don't find any such tasks then tasks having data present in the same rack as the requesting node are scheduled this is known as "Rack local".

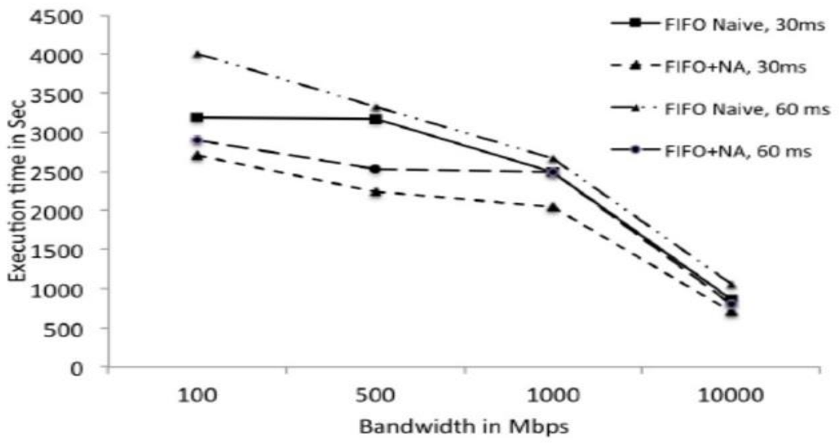


Fig 3.12.2 Comparison of various scheduling techniques

Network awareness is taken into account in case of a non-local map tasks execution wherein it is required to fetch data residing on some other nodes of cluster. Network awareness in Hadoop reduces the data transfer between a cluster to another cluster in event of execution of map task by implementing cluster level locality phenomena. Delay scheduling was utilized to optimize the data locality, because delay scheduling makes sure that in case of a task which is being scheduled on a Tasktracker and that node doesn’t have required data present into that node, processing will be deferred by a pre-defined default amount of time. If one of the task trackers having data becomes available for processing then, task is scheduled on the second Tasktracker. Though this has been a common reason that in case of typical data intensive applications, data split transfer consumes more time in comparison to task processing on the data split.

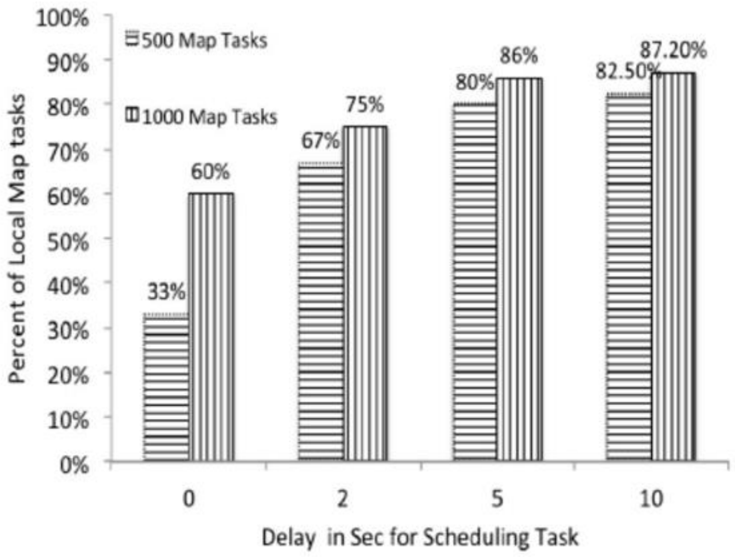


Fig 3.12.3 Result of uses of delay parameter

Delay parameter is very crucial parameter which plays a vital role into establishing an effective scheduling and should be carefully configured. In the Figure 5 talks about different delay amount (in seconds), delay of 0 sec means scheduling without having presence of the delay scheduler. Delay amount of 2 Sec means a non-local map tasks will have to wait for 2 sec to see if Tasktrackers present into some other node having data has placed data processing request.

Since the time taken to finish a task is quite small, hence it is observed that task Tracker get available for processing quite frequently and they place a request for processing of a task further. Hence consideration should also be made into time taken by a task to finish its execution before defining a perfect and suitable delay parameter. below results reflects that network awareness in combination of delay scheduling can be successful into minimizing the data movement across other clusters nodes.

Chapter4

PROPOSED SOLUTION, VAIDATION AND DISCUSSION

Machine learning is an analytical method which is primarily used to automate the analytical solutions and models. This is a subdivision of Artificial intelligence oriented on idea to learn from data or any pattern or output and provide pattern identification and decision making without any human intervention.

In past machine learning was developed and used for pattern recognition by computers can learn without explicitly programmed. Researchers was initially working towards evolution of artificial intelligence to see if computers can learn themselves by data they are provided as training data set.

Basically, in machine learning data inform of input and the output is feedback to algorithm so that it become perfect and can predict based on data provided and independent predict outcome as per input they receive.

Though machine learning started in 1970, but their initial focus was just on image recognition or pattern recognition, but recent food of data as big volume of data, it could be customer sensitive or web crawling or world recognition. It is now time for Machine learning algorithms to looks beyond the traditional usages and harness ability to build complex automated algorithms or solutions for big data.

There are few very popular examples of machine leaning now a days, first one is self-driven google car, recommendation provided to customers by online eCommerce sits e.g., Flipkart or Amazon or fraud detection in banking.

Resurgence of interest in machine learning in current time is due to fact which has made data mining and Bayesian analysis more in demand than ever. Data features now days are like rapidly growing volumes and various types of data getting generated from various channels and over and above to this now a days data computational processing is also very affordable and more sophisticated to store any kind of data, digest and process it faster than ever.

All of these things demand meaningful, quick and reliable business insight from data. Machine learning make this all happen by proving various methods and algorithms which can quickly solve the complex business problem and provides a better business model for analysing vast and more challenging business-oriented data and provides rapid and more accurate results – even on a very large volume of data. Machine learning algorithms also supports developing models basis on business related problem in such a manner that an organization has a better opportunity of identifying opportunities which can help them to focus on growth areas by making them change ready to any unknown risk and eliminate inefficiencies from system also proves insights by highlighting areas wherein they can benefit a lot by paying more attention by suggested business insights along with identification of future growth areas.

Machine learning (ML) can be defined as category of algorithm which makes software applications outcome more reliable and accurate in prediction. The fundamental prerequisites of ML are to building algorithms which take datasets as input then conduct statistical analysis to predict an output and update results as input fresh data for improved training of datasets.

Machine learning procedures are often identical to data mining and predictive modelling procedures. Commonality among them is that they all searching through data to look for highlight trends and reorient programmes as necessary. Many individuals are familiar with machine learning thanks to online shopping and seeing relevant adverts for their purchases. This occurs as a result of recommendation engines' use of machine learning to quickly customise the delivery of web ads.

4.1 Clustering of big sized data set

Clustering is basically an arranging same kind of objects from a large group into small homogeneous groups based on some feature similarity. Maximizing intra-cluster similarities is the primary goal of clustering. The classification depends on the selection of the right attributes and needs to be related to different user concern areas. A proper classification identifies a connection between distinct semi- or unstructured data elements. Data sets are grouped according to user-interesting qualities using a variety of clustering algorithms. Calculating distance, density, and data distribution are important parameters for clustering. An suitable clustering technique is chosen to extract data from the data sets based on the data extraction requirements and the type of data sets. Based on similarities, groups are split up, and after that, labels that help identify them are applied. In a process of selecting of an appropriate clustering algorithm it should be considered whether the algorithm scales enough on the big data set we wanted to do divide based on some commonality among their objects. Big sized datasets can have more than millions of rows, however not all of clustering algorithms does scale efficiently. Many of the clustering algorithms looks at the similarity among all pairs of objects and does the computing. Thus, they push the processing time as the square of the number of objects n, which can be represented as O(n2) in terms of complexity representation.

O(n2) algorithms are not very effective when data size is very large as thus adds processing overheads to process millions of rows. Hence main focus is to explore on the kmeans algorithm, with a complexity of O(n), means this algorithm scales linearly with n. There are various clustering algorithms present to divide large sized data set into smaller data sets having similar objects. Each algorithms gives best result to some specific type of data distribution.

4.1.1 Centroid based clustering

In this type of clustering algorithm, identification of each cluster distance among objects is computed. Thus the object having minimal difference between object and other cluster object is assigned to that cluster. In case of centroid based clustering number of cluster are known in advance which poses challenge or s shown in below picture how similar elements are grouped together based on distance from other groups objects.

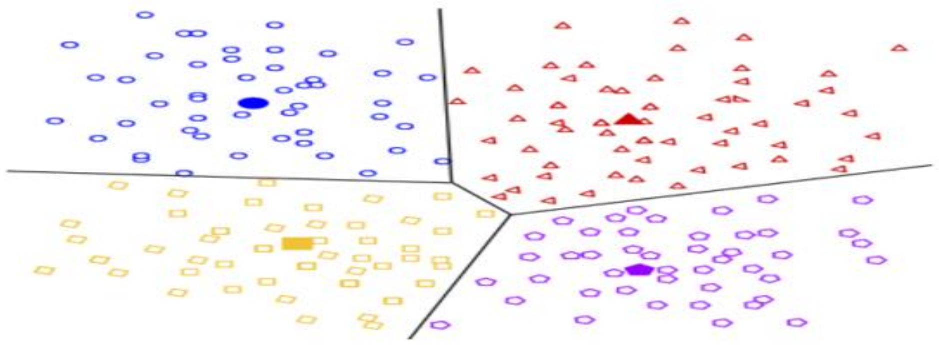


Fig 4.1.1 -Example of centroid based clustering

4.1.2 Distribution based clustering

In this cluster algorithms it is considered that data is composed of some kind of internal distribution between objects e.g., Gaussian distributions and then clusters are divided into based on that distribution wherein there is minimal difference between object and other cluster's object. As shown in below picture distribution-based algorithm divides into three clusters.

In case any objects distance from the distribution's centre increases, its probability to belong that distribution decreases.

As shown below in various show that as distance increases, objects probability to be in that cluster also decreases. This algorithm is helpful when data distribution is known in advance. In a scenario when data distribution of a cluster is not known then it's recommended to use a different algorithm.

