International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 5, Issue 10, October 2016 ISSN 2319 - 4847

Correlating Apache Spark and Map Reduce with Performance Analysis using K-Means

Dr. E. Laxmi Lydia[1,] Dr.A.Krishna Mohan[2], [3]S.Naga Mallik Raj

[1]Associate Professor, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, Andhra Pradesh, India.

[2]Professor & Head of the Department, Department of Computer Science and Engineering, JNTUK Andhra Pradesh, India.

[3]Assistant Professor, Department of Computer Science and Engineering, Vignan's Institute Of Information Technology, Visakhapatnam, Andhra Pradesh, India.

ABSTRACT

Big Data has for some time been the point of interest for Computer Science devotees around the globe, and has increased much more conspicuousness in the late times with the nonstop blast of information coming about because of any semblance of social media and the mission for tech mammoths to access further analysis of their information. This paper talks about two of the correlation of - Hadoop Map Reduce and the as of late presented Apache Spark – both of which give a preparing model to breaking down enormous information. Albeit both of these choices depend on the idea of Big Data, their performance differs altogether in view of the utilization case under usage. This is the thing that makes these two alternatives deserving of analysis as for their fluctuation and assortment in the dynamic field of Big Data. In this paper we contrast these two frameworks along and giving the performance analysis utilizing a standard machine learning algorithm for clustering (K-Means).The look into paper additionally thinks about the performance of the start and MapReduce with the parameters like speed, throughput and energy consumption.

General Terms
Big Data, Machine Learning, K Means.

Keywords
Big data, Hadoop, HDFS, Map Reduce, Spark, Mahout, MLib, Machine learning, K-Means.

1. INTRODUCTION

Apache Hadoop [1] is an open source framework that gives answers for taking care of big data alongside broad handling and examination. It was made by Doug Cutting in 2005 when he was working for Yahoo at the ideal opportunity for the Nutch internet searcher extend. Hadoop has two noteworthy parts named HDFS (Hadoop Distributed File System) [2] and the Map Reduce [3] framework. File System is said to be enlivened by Google's The Google File System (GFS) [4] and gives an adaptable, productive, and reproduction based capacity of data at different nodes that shape a part of a cluster.

HDFS is based on a master slave architecture where "namenode" is the master and "datanodes" are the slave nodes where the genuine data dwells (perhaps repeated data). The replication calculates as a matter of course is of three, however can be arranged according to the need of the client and the utilization sort. The second imperative part, which is Map Reduce is the preparing model for Apache Hadoop which permits effective handling of the imitated data in parallel in view of the previous programming dialect methods of map and reduce. Map is the stage which is actualized to conveyed segments of a dataset to different "mappers" that work in parallel to give the achievability to the quintessence of big data calculation. The yields from these mappers are presented to sorting and shuffling which takes the stream to the following stage, called the "Reduce" stage where data is aggregated to discover the outcome to our underlying issue explanation [5].

Although as of late, the universe of Big Data has seen a dynamic move from this computing model with the presentation and stable release of Apache Spark [6], which gives an easy to use programming interface to decrease coding endeavors and give better performance in a majority of the cases with issues related to big data. Spark gives an alternative to Map Reduce, as well as has choices for SQL like questioning with Shark and a machine learning library called MLib. The performance and working of spark is considerably not quite the same as that of map reduce, but at the same time is reliant on the constraints of parallelism, the sorts of issues in setting, and the assets available. Apache

Spark [7] started as a research extend at UC Berkeley in the AMPLab, was started with a goal to outline a programming model that backings a much more extensive class of applications than MapReduce, while maintaining its automatic fault tolerance.

Spark offers a reflection called Resilient distributed Datasets (RDDs) [8] to bolster these applications productively. RDDs can be put away in memory between questions without requiring replication. Rather, they revamp lost data on disappointment utilizing ancestry: each RDD recollects how it was worked from different datasets (by changes like map, join or groupBy) to reconstruct itself. RDDs permit Spark to beat existing models by up to 100x in multi-pass analytics. RDDs can bolster a wide assortment of iterative calculations, and additionally intuitive data mining and a profoundly effective SQL engine Shark [9].Figure 1 demonstrates the ecosystem system of Hadoop.

