I.J. Information Technology and Computer Science, **2018, 7, 61-70**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.07.07

A study and Performance Comparison of MapReduce and Apache Spark on Twitter Data on Hadoop Cluster

Md. Nowraj Farhan
Department of Computer Science & Engineering, University of Liberal Arts Bangladesh, Dhaka, 1209, Bangladesh
E-mail: nowraj.farhan@gmail.com

Md. Ahsan Habib and Md. Arshad Ali
Faculty of Computer Science and Engineering, Hajee Mohammad Danesh Science and Technology University, Dinajpur, 5200, Bangladesh
E-mail: {ahsan.habib, arshad}@hstu.ac.bd

Received: 17 March 2018; Accepted: 20 June 2018; Published: 08 July 2018

Abstract—We explore Apache Spark, the newest tool to analyze big data, which lets programmers perform in-memory computation on large data sets in a fault tolerant manner. MapReduce is a high-performance distributed BigData programming framework which is highly preferred by most big data analysts and is out there for a long time with a very good documentation. The purpose of this project was to compare the scalability of open-source distributed data management systems like Apache Hadoop for small and medium data sets and to compare it's performance against the Apache Spark, which is a scalable distributed in-memory data processing engine. To do this comparison some experiments were executed on data sets of size ranging from 5GB to 43GB, on both single machine and on a Hadoop cluster. The results show that the cluster outperforms the computation of a single machine by a huge range. Apache Spark outperforms MapReduce by a dramatic margin, and as the data grows Spark becomes more reliable and fault tolerant. We also got an interesting result that, with the increase of the number of blocks on the Hadoop Distributed File System, also increases the run-time of both the MapReduce and Spark programs and even in this case, Spark performs far more better than MapReduce. This demonstrates Spark as a possible replacement of MapReduce in the near future.

Index Terms—Big data, Hadoop, Java Virtual Machine (JVM), MapReduce, Supervised Learning, Apache Spark.

I. Introduction

Due to the current advent of new technologies, mobile devices, and communication media like social networking sites, the amount of data produced every year is growing at an enormous rate and the growth of this rate is also increasing beyond our imagination. Although the amount of data is increasing day by day, most of these data remain unused which can be stored and analyzed for BI to develop new business strategies [1], [12].

As we know, in the modern world, one of the main media of communications is the social networking sites, like Facebook and Twitter, which are so popular and widely used that sometimes we get current news of the world through these sites before any other media. The wide use of these media has made them one of the largest sources of data and one of the most reliable sources of data mining [2]. But, it is actually more or less impossible to store these huge data sets in RDMBSs like MySQL, as there is no specific formats of the data and can be in either text or image formats.

Here comes the power of Big data technologies, which are important in providing more accurate analysis and which may lead to more concrete decision-making resulting in greater operational efficiencies. But before analyzing, we need to capture and store these data. To harness the power of big data, we would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security.

Big data technologies like Apache Hadoop, Apache Flume, and Apache Spark have given us the power to capture, store and analyze this huge amount of data in very efficient and less costly ways. By using these technologies, it is now very easy to process data coming from sources like Twitter and getting the information we want.

II. Background

This section briefly describes the ecosystems of the Big Data infrastructure Apache Hadoop including Hadoop file system, and MapReduce programming model. An Apache data streaming technology that sinks data stream to Hadoop file system, called Apache Flume, is discussed. At last, but not least, Apache Spark, which

is a powerful in-memory distributed processing engine, is described. These are among other existing Apache Hadoop solutions. [11]

A. Relational Database Management System

A relational database is the most popular database model that is used for the storage of relatively small to medium size data and for the access of data using real-time queries. In a relational database, data is organized in tables whose fields are represented as columns and records are represented as rows. A Relational Database Management System (RDBMS) is a system which controls the storage, retrieval, insertion, deletion, modification and security of data in the relational database.

MySQL, especially the Community Edition, is the world's most popular, free, open source and easy-to-use RDBMS implementation. MySQL supports many DBMS features like replication, partitioning, views, MySQL workbench for visual modeling, and also supports features like MySQL connectors for writing applications using different programming languages. MySQL is used by many small to large organizations like Facebook, Twitter, Google and many more to power up their high volume websites. [4]

A typical MySQL deployment includes a server instance to be installed on a single, high-end server machine which accepts queries from local machines or remote hosts. But the problem with it is that data in the database is limited to the storage of the hard drives of the server, and as the data grows, it is needed to increase the number of hard drives, making it costly and more importantly less scalable. That is why with the growth of data this model fails and a DBMS with distributed storage system needs to be deployed.

B. Big Data Analytics

While the MySQL database was designed and vastly used for real-time queries on relatively small and medium data sets, it was not designed for large data sets or Big data analysis, because of the limited capacity of the storage mechanism and the underlying write-optimized "row-store" architecture.

Parallel DBMSs share the same capabilities as traditional, but run on a cluster where the distribution of data is transparent to the end user. Parallel DBMS offers high performance and reliability but much more expensive than traditional single-node RDBMS, because there is no freely available implementation, and yet they have much higher cost in terms of hardware, installation, and configuration. [4], [13].