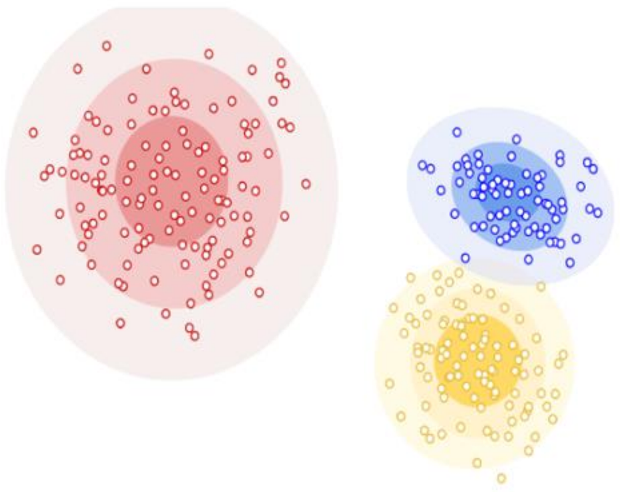


Fig 4.1.2 Example of Distribution based clustering

4.1.3 Connectivity based clustering

Connectivity-based clustering, is also famous as hierarchical clustering, which works on concept of objects is more connected to its closer objects rather than far away objects. Which looks similar to centroid-based clustering but here instead of clusters is grouped basis on set of centres, in connectivity-based testing clusters are formed by computing the maximum distance required to connect partitions of the cluster. Changing this constant will provide various clusters. Connectivity-based clustering does not provide one cluster of data set rather gives a kind of hierarchy which are merged with each other at certain distances. This clustering creates entire clusters on the basis of computation of distances from objects near or far. This has been observed that there may be challenges in case of presence of noise and outliers in datasets because they induce additional clusters, which increases the complexity in clustering or clusters are merge improperly.

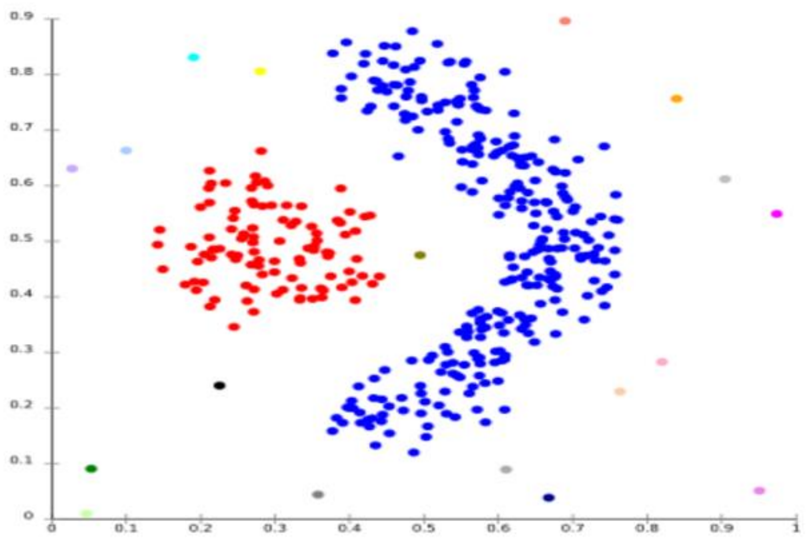


Fig 4.1.3 Example of Connectivity based clustering

4.1.4 Density based clustering

Density-based clustering algorithms picks portions of high density into a cluster. This makes arbitrary-shaped distributions as far as we are able to connect dense areas from large group. However, these algorithms face challenges when data of varying densities and high dimensions is present into parent cluster. However, this issue can be taken care while designing them in such manner that they do not assign outliers to clusters. If outliers are not handled properly while designing, they can result into performance degradation during this clustering method.

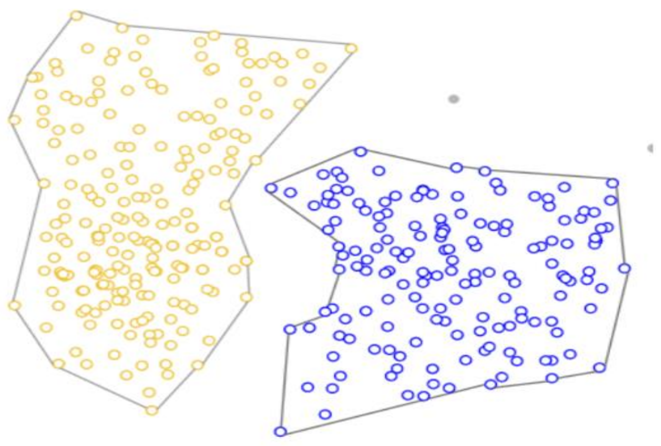


Fig 4.1.4 -Example of density-based clustering

4.2 Finding nearest subgroups using K-Means clustering

This is one of the efficient clustering techniques used to classify big data and partition them into sub groups based on certain similarity between elements of sub-groups which is a tedious task using traditional methods and provides meaningful insight on types of objects and cluster them basis on their characteristics. Various studies have proved that huge data set of can be processed efficiently by this kind of analysis, giving optimized results for variety of data belonging to different scenarios.

The way K-Means algorithm works as below.

- **a)** Specify the K which denotes number of subgroups to be made out of dataset.
- **b)** Initialize centroids by randomly selecting K data points for the centroids without replacement. For best results its always recommended to shuffle dataset before initialization of centroids.
- **c)** Calculate the sum of the squared distance between data points and all centroids.
- **d)** Allocate each object to the nearest cluster (centroid).
- **e)** Determine new set of centroids for the clusters by using the average of the all-data points that belong to each cluster.
- **f)** Continue the iteration until assignment of objects to clusters is not changing at all we call this as convergence.

  Approach K-Means follows to partition datasets is often called as Expectation and maximization. In first one allocation of each object to nearest cluster is focused and in second or maximization calculation of clusters centroid is done iteratively as follows.

  K-means objective function is represented as below:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \| x^i - \mu_k \|^2 \qquad (1)$$

This function states that if wik=1 for an object xi if it belongs to cluster k; else Wik=0. Where the centroid of xi’s cluster is µk .

This basically reduces to a kind of minimization problem of two stage in first stage we tend to minimize the J (final output) for each Wik and make µk fixed, while in second stage we minimize J for each µk keeping Wik fixed.

Thus, first J is computed for each Wik and update each cluster allocation function A-Step. Then we apply another function O-step which differentiates J for each µk and recompute the centroids of clusters post the re-allocation A-step is performed.

A-step is denoted as below.

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

A-step can be described as to allocate each element xi's to the nearest cluster based on minimum sum of squared distance from cluster's centroid.

While O-step commutation is defined as below:

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik}x^i}{\sum_{i=1}^{m} w_{ik}}$$

O-step represents the re-calculation of the centroid of each cluster to for each if the new element allocation.

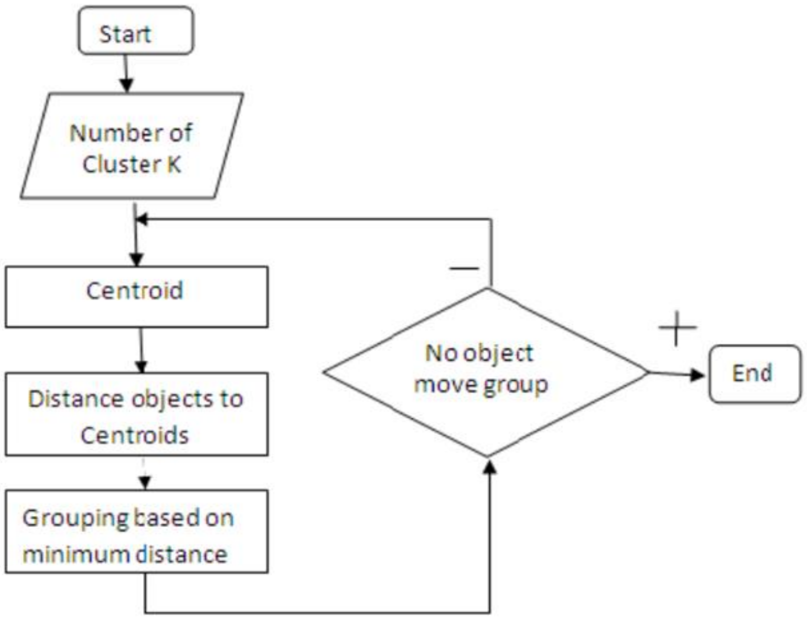
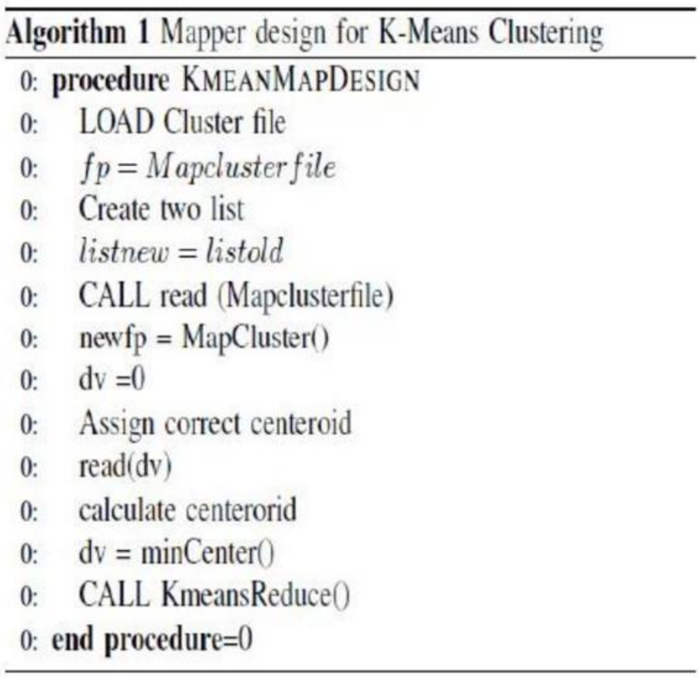

Start
Number of Cluster K
Centroid
Distance objects to Centroids
Grouping based on minimum distance
No object move group
−
+
End

Fig 4.2.1 -Processing of K Means algorithm

While implementing K-Means there are few points to be taken care while implementation as mentioned below:

o It is always recommended to determine the similarity among objects of datasets as K-means used distance as metric and have a mean of zero and a standard deviation of one.

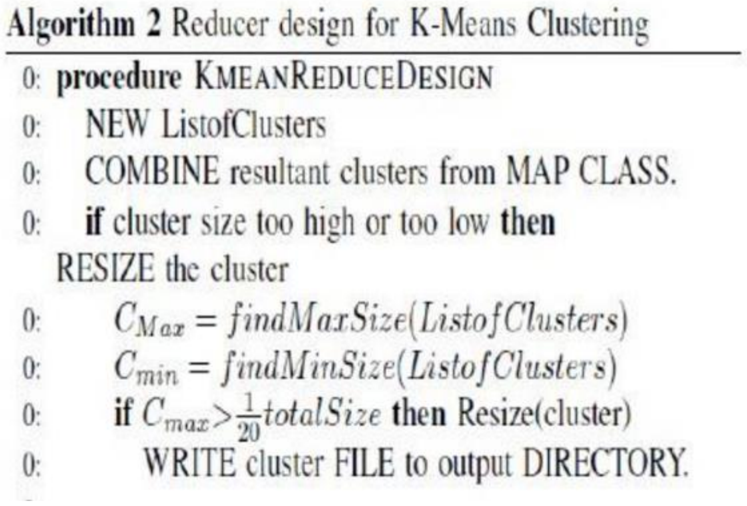
o as K-Means works iteratively random initialization of centroids at the beginning, various initializations may yield different clusters and thus K-Means algorithm may stuck in a local optimum situation. Therefore, it’s is always better to run the algorithm for different initializations of centroids and select the outcome of the one which has given least sum of squared distance.