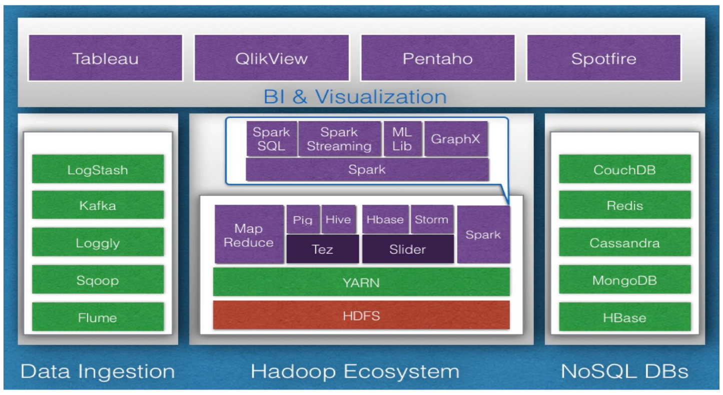


Figure 1: Hadoop Ecosystem

MapReduce

MapReduce is a handling method and a program model for distributed computing in view of java. The MapReduce algorithm contains two essential errands, in particular Map and Reduce. Map takes an arrangement of data and believers it into another arrangement of data, where singular components are separated into tuples (key/value sets). Furthermore, reduce assignment, which takes the yield from a map as an input and joins those data tuples into a littler arrangement of tuples. As the succession of the name MapReduce suggests, the reduce errand is constantly performed after the map work figure 2 demonstrates the handling MapReduce.

The real favorable position of MapReduce is that it is anything but difficult to scale data preparing over numerous computing nodes. Under the MapReduce model, the data preparing primitives are called mappers and reducers. Decaying a data preparing application into mappers and reducers is some of the time nontrivial. Yet, once we compose an application in the MapReduce frame, scaling the application to keep running over hundreds, thousands, or even a huge number of machines in a cluster is only a configuration change. This basic versatility is the thing that has pulled in numerous programmers to utilize the MapReduce model. Every Map assignment/center outputs the same number of spill documents as number of reducers [11][12][13].

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides.
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
- **Map stage** : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
- **Reduce stage** : This stage is the combination of the **Shuffle**stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

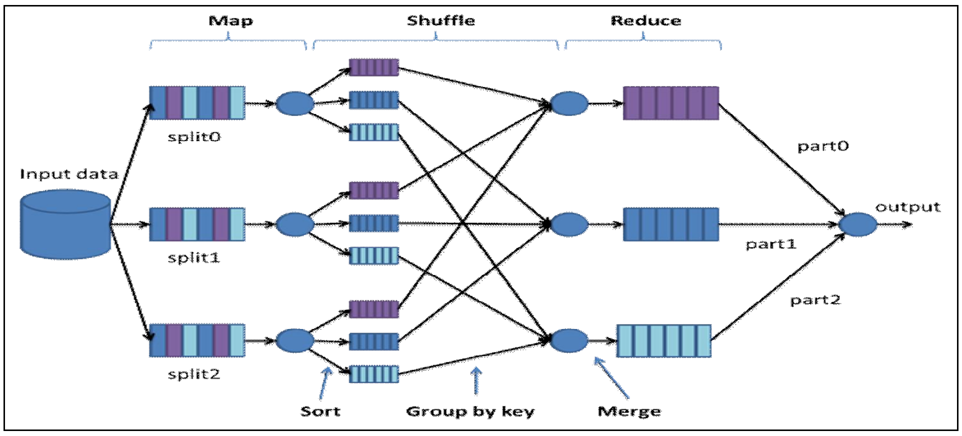


Figure 2: Inputs and Outputs (Java Perspective)

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

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3>(Output).

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

PACHE SPARK

Industries are utilizing Hadoop broadly to analyze their data sets. The reason is that Hadoop framework depends on a straightforward programming model (MapReduce) and it empowers a computing arrangement that is versatile, adaptable, fault-tolerant and cost effective. Here, the primary concern is to keep up speed in handling expansive datasets as far as waiting time amongst queries and

waiting time to run the program. Spark was presented by Apache Software Foundation for speeding up the Hadoop computational computing programming process.