C. Apache Hadoop

One solution to the above problem with traditional RDBMS is Apache Hadoop. Unlike MySQL, Hadoop can be deployed on a cluster of low-end systems, containing only commodity hardware, providing a cost-effective solution for Big data analysis. [4]

Apache Hadoop is a free, Java-based programming framework which supports the processing of large data sets in a distributed computing environment. Hadoop is designed to scale up from single servers to thousands of machines, each offering it's local storage to make an overlay of single storage and offering local computation to make a distributed processing. Such platforms are extremely fault tolerant. [2]

Hadoop is not a single entity, rather contains different components. Two main components of Apache Hadoop is HDFS and MapReduce.

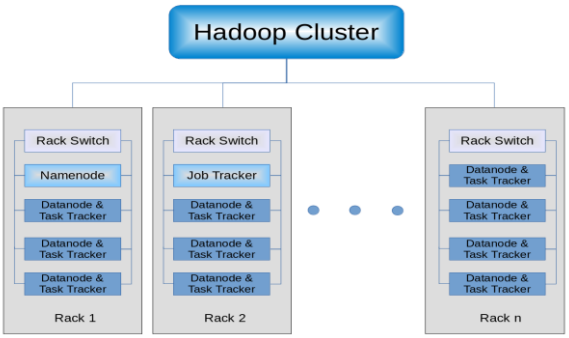

Fig.1. Top level view of a typical Hadoop Cluster

a. Hadoop Distributed File System

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

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

HDFS is written in the Java programming language. An HDFS cluster operates in a master-slave pattern, consisting of a master node or the NameNode and any number of slave nodes or DataNodes. The NameNode is responsible for managing the file system tree, the metadata for all the files and directories stored in the tree, and the locations of all blocks stored on the DataNodes. DataNodes are responsible for storing and retrieving blocks when the NameNode or clients request them.

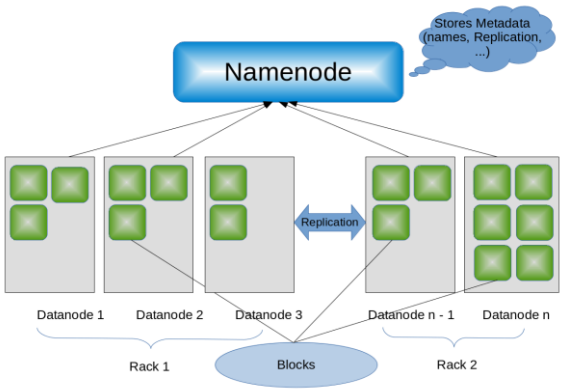

Fig.2. HDFS Architecture

b. MapReduce

MapReduce [9] is a programming model on top of HDFS for processing and generating large data sets using the MapReduce programming paradigm. It was developed as an abstraction of the map and reduce primitives of many functional languages and was designed to compute the large volume of distributed data in a parallel fashion. The abstraction of parallelization, fault tolerance and data distribution allows the user to parallelize large computations easily. The map and reduce model works well for Big Data analysis because it is inherently parallel and can easily handle data-set spanning across multiple machines [14].

Each MapReduce program runs in two main phases: the map phase followed by the reduce phase. The programmer simply defines the functions for the map and reduce phase and Hadoop handles the data aggregations, sorting and message passing between nodes. There can be multiple maps and reduce phase on a single program with possible dependencies between them. [5]

c. Map Phase

The input to the map phase is the raw data. A map function should prepare the data for input to the reducer by mapping the key to the value for each "line" of input. The key-value pairs output by the map function is sorted and grouped by key before being sent to the reduce phase.

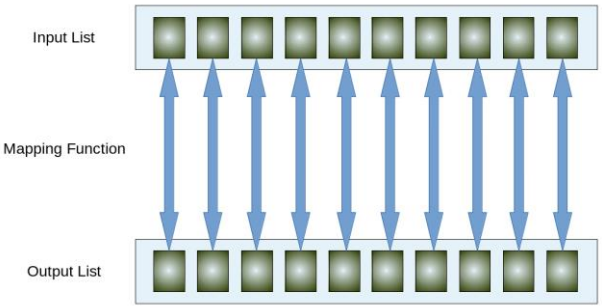

Fig.3. Map Phase

d. Reduce Phase

This stage is the combination of the Shuffle stage and the Reduce stage. The input to the reduce phase is the output from the map phase, where the value is an iterable list of the values with matching keys. The reduce function should iterate through the list and perform some operation on the data. Its job is to process the data that comes from the mapper end. After processing, it produces a new set of output, which is then stored in HDFS.

