Comparison of Standard Computation Against Distributed Computation Using Hadoop Cluster

ABSTRACT

Computing the solutions to real life problems has become the need of the hour and has become a matter of utmost importance. In earlier times, it took days/months to compute and use algorithms to make things work, but as time passed on, computers became faster and more efficient, and these computations took less time. Now a days, new computation techniques are being invented. One such technique which will be discussed in this paper is Distributed Computation using Hadoop. Instead of using a single computer to solve problems, multiple virtual machines will be set up and will be used to prove quantitatively the time taken to execute one job using standard computation is much more than the time taken to execute the same job using Distributed Computation using Hadoop. This will help in faster computation and less time to perform tasks, as compared to standard methods which do not use cloud technology.

Keywords: *Algorithms, Computation techniques, Distributed Computing, Hadoop, Instances, Virtual Computers.*

1. INTRODUCTION

Abbreviations:

HDFS – Hadoop Distributed File System

YARN – Yet Another Resource Negotiator

LTS – Long Term Support

EC2 – Elastic Compute Cloud

SSH – Secure Shell

SCP – Secure Copy Protocol

The motivation for this project was to explore the field of Big data Analytics and to have hands-on experience to understand the importance of Big Data Analytics (i.e., when there is enormous amount of data). Big Data is a field that allows to methodically analyze and extract information from datasets which are deemed to be too large in volume to be processed by conventional data management tools. The motivation is to understand how this field of Big Data Analytics (in this case, using Hadoop) offers an advantage over traditional data management tools in terms of productivity, performance, and cost-effectiveness [2]. The true importance of Big Data Analytics will be realized once there can be comparison of traditional and distributed systems side-by-side in terms of the way in which they process data.

The sections below discuss the components that were involved in the development and working of the proposed method.

1.1. Hadoop

Hadoop Ecosystem is a framework of different tools that provide a substantial performance advantage when carrying out analysis on enormous datasets. Hadoop achieves this performance advantage by performing the processes pertaining to data analysis (Big Data analysis) through distributed computing. Hence, Hadoop is essentially a powerful and highly scalable framework which enables carrying out complex operations on huge sets of data (i.e. Big Data).

The Hadoop ecosystem consists of several distinct components. These are: Hadoop common, Hadoop YARN (Yet Another Resource Negotiator), HDFS (Hadoop Distributed File System) and Hadoop MapReduce [1]. Hadoop common ensures that all the required Java libraries, packages, and scripts along with other necessary files are available to run Hadoop. HDFS gives high throughput read and write access to data, and Hadoop MapReduce provides the structure for parallel processing of large datasets on different interconnected machines/nodes.

1.2. Data Storage in Hadoop

The data storage layer in Hadoop contains the Hadoop Distributed File System (HDFS), which is a file system based on Java which is very useful for storing large volumes of data on the networked machines, thus provided very high read/write throughput throughout the cluster by achieving high throughput from each node. Data pushed to HDFS is automatically duplicated and split into multiple chunks to provide high fault tolerance and availability. HDFS mainly consists of 2 main components:

NameNode: This is the master node which maintains the entire file system, and is also in charge of resource management, job scheduling, and verifying whether the entire job has been completed or not.

DataNodes: Also called slave nodes, they are the nodes which store the actual data and provide the results for any queries made by any client systems. They coordinate all file system operations between themselves.

1.3. Data Processing in Hadoop

Yet Another Resource Negotiator (YARN) and Hadoop MapReduce are situated in the data processing layer. MapReduce is a software programming framework which is the kernel of Hadoop's computing abilities. It enables distributed and parallel processing of large datasets on large clusters of machines. A MapReduce job divides the input data into smaller chunks of data which are then processed parallelly by the "mappers". The "reducers" then put together the output of the mappers to produce a relevant output for the query. The MapReduce framework and HDFS reside on the same nodes, so that tasks can be efficiently scheduled on nodes where the data is already present to prevent overhead. YARN ensure proper resource allocation and resource utilization on Hadoop so that the user does not have to worry about manually increasing the resource utilization of each node to extract maximum value. YARN is also responsible for providing Hadoop with the advantage of scalability in data processing tasks.

The purpose of this review was to understand the overall architecture and structure of the Hadoop ecosystem, and how different layers in this architecture work in tandem to form one big system which is Hadoop. It is clear from the research reviewed that the Hadoop ecosystem architecture is very different from the normal/standard mode of computation that is known. Along with this, there are some hints about why Hadoop could be better at performing data analysis tasks on large datasets than performing the said tasks on a standalone machine. Thus, it is important to conduct more research and studies to effectively quantify the actual advantage
[5] gained by the user when choosing Hadoop over a standard means of computation.