As against a typical conviction, Spark is not an adjusted rendition of Hadoop and is not, generally, reliant on Hadoop on the grounds that it has its own particular cluster administration. Hadoop is only one of the approaches to actualize Spark. Spark utilizes Hadoop as a part of two ways – one is storage and second is preparing. Since Spark has its own cluster administration calculation, it utilizes Hadoop for storage reason as it were.

Apache Spark is an exceptionally quick cluster computing innovation, intended for quick calculation. It depends on Hadoop MapReduce and it extends the MapReduce model to productively utilize it for more sorts of calculations, which incorporates interactive queries and stream handling. The principle highlight of Spark is its in-memory cluster computing that builds the preparing speed of an application. Spark is intended to cover an extensive variety of workloads, for example, batch applications, iterative algorithms, interactive queries and streaming. Aside from supporting all these workload in an individual framework, it lessens the administration weight of keeping up discrete devices.

Evolution of Apache Spark

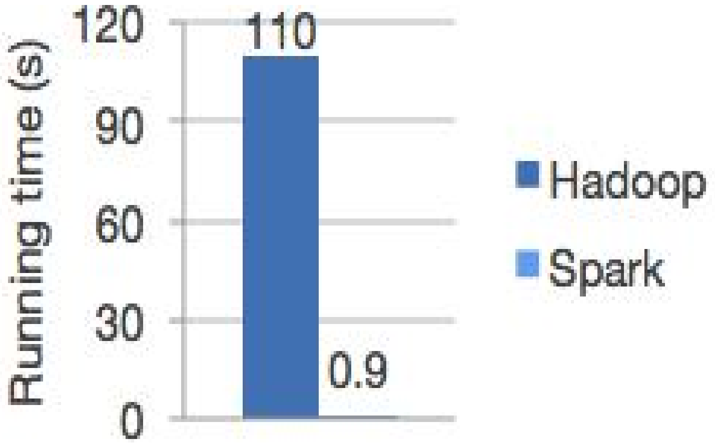
Spark is one of Hadoop's sub extend created in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD permit. It was given to Apache programming establishment in 2013, and now Apache Spark has turned into a top level Apache extend from Feb-2014.

Features of Apache Spark

Apache Spark has following features.

- **Speed:**

Spark enables applications in Hadoop clusters to run up to 100x faster in memory, and 10x faster even when running on disk. Spark makes it possible by reducing number of read/write to disc. It stores this intermediate processing data in-memory. It uses the concept of an Resilient Distributed Dataset (RDD), which allows it to transparently store data on memory and persist it to disc only it's needed. This helps to reduce most of the disc read and write – the main time consuming factors – of data processing.



- **Ease of Use:**

Spark lets you quickly write applications in Java, Scala, or Python. This helps developers to create and run their applications on their familiar programming languages and easy to build parallel apps. It comes with a built-in set of over 80 high-level operators.We can use it interactively to query data within the shell too.

Word count in Spark's Python API

```
datafile = spark.textFile("hdfs://...")
datafile.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda x, y: x+y)
```

- **Combines SQL, streaming, and complex analytics.**

  In addition to simple "map" and "reduce" operations, Spark supports SQL queries, streaming data, and complex analytics such as machine learning and graph algorithms out-of-the-box. Not only that, users can combine all these capabilities seamlessly in a single workflow.
- **Runs Everywhere**

  Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, S3.

Spark Built on Hadoop

The following diagram shows three ways of how Spark can be built with Hadoop components.