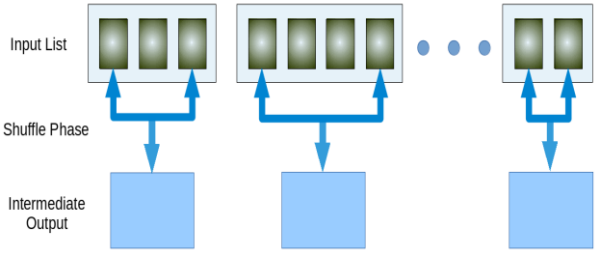

Fig.4. Combiner Phase

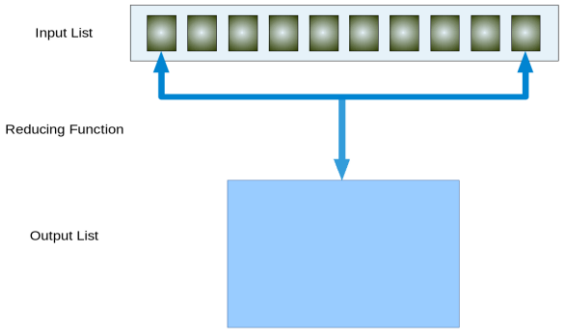

Fig.5. Reduce Phase

D. Apache Flume

Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into HDFS. It has a simple and flexible architecture based on streaming data flows, and is robust and fault tolerant with tunable reliability mechanisms for fail over and recovery. It uses a simple extensible data model that allows for online analytic application.

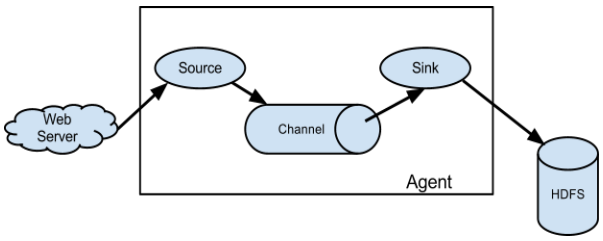

Fig.6. Flume Architecture

A Flume event is defined as a unit of data flow having a byte payload and an optional set of string attributes. A Flume agent is a Java Virtual Machine (JVM) process that hosts the components through which events flow from an external source to the next destination (hop). A Flume source consumes events delivered to it by an external source like a web server. The external source sends events to Flume in a format that is recognized by

the target Flume source. When a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it's consumed by a Flume sink. The sink removes the event from the channel and puts it into an external repository like HDFS or forwards it to the Flume source of the next Flume agent (next hop) in the flow. The source and sink within the given agent run asynchronously with the events staged in the channel.

The events are staged in a channel on each agent and then delivered to the next agent or terminal repository (like HDFS) in the flow and removed from a channel only after they are stored in the channel of next agent or in the terminal repository. This is how the single-hop message delivery semantics in Flume provide end-to-end reliability of the flow. Flume uses a transactional approach to guarantee the reliable delivery of the events.

E. Apache Spark

Apache Spark is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. The spark engine runs in a variety of environments, from cloud services to Hadoop Clusters. Spark supports a variety of popular development languages including Java, Python and Scala.

Apache Spark provides an elegant, attractive development API and allows data workers to rapidly iterate over data via machine learning and other data science techniques that require fast, in-memory data processing. Spark is 10-100 times faster than MapReduce delivering faster time to insight on more data, resulting in better business decisions and user outcomes.

Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone, Hadoop YARN or Apache Mesos. Spark can interface with a wide variety of distributed storages, including HDFS, Cassandra, OpenStack Swift, and Amazon S3.

Spark has two key concepts: Resilient Distributed Dataset (RDD) and directed acyclic graph (DAG) execution engine.

a. Resilient Distributed Dataset (RDD)

RDD is a distributed memory abstraction. It allows in-memory computation on large distributed clusters with high fault-tolerance. Spark has two types of RDDs: parallelized collections that are based on existing programming collections (like list, map, etc.) and files stored on HDFS. RDD performs two kinds of operations: transformations and actions. Transformations create new datasets from the input or existing RDD (e.g. map or filter), and actions return a value after executing calculations on the dataset (e.g. reduce, collect, count, saveAsTextFile, etc.). Transformations are the lazy operation that define only the new RDD while actions perform the actual computation and calculate the result or write to the external storage.

Directed acyclic graph (DAG) execution engine: Whenever the user runs an action on RDD, a directed acyclic graph is generated considering all the transformation dependencies. This eliminates the traditional MapReduce multi-stage execution model and also improves the performance.

III. Related Works

A. Text Mining

"Sifting through vast collections of unstructured or semi structured data beyond the reach of data mining tools, text mining tracks information sources, link isolated concepts in distant documents, maps relationships between activities, and helps answer questions." [6]

Text mining is a mechanism to find meaningful information from large amount of structured, semi-structured or unstructured text data. This encompasses the combination of human linguistic capacity and computational power of computers. The linguistic capacity includes the ability to differentiate variations in spelling, filter out noisy data, and understand abbreviation, synonyms etc. and finding the contextual meaning. The computational power of computers include the ability to process large amount of data at high speed. Some applications of text mining are: classification, clustering, information gathering etc. there are several types of algorithm for text mining, which can be categorized into two types: supervised learning and unsupervised learning. [7]

a. Supervised Learning

Supervised learning is a technique where the learning process is supervised by correct data before any prediction is made on the target data. It consists of finding the relationship between the predictor and the target attribute. If the algorithm can predict a categorical value, then it is called a classification function. If the algorithm can predict a numerical value, then it is called regression.