Also, here is a look at the Advantages and Disadvantages of Distributed Computing: (Refer Table 1)

1.4. Problem Statement

Create a Hadoop cluster, and perform data analysis/machine learning tasks on the created cluster. Finally, the obtained speedup will be compared with the standard computational power available.

1.5. Research Objectives

To quantify the speedup and performance improvement achieved by running a data analytics task on Hadoop versus running it on a standalone machine (standard computational means).

2. RELATED WORK

In [1], Mehta *et al.*, gave an overview on the newly invented and still in process of establishment technique of Hadoop Ecosystem. It also takes Hadoop as a combination of HDFS and MapReduce. It also considers the Hadoop modules for which there exist a variety of other projects that provide specialized services and are broadly used to make Hadoop more accessible and more user-friendly, which all together comprise of the Hadoop Ecosystem. In [2], helps to solve big data problems using Hadoop. Mainly this publication deals with the core of Hadoop, and it's working. It also talks about operational aspects of Hadoop and its applications in various fields nowadays. In [3] covers the characteristics in HDFS and MapReduce. This book attempts to build an open-source web program and helps to manage computations running on even a couple of computers. Once GFS and MapReduce papers were published by Google, the route became clear. In [4] Vliet, Programming Amazon EC2, O'Reilly Media, Inc., 2011 gave practical approaches for creating applications with AWS EC2 (Elastic Compute Cloud) and a number of other AWS tools, with an emphasis on crucial problems consisting of load balancing, monitoring, and automation. It helped to realize the application's roadmap and become aware of the AWS services needed. It gave a concept of how to create and run this software as part of the planning and implementation process. It helped relocate simple web applications to the cloud with EC2 and Amazon S3(Simple Storage Service). It gave explanations as to how Auto Scaling and Load Balancing provided by EC2 helps to meet traffic demand. It helped to discover the right equipment to reduce downtime, enhance uptime, and manipulate this decoupled system. In [5], Fu *et al.*, discussed about the changes in performance of a job in case more nodes are added to a Hadoop Computation, they found that this will result in faster computation of a job. They also discussed and evaluated three methods for data redistribution in this use case and discuss their advantage and disadvantages.

In [6], Park *et al.*, deduced a fast and scalable distributed algorithm called PACC (Partition Aware Connected Components) for performing connection component computation dependent on three key techniques which are, two-step processing of partitioning & computation, edge filtering, and sketching. In [7], n Husain *et al.*, went through previous work in the domain of distributed computation and gave an overview of what Hadoop Cluster is capable of and how it overcomes many problems in the field of Big Data Analytics, and also discusses about the various drawbacks it comes with which can later be improved in the near future. In [8], Wei *et al.*, developed a Hadoop Spark distribute framework supported on big-data technology, to accelerate the computation of typhoon rainfall prediction models. This study exploited deep neural networks (DNNs) and multiple linear regressions (MLRs) in machine learning, to ascertain rainfall prediction models and for rainfall prediction accuracy. For big-data technology, the Hadoop Spark distributed cluster-computing framework was the used. In [9], Zhang *et al.*, proposed MCRS as recommendation model and recommendation algorithm. MCRS is enforced on distributed computation framework. The basic algorithm of MCRS is distributed association rules mining algorithm. In [10], Alarabi *et al.*, presented an expansive study on ST-Hadoop; the first full-fledged open-source MapReduce framework with a provincial support for spatio-temporal data. In [11], Lei *et al.*, analyzed AIS data, a big data framework run on Hadoop, which extended the data type, storage, computing, and operation layer of traditional Hadoop to include trajectory data. In [12], Akaash *et al.*, gave an overview on Spark and Hadoop architecture, their differences and compared their performance. In [13], Li *et al.*, deduced improved fringe image processing method based on the Hartley transform because the traditional Fourier transform is complex and takes a long time to get implemented. For more speed and computational power, they even used Hadoop. In [14], Nagesh *et al.*, identified the factors affecting the performance of frequent item mining algorithm based on Hadoop MapReduce technology and proposed it as an approach for increasing the performance. In [15], Hemant Kumar Reddy *et al.*, employed a data location aware application scheme that optimizes performance by reducing runtime overhead of data transfer among clusters.