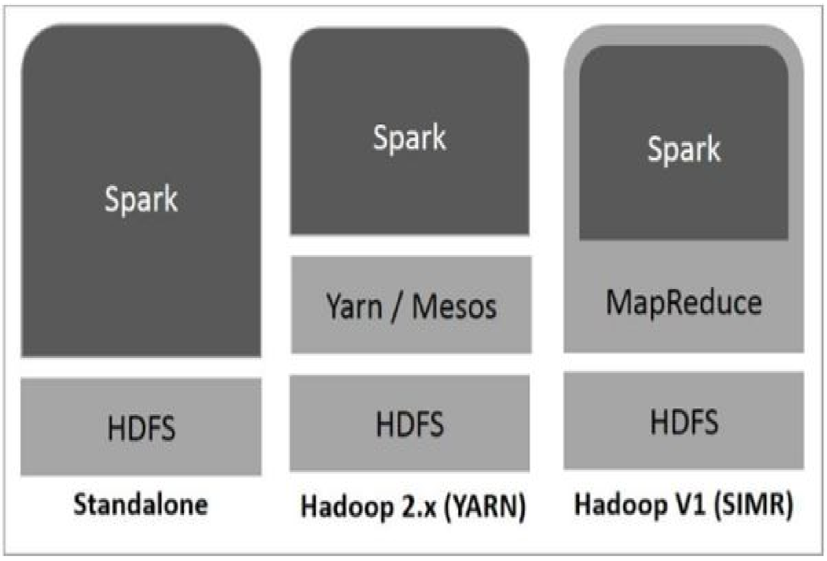


Figure 3 : 3 ways of spark to built with Hadoop components.

There are three ways of Spark deployment as explained below.

- **Standalone** − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
- **Hadoop Yarn** − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- **Spark in MapReduce (SIMR)** − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Components of Spark

The following figure 4 depicts the different components of Spark.

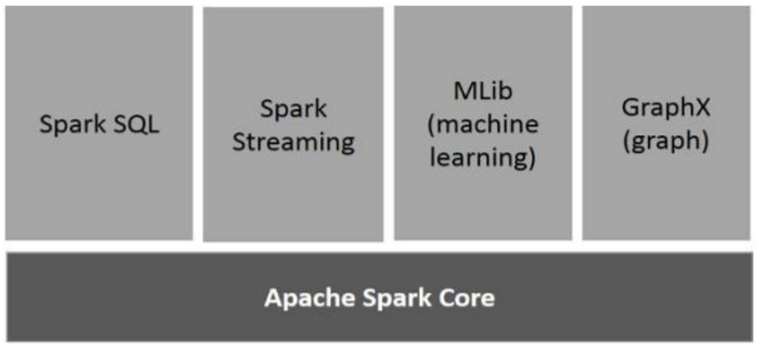


Figure 4: components of Spark

- **Apache Spark Core**

  Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

- **Spark SQL**

  Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

- **Spark Streaming**

  Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

- **MLlib (Machine Learning Library)**

  MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of **Apache Mahout** (before Mahout gained a Spark interface).

- **GraphX**

  GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

2. DIFFERENCE BETWEEN MAPREDUCE AND SPARK

Spark stores data in-memory while Hadoop stores data on disk. Hadoop utilizes replication to accomplish fault tolerance though Spark utilizes distinctive data stockpiling model, resilient distributed datasets (RDD), utilizes a sharp method for guaranteeing fault tolerance that minimizes network I/O.

- Iterative Algorithms in Machine Learning
- Interactive Data Mining and Data Processing
- Spark is a fully Apache Hive-compatible data warehousing system that can run 100x faster than Hive.
- Stream processing: Log processing and Fraud detection in live streams for alerts, aggregates and analysis
- Sensor data processing: Where data is fetched and joined from multiple sources, in-memory dataset really helpful as they are easy and fast to process.

a) **i) Hadoop vs Spark Performance**

Hadoop Spark has been said to execute batch processing jobs close around 10 to 100 times quicker than the Hadoop MapReduce framework just by simply by eliminating the quantity of reads and writes to the disc.

In the event of MapReduce there are these Map and Reduce assignments consequent to which there is a synchronization obstruction and one needs to protect the information to the disc. This component of MapReduce framework was produced with the expectation that if there should be an occurrence of disappointment the jobs can be recuperated however the downside to this is, it doesn't influence the memory of the Hadoop group to the most extreme.

b) ***Nevertheless with Hadoop Spark the idea of RDDs (Resilient Distributed Datasets) gives you a chance to spare information on memory and safeguard it to the circle if and just in the event that it is required and also it doesn't have any sort of synchronization hindrances that could back off the procedure. Therefore the general execution engine of Spark is much quicker than Hadoop MapReduce with the utilization of memory.***

c) **ii) Hadoop MapReduce vs Spark-Easy Management**

It is presently simple for the associations to streamline their infrastructure utilized for data processing as with Hadoop Spark now it is conceivable to perform Streaming, Batch Processing and Machine Learning all in similar cluster.