b. Unsupervised Learning

Unlike the supervised learning this technique doesn't require training to yield output. It only uses the predictor attribute values to gain understanding of the structure and relationship of the data. Finding the number of market segments, determining the themes of news etc. can be examples of unsupervised learning. Algorithms under unsupervised learning are feature extraction, clustering etc.

c. Document clustering

Document clustering can be defined as clustering of documents. Clustering is a process of understanding the similarity and dissimilarity between the given objects. Another way to look at clustering is, dividing objects into different meaningful subgroups based on their category, more specifically containing common characteristics. There are various clustering, which can be used to the purpose of document clustering. One of the most popular

clustering algorithm is the K-Means algorithm, which is often used to cluster large amount of data, even inside HDFS very efficiently. [8]

B. Hadoop Cluster Setup

Before working with big data, it is important to setup a Hadoop cluster, because the right resources allow to optimize the environment for the working purposes. However, it is not a simple task as optimizing a distributed environment and its related software can have its complexities.

Hadoop cluster has two types of machines:

- Master: Contains the HDFS NameNode, the MapReduce JobTracker
- Slave: Contains the HDFS Datanodes, the MapReduce TaskTrackers

There are several companies which provides platforms for setting up Hadoop in an efficient way and has very good monitoring system. One of which is Hortonworks, which lets users to setup Hadoop using Hortonworks Data Platform (HDP). Hortonworks recommends setting up master and slaves on separate computers as the master may be decommissioned from the cluster very often for maintenance and the slaves need to maintain a flawless workload without interrupting the master.

As Hortonworks suggests, cluster of three or more nodes should have a dedicated NameNode/JobTracker and the rest of the nodes should be used as slave nodes. For a NameNode failure, it is suggested to keep a secondary NameNode, which will hold all the information the NameNode has. That leaves the remaining hosts for Slave nodes, each running a DataNode and TaskTracker. [13]

C. Extracting Twitter Data

Recently, companies have discovered that social media analytics is crucial, especially for customer feedback and building goodwill. The analytics allow marketers to identify sentiment and detect trends in order to better accommodate the customer. There have been significant examples where companies, such as the airline industry, have used such analytical tools to reach customers based on feedback received. [3]

Tweets are the most up to date data information of current events. But they are also fragmented and noisy, motivating the need for systems that can extract, aggregate and categorize important events. There are many ways to extract tweets from twitter. TWICAL was the very first open-domain event-extraction and categorization system for twitter [14]. Programming languages like Java and Python can also be used to extract data from twitter, but has many drawbacks. The current and one of the most used techniques for retrieving tweets from twitter is to use the twitter API. And to use twitter API most efficiently it is better to use it with Hadoop and its ecosystems. For getting raw data from twitter using Hadoop streaming tool like Apache Flume is very flexible, reliable and fault tolerant [8]. Apache Flume needs initial configuration and define what type of data should be streamed from twitter. After that Flume will be able to stream data from twitter to the desired storage uninterruptedly [15].

IV. Research Methodology

Conducting this project was a combination of multiple steps. First of all a few research papers, journals and on-line articles were read to get the idea of what has already been done and what to do to make the project more interesting. The second step was to setup the server on a single node cluster with a small dataset to test the system. Then for the final approach, a large cluster was setup and a large dataset was uploaded from twitter into the cluster and few other steps were done for analyzing the data and get the output.

A. Descriptive Methodology

Big data, as a new feature in the fields of technology, still doesn't have that much work done on this. Although there are research activities in Big Data acquisition, storage, and processing, the knowledge of this domain is not properly spread among common people, and assumed that only software industry giants talk about this domain. After an initial field survey it was found that in Bangladesh very few companies work on big data, and those who are working with these are also struggling to find the proper solution to problems caused during the continuation of their projects.

Internet, the source of all information, has also quite a few good documentation of how to work with big data technologies step by step, from where to retrieve data, how to retrieve those data and so on. But the problem with those documentations is that they vary from version to version of different big data technologies like Apache Hadoop, and solution to a problem that works on a version of Apache Hadoop doesn't work with other versions.

A number of research papers, on-line articles and previous projects on different topics were reviewed to come up with an appropriate solution of how to dominate over the problems of different versions of big data technologies, although it was not a complete success, it paved the way to complete the project. After that it was time to find a way to collect the large amount of data and to work on that data find our answers.

B. Applied Methodology

The main aim of this project is to surf the areas of different big data technologies to find out an easy and efficient way to store huge amount of data, analyze those data to find the most tweeted programming languages andcompare the efficiency of different big data technologies. BigData performance measurement is quite challenges due to three V (volume, velocity, and variety). Performance challenges - due to volume includes scalability, and impact on networking, due to velocity includes access latency, and response time, due to variety

includes various types of data format [10]. In our study, we concentrate on the volume related issues only.

a. Environment Setup

First of all, we setup Hadoop and Spark on a single machine for testing purposes. But, to provide fair and controlled environment for our analysis we needed to setup Hadoop and Spark on a distributed cluster.