Algorithm 1 Mapper design for K-Means Clustering

```
0: procedure KMEANMAPDESIGN
0:    LOAD Cluster file
0:    $fp = Mapclusterfile$
0:    Create two list
0:    $listnew = listold$
0:    CALL read (Mapclusterfile)
0:    newfp = MapCluster()
0:    dv =0
0:    Assign correct centeroid
0:    read(dv)
0:    calculate centerorid
0:    dv = minCenter()
0:    CALL KmeansReduce()
0: end procedure=0
```

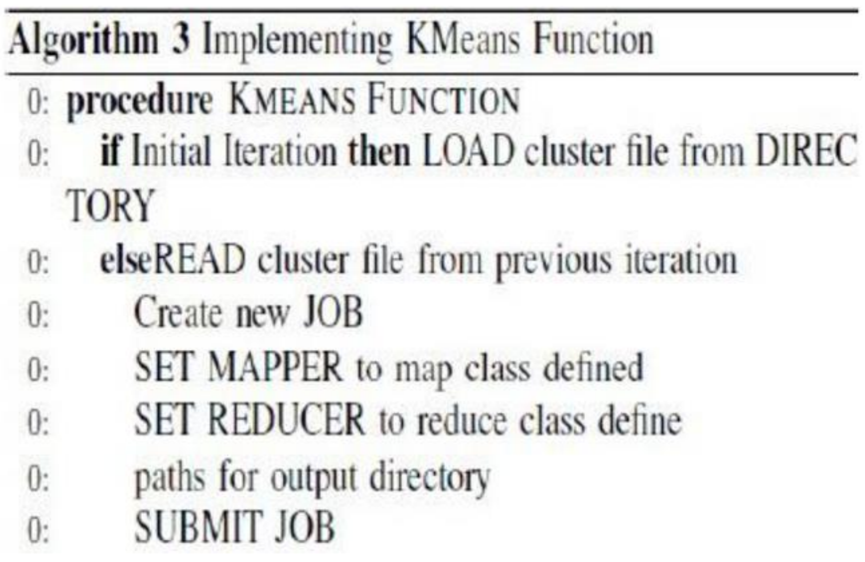
Fig 4.2.2 -Mapper design for K-Means Clustering



Algorithm 2 Reducer design for K-Means Clustering

```
0: procedure KMEANREDUCEDESIGN
0:    NEW ListofClusters
0:    COMBINE resultant clusters from MAP CLASS.
0:    if cluster size too high or too low then
   RESIZE the cluster
0:       $C_{Max} = findMaxSize(ListofClusters)$
0:       $C_{min} = findMinSize(ListofClusters)$
0:       if $C_{max} > \frac{1}{20} totalSize$ then Resize(cluster)
0:          WRITE cluster FILE to output DIRECTORY.
```

Fig 4.2.3 -Reducer design for K-Means Clustering



Algorithm 3 Implementing KMeans Function

```
0: procedure KMEANS FUNCTION
0:     if Initial Iteration then LOAD cluster file from DIREC
   TORY
0:     elseREAD cluster file from previous iteration
0:         Create new JOB
0:         SET MAPPER to map class defined
0:         SET REDUCER to reduce class define
0:         paths for output directory
0:         SUBMIT JOB
```

Fig 4.2.4 -Implementation of K-Means Clustering

4.3 Evaluate K-Means performance using MapReduce

K-Means clustering is very popular and used for variety of applications to know various data insights as market segmentation clustering, clustering of document clustering, image clustering and image compression.

Here I have used image compression datasets for evaluation of best K value and how this has minimized sum of squared distance from centroids.

Here K-Means is implemented to compress an image. The dimensions of image are (396, 396, 3). For each for each pixel location there are three 8-bit int intensity for red, green, and blue colour values. Objective is to minimize the number of egers used to specify the colours to thirty and show compressed image using those thirty colours. In order to determine which colours to use K-Means algorithm on the image is applied wherein each image pixel is considered as a data point. That means reshaping of image will allow us to represent the image using the thirty centroids for each pixel and would compress the size of the image by a factor of six. The original size of image size was 396 * 396 *24 = 3,763,584 bits and post compression new size of image comes out to be 30 * 24 + 396 * 396 * 4 = 627,984 bits. This big difference is size is due to the fact that we are using centroids as a lookup for colours of pixels which will reduce the size of each pixel location to 4-bit from original 8-bit.

We have used sklearn implementation of kmeans which shows results as below which shows the comparison between the original image and the compressed image looks quite similar to original image because most if the features of the image. The original image was unchanged and retained.

As there are some standard methods to evaluate algorithm performance in supervised learning, K-means is not able to eb evaluated using those methods. However, there are two metrics below which are used for K-Means performance evaluation.

- Elbow method
- Silhouette analysis

4.3.1 Elbow method

Elbow method is used to determine what is a good value of k by computing the sum of squared distance (SSE) between data objects and their initial allocated clusters centroids. Value of k is picked up when we observe that sum of squared distance starts to flatten out and forming an elbow. As per image compressor data set we can observe and compute SSE value for different values of k to see for what value of K curve is flattening out and forming an elbow

As per the graph below we can see that from k=2 it started flattening thus makes k=2 a good choice for our clustering solution. Sometimes it’s still hard to figure out a good number of clusters to use because the curve is monotonically decreasing and may not show any elbow or has an obvious point where the curve starts flattening out.

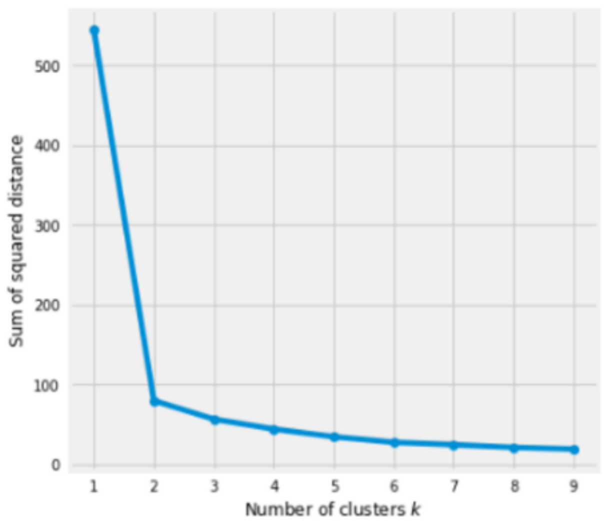


Fig 4.3.1 Result of Elbow analysis for K=2

4.3.2 Silhouette Analysis

Silhouette analysis is an alternative method used to determine the coefficient of separation among clusters. For each cluster:

Calculate ai which is the average distance of all data objects in the same cluster.

Calculate bi which is the average distance of all data objects in the nearest cluster.

Compute the coefficient:

$$\frac{b^i - a^i}{max(a^i, b^i)}$$

Value of coefficient ranges between -1 and 1

If value of coefficient is 0 -This implies the sample is very close to the nearest clusters.

It values of coefficient is 1 –This implies the sample is quite far from the nearest clusters.

It values of coefficient is -1 –This means the sample is allocated to an incorrect cluster.

Therefore, it's always desirable to find coefficients big and close to 1 so that it can be allocated to correct clusters.

For different vales of K silhouette analysis has produced two result points as below.

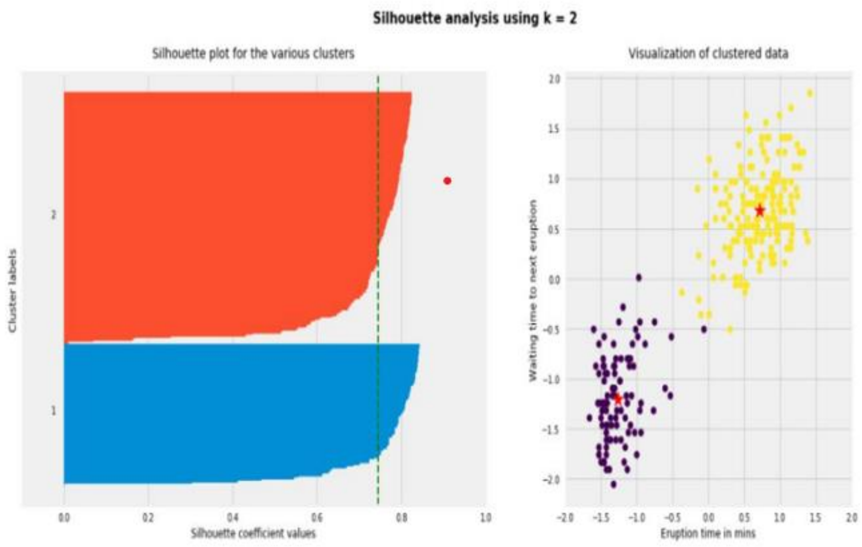


Fig 4.3.2.1 -Result of Silhouette Analysis for K=2

As the plots below in figures shows that K=2 has the best average silhouette score approx. 0.75 and all clusters are more than average tells that it is actually a best choice for this data set. Also, the thickness of the silhouette plot indicates size of cluster. The plot shows that first cluster is almost two times than cluster two . When we have increased K to 3 the average silhouette score decreased drastically to the level of 0.48 and this has gown down more for k=4 which is 0.39 respectively. Other than this thickness of silhouette plot is also fluctuating. We can say that if silhouette score is more than 0.5 that value of K will be yield best result for K-means clustering.

