IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 10, October 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

# Comparing the Word count Execution Time in Hadoop & Spark

**Z. Priyanka Reddy[1], P.N.V.S. Pavan Kumar[2]**

[1] Computer Science and Engineering Dept, GPREC,
Kurnool (District), Andhra Pradesh-518007, INDIA. priyankareddy.z5b8@gmail.com

[2] Asst.Prof. in Computer Science and Engineering Dept, GPREC,
Kurnool (District), Andhra Pradesh-518007, INDIA. pegatraj@gmail.com

**Abstract**

Hadoop is a very fashionable general rationale framework for many different classes of data-intensive applications. However it is not superior for fast word count operations because of the charge paid for the data reloading from disk at each iteration ie HDFS. As an promising framework Spark which is designed to have a universal cache mechanism, can achieve better performance in response time since the in memory access over the distributed machines of cluster will proceed during the entire iterative(word count) process. Even though the performance on time has been evaluated for Spark over Hadoop. In this work conducted extensive experiments for word-count operations to compare the performance in both time and memory cost between Hadoop and Spark. It found that although Spark is in general faster than Hadoop in running operations, it has to pay for more memory consumption. Also, its speed advantage is weakened at the moment when the memory is not sufficient enough to store newly created intermediate results.

***Keywords:*** *Hadoop, HDFS, mapreduce, wordcount, spark*

## 1. Introduction

### Big Data

The term “Big Data” refers to the continuous massive expansion in the data volume and variety as well as the velocity of data processing. The Big Data[4] era is in full force today because of changing of technology through their instrumentation. The term Big Data can be interpreted in many different ways and it conforming to the volume, velocity, and variety attributes. Big Data solutions are for analyzing not only raw structured data but also analyze the semi-structured data from wide variety of sources Big Data solutions are iterative and exploratory analysis and it is a reciprocal of traditional analysis and it can be existing in better business outcomes and Big Data is well suited for solving information but don't natively fit within traditional database.

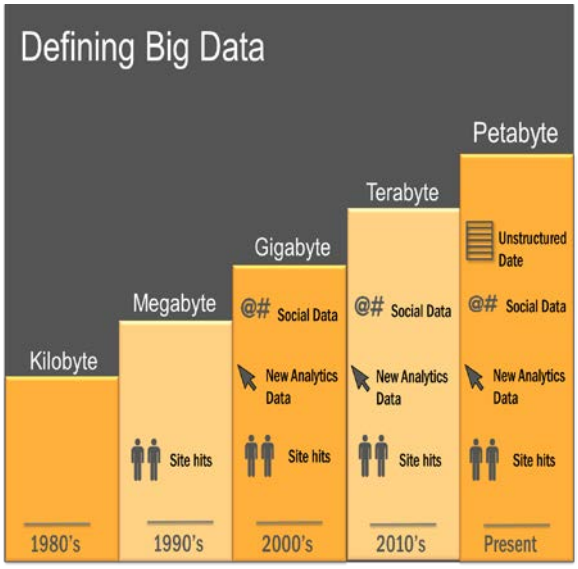



**Fig: 1. Understanding of Big Data**

It is clearly observed in the fig:1, in previous years the data can be in kilobytes and peta bytes only but present generation the data can be stored in very large amount of data at time it can be processed consequently.

### Characteristics of Big Data

Big Data is not a large amount of data it have the three characteristics Volume, Velocity, and Variety

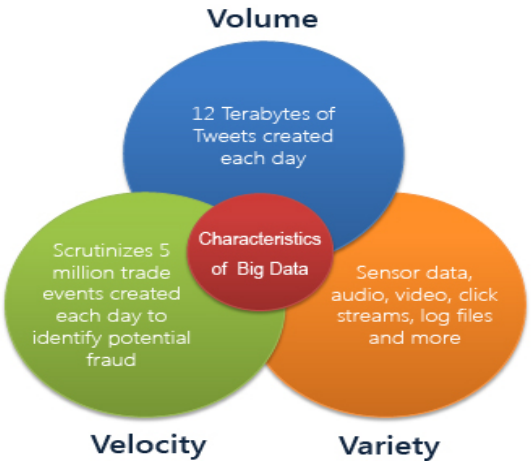



**Fig:2. 3Vs of Big Data**

Volume: Here Volume refers to the scale of the data and processing needs. And volume is nothing but gigabytes it is probably not Big Data, but at the terabyte level and beyond it may very well. I mean Exabyte's of data that big global enterprises have. There is a natural tendency for companies to store data of all stores like a financial data and medical data and environmental data and so on.