We installed Hadoop using Hortonworks Ambari Server on a "seven node" cluster, where the Ambari Server was installed on a single machine to monitor the overall cluster operations. Hadoop was installed on the master node of the Ambari Server, and rest of the machines were kept as Datanodes.

Hadoop version 2.6.0 was used running on Java 1.8.0 which includes both the HDFS and MapReduce in distributed configurations. First, we configured the HDFS NameNode and MapReduce JobTracker on the Ambari master node along with five Datanodes as HDFS Datanodes and MapReduce TaskTrackers. One important thing to mention here was that, two nodes on the cluster, the one with the Ambari Server and another one which was used as NameNode, did not add computing power for MapReduce or Spark in this experiment. The first one was used just for monitoring purposes and the second one was the NameNode, whose only purpose was to maintain the file system tree, the meta data for directories in the tree, and the DataNodes on which all the blocks for a given file were located.

We left the default replication factor of three per block, and the number of map tasks and reduce tasks were also kept as default, means that MapReduce will set the number of map tasks and reduce tasks based on the number of blocks on HDFS.

We installed Apache Spark version 1.3.0 and configured it to run on the same HDFS described above.

b. Data Retrieving and Storage

Now that the environment setup was complete, it was time to retrieve data from twitter and start analyzing those data. For the purpose of this project, Tweepy library of the python programming language was chosen, as it gives an easy way to retrieve data from twitter and store them in our local storage. But, to retrieve data from twitter we first need to create a "Twitter App" using the Twitter Application Management System. After that some secret codes, access token, access secrete were provided to retrieve the necessary data from twitter. A python program was written to extract data from twitter into our local storage. When the program was executed it automatically retrieved data from twitter based on the filtering mechanism mentioned inside the code. The retrieved data was written as JSON format into a text file.

The problem with the above mentioned technique was that, it was a very slow process and it did not have any proper way to handle exceptions, like automatic disconnection while getting data. And we had to restart the program to retrieve data again. Another problem was that, we needed to write our data to a new file as the old one grows to a size of one gigabyte. Due to these problems, it was very hectic job to get data continuously as we had to monitor the retrieval flow of data all the time, in case it disconnects from the Internet or it needed a new file to write data on. Another huge problem with this process was that, we had to copy this huge amount of file into HDFS manually.

To solve this problem another technique was used, Apache Flume, which would give us a way to write data directly into HDFS in a fault tolerant way, meaning that it will automatically recover from different failures, like disconnection from the Internet.

Apache Flume is a member of Hadoop framework which is used to retrieve data from the Internet directly to HDFS and is very reliable and fault tolerant as it can handle different problems like the ones mentioned above. There are many ways to stream on-line data into HDFS using Flume either by writing a program or creating a Flume configuration file. We used the second method to stream twitter data into HDFS. We needed to use the secret codes which we got after creating the Twitter App. A configuration file was created with information like how we wanted to retrieve data and how to filter those data to only get the relevant information from twitter. After running Flume for about a week we had enough data to analyze them.

c. Analyze Data with MapReduce

Now that we had a reasonable amount of data on HDFS it was time to analyze them. Various tools are available to analyze big data. Our first choice was MapReduce, which is a Hadoop component to analyze big data using Java programming language.

The files uploaded in HDFS was in the format of a JSON file, which contained all the information about a tweet, for example, user ID of the person who posted the tweet, timestamps, location and so on. But we only wanted the text containing the original tweet. That is why, a simple MapReduce code was written using Java, which would clean the JSON file to erase all the unnecessary information and only the text was taken. After that it was quite simple to count the number of tweets containing different programming languages and the output was again written back to HDFS. To be more specific, we used the word count algorithm with a little bit of modification, which was a simple and easy to understand way to do the desired job. It was intended to use different clustering algorithm like K-Means, which would be more efficient but complex to understand and might fail in case of tweets which have different programming languages.

Next it was the time to compare the performance of how well does MapReduce use the parallelization of the cluster. So, we manually switched off the slave nodes one by one and calculated the running time of the MapReduce program on each steps and draw a graph on that.

d. Analyze Data with Apache Spark

One problem with MapReduce, as discussed above, is that it is comparatively very slow, and as the data grows it becomes slower. To solve this, another big data tools was chosen, Apache Spark, which is supposed to work

"10 to 100 times faster than MapReduce" for in memory processing and 10 times faster for data on disks or HDFS.

First of all we cleaned the JSON files as we did with MapReduce, and took only the text part of the tweet. Then we wrote a Spark based Java program to count the tweets containing different programming languages.