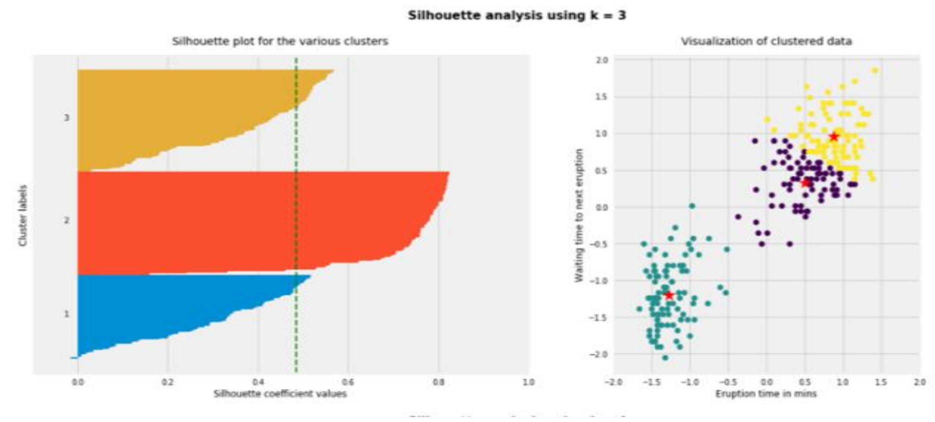


Fig 4.3.2.2 -Result of Silhouette Analysis for K=3

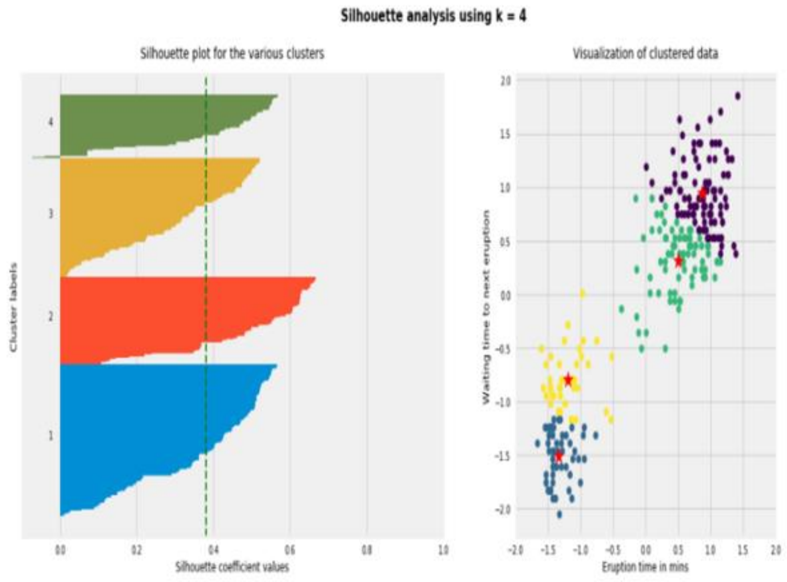


Fig 4.3.2.3 -Result of Silhouette Analysis for K=4

4.4 Drawbacks of K-Means clustering

This has been observed and provided by various researchers that K-Means algorithm works well in capturing structure wherein shape of cluster is like spherical. As K-Means always tries to make spherical shape around the centroid. This means if clusters have some other than spherical shape K-Means performs poorly in clustering the data.

This has been shown in below few examples where it does not able to discriminate well First, K-Means algorithm doesn't allocate an object which is far from each other to be in same cluster while they are actually in same cluster. Below example shows object on two lines which explains how k-means tries to group half of the data points of each horizontal lines together.

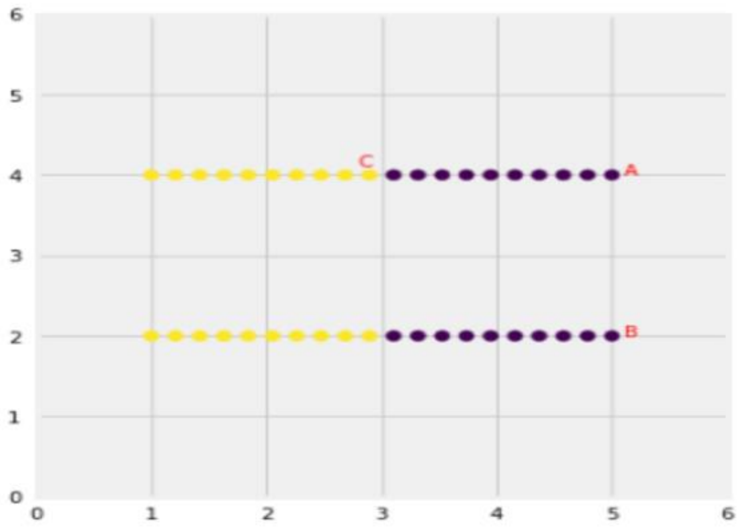
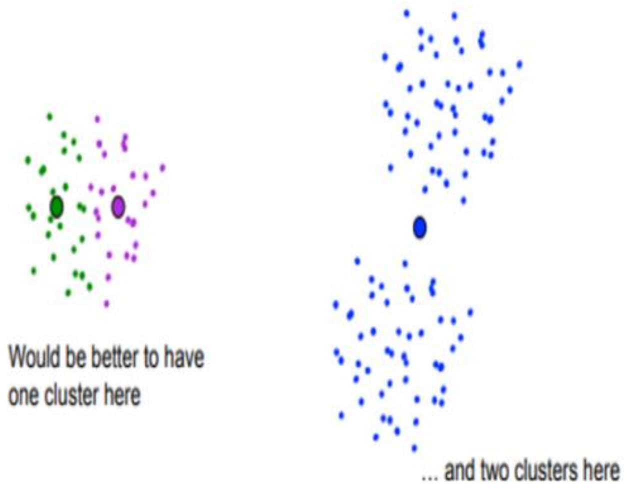


Fig 4.4.1 K-Means clustering drawback

K-Means assumes that point ‘B’ is close to point ‘A’ in comparison to point ‘C’ because of non-spherical shape. Therefore, it allocates points ‘A’ and ‘B’ in same cluster while allocates ‘C’ in a different cluster.



Would be better to have one cluster here
... and two clusters here

Fig 4.4.2 Local optimum problem with K-Means

4.5 MapReduce processing using Parallel K-means clustering

The K-Means technique, which was first devised and suggested by MacQueen [32], is the most straightforward and well-known clustering procedure to divide a given data set into a pre-specified number of clusters [33–35]. Iterations over data sets are necessary until a solution is reached. The K-Means algorithm frequently reaches a local best. An incremental-Means (IKM) technique [36] was devised to address the local optimum problem and converge to the global optimum, however this is more difficult. As traditional K-Means algorithm needs multiple iterations on data sets hence this should be completely present into main memory for faster data accessing, but this become difficult when size of data is very large [34].

As a result, several researchers used parallelism in the MapReduce-based K-Means technique to effectively cluster large amounts of data. A framework that employs the parallel programming technique of MapReduce with K-Means was created by Chu et al. [19]. Using MapReduce, Zhao et al. [20] built the PK-Means algorithm.

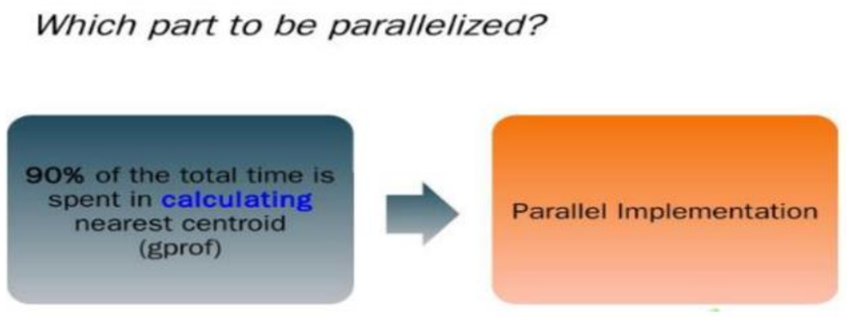

Which part to be parallelized?
90% of the total time is spent in calculating nearest centroid (gprof)
Parallel Implementation

Fig 4.5.1 -How parallelism an improve K-Means processing

There are three key features of K-means which PK-means tries to overcome; these are mentioned as below.

1) Initialization of centroids
2) Number of clusters K
3) Number of iterations -I

Thus PK-means focus upon above mentioned parameters and find a suitable values for each of parameters as listed below.

1) Identify a good set of initial centroids
2) Explore parallelism during initialization.
3) Ensure output is best in quality by minimizing intra cluster distance and maximizing inter cluster distance.

Below picture shows how Parallel K-means can reduce most time-consuming step and optimize its performance into clustering large sized data set.

However most intensive calculation in parallel K-means happens for data set scanning on memory as same has been shown in below figure which demonstrate the flow of processing in case of parallel execution and how it does spt an opportunity to implement the parallel processing.

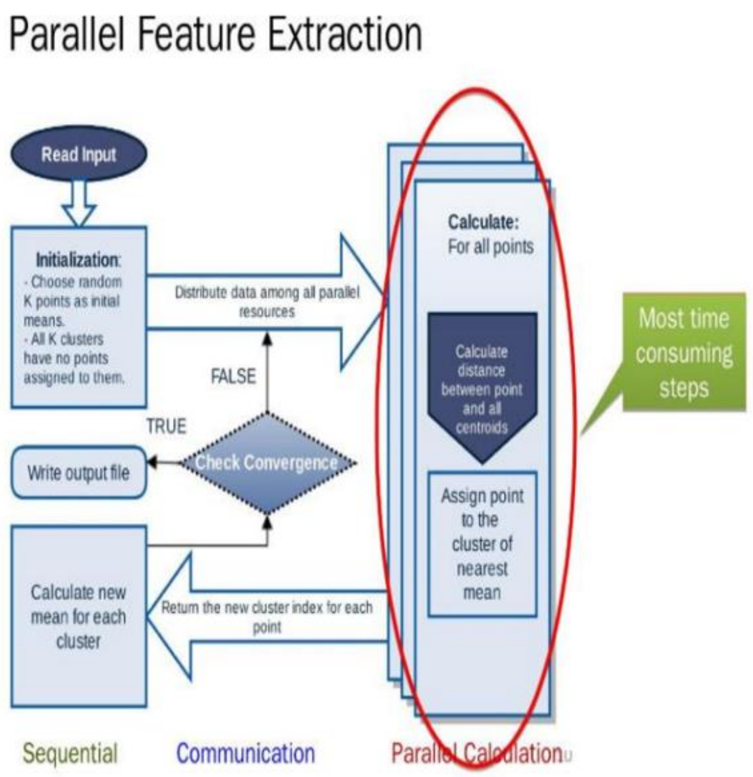

Parallel Feature Extraction
Read Input
Initialization:
- Choose random K points as initial means.
- All K clusters have no points assigned to them.
Distribute data among all parallel resources
FALSE
TRUE
Write output file
Check Convergence
Calculate new mean for each cluster
Return the new cluster index for each point
Calculate: For all points
Calculate distance between point and all centroids
Assign point to the cluster of nearest mean
Most time consuming steps
Sequential
Communication
Parallel Calculation

Fig 4.5.2 -Pictorcial represenation of parallel execution of K-Means

4.6 Implementation of MapReduce using PK-Means

There has been improvement made into MapReduce to implement the parallel K-Means processing as described for each of the component in detail in below. The input dataset is stored on Hadoop distributed file system on different nodes as a sequence file of pairs, each of them represents a record in the dataset. Offset of byte holds information about data file starting point which is represented by key and strings of the content of record. The dataset is divided and distributed to all mappers' processes. Hence this gives an opportunity for parallel computations of distance. Parallel K-Means construct a global variant centre for each map task, which is a kind of array.

