Indonesian Journal of Electrical Engineering and Computer Science
Vol. 13, No. 1, January 2019, pp. 392~404
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v13.i1.pp392-404

Performance evaluation of cloud service with hadoop for twitter data

P Ganesh[1], K Sailaja Kumar[2], D Evangelin Geetha[3], T V Suresh Kumar[4]
[1]Dept. of MCA, BMSIT&M, India
[2,3,4]Dept. of Computer Applications, MSRIT, India

Article Info

Article history:

Received Jul 18, 2018
Revised Aug 21, 2018
Accepted Nov 18, 2018

Keywords:

Big Data
Cloud computing
Hadoop
MapReduce
Performance analysis

ABSTRACT

In the era of rapid growth of cloud computing, performance calculation of cloud service is an essential criterion to assure quality of service. Nevertheless, it is a perplexing task to effectively analyze the performance of cloud service due to the complexity of cloud resources and the diversity of Big Data applications. Hence, we propose to examine the performance of Big Data applications with Hadoop and thus to figure out the performance in cloud cluster. Hadoop is built based on MapReduce, one of the widely used programming models in Big Data. In this paper, the performance analysis of Hadoop MapReduce WordCount application for Twitter data is presented. A 4-node in-house Hadoop cluster was setup and experiment was carried out for analyzing the performance. Through this work, it was concluded that Hadoop is efficient for BigData applications with 3 or more nodes with replication factor 3. Also, it was observed that system time was relatively more compared to user time for BigData applications beyond 80GB. This experiment had also thrown certain pattern on actual data blocks used to process the WordCount application.

Copyright © 2019 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

P Ganesh,
Dept. of MCA,
BMSIT&M, India.
Email: pganesh@bmsit.in

1. INTRODUCTION

Today, all of us experience every one talking about cloud computing [1] and its associated services. Due to its flexibility in hosting services and high availability, cloud computing is in demand everywhere. The data intensive society is extensively relying on cloud for plethora of services that are supported by it [2]. In addition, extensive availability of internet services is a booster for cloud service demand. In every walk of life, today, enormous amounts of data are being generated (Weather forecasting, Transport, Super markets, Communication, Engineering Applications, Scientific Experiments, Online sales, Social networking etc.). Cloud network is used to store, process and retrieve this large data through high performance parallel storage systems. Cloud computing has had advanced as an effective model to process Big Data applications [3]. Hadoop [4] has been developed to process Big Data in cloud landscape. Hadoop is open source structure containing MapReduce [5], HDFS and HBase. MapReduce is the core component of Hadoop which process large amounts of datasets in parallel by distributing the work into a set of autonomous tasks. Hadoop Distributed File System (HDFS), is a highly reliable storage. Hadoop can be used to process large data, which uses parallel computing method to accelerate the speed of data processing and protect the data storage through maintaining multiple copies of the data [6]. At the same time, with the increase of user network, HDFS, Hadoop's flagship filesystem, can be horizontally scalable. Also, it is designed to be deployed on low-cost hardware. HDFS is capable of supporting high throughput, fault-tolerant, streaming data of very large files [7]. Name node and Data node combination in Hadoop, is designed to deal effectively with the large volumes of

data processing. Files in HDFS are divided into various number of blocks of data, with minimum size being 64 MB [8]. In our experiment, each block size was 128 MB. It is imperative to say that better the size of each block better will be the performance. The experiment with 4-node Hadoop cluster was setup to process Twitter data. R is widely used open environment for handling large-size data like Twitter. To extract required and sufficient large dataset from Twitter, R streaming API was used. Using Python, the extracted Twitter data was made into various chunks. The experiment was conducted and analyzed for Twitter data ranging from 10 GB to 100 GB, in multiples of 10 GB. Time command under Linux platform, provides three details of process execution namely real time, user time and system time. User time specifies the amount of time spent within the process or application in user mode. System time denotes the time spent in kernel within the process or application. Real time is the actual time spent by the process to complete its execution. In general, real time includes user time, system time, waiting time, if any and time spent for other associated activities. As real time depends on various activities of a process execution, user and system times only were considered in the analysis, as is obvious. The rest of the paper is organized as follows. In section II related research contributions are highlighted. In section III, the notion of Big Data is discussed while in section IV, the evolution of data at its current form with regard to its size is discussed. Section V discusses the need for Hadoop system in solving Big Data instances. Hadoop experimental environment is elaborated in section VI with its results and associated discussions are being explained in section VII.