Variety: It refers to the different layouts of data like images, documents, videos, streams etc. Here data is become complex because of it include not only structured traditional data, but also semi-structured and unstructured

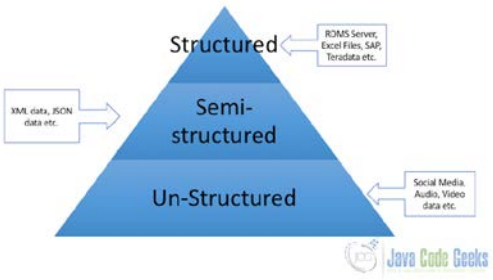



**Fig:3. Three types of Data**

**Structured data**: It is describe the data that resides in fixed field means files records tables and it is standard database and data contained in relational databases and spreadsheets. The data configuration and consistency allows it to respond to simple quires to arrive at usable information based on organization's parameters.

Semi-structured data: It is a database model where there is no separation between the data and schema, it is not a fixed schema and it contains tags or other markers to enforce hierarchies of records and fields within the data.

Unstructured data: It is refers to information that either does not have a predefined data model or is not organized in pre-defined manner. This type of data cannot easily be indexed into relational tables for analysis or querying like images, audio, and video files.

Velocity: Generally velocity refers to the speed of the data processing. And conventional understanding of velocity typically how quickly the data arrives and stored and how quickly it can be retrived and velocity should also be applied to data in motion. The speed at which the data is flowing is found. The various information streams and the increase in sensor network deployment have led to a constant flow of data at a pace that has made it impossible for traditional systems to handle.

## Hadoop

Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library. Hadoop has its origins in Apache Nutch, an open source web search engine. Hadoop[5] was made its own top-level project at Apache in January 2008, which was developed by Doug Cutting after joining his team in Yahoo.

- Hadoop distributes data and computes across a large number of computers.
- Hadoop is an open source, horizontally scalable system for reliably storing and processing massive amounts of data across many commodity servers.
- Hadoop is a distributed file system and how data is retrieved and distributed

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 10, October 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

## Components of Hadoop

There are primarily two major components of Hadoop. They are

1. MapReduce
2. HDFS (Hadoop Distributed File System)[6]

## MapReduce

MapReduce is a programming model for large scale data processing and MapReduce Programs write in various languages like Java, Ruby, Python, C++ etc one important point is the MapReduce programs are inherently parallel. MapReduce[15] comes into its own for large datasets. Here Datasets is nothing but a data can be stored as a semi-structured and record-oriented and data format supports a rich set of meteorological elements, many of which are optional or with variable data lengths.

- Restricted parallel programming model meant for large clusters.
- Map Reduce divides the workload into multiple independent tasks and schedule them across cluster nodes.
- In a Map Reduce cluster, data is distributed to all the nodes of the cluster as it is being loaded in.

How to map and reduce the data?

MapReduce works by breaking the processing into two phases they are map phase and reduce phase and each phase has key-value pairs as input and output.

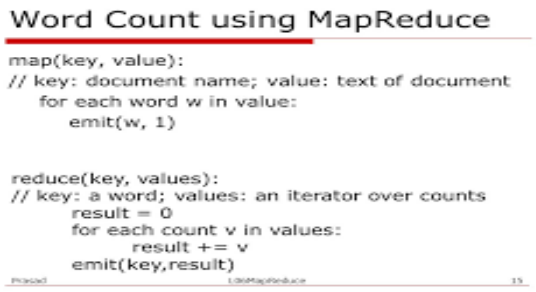



**Fig:4. Map-reduce algorithm**

Mapper:-

- Grab the relevant data from the source (parse into key, value).
- Generally Mapper maps input key value pairs to set of intermediate key value pairs and it transforms input records into intermediate records.
- The transformed intermediate records do not need to same type as the input records. And given input pair may map to zero or many output pairs.

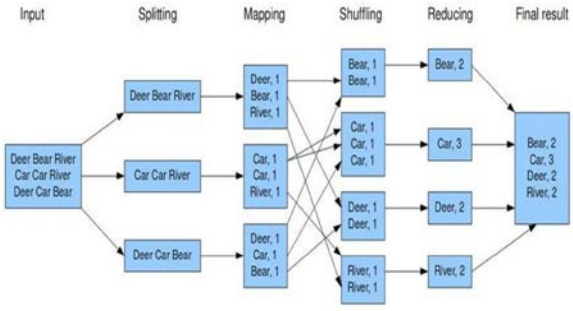
**Fig:4. Map Reduce Process**

Shuffling process (SORT):-

- Fetch the relevant partition of the output from all mappers.
- Sort by keys (different mappers may have output the same key).