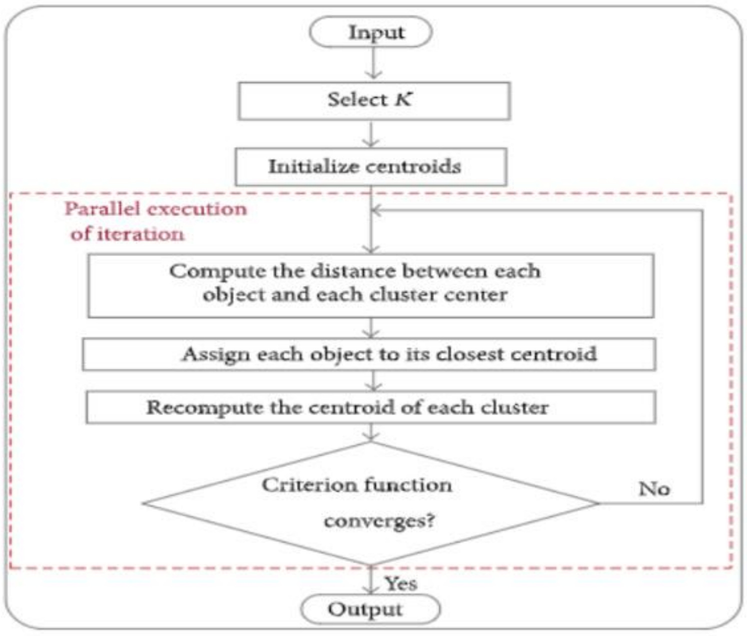
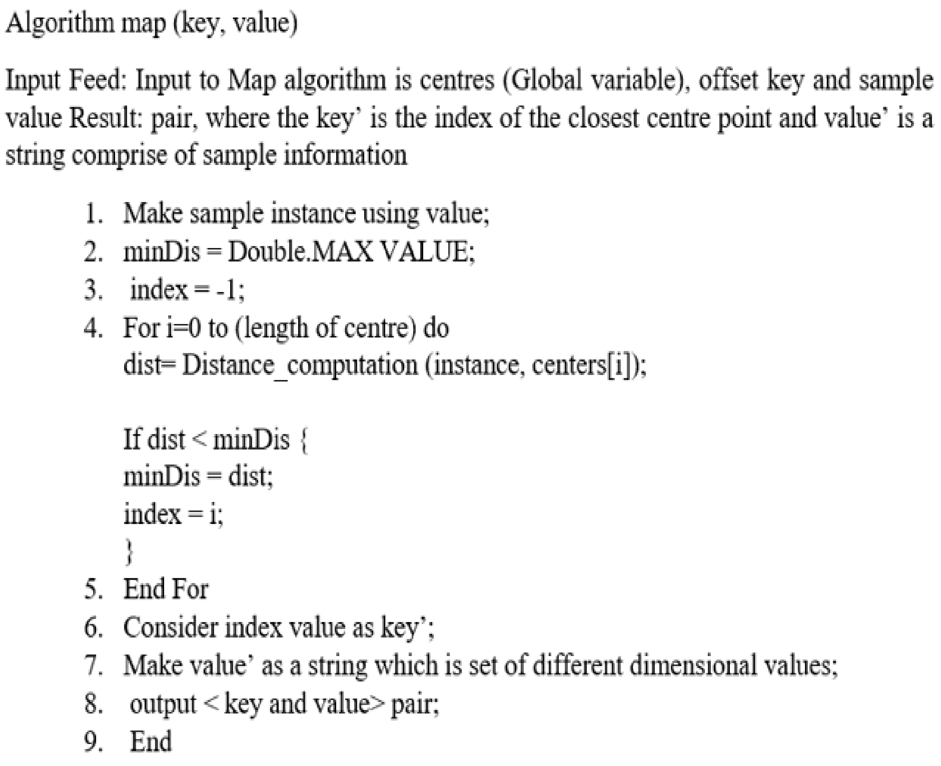

Input
Select K
Initialize centroids
Parallel execution of iteration
Compute the distance between each object and each cluster center
Assign each object to its closest centroid
Recompute the centroid of each cluster
Criterion function converges?
No
Yes
Output

Fig 4.6 -Flow diagram of PK- Means algorithms

Detailed implementation of Map reduces, combiners pseudo code used for Parallel K-means algorithms are shown as below.

4.6.1 Map function algorithm



Algorithm map (key, value)

Input Feed: Input to Map algorithm is centres (Global variable), offset key and sample value Result: pair, where the key' is the index of the closest centre point and value' is a string comprise of sample information

```
1. Make sample instance using value;
2. minDis = Double.MAX VALUE;
3.  index = -1;
4. For i=0 to (length of centre) do
   dist= Distance_computation (instance, centers[i]);

   If dist < minDis {
   minDis = dist;
   index = i;
   }
5. End For
6. Consider index value as key';
7. Make value' as a string which is set of different dimensional values;
8.  output < key and value> pair;
9.  End
```

In Above Map Algorithms Step 2 and Step 3 basically does the initialization of intermediate variable minDis and index. In Step function Distance compute (instance, centers[i]) is used to find the closest centre by obtaining the distance between instance and the centre point centers[i]. Once processing of each map task is finished task combiner is executed, which combines the same map task intermediate data.

During the combination of map tasks performed by combine function, which partially sums the assigned values to the same cluster and further it does calculate the mean value of each object in that cluster by computing distance.

4.7 Experimental set up for Parallel K-Means using MapReduce

So, we attempted to propose and test an efficient parallel k-means clustering algorithm using MapReduce. We used three ways to test the performance capability up, processing up and volume up to measure the performances of our proposed algorithm.

To conducted experiment, we have set up a cluster of computers, each node has processor of 2.8 GHz cores and 4GB RAM configuration and for software version 0.17.0 of Apache Hadoop was used along with Java 1.5.0 14 for all experiments used for MapReduce system.

4.8.1 Comparison between sequential K means and parallel K-means

As shown below that parallel K-means gives better result than the sequential K-means algorithms to handle the large sized data.

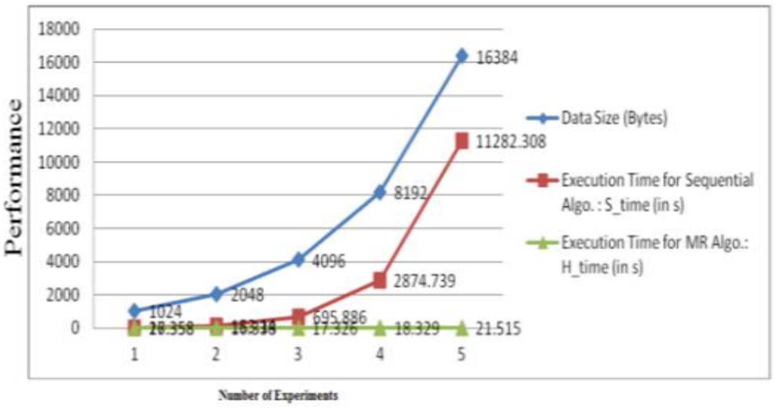


Fig 4.7.1.1 Performance of sequential and parallel K-Means

4.7.2 Performance of PK-Means by Increasing data set size and keeping Number of nodes constant

To conduct this experiment, we have kept number of nodes constant to two and increased size of data set from 256 MB to 2048 MB to measure performance of parallel K-Means.

Results shown below for word count program has shown improved performance.

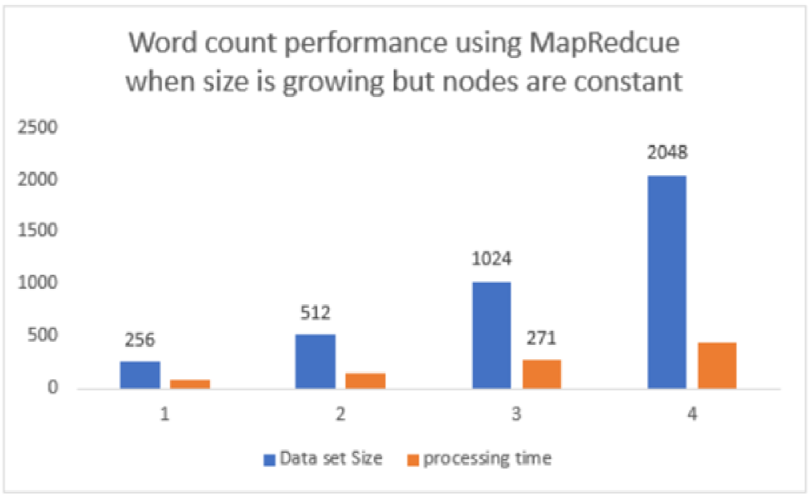


Fig 4.7.2.1 wordcount performance -size of data is growing but number of nodes are constant

Similar results shown below for grep count program has shown improved performance.

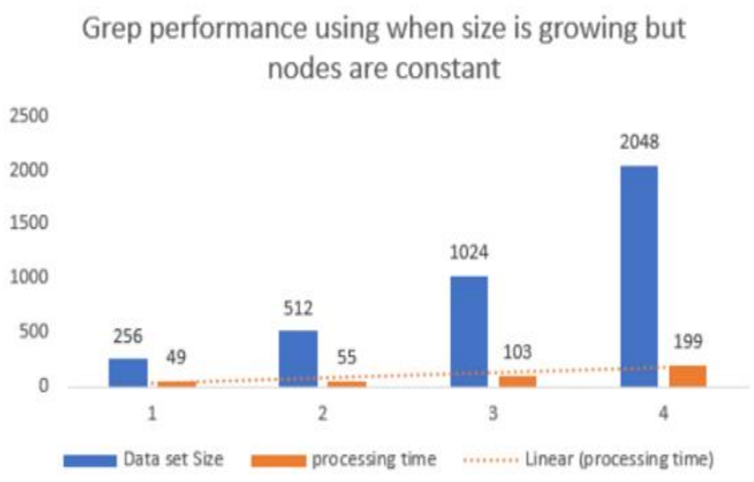


Fig 4.7.2.2 - Grep performance when size of data is growing but number of nodes are constant

Result of Parallel -K-means algorithm has shown improved results as below when size of input data was increased keeping the number of nodes constant.