2. RELATED WORK

MapReduce has been considered as a better and simplified approach for parallel and distributed processing of large data sets. Hadoop is one of the extensively used open source MapReduce implementation. Several authors have published material on Hadoop and its performance. In [9], author describes running Hadoop on Amazon. E.Vianna et al in [10] assesses Hadoop execution time through virtual machines. In [11], Gohil et al presents Hadoop performance through WordCount, TeraSort and other applications of MapReduce using Amazon EC2 instances and their analysis was carried out through an in-house Hadoop cluster setup. The authors concentration was mainly to reduce the network I/O latency and have a dedicated cluster. Mukhtaj Khan et al though [12] estimates the job completion time and resource provisioning. The work of Vladyslav Taran et al in [13] estimates the performance of distributed computing on the basis of Hadoop and Spark frameworks. In [14], Wenhui Lin et al, proposes a node performance measurement for Hadoop. Ankita Jain et al [15] presents an approach to improve Hadoop performance by proper tuning of MapReduce configuration parameters. The need for more analysis of high performance of big data applications was reiterated by Satish Londhe et al in [16]. Sachin Arun Thanekar et al [17] highlights the advantages of Hadoop in processing the high volimes of data through comparison of Hadoop and RDBMS. In [18], Archana R. A. et al proposes a data security system for big data applications through masking technique. All their publications were focused around different aspects of analyzing Hadoop performance. Current work of ours supplements performance analysis for Hadoop.

3. BIG DATA

Data are everywhere and we live in the Data age [19]. The journey of data explosion has just begun in the recent past. As per the survey the size of the digital universe will be 44 zettabytes by 2020. This flood of data is coming from many sources like stock markets, air travel, social media, sales, healthcare, governance among others. Also, data can be structured, unstructured, roughly correlated, discrete or disjoint. It is not ending here. The data tends to expand rapidly in future. But how this data is being put into use, is the challenge, as Big Data has become a household term today. The analysis of Big Data leads to interesting benefits, conclusions, inventions, relationships, business opportunities among others [20]. As the data gets piled, managing and analyzing it in the optimal manner become more critical to leverage its benefits. Due to the various hidden advantages of Big Data, analytics is being referred everywhere and obviously embedded into our daily lives. The significance, prominence and effect of analytics are now higher than ever before and as more and more data are being collected and that there is considerable value in knowing what is hidden in data, analytics will continue to develop. Cloud plays a key role in the Big Data world, by providing horizontally expandable and optimized infrastructure that supports practical implementation of Big Data. Figure 1 shows the Big Data analytics process model. Initially, a detailed description of the problem-in-hand to be solved with analytics is desired. Next, all source data need to be identified that could be of potential interest. All the relevant data is later gathered. This is followed by a data cleaning step to get rid of all inconsistencies, such as missing values, duplicate data etc. During analytics phase, an analytical model is estimated on the pre-processed and transformed data. Finally, once the model is built, it can be interpreted and evaluated for usage.