**Reducer:-**

- Input is the sorted output of mappers.
- It reduces a set of intermediate values which share a key to a smaller set of values.
- Call the user Reduce function per key with the list of values for that key to aggregate the results.




Mapper: map (k, v) → list (k1, v1)

Reducer: reduce (k1, list (v1)) → v2

Generally, the map input key and value types (K1 and V1) are different from the map output types (K2 and V2). Here reduce input must have the same types as the

map outputs and the reduce output types may be different again (K3 and V3).

**HDFS (Hadoop Distributed File System)**

HDFS refers to "Hadoop Distributed File System". HDFS is designed for the storing of very large files with streaming data access patterns, running on clusters of commodity hardware. The key part of HDFS is its architecture itself.

**HDFS architecture**

- The HDFS is designed in the form of a master and slave pattern.
- Here, the master is the NameNode and the slave is the DataNode.

**NameNode:**

Here NameNode acts as an HDFS daemon and it control all the data node's file system operations and block mapping and internally name maintains the file system tree which is stored on the local disk in the form of two files. They are:

**NameSpace image:**

- The purpose of this is that it captures the snapshot of what the file system comprises of.
- The below is how a namespace image would be in the name node.
- This consists of the file_name, replication factor and block_ids.

**Edit log:**

The purpose of edit log is that it maintains the record of all transactions.

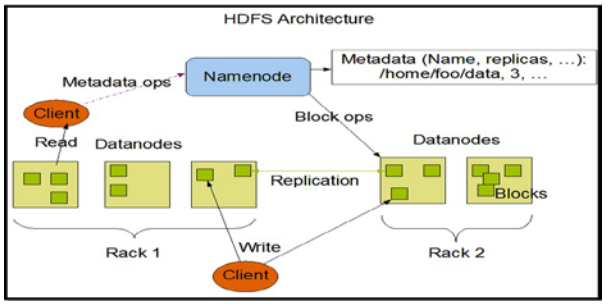



**Fig:5. HDFS Architecture**

## Apache Spark

Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project.

Spark has several advantages compared to other big data and MapReduce technologies like Hadoop and Storm.

Spark gives us a comprehensive, unified framework to manage big data processing requirements with a variety of data sets that are diverse in nature (text data, graph data etc) as well as the source of data (batch v. real-time streaming data).Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk. It is quickly write applications in Java, Scala, or Python. It comes with a built-in set of over 80 high-level operators. And you can use it interactively to query data within the shell. In addition to Map and Reduce operations, it supports SQL queries, streaming data, machine learning and graph data processing. Developers can use these capabilities stand-alone or combine them to run in a single data pipeline use case.

In this first installment of Apache Spark article series, we'll look at what Spark is, how it compares with a typical MapReduce solution and how it provides a complete suite of tools for big data processing.

## Hadoop and Spark

Hadoop as a big data processing technology has been around for 10 years and has proven to be the solution of choice for processing large data sets. MapReduce is a great solution for one-pass computations, but not very efficient for use cases that require multi-pass computations and algorithms. Each step in the data processing workflow

has one Map phase and one Reduce phase and you'll need to convert any use case into MapReduce[7] pattern to leverage this solution.

The Job output data between each step has to be stored in the distributed file system before the next step can begin. Hence, this approach tends to be slow due to replication & disk storage. Also, Hadoop solutions typically include clusters that are hard to set up and manage. It also requires the integration of several tools for different big data use cases (like Mahout for Machine Learning and Storm for streaming data processing).

It is string together a series of MapReduce[7] jobs and execute them in sequence. Each of those jobs was high-latency, and none could start until the previous job had finished completely.Spark[1] allows programmers to develop complex, multi-step data pipelines using directed acyclic graph (DAG) pattern. It also supports in-memory data sharing across DAGs, so that different jobs can work with the same data.Spark runs on top of existing Hadoop Distributed File System (HDFS) infrastructure to provide enhanced and additional functionality. It provides support for deploying Spark applications in an existing Hadoop v1 cluster (with SIMR –Spark-Inside-MapReduce) or Hadoop v2 YARN cluster or even Apache Mesos. Spark as an alternative to Hadoop MapReduce rather than a replacement to Hadoop. It's not intended to replace Hadoop but to provide a comprehensive and unified solution to manage different big data use cases and requirements.