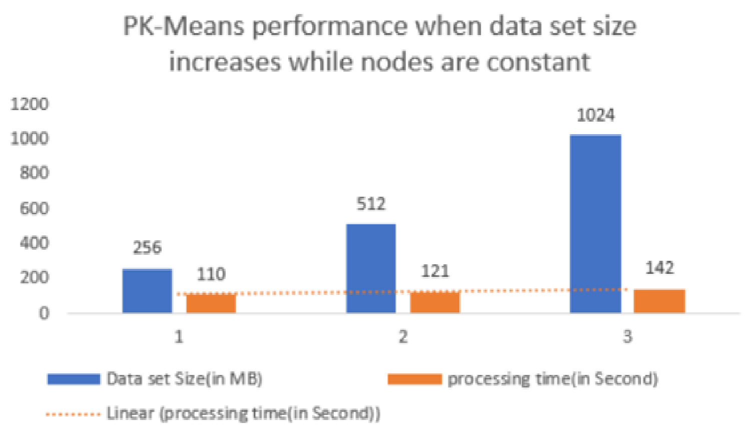


Fig 4.7.2.3 PK-Means performance-data set size grows but nodes constant

4.7.3 Experiment to keep data set size constant and increase number of nodes

Performance of parallel K-Means is evaluated for scenario by increasing the number of nodes to provide high processing capability and keep the size of data sets constant. Thus, any good parallel processing algorithm will yield linear speedup. A system with n times of number of computers gives a speedup of n. However due to increase in communication cost makes linear increase in processing capability is a challenging task as shown below.

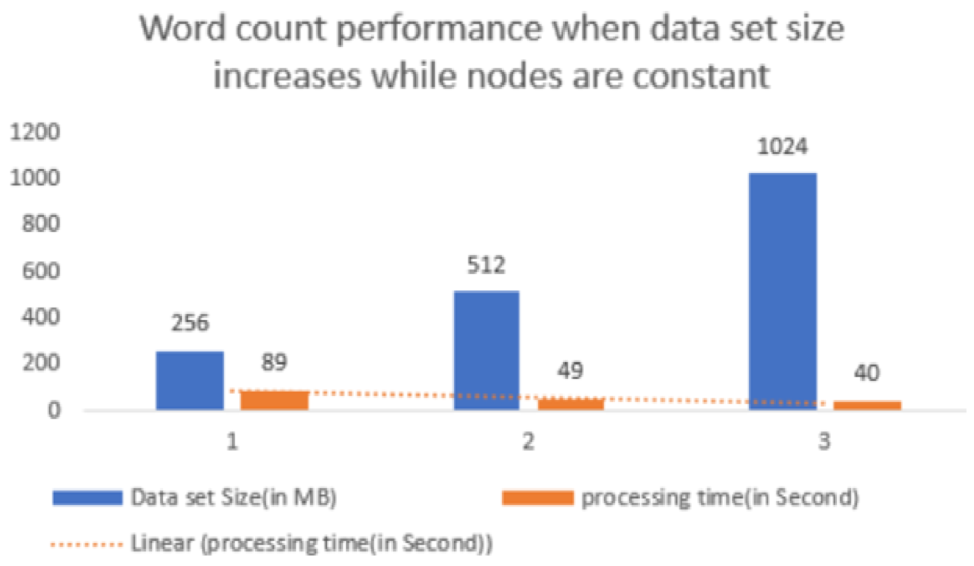


Fig 4.7.3.1 wordcount performance when size of data is constant but number of nodes are growing

Similar result has been conducted for Grep search when data set size grows and nodes are constant.

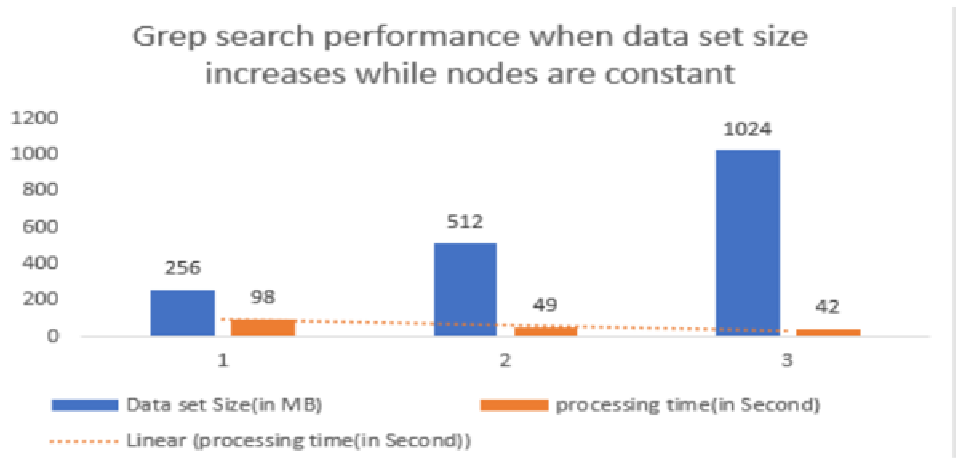


Fig 4.7.3.2- Grep performance when size of data is growing but number of nodes are constant

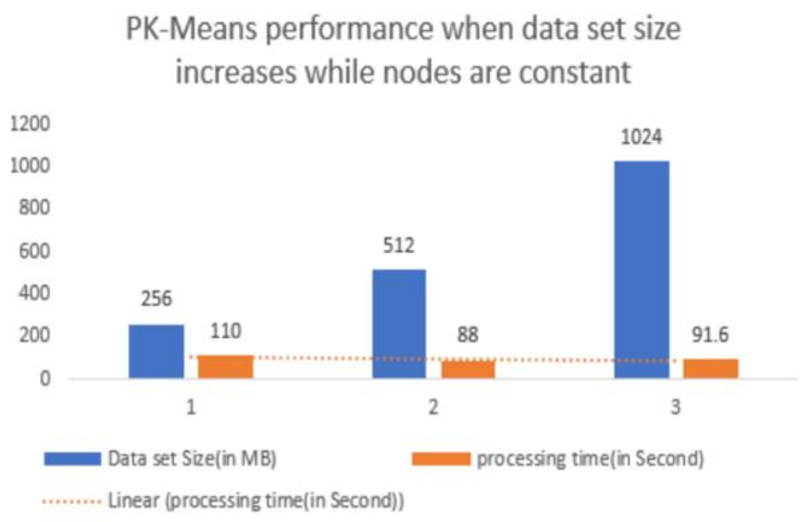


Fig 4.7.3.3 – PK-Means performance when size of data is growing but number of nodes are constant

We have conducted this experiment of varying the processing capability on different sized datasets and nodes. The numbers of computers were changed from 2 to 4. The size of the dataset was ranging from 256 MB to 2GB and results obtained are plotted against the nodes and dataset size to know the relationship between both these parameters of processing capability and varying datasets size.

As per above figure result indicates that PKMeans yields a better performance result. Capability-up in parallel k-means provided better result on large sized datasets, this provides us a faster computing algorithm for large sized data sets. Therefore, PKMeans algorithm makes itself a best candidate of processing of vast datasets efficiently and gives better results in heterogeneous environments. This has been observed that In case of PK-Means experiment that processing time first decreases and then increases due to fact that size of data set is constant while communication among other nodes is increasing.

4.7.4 Experiment when both dataset size and number of nodes are increasing

In this experiment we have tested wordcount and PK-means algorithm processing capability when both dataset size and number of nodes have increased simultaneously. This can be defined as the capability of an n-times larger system to execute n-times bigger job in the similar run-time as the actual system. To prove efficacy and show how effective PKMeans algorithm manages bigger sized datasets efficiently in a case wherein additional nodes are available for processing, scale-up experiments were performed by increasing the dataset size in same preposition of number of nodes in the cluster. Various datasets are used to conduct processing starting from 1Gb and then increase by 1 GB up to 4 GB on one node and then adding one node till 4 nodes respectively.

Similar experiments have shown that PK-Means algorithm outshines all other in terms of scale up as well due to element of parallel processing. In Size-up analysis performance was evaluated by making machines number constant and increase just size of datasets by some factor let's say n. Size up evaluates how much time a job takes to complete on a system by consuming and processing the data set of increasing in size from previous iterations.

To compute the size-up performance, numbers of nodes were changed from one to four respectively. Fig-4.8.4.1 shows the size-up performance on different kind of experimental set-up improved performance shown by PK-Means in terms of size-up analysis.

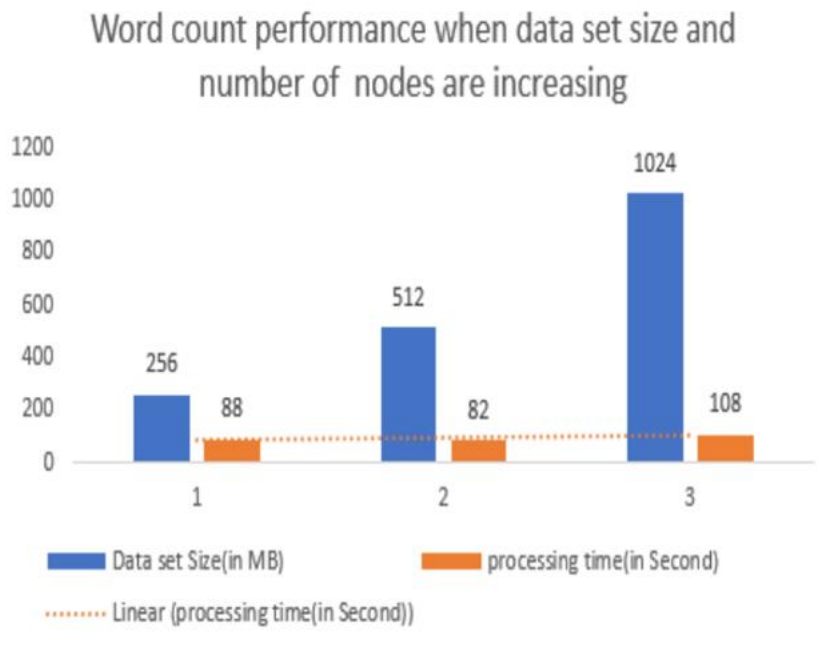


Fig 4.7.4.1 Wordcount performance when size of data and number of nodes are growing

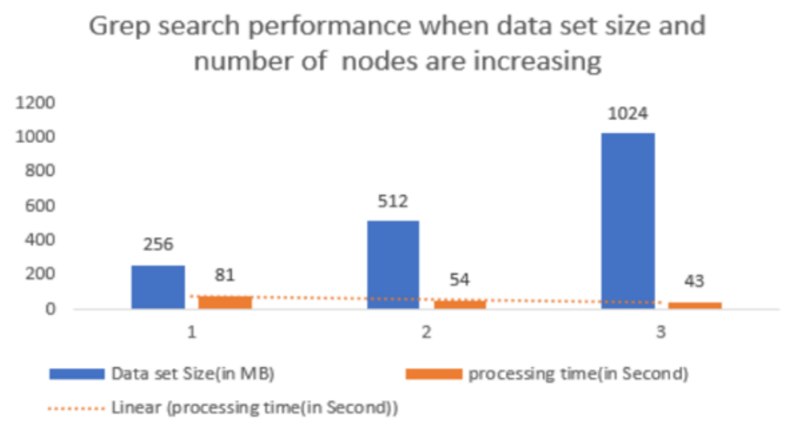


Fig 4.7.4.2 Grep search performance when size of data and number of nodes are growing