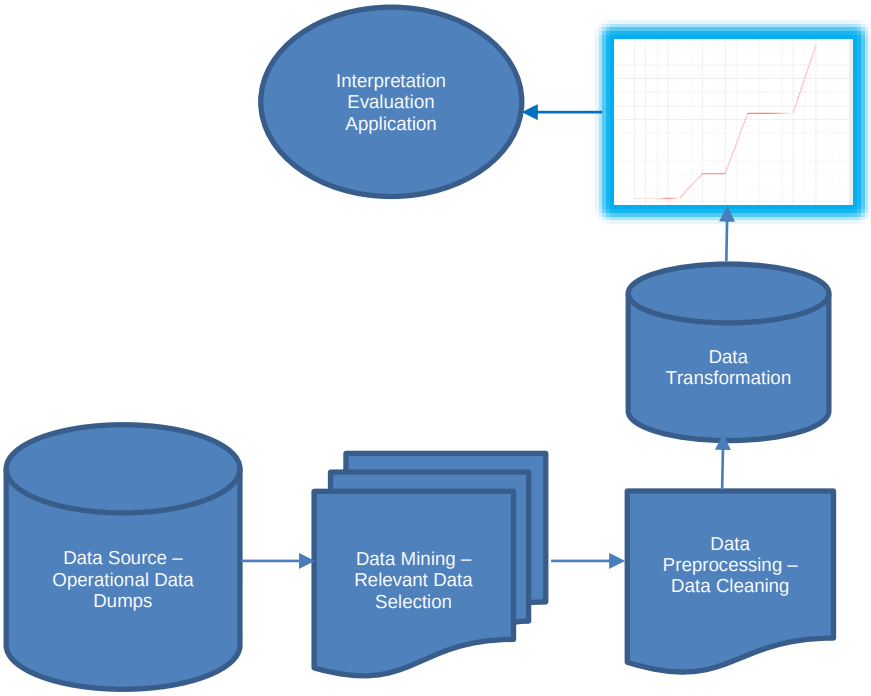


Figure 1. Big Data Analytics Process Model

4. DATA SIZE

Data in order to qualify to become Big Data needs to satisfy three characteristics as per Gartner [21]. They include Volume, Velocity and Variety. Volume specifies the amount of data being generated. In general, Big Data is associated with this aspect majorly. Velocity refers to the frequency with which the data is generated or used. Variety identifies the diverse collection of data, not specified to a specific format or type. Also Big Data size ranges from several GB to several billions of GB. Hadoop, as designed, effectively deals with huge volumes of Hadoop, one need to specify the given task as MapReduce job. As such, in the data ranging from hundreds of GB to millions of GB. To take advantage of parallel processing which is facilitated by experiment, about 158.8 GB of Twitter dataset comprising variety of tweets, over a given time period, was extracted. The same dataset was processed using WordCount, a MapReduce application. Figure 2 highlights on how the data size growth is influenced by various activities.

Figure 2. Big Data growth prediction

5. HADOOP (HDFS)

When non-uniform data develops to large dimensions, a distributed approach to analyze such data needs to be considered [22]. MapReduce has emerged as the optimal model of choice for handling "Big Data" problems. MapReduce frameworks such as Hadoop offer both storage and processing capabilities for all sorts of data. MapReduce begins with the notion of splitting an input dataset over a set of dedicated machines, called workers, and processes these data splits in parallel with user-defined map and reduce functions. MapReduce abstracts away from the users the details of data distribution, parallelization, scheduling and fault tolerance. Various phases of MapReduce job execution are Input splits, Mapper, Combiner, Partitioner, Shuffling, Sorting, Reducer. Figure 3 shows the Hadoop (MapReduce) job execution flow of map and reduce tasks.

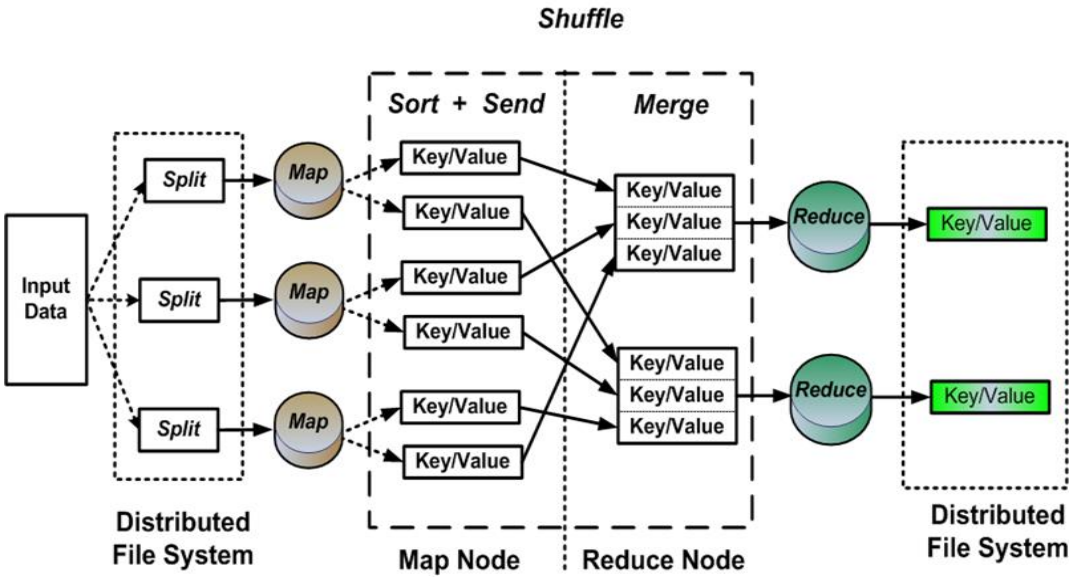


Figure 3. Hadoop (MapReduce) Job execution flow

Apache Hadoop is the leading open source MapReduce implementation to solve Big Data problems. Hadoop is built around two central components, namely, HDFS and the Hadoop MapReduce Framework. They perform data management and job execution responsibilities respectively. HDFS is designed with a goal to provide a quick and automatic fault-recovery. The strategy of HDFS architecture is to store and retrieve a huge amount of data. HDFS always wants to work with huge data sets. A MapReduce-based application seamlessly suits in this model. An HDFS cluster has two types of nodes operating in a master-worker pattern, namely, *namenode* (master) and a number of *datanodes* (workers). Tasks of namenode include managing file system namespace, regulating client's access to files, ensuring data nodes are alive and maintaining replicas as per replication factor. Tasks of datanode involve creation and deletion of blocks as per replication factor, managing data storage, communicating to namenode on the health of HDFS through heartbeats.