## 2. Problem Statement

### 2.1 Existing System

Amount of data generated every day is expanding in drastic manner. Big data is a popular term used to describe the data which is in zetta bytes. Government, companies many organizations try to acquire and store data about their citizens and customers in order to know them better and predict the customer behavior. Social networking websites generate new data every second and handling such a data is one of the major challenges companies are facing. Data which is stored in data warehouses is causing disruption because it is in a raw format, proper analysis and processing is to be done in order to produce usable information out of it.

### 2.2 Proposed System

New tools are being used to handle such a large amount of data in short time. Hadoop was introduced as a solution to handle processing, storing and retrieving the big data. It is important for processor architects to understand what processor micro-architecture parameters that affect performance. Hadoop framework consists of several applications developed using MapReduce framework one of these applications is Word Count. MapReduce is a programmable framework for pulling data in parallel out of a cluster. Parallel processing is when a program executes multiple code paths simultaneously for a massive amount of data.

## 3. Experimental Environment

### 3.1 Gathering Single Node-Cluster Architecture

The experimental single node-cluster is composed of one computer. One of them is designated as master, and the other as slaves. The operating system Ubuntu 14.04.2 (GNU/Linux 3.5.0-28-generic x86 64) for the computer. All of the computers have 4 cores, but master and slaves have different frequencies. Apache Hadoop is a cluster manager that provides efficient resource isolation and sharing across distributed frameworks. It can run Hadoop, Spark, MPI, Hypertable and other frameworks. It will to manage the resources of our experimental cluster. where each slave contributes 1 CPU cores and 8GB memory to

HDFS. So the total resources managed by HDFS are 1CPU cores and 21 GB memory. When a job is submitted to Hadoop or Spark, It will allocate resources to it according to its demanding needs.

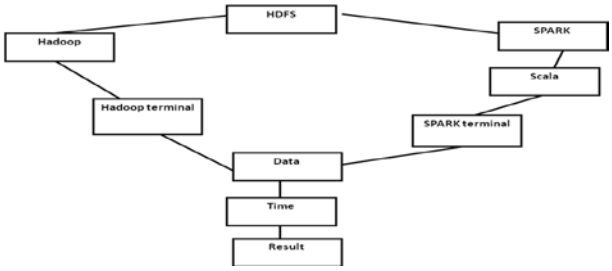



**Fig: 6. Hadoop and Spark Architecture**

Hadoop 2.6.0 and Spark 1.6.0 for all the experiments and Spark 1.6.0 are relatively new stable release. Although Hadoop 2.6.0 is the most recent stable release as of writing this paper, only Hadoop 0.20.205.0 is ported to run on mesos. Hadoop 0.20.205.0 in order to let Spark and Hadoop share the same experimental conditions.

### 3.2.DatasetDescription

Data size and data connectedness are two important properties of a specific dataset. In terms of graph dataset, the number of nodes reflects its size and the number of edges reflects its internal connectedness. These two properties have great impact on the running time and memroy usage of the mapreduce algorithm. Also install Hadoop 1.1.2 separately to repeat the experiment, and find that the new version Hadoop does have performance improvement and the improvement is a constant under different graph datasets. Due to space limitations, it will not show the results here. So we will choose different node number and edge number as our experimental datasets.

## 3.3 Apache Spark with Scala

Scala is a Scalable language. It is a modern multi-paradigm programming language designed to express common programming patterns in a concise, elegant, and type-safe way. Scala has been created by Martin Odersky and he released the first version in 2003. Scala smoothly integrates the features of object-oriented and functional languages. This tutorial explains the basics of Scala in a simple and reader responsive way.

scala is integrated with Apache Spark. It is easy to access the programming lambda expressons by using scala. It is integrated with Scala IDE.

## 4. Implementation and Results

### 4.1 Create jar and Data loading into HDFS

Hadoop platform when iteration number is fixed, the shapes of memory usage plots are similar with the increase in the size of dataset. Differences exist in two aspects. When dataset size is larger, the gradual increment between two consecutive iterations is bigger and the three peaks in each iteration are higher. When the increment reaches memory limit of a slave, the increase stops and the height of the three peaks does not change.