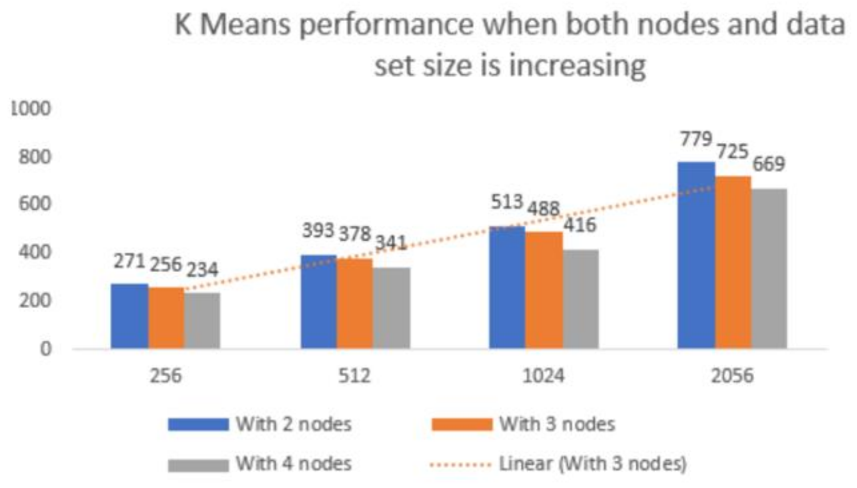


Fig 4.7.4.3 Parallel K-Means performance when size of data and number of nodes are growing

In this experiment, the performance of MapReduce application has been evaluated for execution time and number of nodes in Hadoop cluster. This also evaluates how MapReduce programming model's execution time decreases when number if nodes increase, and this has been proved that Parallel K-Means performs well and efficiently and the results totally depend on the size of Hadoop cluster. Though performance of PK-Means Algorithm can be improved by introduction of new more level of parallel approach to have data commutation in parallel manner across various pre-processing steps to increase convergence in parallel K-means.

pre-processing steps to increase convergence in parallel K-means.

4.8 Two phase parallel K-means algorithms

K-means (KM) is one of the most widely used algorithms to cluster data because of its dataset is not too large. Thus K-means algorithm have pitfall that it converges to local optimums because due to initialization of cluster centers randomly.

There are three approaches to overcome local optimum of K-Means problem as below.

1. Define a good initialization of clusters centroids.
2. Random Initialization of cluster centers by the help of a modified learning mechanism.
3. Gradually increase the number of clusters from 1 to the K using an appropriate mechanism to initialize new clusters.

Several researchers have tried using first approach to find a good initialization for clusters in K-means algorithms. An appropriate heuristic can yield a good initialized position. Centers of clusters can be initialized using a k dimensional -tree by performing a density estimation of the data.

These methods didn't achieve the desired results to identify a good initialization of centroids albeit they have added complexity into system. In second approach to overcome the local optimum has been made by various researchers by using a modified learning methodology after initialization of clusters centroids. Fritzke [8] proposed a new jumping operation to obtain convergence Similar to Fritzke's work Patane, G., & Russo, M.[] have used the utility index to overcome the problem of local optimum [9].

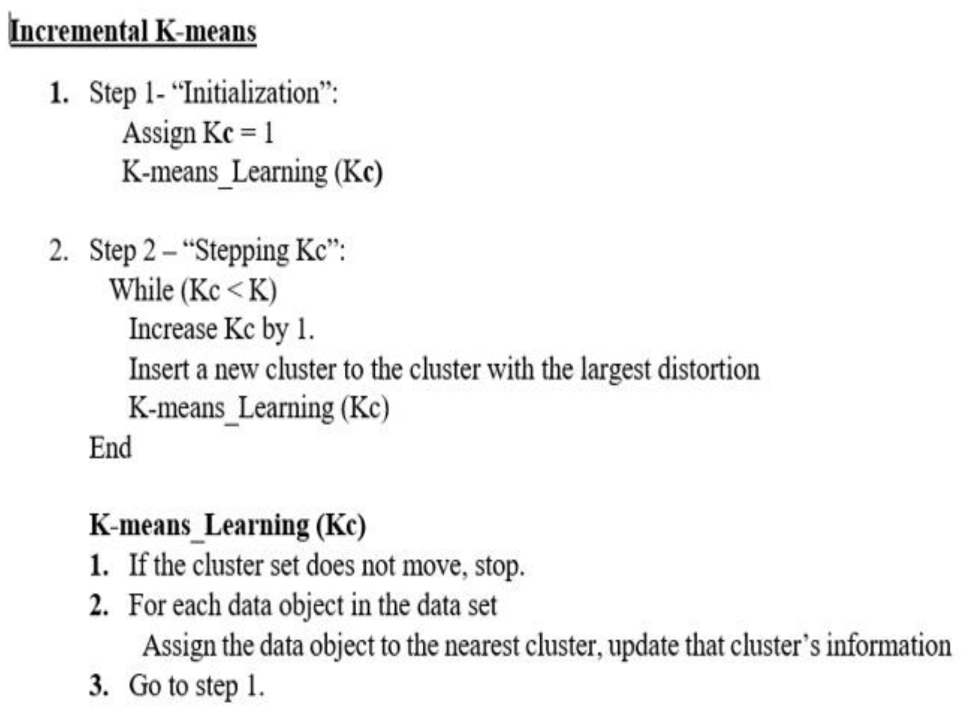
A new updating method was introduced by Chinrungrueng and Sequin [10] for underlying object distribution. All these improvements didn't work well into Random Initialization of cluster centres to reach the global optimum.

Third approach is based on incremental method that rather than initializing K clusters, initialize one cluster initially then numbers of clusters are added by one more ranging from one to Every time a new cluster is added into existing set of clusters.

4.8.1 Incremental K-means algorithms

The Incremental K-means algorithm (IKM) increases cluster number from 1 to K. Primary objective of this algorithm is explained in IKM, K-means algorithm is iterated K times and after successful execution of each iteration current clusters number Kc is incremented by 1 which is second step of algorithm. Thus, to get a suitable position for the recently added cluster, new cluster is allocated to set of clusters who has maximum level of distortion error. Therefore, experimental results have shown that IKM converges to global optimum on data sets but this increases the complexity of IKM algorithm which is (O (K^2n)) while complexity of K-Means is (O(Kn)).

The algorithms used for Incremental K-means implementation is mentioned as below



Incremental K-means

```
1. Step 1- "Initialization":
       Assign Kc = 1
       K-means_Learning (Kc)

2. Step 2 – "Stepping Kc":
     While (Kc < K)
        Increase Kc by 1.
        Insert a new cluster to the cluster with the largest distortion
        K-means_Learning (Kc)
   End

   K-means_Learning (Kc)
   1. If the cluster set does not move, stop.
   2. For each data object in the data set
          Assign the data object to the nearest cluster, update that cluster's information
   3. Go to step 1.
```

4.9 Two phase parallel K-Mean algorithm -2PParK-means

As a part of this research an advanced level incremental K-Means algorithm called as Two-Phase parallel K-means or 2PParK-means has been proposed and evaluated on large data set over heterogeneous cluster. The proposed two-Phase parallel K-means uses existing feature of K-means to process large data sets by scaling few of its characteristic due to its processing simplicity. While processing of large data set K-means algorithm does many iterations on data set. Hence in case of K-Means clustering faster execution using MapReduce its essential to load entire data set into computer memory for faster accessibility of data, which is doable up to certain size of data sets. However, for more than 100 GB loading them into memory is not possible. 2PParK-means is proposed to improve this challenge.

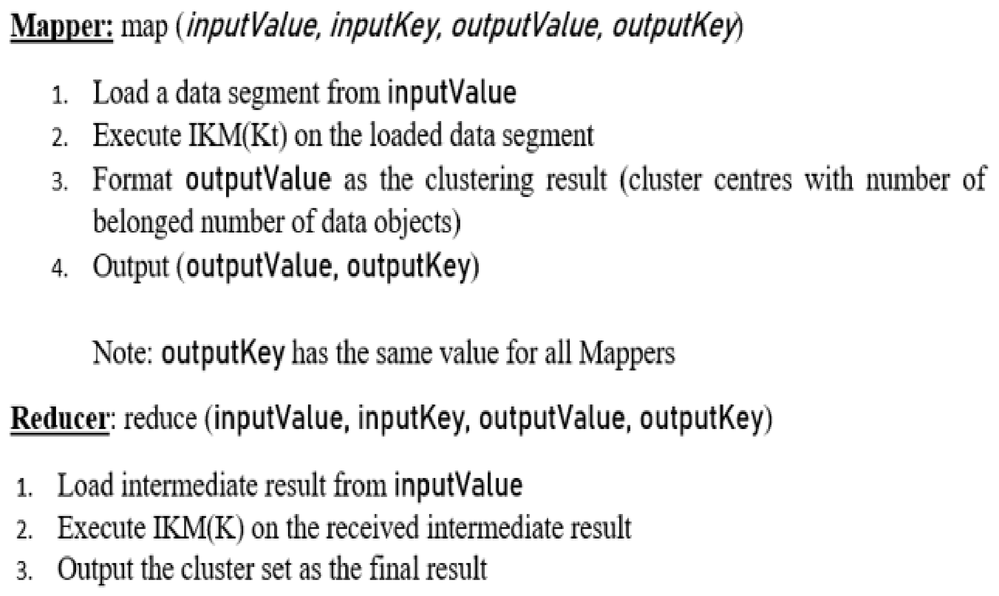
2PParK-means does execution of large sized data set following two level processing approach. On first level of processing 2PParK-means algorithm loads and processes all data portions one after other wherein each portion of data is divided in such a manner that the each of the data portion contains data objects. Level one processing gives an interim cluster set which is used for further processing in next level. The K-means algorithm is applied on first level due to its simplicity and convenience. In second level of processing 2PK-means uses all the intermediate clusters to generate final clusters by applying incremented K-Means clustering.

Input: *K* and *Kt* being the expected number of clusters and the temporary number of clusters for Mappers

L being the length of the data segment

Output: the cluster set

Algorithm:



```
Mapper: map (inputValue, inputKey, outputValue, outputKey)

    1. Load a data segment from inputValue
    2. Execute IKM(Kt) on the loaded data segment
    3. Format outputValue as the clustering result (cluster centres with number of
       belonged number of data objects)
    4. Output (outputValue, outputKey)

       Note: outputKey has the same value for all Mappers

Reducer: reduce (inputValue, inputKey, outputValue, outputKey)

  1. Load intermediate result from inputValue
  2. Execute IKM(K) on the received intermediate result
  3. Output the cluster set as the final result
```

Two-Phase parallel K-means or 2PParK-means, is further executed on Hadoop MapReduce framework to achieve greater performance on heterogeneous nodes. In event of Mapper is available, intermediate clustering result of all Mappers.