HDFS architecture is portrayed in figure.4. A Hadoop JobTracker, available on the master node is accountable for resolving job details such as mappers/reducers, monitoring the job progress and worker status [23]. When a dataset is placed into the HDFS, it is split into a number of data chunks and distributed throughout the cluster. Each worker, hosting a data split, runs a process called datanode. A TaskTracker is responsible for processing the data splits of the local datanode. A client accesses the filesystem by communicating with the namenode and datanodes. HDFS stores files as blocks and distributes them across the entire cluster. Files in HDFS are divided into various number of blocks of data, with minimum size being 64 MB. The block size can be any other (better) value than 64MB. Better the size of each block better will be the performance. To ensure high availability and fault-tolerance, blocks are replicated sufficiently numerous times. As a default, replication factor for a block is set to 3. It is witnessed that changing the block size and replication factor, affects the Hadoop performance [24]. However, as a rule of thumb, the replication factor should not be more than the number of nodes in the cluster. HDFS is designed with the property to be portable from one platform to other. This facilitates widespread acceptance of HDFS. Hadoop offers performance enrichments consenting for high throughput. Hadoop processes larger data sets at affordable costs, provides hardware fault-tolerance, ensures high availability of services among others.

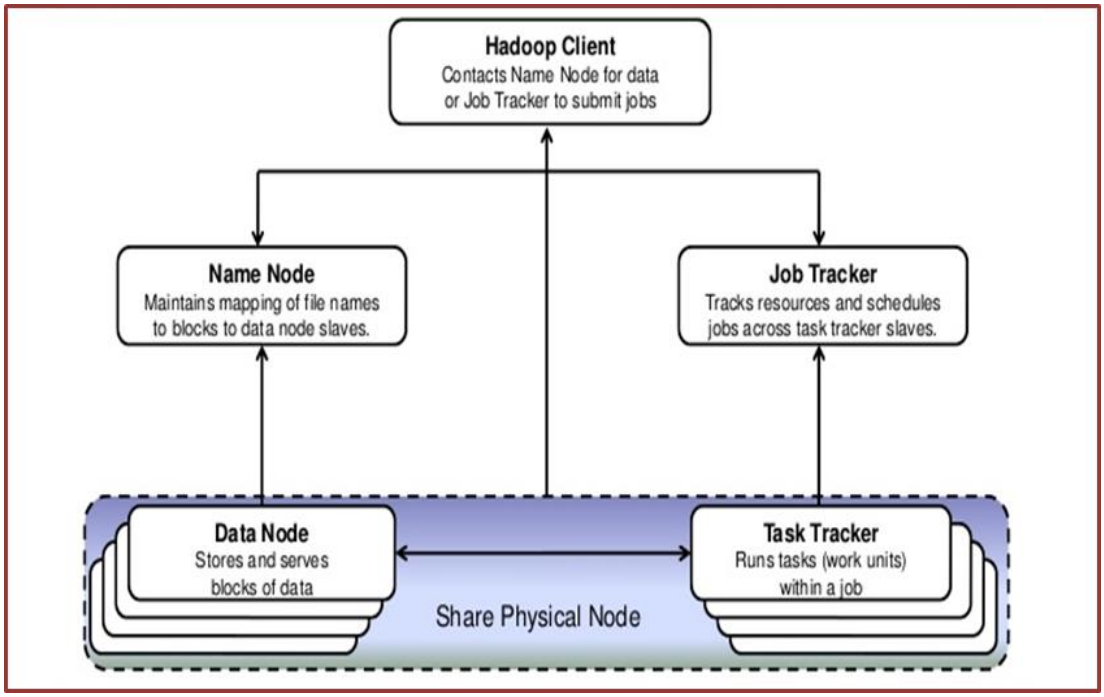


Figure 4. HDFS Architecture

6. EXPERIMENTAL ENVIRONMENT

The Hadoop cluster environment was set up and used for analysis as detailed:

6.1. Cluster setup

An in-house Hadoop cluster with four nodes (1 master and 3 slaves) using Intel Core i5 machines was setup as depicted in figure.5. The specifications and configurations of the cluster are listed in Table1. All the four nodes were with similar specifications and configurations. The stages involved during cluster installation and configuration are presented below. The master node was also considered as namenode and the remaining 3 nodes as datanodes. The namenode was also used as a datanode. The data block size of the HDFS, was 128 MB. Replication factor for data blocks was set accordingly during the course of experiment. It was set as 1 for 1node, 2 for 2nodes, 3 for 3 nodes and 3 for 4nodes. The WordCount application was analysed with Twitter datasets for the datasets ranging from 10GB to 100GB.