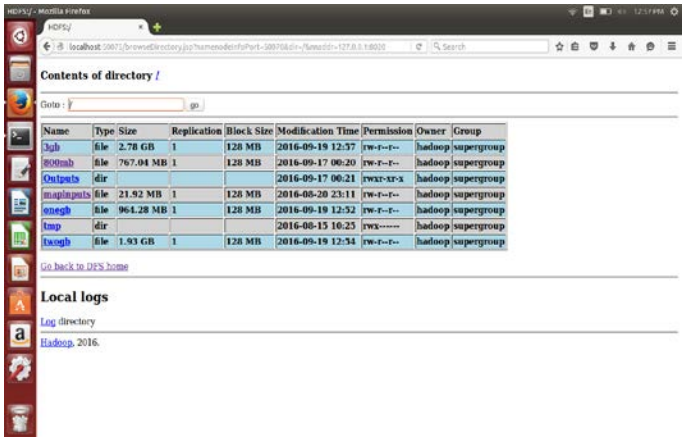



**Fig: 7.After creating the jar load the data into HDFS browser**

In this HDFS browser load the data and executing taking so much of time. In large environment it will take more time.

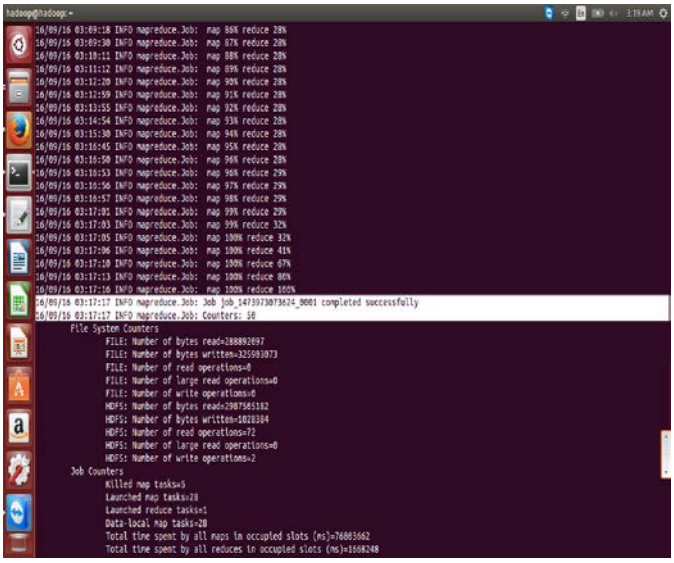


**Fig:8. Running the job and completed**

## 4.2 Generate RDD's By using Apache Spark and Scala

Spark platform when dataset is fixed, memory usage only exhibits occasionally gradual increase with the increase in the number of iterations. The reason for periodicity is the same as that for Hadoop. However, there are no peaks in each cycle. The gradual increment between two consecutive map-reduce is because of the creation of new auxiliary RDDs. Spark uses RDD[23] for data cache and RDD is a read-only data structure.

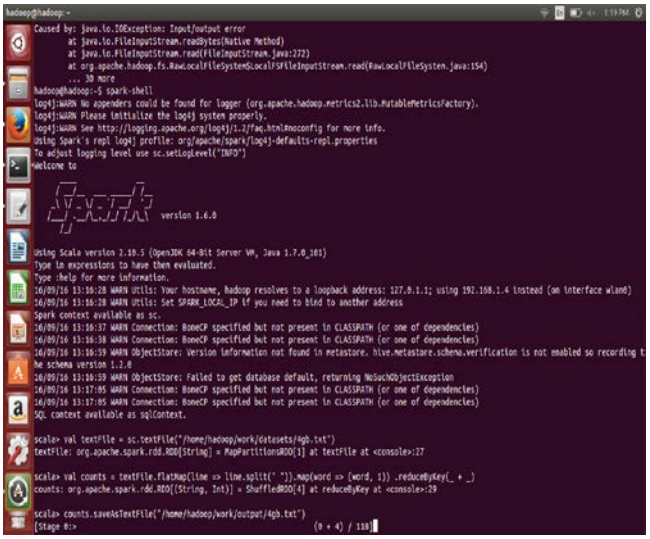


**Fig: 9.Data running in Spark environment**

For Spark platform when iteration number is fixed, the shapes of memory usage plots are similar with the increase in the size of dataset. When dataset size is larger, the gradual increment between two consecutive iterations is bigger. When the increment reaches memory limit of a slave, the increase stops. When the dataset size is too large, the program cannot run without enough memory. In Hadoop platform, iterative operations are implemented through multiple MapReduce computations. Each iteration corresponds to one or more MapReduce computations. When a Map or Reduce computation ends, the memory used by it will be released. Whereas, in Spark platform, iterative operations are implemented in a single Spark program. So there is no memory release phenomenon in Spark implementation. From the experimental memory plots, It can also find that Spark is extremely memory consuming. For a dataset whose size is 300M-4GB, the total 21GB memory of the cluster will be occupied.