Performance evaluation of 2PParK-means

The proposed 2PPar-KMeans algorithm's, performance is compared for increasing number of nodes in cluster to check how it performed in comparison to previous speed-up ratio. The experiments are conducted on a heterogeneous cluster which has different computing capacity and configuration. Two nodes have a 2.8GHz CPU and 2GB and other two nodes have 1.7 GHZ and 4 GB RAM with Hadoop version 2.7.7 and Java 10 version are used as the MapReduce system for all experiments.

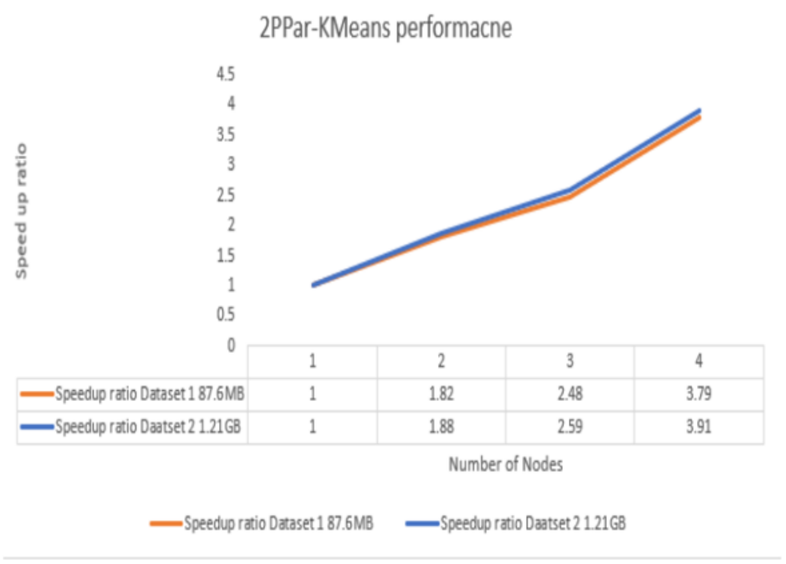
We have used forest cover type data set provided by UCI Machine Learning Repository. [26]. Forest cover type was into 30 x 30-meter cell and this was initially determined from US Forest Service. Data was then mapped US Geological Survey to find the required independent variables. The data which we get was in raw form (not scaled) and have binary columns of data for qualitative independent variables such as wilderness areas and soil type.

There are 581,012 objects with 52 attributes in datasets. However, for our experiments, we have used just the first 4 attributes are used. We have extended data set by introduction of noise to objects to create two test data sets.

The Modified dataset with noise has approx. 2,124,012 data objects, 4 attributes, and a size of 84.58MB. The other dataset with noise has enlarged data set 2 has 28,040,300 data objects, 4 attributes, and a size of 1.21GB.

The algorithm is evaluated with increasing number of nodes (1, 2 and 4). The number of data objects in one data segment is varied from 0.1% to 1% the size of data sets. Number of clusters K and Kt are selected as 10. The speed-up ratios for 2PPar-KMeans algorithms are computed by increasing cluster processing capability by adding nodes.

As shown in below picture a comparison of processing of 2PPar-KMeans on different nodes with varying dataset size. It is clearly shown that for second dataset size has increased but speed up ratio remain almost remains constant and proves that it provides faster result for large size data set processing as well.



	1	2	3	4
Speedup ratio Dataset 1 87.6MB	1	1.82	2.48	3.79
Speedup ratio Daatset 2 1.21GB	1	1.88	2.59	3.91

Fig 4.9- Performance results of 2PPar-KMeans

Chapter 5

CONCLUSION AND RECOMMEDNATION

In this thesis a new incremental version of K-Means has been proposed and tested on various scenarios and compared its performance with other similar algorithms on heterogeneous network to ensure it does perform best among the existing algorithms used for clustering.

5.1 Conclusion of research work

In the course of carrying out research work object was to find out what causes the performance degradation into processing of a big data into a heterogeneous environment and look at how different researchers have looked at ways to overcome those challenges and either proposed a new algorithm or did few amendments into existing algorithms by finetuning few of the parameters which were causing lacklustre performance.

Initial reasons observed for poor performance into heterogeneous cluster was due to varying nodes processing capability and different time required to process data into various nodes because of data movement between various nodes when Hadoop splits data and allocates respective task for processing to underline slave nodes .Hence there may be situation when a poorly configured nodes may take long time to process and thus master nodes keep waiting for that nodes to finish processing .This slowness into processing of poorly configured node causes performance degradation. To overcome this speculative execution was introduced which aborts the task running more than average time and restarts that task to some other idle or less occupied node. This issue was addressed by LATE or longest approximate time to end algorithm. This algorithm performs well in homogeneous environment but this doesn’t perform well in case of heterogeneous environment.

As most of time is spent in data movement in case map reduce processing for large data sets. This K-means was chosen to get the faster processing but K-means also have below trade-offs.

This has been observed and provided by various researchers that the k-Means algorithm works well in capturing structures where the shape of the cluster is spherical. As K-means always tries to make the shape of a sphere around the centroid. This means if clusters have some other than spherical shape k-Mean performs poorly in clustering the data.

In order to overcome the trade-off of local optimum some amendments were done and parallel K-mean was proposed and tested on various scenarios to measure the performance of algorithm on datasets of different sizes.

By employing parallel K-means algorithms computational time is reduced. There are several versions of parallel K-means algorithms proposed but here one of them is studied which is incremental parallel K-means algorithms. This version uses the processing capability of multiple computing nodes to provide faster results for the clustering process of the K-means.

However various parallel versions of K-means follow different methods to induce parallel processing. Some of them uses MapReduce framework. In the case of MapReduce processing, a Mapper process is invoked multiple times and this makes several iterations, this iterative nature causes the huge communication cost among nodes while processing of large dataset. The proposed Two-Phase K-Means (2PK-means) is based on concept to boost K-means to work faster on big sized data sets. As K-means algorithm needs many data revisits and to increase the processing on big datasets data should be entirely loaded into computer main. This is nearly impossible when size of data is huge like in more than 100 GB or in TB. To take care of such high-speed processing demand 2PKmeans was proposed to mitigate this trade-off.

The proposed 2PK-means does processing in two steps to ensure faster and convenient execution of big data. In first step of 2PK-means sequentially loads and processes all data partitions and generates an interim cluster set which is feed forwarded to next step as an input. In first step K-means algorithm is used due to its simplicity while in step two 2PK-means using all intermediate cluster datasets creation of final result is done using incremental K-Means. This methodology sures that only single data scan is required for processing of the large data set. This makes this algorithm suitable for even a node with limited processing capability.

5.2 Recommendations

As experienced while writing this thesis there is still scope for improvement in the computation of distances for data present on various nodes. Hence focus in future studies can be given to the optimization of distance metric and initialization strategy of various versions of K-Means implementations to enhance the processing of clustering algorithm for processing of big data.

There are future areas to improve the efficiency and scalability by implementing of optimal K-Means algorithm into a MapReduce framework of the Hadoop environment. Future studies can be focused on the optimization and parallelism of K-Means which can reduce the consumption of memory and minimize the computational cost while processing bulkier datasets involved in heavy calculations, thereby significantly improving the accuracy, efficiency, and scalability.

REFRENCES

[1] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2012.

[2] F. Ohlhorst, Big Data Analytics: Turning Big Data into Big Money, Hoboken, N.J, USA: Wiley, 2013.

[3] Apache Hadoop, http://hadoop.apache.org.

[4] F. Li, B. C. Ooi, M. T. Özsu and S. Wu, "Distributed data management using MapReduce," ACM Computing Surveys, 46(3), pp. 1-42, 2014.

[5] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein and C. Welton, "MAD skills: New analysis practices for Big Data," VLDB Endowment, 2(2), pp. 1481-1492, 2009.

[6] K. Grolinger, W. A. Higashino, A. Tiwari and M. A. Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores," Journal of Cloud Computing: Advances, Systems and Application, 2, 2013.

[7] X. Su and G. Swart, "Oracle in-database hadoop: When MapReduce meets RDBMS," Proc. Of the 2012 ACM SIGMOD International Conference on Management of Data, 2012.

[8] Apache Cassandra, http://www.datastax.com/docs.

[9] J. Lin, "Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail!" Big Data, 1(1), pp. 28-37, 2013.

[10] Apache Mahout, https://mahout.apache.org/.

[11] X. Zhang, Yuhong Feng, S. Feng, Jianping Fan, Zhong MingComputer Science2011 International Conference on Cloud and Service Computing2011,"An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments".

[12] J. Fan, F. Han and H. Liu, "Challenges of Big Data analysis," National Science Review, in press, 2014.

[13] D. Logothetis and K. Yocum, "Ad-hoc data processing in the cloud," Proc. of the VLDB Endowment, 1(2), pp. 1472-1475, 2008.

[14] Dazhao Cheng, Jia Rao, Yanfei Guo, Xiaobo Zhou,"Improving MapReduce Performance in Heterogeneous Environments with Adaptive Task Tuning",Published in: IEEE Transactions on Parallel and Distributed Systems (Volume: 28 , Issue: 3 , March 1 2017)

[15] Afrati, Foto N., and Jeffrey D. Ullman. "Optimizing joins in a map-reduce environment." In Proceedings of the 13th International Conference on Extending Database Technology, pp. 99-110. ACM, 2010.

[16] HadoopDB Project. Web page: www.db.cs.yale.edu/hadoopdb/hadoopdb.html

[17] Improving Map Reduce Performance through Data Placement in Heterogeneous Hadoop Clusters- Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding,

[18] “Big data: science in the petabyte era,” Nature, vol. 455, no. 7209, pp. 1–136, 2008.View at: Google Scholar

[19] P. Zikopoulos, C. Eaton, D. deRoos, T. Deutsch, and G. Lapis, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill, New York, NY, USA, 2011.

[20]H Hu, Y Wen, T Chua, X Li, “Towards Scalable Systems for Big Data Analytics: IEEE 2014.

[21] Wolf, J., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., Wu, K.-L., Balmin, A.: Flex: A Slot Allocation Scheduling Optimizer for Mapreduce Workloads. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 1–20. Springer, Heidelberg (2010)

[22] An Empirical Analysis of Scheduling techniques for Real-time cloud based data processing-linh T.X. Phan Zhuoyao zhang, Qi Zheng Boon Thau Loo University of Pennsylvania

[23] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. Starfish: A self-tuning system for big data analytics. In Proc. Conference on Innovative Data Systems Research (CIDR) (2011).