Table 1. Experimental setup of Hadoop Cluster

Intel Core i5 Nodes	CPU	4 Cores
	Processor	3.30 GHz
	Hard Disk	500 GB
	Connectivity	Gigabit Ethernet LAN
	Memory	8 GB
Software	Operating System	Ubuntu 16.04 LTS
	JDK	1.8.0_171
	Hadoop	2.7.6

Step1: Each node was installed with Ubuntu 16.04
Step2: JDK 1.8.0_171 was installed on each node
Step3: Hadoop-2.7.6 was installed on all nodes
Step4: Nodes were classified as Master and Slaves and configured accordingly
Step5: Configure **ssh** and establish the connection among all 4 nodes

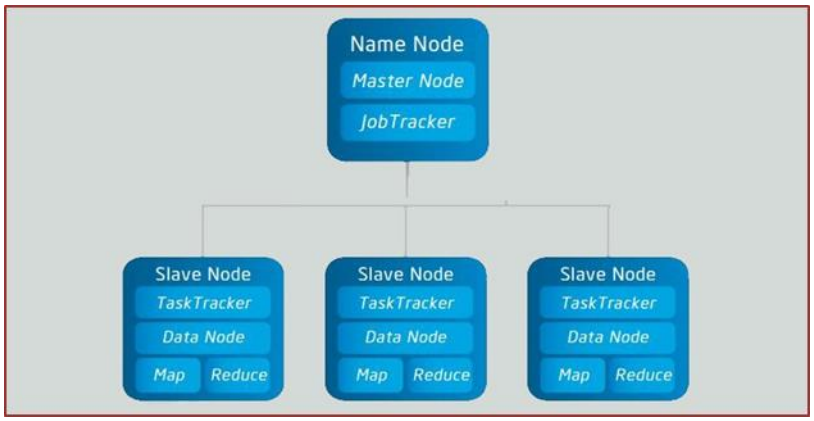


Figure 5. Hadoop cluster with 4 nodes (1Master and 3 Slaves)

6.2. Data gathering

Members of Twitter keep generating millions of tweets every second on average. These tweets are from different geography, pertain to various contexts, refer variety of categories. Big Data actually refers to data that are larger in size, have variety of details and generated frequently. Twitter data fulfils all these characteristics. Hence, Twitter dataset up to 158.8 GB, was extracted in one stretch during the period from 17th January 2017 to 11th February 2017 as shown in the figure.6. The tweets extracted in this manner was used in the experiment for analysis of Hadoop's performance. The process used to extract Twitter data was as below:

Step1: Twitter account created with valid credentials
Step2: R packages from Rstudio were installed
Step3: An R application to extract tweets was created
Step4: Post successful authentication, tweets were extracted over the continuous period of 26 days
Step5: The extracted tweets were stored into .csv file