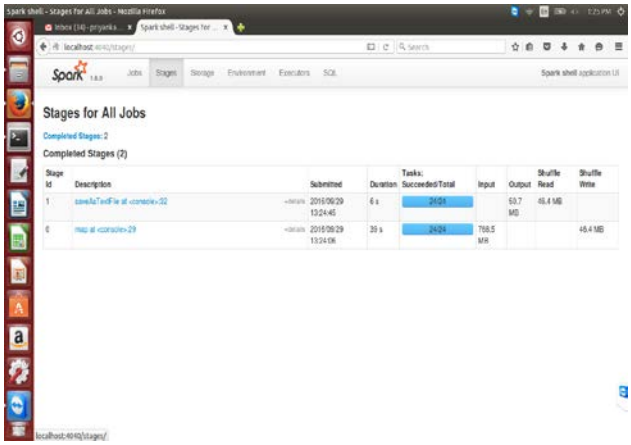


**Fig: 10. Data Jobs running in Apache Spark**

## 4.3 Apache Hadoop and Spark Comparison

Before get into further discussion on what empowers Apache Spark over Hadoop MapReduce let us have a brief understanding of what actually Apache Spark is and then move on to understanding the differences between the two. Apache Spark is an open source standalone project that was developed to collectively

function together with HDFS. Apache Spark by now has a huge community of vocal contributors and users for the reason that programming with Spark using Scala is much easier and it is much faster than the Hadoop MapReduce framework both on disk and in memory. Hadoop and Spark is just the apt choice for the future big data applications that possibly would require lower latency queries, iterative computation and real time processing on similar data. Hadoop Spark has lots of advantages over Hadoop MapReduce framework in terms of a wide range of computing workloads it can deal with and the speed at which it executes the batch processing jobs.

Apache Spark executes the jobs in micro batches that are very short say approximately 5 seconds or less than that. Apache Spark has over the time been successful in providing more stability when compared to the real time stream oriented Hadoop Frameworks. however every coin has two faces and yeah so does Hadoop Spark comes with some backlogs such as inability to handle in case if the intermediate data is greater than the memory size of the node, problems in case of node failure and the most important of all is the cost factor. Hadoop Spark makes use of the journaling for providing resiliency in case there is a node failure by chance as a result it can conclude that the recovery behavior in case of node failure is just similar as that in case of Hadoop MapReduce except for the fact that the recovery process would be much faster. Spark also has the spill to disk feature incase if for a particular node there is insufficient RAM for storing the data partitions then it provides graceful degradation for disk based data handling. When it comes to cost, with street RAM prices being 5USD per GB, it can have near about 1 TB of RAM for 5K USD thus making memory to be a very minor fraction of the overall node costing. One great advantage that comes coupled with Hadoop MapReduce over Apache Spark is that in case if the data size is greater than memory then under such circumstances Apache Spark will not be able to leverage its cache and there is much probability that it will be far slower than the batch processing of MapReduce.

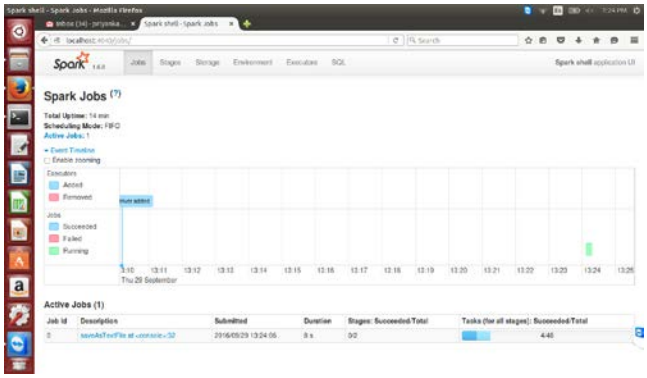



**Fig: 11. Running the Spark jobs**

However the current trends are in favor of the in-memory techniques like the Apache Spark as the industry trends seem to be rendering a positive feedback for it. So to conclude with it can state that, the choice of Hadoop MapReduce vs. Apache Spark depends on the user-based case and we cannot make an autonomous choice.