A large portion of the continuous applications utilize Hadoop MapReduce for generating reports that assistance in discovering answers to chronicled questions and afterward through and through postpone an alternate framework that will manage stream processing in order to get the key measurements progressively. Accordingly the associations should oversee and keep up particular systems and after that create applications for both the computational models.

However with Hadoop Spark every one of these complexities can be dispensed with as it is conceivable to actualize both stream and cluster preparing on similar system so it disentangles the development, deployment and support of the application. With Spark it is conceivable to control various types of workloads, so if there is a communication between different workloads in similar process it is simpler to oversee and secure such workloads which come as a restriction with MapReduce**.**

iii) Spark vs Mapreduce–Real Time Mechanism to Process Streams

In case of Hadoop MapReduce you simply get the opportunity to handle a batch of put away data however with Hadoop Spark it is too conceivable to alter the data progressively through Spark Streaming.

With Spark Streaming it is conceivable to go data through different software functions for example performing data analytics as and when Developers can now too make utilization of Apache Spark for Graph processing which maps the connections in data among different elements, for example, individuals and items. Associations can likewise make utilization of Apache Spark with predefined machine learning code libraries so machine learning can be performed on the data that is put away in different Hadoop clusters.

iv) Spark vs MapReduce –Caching

Spark guarantees bring down latency computations by caching the halfway results over its memory of distributed workers not at all like MapReduce which is disk arranged totally. Hadoop Spark is gradually ending up being a gigantic profitability help in contrast with composing complex Hadoop MapReduce pipelines.

v) Spark vs MapReduce- Ease of Use

Writing Spark is always compact than writing Hadoop MapReduce code. Here is a Spark MapReduce example-The below images show the word count program code in Spark and Hadoop MapReduce.If we look at the images, it is clearly evident that Hadoop MapReduce code is more verbose and lengthy.

2.1 Reasons to choose Spark

Spark utilizes the idea of RDD which allows us to store data on memory and persevere it according to the necessities. This allows an enormous increment in batch processing job execution (upto ten to hundred times as much as that of customary Map Reduce).

Spark additionally allows us to reserve the data in memory, which is gainful in the event of iterative algorithms, for example, those utilized as a part of machine learning.

Customary MapReduce and DAG motors are problematic for these applications since they depend on non-cyclic data flow: an application needs to keep running as a progression of particular jobs, each of which peruses data from stable storage (e.g. a distributed file system) and composes it back to stable storage. They acquire critical cost stacking the data on every progression and composing it back to imitated storage.

Spark allows us to perform stream processing with substantial info data and manage just a chunk of data on the fly. This can likewise be utilized for online machine learning, and is very proper for utilize cases with a prerequisite for constant examination which happens to be a practically omnipresent necessity in the business.

Specifically, MapReduce is wasteful for multi-pass applications that require low-latency data sharing over multiple parallel operations. These applications are entirely normal in examination, and include:

- Iterative algorithms, including many machine learning algorithms and graph algorithms like PageRank.
- Interactive data mining, where a user would like to load data into RAM across a cluster and query it repeatedly.
- Streaming applications that maintain aggregate state over time.

3. MACHINE LEARNING AND K-MEANS

3.1 Machine Learning Introduction

Machine learning is an active branch of artificial intelligence that allow computers to learn new patterns and instructions from data rather than being explicitly coded by a developer. Machine learning allows systems to enhance themselves based on new data that is added and to generate more efficient new patterns or instructions for new data [14].