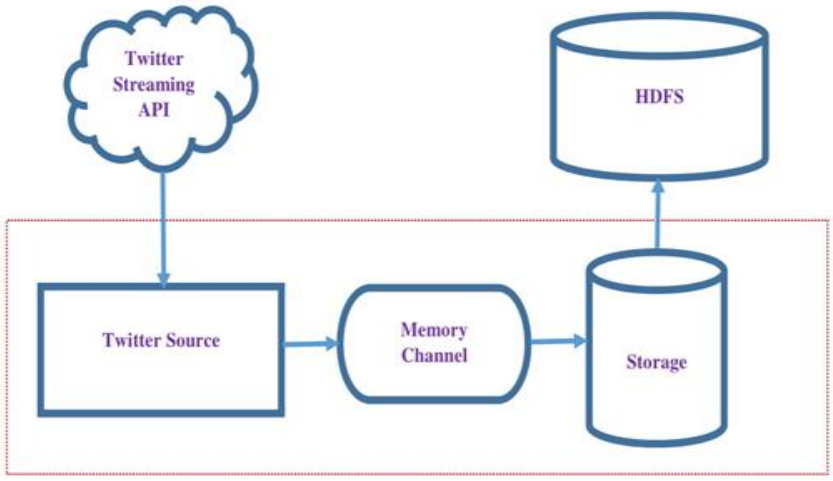


Figure 6. Twitter tweets extraction

6.3. Data pre-processing

Dataset compiled from the Twitter was a combination of text, images, audio files, video files. Using Python script, individual data chunks containing text data ranging from10 GB till 100 GB were created. Figure.7 contains the Python script for chunks creation. The main reason behind creating chunks from 10 GB to 100 GB was to compare the performance of Hadoop with less data size and Big Data size. In addition, these range of chunks help us to analyze the performance with a certain scale of uniform data size.

```
import sys

gb=sys.argv[1]
filename=sys.argv[2]

bytes = 1024*1024*1024*int(gb)

print "Creating ", gb,"GB chunk with filename ",filename

with open("tweets5.csv","rt") as f:
        with open(filename,"wt") as g:
                size=0
                i=1
                for line_no, line in enumerate(f):
                        if size + len(line) <= bytes:
                                g.write(line)
                                size = size + len(line)
                        elif size + len(line) > bytes:
                                strln=bytes-size
                                g.write(line[0:strln])
                                print "Done"
                                exit(0)
                        if size >= (bytes*(i*0.1)):
                                print i*10, "%"
                                i=i+1
```

Figure 7. Python script for Twitter data chunks creation

6.4. Performance analysis using MapReduce

The experiment was conducted using 1node, 2nodes, 3nodes and 4nodes. MapReduce framework of Hadoop was used with a simple, single and standalone WordCount application. While running this application no other application was running to ensure proper execution of the WordCount program with less latency. WordCount application is meant for counting the number of similar words and their occurrence in the given input file. The output of the WordCount is a file with relevant data for analysis. The current experiment was conducted to gather the performance statistics of the WordCount application execution for various data sizes. By combining time command with WodCount program, various time stamps of the data processing were gathered. The response time for loading and execution of each dataset was noted and tabulated. A sample workflow [25] of WordCount application is shown in Figure 8.

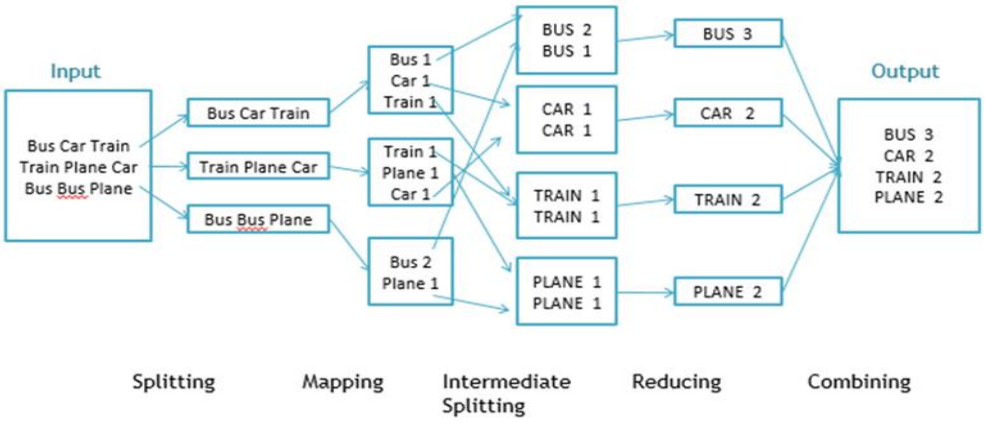


Figure 8. WordCount application workflow using MapReduce

7. RESULTS AND DISCUSSION

As an initial step to start the Hadoop cluster, name nodes were formatted. HDFS and YARN services were initiated and confirmed for their availability. The required datasets were loaded into HDFS. The WordCount application was run considering 1node, 2nodes, 3nodes and 4nodes repeatedly, for all the proposed datasets, as mentioned above. Throughout this experiment the replication factor (RF) was kept changing as per the rule of thumb and HDFS default replication notion. It was set as 1 for 1node, set as 2 for 2 nodes and set as 3 for both 3 nodes and 4 nodes. Table 2 and Table 3 provides the details of time stamps under varying replication factors along with varying datasets considering different node specifications. For every dataset, under each node trial, the user time, system time, real time and response time of the WordCount application were obtained using *time* command. The user, system and response time of various input datasets under 1node and 2nodes are detailed inTable 2. The user, system and response time of various input datasets under 3nodes

and 4nodes are detailed in Table 3. Table 2 contains the total user, system and response time spent for both loading and executing the various datasets, with 1node and 2nodes. Table3 contains the total user, system and response time spent for both loading and executing the various datasets, with 3nodes and 4nodes.