After that we compared the performance of Spark by reducing the number of slave nodes one by one and taking notes of the running time of the Spark program that we wrote to count the tweets. Finally, we drew a comparison graph based on the results.

e. Performance Comparison

As discussed above, we had analyzed our dataset using different tools, like MapReduce and Apache Spark. So, we wanted to compare the performance between two tools, how well these two stand against each other and which one should be preferable for the analysis purpose of big data.

We worked on single node and multi node cluster. Big data tools like Apache Hadoop, are supposed to work better on a cluster rather than a single machine. The larger the cluster the better the performance. So, we also tried to look at the comparison of how the cluster outperforms the computation power of a single machine.

V. Result and Analysis

The first goal with this project was to count the number of tweets related to different programming languages. So, the first step was to use different big data tools and generate the desired results. It was chosen to work on different platforms, to single node and cluster, to different data sets of different size and to different tools. After that it was time to run the computation on MapReduce and Spark on different size of data sets and results are noted down.

As stated above, we wanted to count the number of tweets containing different programming languages, it was done using both MapReduce and Apache Spark. Then the top four programming languages were picked that were tweeted by most of the people and drew a graph based on the result.

Then the running time and other necessary information of both MapReduce and Apache Spark programs were calculated, that we executed on our data sets, gathered from streaming on Twitter using Apache Flume, and drew a comparison graph on that.

After that the size of our data sets were changed and the programs were ran again on both MapReduce program and Spark program. Then the running time of those programs were calculated and the results were noted down. The result of the Table 1 is shown graphically in the Fig. 8. One more thing to notice was that, by default MapReduce and Spark creates map tasks based on the number of blocks on HDFS.

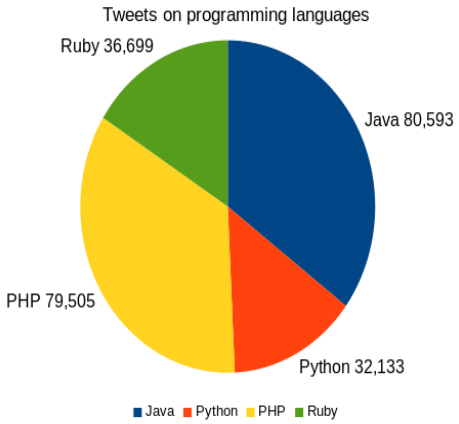

Fig.7. Top 4 most tweeted programming languages

So, the more blocks on HDFS the more map tasks the programs create. So, we ran both MapReduce and Spark on a 12 gigabytes of HDFS files containing different blocks, one with 7725 blocks and the other with 396 blocks that means both MapReduce and Spark created 7725 and 396 map tasks for those files.

Table 1. Run-time (minutes) of MapReduce and Apache Spark on Different size of data.

Data size(GB)	Run-time (MapReduce)	Run-time (Spark)
5.8	16.1	1.56
12.4	46.53	4.14
43.4	119.53	14.20

And we got a surprising results from the change of the number of blocks on HDFS.

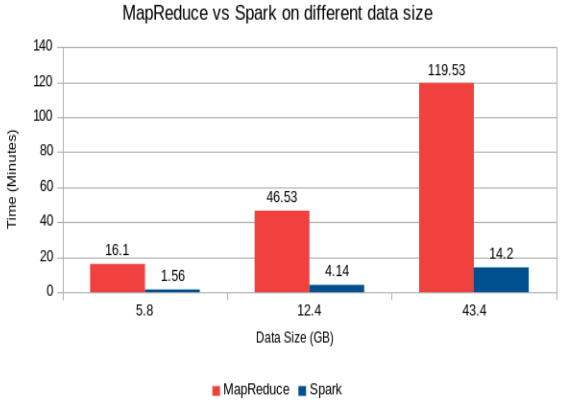

Fig.8. Run-time of MapReduce and spark on different data sizes

MapReduce took a very long time on the file with too many blocks, and outperforms itself by a huge range when the number of blocks was reduced on the same size of file. Similar result was observed from Spark too, but in case of Spark the result was not that much different when we reduced the number of blocks, as we got from

MapReduce. We can see the result from the table below.

Table 2. Run-time (minutes) of MapReduce and Apache Spark with the change of number of blocks on data size of 12.4GB.

# Blocks	Run-time (MapReduce)	Run-time (Spark)
7725	46.53	4.14
396	19.48	3.19

Bellow figure displays the results for different map tasks per DataNode. In this case, we observed that MapReduce with less map task per DataNode slightly better performs than more map tasks per DataNode.

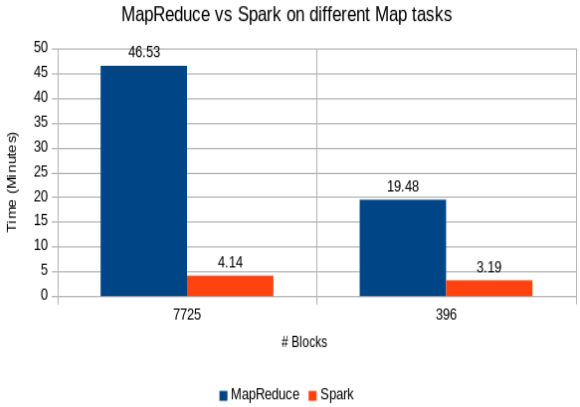

Fig.9. Comparison between MapReduce and Spark with the change of map tasks

Finally, the nodes from the cluster were reduced one by one and the running time of those programs were calculated, and the results were noted down.