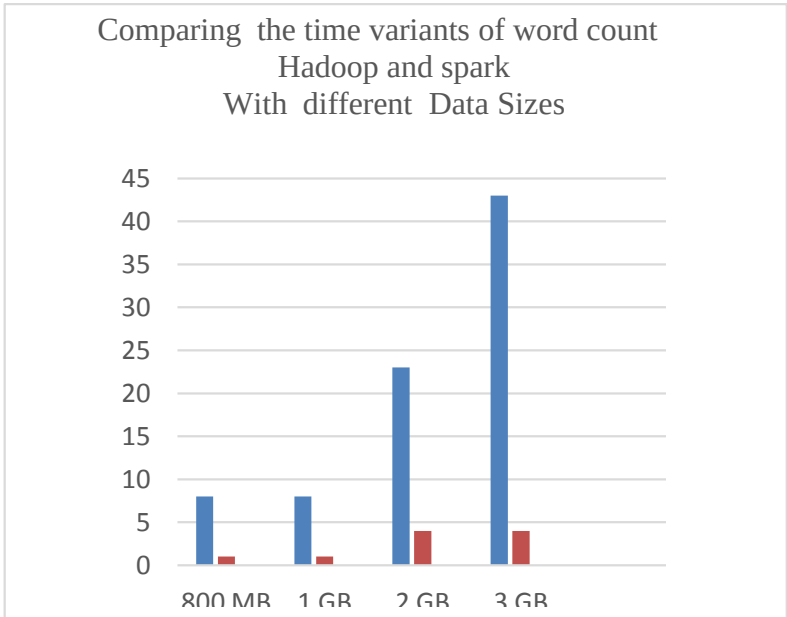



**Fig: 12. Comparison between Hadoop and Spark**

## 5. Related Work

As a new programming model MapReduce [2] is fitting for processing and generating large data sets in a scalable, reliable and fault-tolerant manner. Apache Hadoop [3] provides an open source implementation of MapReduce. Performance of MapReduce has been deeply studied in [17] [18]. The authors show that with proper implementation, performance of MapReduce can approach traditional parallel databases while achieving scalability, flexibility and fault-tolerance[22].Spark [1] was developed recently to optimize iterative and interactive computation. It uses caching techniques to dramatically improve the performance for repeated operations. The main abstraction in Spark is called resilient distributed dataset (RDD), which is maintained in memory across iterations and fault tolerant. Spark can outperform Hadoop by 10x in iterative machine learning jobs, and can be used to interactively query a 39 GB dataset with sub-second response time. Other similar works include Twister [7] and HaLoop [8]. Compared with Hadoop, Twister outperforms Hadoop in parallel efficiency; HaLoop averagely reduces query runtimes by 1.85, and shuffles only 4% of the data between mappers and reducers. Some efforts focus on iterative graph algorithms, an important class of iterative operations. Pegsus [19] unifies many iterative graph algorithms as Generalized Iterated Matrix-Vector multiplication (GIM-V). By exploiting special properties of matrix, it can achieve more than 5x faster performance over the regular version. Pregel [20] chooses a pure message passing model to process graphs. Distributed GraphLab [21] is similar to Pregel. The key difference between Pregel and GraphLab is that Pregel has a barrier at the end of every iteration, whereas GraphLab is completely asynchronous. Experiments show that applications created using Distributed GraphLab outperform equivalent Hadoop/MapReduce implementations by 20-60x.

## 6. Conclusion and Future work

For different experiment settings, it find that although Spark is in general faster than Hadoop, it is at the cost of significant memory consumption. If speed is not a demanding requirement and do not have copious memory, it's better not choose Spark. In this case, as long as it has enough disk space to accommodate the original dataset and intermediate results, Hadoop is a good choice. For a specific iterative operation, if the application is time sensitive, Spark should be considered. But enough memory is a necessary condition for the operation in order to secure Spark's performance advantage. The problem is that it is hard to determine the accurate amount of memory for iterative operations running on Spark. Exactly how much memory is enough depends on the particular iterative algorithm and the size of the dataset it processes. Intermediate results during iterations are stored in memory as RDDs. Because of the read only nature of RDD, new RDDs will always be created in each iteration. If the size of intermediate results is at constant level, the increase of memory usage between two consecutive iterations is not significant. If the size of the intermediate results is proportional to the size of the input dataset, the increase of memory usage between two consecutive iterations is significant. As part of our future work, the plan to find a prediction model for the tradeoff of response time and memory usage for iterative operations. Also we will explore the influence of graph structures on response time of iterative operations.