Table 2. Response times-1node and 2nodes

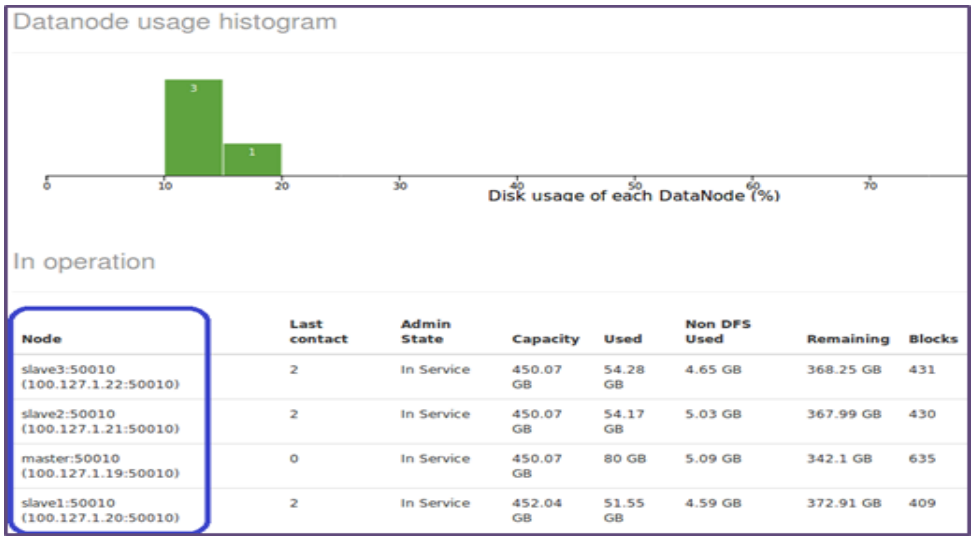
Time in Sec - With Loading and Execution – 1node and 2nodes					
Size	RF	Nodes	User Time-sec	System Time-sec	Total Response Time-sec
10 GB	1	1node	493.260	67.372	560.632
	2	2nodes	496.488	70.824	567.312
20 GB	1	1node	1049.136	158.388	1207.524
	2	2nodes	1018.488	157.300	1175.788
30 GB	1	1node	1565.604	241.816	1807.420
	2	2nodes	1568.484	250.312	1818.796
40 GB	1	1node	2146.484	347.296	2493.780
	2	2nodes	2132.420	352.872	2485.292
50 GB	1	1node	2688.968	438.920	3127.888
	2	2nodes	2668.156	453.100	3121.256
60 GB	1	1node	3229.856	537.168	3767.024
	2	2nodes	3280.384	578.888	3859.272
70 GB	1	1node	3766.040	648.496	4414.536
	2	2nodes	3745.776	683.488	4429.264
80 GB	1	1node	4357.408	751.204	5108.612
	2	2nodes	4358.792	765.552	5124.344
90 GB	1	1node	4890.358	851.204	5741.562
	2	2nodes	4888.920	899.108	5788.028
100 GB	1	1node	5421.676	976.860	6398.536
	2	2nodes	5443.860	981.788	6425.648

Table 3. Response times-3nodes and 4nodes

Time in Sec - With Loading and Execution – 3nodes and 4nodes					
Size	RF	Nodes	User Time-sec	System Time-sec	Total Response Time-sec
10 GB	3	3nodes	503.752	69.744	573.496
	3	4nodes	489.880	68.216	558.096
20 GB	3	3nodes	1040.032	163.556	1203.588
	3	4nodes	1017.012	155.700	1172.712
30 GB	3	3nodes	1574.636	249.504	1824.140
	3	4nodes	1569.600	243.852	1813.452
40 GB	3	3nodes	2116.860	356.568	2473.428
	3	4nodes	2111.840	349.744	2461.584
50 GB	3	3nodes	2689.428	455.608	3145.036
	3	4nodes	2658.112	453.988	3112.100
60 GB	3	3nodes	3248.084	559.948	3808.032
	3	4nodes	3236.100	562.072	3798.172
70 GB	3	3nodes	3776.560	675.068	4451.628
	3	4nodes	3759.456	661.392	4420.848
80 GB	3	3nodes	4378.448	773.124	5151.572
	3	4nodes	4327.928	774.920	5102.848
90 GB	3	3nodes	4941.120	902.652	5843.772
	3	4nodes	4923.052	872.664	5795.716
100 GB	3	3nodes	5517.660	999.208	6516.868
	3	4nodes	5477.084	1009.264	6486.348