3. METHODS

3.1. System Architecture

As stated before, the system architecture for this project consists of the following 4 instances/virtual machines, pro- cured from Amazon Web Services, specifically their EC2 product (Elastic Compute Cloud) [16]:

HadoopNameNode – This is the master node which runs the processes of NameNode and JobTracker.

SecondaryNameNode – This is a backup node for the master node, and its job is to store a snapshot of the contents of HadoopNameNode as backup.

Slave1 – This is the first slave node, which runs 2 processes namely: TaskTracker and DataNode. TaskTracker reports to JobTracker in the master node, while DataNode reports to NameNode in the master node.

Slave2 – This is the second slave node, which runs 2 processes namely: TaskTracker and DataNode. TaskTracker reports to JobTracker in the master node, while DataNode reports to NameNode in the master node [17-18].

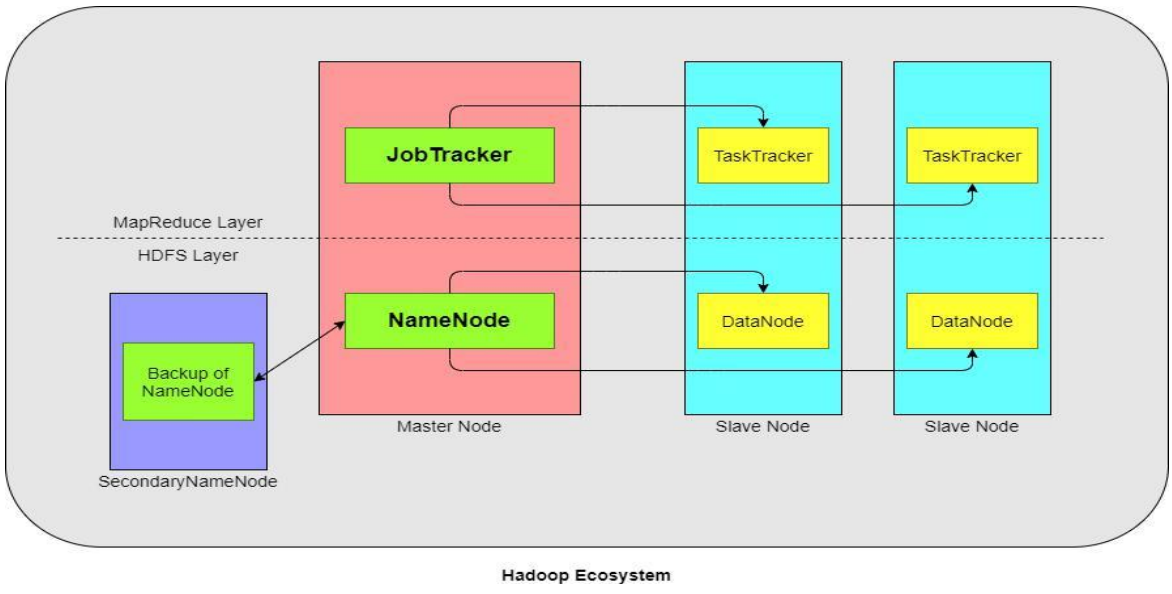

Figure 1 System Architecture

The JobTracker and the two TaskTracker processes are part of the MapReduce layer, while NameNode and the two DataNode processes are part of the HDFS layer.

This sums up the whole architecture used for this project.

(**Refer Figure 1** for system architecture)

3.2. Algorithms/Techniques

Following are few of the techniques and algorithms used while making the initial part of this project:

SSH (Secure Shell) – This is a cryptographic network protocol used to use network services securely over an un- secured network. Typically, its usages include login, remote command-line, and remote command execution (all of which is used in this project). SSH achieves this secure channel by employing a client-server architecture, coupling an SSH client with an SSH server. The SSH client that is used in this project is PuTTY [19].

SCP (Secure Copy Protocol) – This is a method to move computer file securely between a remote host and a local host or among 2 remote hosts. It is primarily built upon the SSH protocol. SCP is used in this project to transfer the Hadoop config files between machines to not have to make the same changes in the config files repeatedly, instead required changes can be made on one host, and then SCP can be used to transfer the files to the other host. The SCP client that is used in this project is WinSCP [20-21].

MapReduce – This is a computing/processing approach and a program model for distributed computing primarily built on Java [22]. It contains two essential tasks, specifically Map and subsequently, Reduce. Map takes a type of data and translates it into another type of data (tuples or key/value pairs) and Reduce aggregates the said tuples to provide an output. MapReduce programs will be used in this project to run jobs on the Hadoop cluster.