Table 3. Run-time (minutes) of MapReduce on Different Datanodes on data size of 12.4GB.

# Datanodes	Run-time (Minutes)
5	46.53
4	60.06
3	79.19
2	*
1	*

* Unable to perform operation.

The result of the above Table 3 is show in the following Fig. 10.

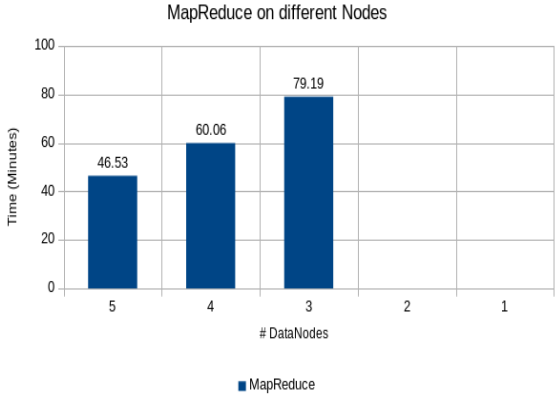

Fig.10. Run-time of MapReduce on different nodes

From the above figure it is clear that with reduced nodes on the cluster the run-time of MapReduce program also increases and at a point it cannot perform the computation at all. The reason is that, as explained earlier, data into HDFS are copied into different nodes as blocks for increasing fault tolerance, and MapReduce program could not find the required block and as a result it failed proceed with the execution. Increasing the replication in HDFS should have solved this problem.

Table 4. Run-time (minutes) of Apache Spark on Different Datanodes on data size of 12.4GB.

# Datanodes	Run-time (Minutes)
5	4.14
4	4.50
3	3.53
2	7.26
1	*

* Unable to perform operation.

The result of the above Table 4 is show in the following Fig. 11.

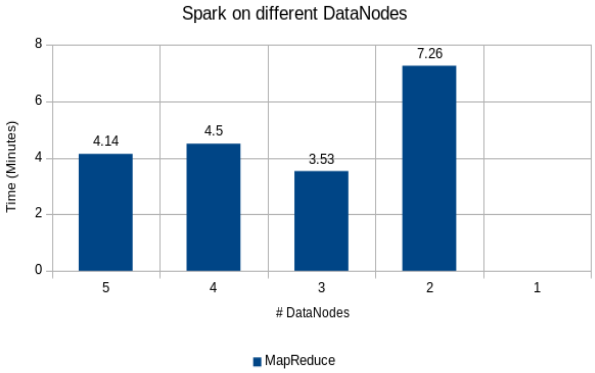

Fig.11. Run-time of Spark on different nodes

Fig. 11 shows that for computing tasks on three nodes, Spark performs better than the one with four or five nodes. The possible reason could be that it needs a lot of time to search for blocks in HDFS as they are scattered across all the nodes into the cluster, and with three active nodes it needed to search on a small cluster and after finding all the needed data it could execute the program immediately.

From these results, it is evident that both MapReduce and Spark works better on distributed mode rather than a single machine. And the performance of these frameworks also depend on the number of blocks on HDFS. So, to get better performance it is better to keep the number of blocks as less as possible.

It can also be concluded that Spark outperforms MapReduce by a dramatic range. And as the data size increases it is even better to use Spark rather than MapReduce, because the run-time of MapReduce also grows with the increase of data size due to its inherent nature of using disk I/O for all the steps of computation. From the above result it is also noticeable that in case of node failure Spark is more fault tolerant than MapReduce.

VI. Conclusion

This project tried to explore various fields of big data technologies, like Apache Hadoop, Apache Flume and Apache Spark. Hadoop can store very large amount of data in its file system called HDFS in a fault tolerant manner, and can scale up to any size as needed. Another Hadoop component, MapReduce, is capable of processing large data sets in a parallel fashion, and obviously, in a fault tolerant manner. Another big data technology, Apache Spark, which is "10 to 100 times faster than MapReduce", was one of the main point of concentration in this project. All of these frameworks were installed on a cluster and their working methodology was monitored using Hortonworks Ambari Server. And finally with the help of some simple algorithms, MapReduce and Spark programs were written through which the data sets on HDFS, which were of different sizes, were analyzed and the performance between MapReduce and Spark were compared, only to find out the performance characteristics of Apache Spark compared to traditional MapReduce approach.