3.2 K-Means Algorithm

K Means clustering is a non-hierarchical approach of collection things into various number of clusters/gatherings. The quantity of clusters/gatherings is characterized by the client which he picks in view of his/her utilization case and data being referred to K-Means works by framing group of data focuses by minimizing the total of squared separations between the data focuses and their centroids. A centroid is a main issue to a gathering of data focuses in the dataset. There are different methods for picking introductory centroid, yet as a rule it is done utilizing arbitrary allotment. The algorithm [14] is as per the following:

1. Firstly, select randomly chosen ‘k’ cluster centroids.
2. Cluster Assignment: In this step, assign each of the data points in the dataset to one of the centroids, selecting centroid which is closest to the data-point.
3. Centroid Movement: For each centroid, compute the average of all the data-points that are allocated to each centroid. This computed average is the new value of the particular centroid.
4. Calculate the sum of square of distance that each centroid has moved from its previous value, repeat steps 2 and 3 until this value is not less than or equal to threshold value (usually 0.01) or the number of iterations reaches maximum iterations specified, either of which is satisfied.

4. COMPARISON

With a specific end goal to arrive at a decision about the viable comparison of Apache Spark and Map Reduce, we played out a near examination utilizing these frameworks on a dataset that permits us to perform clustering utilizing the K-Means algorithm.

4.1 Dataset Description

The Data Set incorporates sensor data of size 1240 MB gathered throughout the years, and incorporates latitude and longitude estimations of the individual records. A specimen of the data records is appeared as beneath: The data record contains:

1. Date
2. Device Name
3. Device ID
4. Status
5. Latitude
6. Longitude Sample Record:

4.2 Performance Analysis and Description

Post working on the K-Means algorithm on the described data set, we achieved the following results for comparison (shown in the table1 and 2). To gain a varied analysis, we considered 64MB, 1240 MB with a single node and 1240MB with two nodes and monitored the performance in terms of the time taken for clustering as per our requirements using K-Means algorithm. The machines used had a configuration as follows:

- 4GB RAM
- Linux Ubuntu
- 500 GB Hard Drive

The results clearly demonstrated that the performance of Spark end up being impressively higher as far as time, where each of the dataset measure results in an abatement in the handling time of up to three times when contrasted with that of Map Reduce. Despite the fact that there exists a minor vacillation in this outcome, this is because of the irregular way of the K-Means algorithm and does not influence the examination to an expansive degree.

Table 1. Results for K-Means using Spark (MLib)

Dataset Size	Nodes	Time (s)
62MB	1	18
1240MB	1	149
1240MB	2	85

Table 2. Results for K-Means using Map Reduce (Mahout)

Dataset Size	Nodes	Time (s)
62MB	1	44
1240MB	1	291
1240MB	2	163

Figure 5 and 6 shows the graphically performance analysis, figure5 demonstrates that spark is faster than map reduce by taking the datasets, nodes and time as consideration. Our results for this analysis shows that Spark is a very strong contender and brought changes with concern to the in-memory processing