Table 1. Advantages and Disadvantages of Distributed Computing

Advantages	Disadvantages
Since computing takes place independently on every node, it is very handy and reasonably priced to feature extra nodes and capability anyplace necessary.	A distributed system like a Hadoop cluster must decide which jobs have to run when and the job schedulers used in these systems (Hadoop uses YARN scheduler) have certain limitations which can lead to incomplete hardware usage or wastage (inefficient scheduling).
Most distributed systems have a certain degree of redundancy as they are made up of multiple nodes that work together. If configured correctly, the system should not face any disruption in case a single machine fails (i.e., Reliability).	In case of a very widely distributed system with many nodes, it can take a longer time for the nodes to communicate which in turn can slow down performance. Thus, the issue of latency arises.
Workloads can be broken up and thus performance is greatly increased, and mostly performance in Hadoop clusters is less dependent on the CPU usage of each node, the main factor which determines performance is the memory usage.	Gathering, processing, and monitoring the hardware usage data is difficult when using distributed systems (such as a Hadoop cluster). This is especially difficult when dealing with larger clusters.

3.3. Design Methodology

The design methodology used in this project was the Waterfall model [2], as the requirements were mapped out first, which were – Four nodes (virtual machines/instances) to run the HDFS layer of Hadoop, and three nodes to run the MapReduce layer of Hadoop. Thus these requirements were used to proceed with the design of the system architecture (Four AWS EC2 instances procured, and Hadoop configured according to the requirements) and now this paper shall be moving on to the implementation phase as regular jobs are run on the cluster [23-25].

3.4. Setting up a Hadoop Cluster

The steps what were followed in setting up the Hadoop cluster were:

Acquiring and launching 4 instances of Amazon EC2 (Elastic Compute Cloud) [5] virtual machines through an AWS (Amazon Web Services) account, with the said virtual machines running Ubuntu 18.04 LTS and 8 GB of memory [26]. These instances were named NameNode

(master), SecondaryNameNode, Slave1(data node) and Slave2(data node).

Then client access is set up to the previously created EC2 instances, by using password-less SSH (Secure Shell) access among servers to set up the cluster. This gives remote access from master server to slave servers which enables master server to remotely and conveniently start the data node and task tracker services on the slave servers [27-28].

Then connect to the said EC2 instances, starting with the master node – NameNode, and then similarly connect to all the other nodes. Then enable public access in the nodes so that all the nodes can be accessed from the master node.

Then install Java on all the nodes by using the following command: sudo apt install openjdk-11-jdk [29].

Then download Hadoop v1.2.1 from the Apache download page, and then we unzip the files, look through the package content, and rename the hadoop-1.2.1 directory to just hadoop.

Then setup the environment variables so that we do not have to specify the entire paths for Java and Hadoop whenever we run a command pertaining to them.

Then add the AWS EC2 key pair provided, to the ssh-agent program (ssh-agent is a background program that takes care of passwords for SSH private keys), to enable password-less SSH on the servers [30].

Then edit the required Hadoop config files (specifying the master node, the slave nodes etc.) on all the servers to get the cluster up and running [31].

Then start up all the hadoop daemons from the master node (A daemon is a process that runs in the background on a multi-tasking operating system) using the command: start- all.sh. This will start:

NameNode, JobTracker and SecondaryNameNode die- mons on the master node and SecondaryNameNode daemons on the SecondaryNameNode, and also DataNode and TaskTracker daemons on the slave nodes.

Now, a big text file with multiple words is uploaded onto the Hadoop cluster. The data in question is – "The Project Gutenberg eBook of The Adventures of Sherlock Holmes, by Arthur Conan Doyle" and then implement the Hadoop Mapper and Reducer classes for this file in order to find out the word count of each word. The written code is exported as a .jar file and also uploaded to the Hadoop cluster for execution, and then the job is executed on the cluster [32].

Simultaneously, standard Java code is written and executed to process this file and find out the word count on a standalone machine without Hadoop (thus no mappers or reducers needed).

4. RESULTS

Now, having set up the Hadoop cluster, it can be verified that it is running by visiting the various webUI pages of the different processes:

The NameNode process webUI runs on port number 50070 on the master node.

The JobTracker process web UI runs on port number 50030 on the master node.

The TaskTracker processes run on port number 50060 on both the slave nodes.