References

[1] Marissa Rae Hollingsworth, "Hadoop and Hive as Scalable Alternatives to RDBMS- A Case Study", January 2012. Available: http://scholarworks.boisestate.edu/cs_gradproj/2/. [Accessed: 21 – Dec – 2017]

[2] Jodi Blomberg, "Twitter and Facebook Analysis: It's Not Just for Marketing Anymore", 2012. Available: http://support.sas.com/resources/papers/proceedings12/309-2012.pdf. [Accessed: 11 – Dec – 2017]

[3] Vora, M.N, "Hadoop-HBase for large-scale data", December 2011. Available: http://ieeexplore.ieee.org/document/6182030/?reload=true. [Accessed: 1 – Jan - 2018]

[4] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing'" July 2011. Available: https://www.usenix.org/node/162809. [Accessed: 23 – Nov - 2017]

[5] Penchalaiah.C, Sri.G.Murali, Dr.A.SureshBabu, "Effective Sentiment Analysis on Twitter Data using: Apache Flume and Hive". Available: http://www.ijiset.com/v1s8/IJISET_V1_I8_14.pdf. [Accessed: 3 – Dec - 2017]

[6] Weiguo Fan, Linda Wallace, Stephanie Rich, Zhongju Zhang, 'Tapping into the Power of Text Mining' 2005. Available: https://cacm.acm.org/magazines/2006/9/5835-tapping-the-power-of-text-mining/abstract. [Accessed: 14 – Jan - 2018]

[7] Dipesh Shrestha, "Text Mining With Lucene And Hadoop: Document Clustering With Feature Extraction", 2009. Available: https://pdfs.semanticscholar.org/36ce/71c9ff15cc46b32ab35d30d4b3b1c58cbfc6.pdf. [Accessed:]

[8] Alan Ritter, Mausam, Oren Etzioni. "Open Domain Event Extraction from Twitter", 2012. Available: http://www.cse.iitd.ac.in/~mausam/papers/kdd12.pdf. [Accessed: 7 – Feb – 2018]

[9] Dean J and Ghemawat S, "MapReduce simplified data processing on large clusters", Communications of the ACM 51:107-113, 2008.

[10] Pankaj Deep Kaur, Amneet Kaur, Sandeep Kaur,"Performance Analysis in Bigdata", IJITCS, vol.7, no.11, pp.55-61, 2015. DOI: 10.5815/ijitcs.2015.11.07

[11] Luis Emilio Alvarez-Dionisi,"Toward Grasping the Dynamic Concept of Big Data", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.7, pp.8-15, 2016. DOI: 10.5815/ijitcs.2016.07.02

[12] D. Newberry, "The role of small and medium-sized enterprises in the futures of emerging economies", Technical report, World Research Institute, 2006. Available: http://earthtrends.wri.org/features/view_feature.php?fid=69&theme=5. [Accessed: 5 - March - 2014].

[13] Big Data Working Group, "Big Data Analytics for Security Intelligence", Cloud Security Alliance, pp. 1-22, 2013.

[14] Dean, J and Ghemawat, J, "MapReduce: Simplified Data Processing on Large Clusters", In the Proceedings of the 6th Symposium on Operating Systems Design and Implementation, pp. 137-149, 2004.

[15] E. Benson, A. Haghighi, and R. Barzilay, "Event discovery in social media feeds" Proc. Of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 389-398, 2011.

Authors' Profiles

Md. Nowraj Farhan received his Bachelor of Science in Computer Science and Engineering from University of Liberal Arts Bangladesh in 2015. He is pursuing M.Sc in the area of software Engineering/ Information & Communication Systems. His main research interests include Software Engineering, Data Mining and Big Data Analytic.

Md. Ahsan Habib is currently an Assistant Professor of department of Computer Science and Engineering in Hajee Mohammad Danesh Science and Technology University, Bangladesh. He received his Master of Engineering in Information and Communication Technologies from Asian Institute of Technology (AIT), Thailand in the year of 2007, and the Bachelor of Science in Computer Science and Engineering (CSE) from Shahjalal University of Science and Technology (SUST), Sylhet-3114, Bangladesh in the year of 2003. In his academic career, he served as a faculty member of Computer Science & Engineering in several universities in Bangladesh including Asian University of Bangladesh, and University of Liberal Arts Bangladesh. Besides this, he was active in the software development industry in home (CTO in G5-Technologies Ltd., Bangladesh) and abroad (as a Senior Software Engineer in iSoftel – Thailand Co. Ltd., and as a Software Development Manager in Mobile-Technologies Ltd. in Thailand). His research interest lies in the area of Machine Learning, Data Mining, Big Data Analytics, and Computer Security. He is a member of IEEE.

Md. Arshad Ali was born in 1986. He received the Master of Engineering from Okayama University, Japan in 2018. He worked as an assistant professor in Hajee Mohammad Danesh Science and Technology University (HSTU), Bangladesh. Currently, he is pursuing his Ph.D. in the field of information security and cryptography at Okayama University, Japan. His research interest includes information security, AES, pseudo-random sequence, and homomorphic encryption. He is a member of IEEE.

How to cite this paper: Nowraj Farhan, Ahsan Habib, Arshad Ali, "A Study and Performance Comparison of MapReduce and Apache Spark on Twitter Data on Hadoop Cluster", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.7, pp.61-70, 2018. DOI: 10.5815/ijitcs.2018.07.07