## References

[1] M. Zaharia, M. Chowdhury, S. S. Michael J. Franklin, and I. Stoica, "Spark: Cluster computing with working sets," In HotCloud, June 2010.
[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," In OSDI, 2004.
[3]Apache Hadoop. [Online]. Available:http://hadoop.apache.org
[4].Ms.Vibhavari Chavan, Prof.Rajesh, N.Phursle, "survey paper on big data",(IJCSIT),vol.5 ,2014.
[5]."Welcome to Apache Hadoop". Hadoop.apache.org. Retrieved 2015-12-16.

[6]. Serge Blazhievsky, "Introduction to Hadoop,Map Reduce and HDFS for Big Data Application". SNIA Education, 2013.
[7] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative MapReduce," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, pp. 810–818.
[8] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "HaLoop: Efficient iterative data processing on large clusters," in Proceedings of the VLDB Endowment, September 2010, pp. 285–296.
[9].Dr.A.J Singh, vibhav sarjolta MapReduce word count : execution and effects of altering parameters,IJIRCCE,vol.3,issue 10,october 2015
[10].Mahesh Maurya, Sunita Mahajan,"performane analysis of MapReduce programs on Hadoop cluster, IEEE, 2012.
[11].Trong -Tuan Vu, fabric Huet," A light weight continuous jobs mechanism for MapReduce frameworks", IEEE, 2013.
[12].Nandan Mirajkar1, Sandeep Bhujbal2 , Aaradhana Deshmukh3, "Perform word count Map-Reduce Job in Single Node Apache Hadoop cluster and compress data using Lempel-Ziv-Oberhumer (LZO) algorithm".
[13].Herodotos Herodotou, Hadoop performance models, Cornell University Library, Report Number CS-2011-05, arXiv: 1106.0940v1 [cs.DC].
[14].JOSEPH A.ISSA, "Performance evaluation and estimation model using regression method for Hadoop word count", IEEE volume 3, 2015.
[15] jeffrey dean and sanjay ghemawat,"MapReduce: Simplified Data Processing on Large Clusters", USENIX Association OSDI '04: 6th Symposium on Operating Systems Design and Implementation.
[16].Joseph Issa, Abdallah Kassem, "Disk I/O Performance-per-Watt Analysis for Cloud Computing", (IJCA), vol. 97, July 2014.
[17] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of MapReduce: An in-depth study," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 472–483, 2010.
[18] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: a survey," ACM SIGMOD Record, vol. 40, no. 4, pp. 11–20, 2012.
[19] "Pegasus: A peta-scale graph mining system implementation and observations," in Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on. IEEE, 2009, pp. 229–238.
[20]"Pregel: A system for large-scale graph processing," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 135–146.
[21] "Distributed GraphLab: A framework for machine learning and data mining in the cloud," Proceedings of the VLDB Endowment, vol. 5, no. 8, pp. 716–727, 2012.
[22].Vishal S Patil, Pravin D. Soni, "hadoop skeleton & fault tolerance in hadoop clusters", International Journal of Application or Innovation in Engineering & Management (IJAIEM)Volume 2, Issue 2, February 2013 ISSN 2319 – 4847 [8] Sanjay Rathe.
[23].V.Srinivas Jonnalagadda, P.Srikanth, Krishnamachari Thumati, Sri Hari Nallamala," A Reviw study of apache spark in bigdata proceesing",International Journal of Computer Science Trends and Technologies(IJCST)Volume 4, Issue 3, May-June 2016.

**First Author: Priyanka Reddy** was born in Andhra Pradesh, India. She received the B.Tech Degree in Computer Science and Engineering from Jawaharlal Nehru Technological University Anantapur branch, India in 2014 and M.Tech Degree in also same branch and University. Her research interests are in the area of Semantic Web and Big Data Analytics.

**Second Author**: **P.N.V.S. Pavan Kumar M.Tech[Ph.D],** was born in Andhra Pradesh, India. He is working as Asst.Prof, in Computer Science & Engineering Dept, GPREC Kurnool(district),A.P.518007,INDIA.His research interest s are in the area of big data .

.