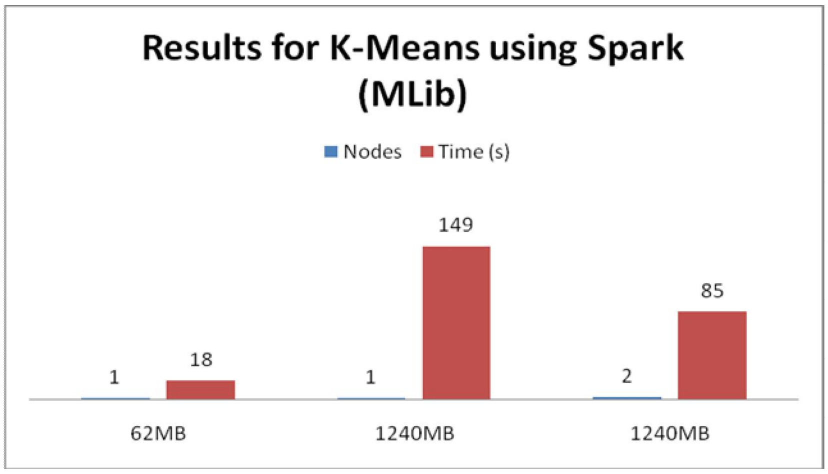


Figure 5: Results for K-means using Spark

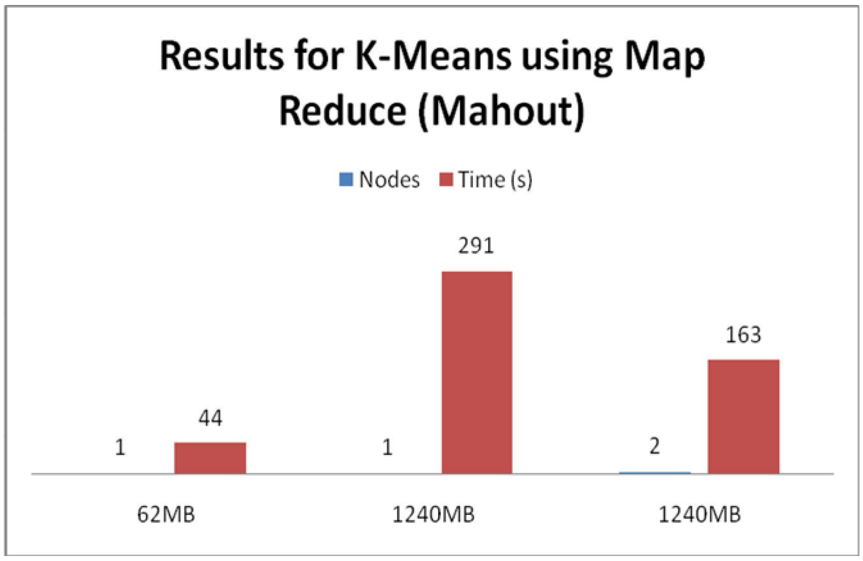


Figure 6: Results for K-means using Map Reduce (Mahout)

5. CONCLUSION

This paper gives a review of both the frameworks and additionally looks at these on different parameters took after by an execution analysis utilizing K-Means algorithm. Our outcomes for this analysis demonstrate that Spark is an extremely solid contender and would realize a change by utilizing as a part of memory preparing. Watching Spark's capacity to perform bunch handling, streaming, and machine learning on similar cluster and taking a gander at the present rate of reception of Spark all through the business, Spark will be the accepted structure for an expansive number of utilization cases including Big Data processing.

6. FUTURE WORK

Although most of the algorithms on Mahout till now have been based on Map Reduce, Spark’s consistent improvements and increasing user base has lead Mahout to adopt Spark for their base framework replacing Map Reduce for their future implementations. This is one of the many instances where Spark is proving out to gain predominance over Map Reduce.

REFERENCES

[1] Apache Hadoop Documentation 2014 http://hadoop.apache.org/.

[2] Shvachko K., Hairong Kuang, Radia S, Chansler, R The Hadoop Distributed File System Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium

[3] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In OSDI'04: Sixth Symposium on Operating System Design and Implementation, 2004.

[4] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York, 2003.

[5] HortonWorks documentation 2014 http://docs.hortonworks.com/HDPDocuments/HDP1/HD P-1.2.4/bk_getting-started-guide/content/ch_hdp1_getting_started_chp2_1.html

[6] Apache Spark documentation 2014 https://spark.apache.org/documentation.html.

[7] Apache Spark Research 2014 https://spark.apache.org/research.html.

[8] Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley, 2011

[9] Reynold Xin, Joshua Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, Ion Stoica. Shark: SQL and Rich Analytics at Scale. SIGMOD 2013. June 2013.

[10] Tom White, Hadoop the definitive guide chapter 06

[11] Spark Internals - Spark Summit 2014 http://spark-summit.org/wp-content/uploads/2014/07/A-Deeper-Understanding-of-Spark-Internals-Aaron-Davidson.pdf

[12] Spark Job Flow – Databricks https://databricks-training.s3.amazonaws.com/slides/advanced-spark-training.pdf

[13] Aaron Davidson, http://www.cs.berkeley.edu/~kubitron/courses/cs262a-F13/projects/reports/project16_report.pdf, Andrew Or. Optimizing Shuffle Performance in Spark. Technical Report

[14] Machine Learning, Wikipedia, 2014 http://en.wikipedia.org/wiki/Machine_learning

[15] Machine learning with Spark - Spark Summit 2013 https://spark-summit.org/2013/exercises/machine-learning-with-spark.html