The task using Hadoop is completed in approximately 22 seconds: 6 seconds for the 2 mappers each (Refer Figure 2) + 10 seconds for the reducer (Refer Figure 3). IntWritable and LongWritable data types are used which are only available in Hadoop and are optimized for serialization and are thus much faster than the standard Java Integer and Long data types.

On the other hand, the standard program using all standard Java classes, methods, and data types (Integer and Long) takes 353610 milliseconds just to read the text file. This is equal to roughly 353 seconds (Refer Figure 4).

Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_202106121149_0003_m_000000	100.00%	hdfs://ec2-3-88-217-29.compute-1.amazonaws.com:8020/test/big.txt:0+3244333	12-Jun-2021 11:59:39	12-Jun-2021 11:59:46 (6sec)		17
task_202106121149_0003_m_000001	100.00%	hdfs://ec2-3-88-217-29.compute-1.amazonaws.com:8020/test/big.txt:3244333+3244333	12-Jun-2021 11:59:39	12-Jun-2021 11:59:46 (6sec)		17

Figure 2 Mapper Execution Time

Task	Complete	Status	Start Time	Finish Time	Errors	Counters
task_202106121149_0003_r_000000	100.00%	reduce > reduce	12-Jun-2021 11:59:46	12-Jun-2021 11:59:56 (10sec)		15

Figure 3 Reducer execution time

Figure 4 Time taken by standard Java code to read the file

5. DISCUSSION

5.1. Source of input data

The input text file for this project is the e-book "The Adventures of Sherlock Homes by Arthur Conan Doyle" obtained from the Project Gutenberg library of free e-books.

This e-book was chosen due to the fact that it contains roughly 128458 lines of text, which was considered to appropriate to carry out a comparison on.

5.2. Comparative Analysis

Table 2. Comparative Analysis of results

METHOD	LANGUAGE	NUMBER OFNODES	SIZEOFINPUT (in Megabytes)	TIMETAKEN (in milliseconds)
Method using Java	Java	1	7	353610
Method using Hadoop	Java Hadoop	4	7	22000
Gohil P. *et al.*	Java Hadoop	5	600	192330
Gohil P. *et al.*	Java Hadoop	7	600	163670
R. Yadav *et al.*	Java	1	530	25840
R. Yadav et al	C#	1	530	22557
R. Yadav *et al.*	Java Hadoop	11	500	2143

The rows in bold denote the methods which use Java Hadoop (Distributed Computation).

In Gohil P. *et al.*, Java Hadoop is used and the number of nodes used in two cases are 5 and 7, with the time decreasing as the nodes are increased (approximately 192 seconds for 5 nodes and 163 seconds for 7 nodes, on a file of size 600MB).

R. Yadav *et al.* uses two languages in addition to Java Hadoop, with a clear decrease in execution time seen when comparing Hadoop to C# and Standard Java (approximately 22 seconds for C# and 25 seconds for Java, while Java Hadoop takes 2 seconds).

Thus, in all the cases, Java Hadoop using the MapReduce Framework greatly exceeds the time taken by standard single node execution. Also, it is noticed that as the number of nodes is increased, the execution time decreases which is in line with the findings of this paper as well.

6. CONCLUSION

6.1. Conclusion

Thus, it is found that the time taken by the Hadoop cluster to give the required output is far less than the time required by the standalone machine even for a file which is considered small by today's standards. This can be

extrapolated with bigger file sizes and larger number of nodes in the cluster, subject to cost constraints.

Clearly, the MapReduce distributed computation engine is superior in terms of time taken to complete a job when compared to standalone computation systems. Further, the results are quantified using the screenshots below. The performance is expected to increase much more rapidly in the case of Hadoop as higher specification machines and a greater number of nodes are obtained. This is because IntWritable and LongWritable data types are used which are only available in Hadoop and are optimized for serialization and are thus much faster than the standard Java Integer and Long data types.

Thus, the advantage offered by the special Hadoop file I/O tools over standard tools when trying to perform tasks on files with greater amounts of data is clearly observed.

6.2. Future Plan

As it is known Hadoop is completely written in Java, a language widely used by end users, engineers as well as cyber criminals hence it can lead to numerous security breaches. In future with more developments, more programming languages can be used to write in Hadoop, even few languages specific to Hadoop can be created which will help to curb the afore- mentioned problem. Hadoop only ensures that the data job is completed, but it does not guarantee when the job will be complete. This can also be something which can be worked on in near future.