7.1. Number of nodes

From the timestamp details gathered in Table 2, it is prudent that the total response time with 1node are less compared to total response time with 2nodes for most of the data sizes. This is obvious due to the fact that multiple read/write operations under 1node are quite less compared to the read/write operations under 2nodes. It can be recalled that Hadoop is designed to work efficiently for parallel and distributed processing. At the same time, as seen in Table 3, the total response time obtained with 3nodes is more than the total response time obtained with 4nodes. As per Hadoop architecture and design principle, Big Data problems are solved better with 3nodes and above. Thus, it is witnessed that the total response time is better and consistent with 3nodes and 4nodes, as is expected. This difference in response time between 3nodes and 4nodes is quite evident, practical and reasonable. Hence more the number of nodes, better performance from Hadoop can be expected. A snapshot of datanodes and their usage to process 50 GB data with 4nodes is shown in figure.9. Henceforth, the discussion and analysis of performance in the remaining part of the paper will be based on 3nodes and 4nodes.



Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks
slave3:50010 (100.127.1.22:50010)	2	In Service	450.07 GB	54.28 GB	4.65 GB	368.25 GB	431
slave2:50010 (100.127.1.21:50010)	2	In Service	450.07 GB	54.17 GB	5.03 GB	367.99 GB	430
master:50010 (100.127.1.19:50010)	0	In Service	450.07 GB	80 GB	5.09 GB	342.1 GB	635
slave1:50010 (100.127.1.20:50010)	2	In Service	452.04 GB	51.55 GB	4.59 GB	372.91 GB	409

Figure 9. Hadoop datanode usage with 4nodes to process 50GB data

7.2. Data Size

From Table 3, it is evident that the total response time for processing the data is less with 4nodes compared to processing of the data with 3nodes. This result is expected in Hadoop as it is designed to distribute the task and process the data in an efficient manner. Also, beyond 70 GB, it may be observed that the difference in total response time between 3nodes and 4nodes is significant and obvious. This is where it can be concluded that our Hadoop model is efficient in processing large data size applications. Thus a conclusion can be drawn that larger the data size better will be the Hadoop's performance and Hadoop is best suited for processing Big Data. The total response time, depicted in Figure 10, corresponds to the above discussion. Here the total response time with 4nodes, as shown with red line, is significantly less than the total response time with 3nodes, as shown in green line, for the Big Data sets.

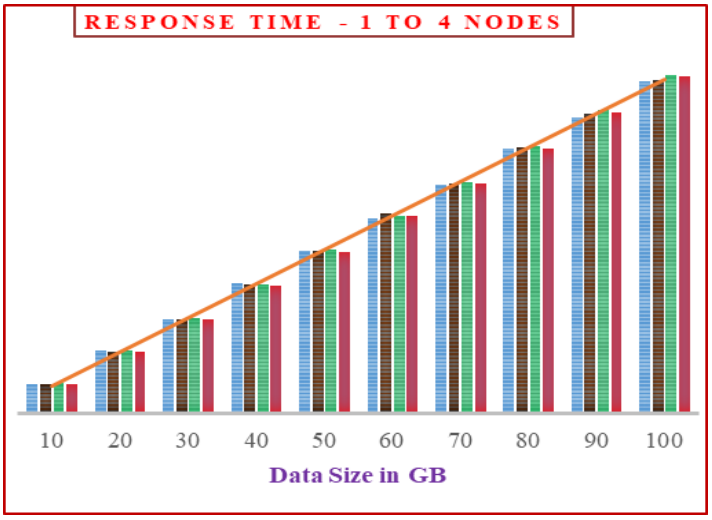


Figure 10. Graph for total response time – 1node, 2nodes, 3nodes and 4nodes

7.3. Replication Factor

Replication factor ensures fault-tolerance, in case of data damage while processing. This value enables HDFS to keep corresponding number of replicas of input data in the available data nodes. In Hadoop literature it is recommended to have replication factor of 3 for all Big Data applications in order to guarantee better performance. However, the rule of thumb points to have replication factor not more than the number of nodes.

From the Table 2, where the replication factor is less than 3, for 1node and 2nodes, it is evident that the performance is not satisfactory and the result is as expected. At the same time, as seen in Table 3, the replication factor 3 for 3nodes and 4nodes, provides expected performance for Big Data applications. The screen shots for replication factor 2 with 90GB dataset and replication factor 3 with 50GB dataset are shown in Figure 11 and Figure 12, respectively. It is concluded that maintaining the default replication factor 3, is always recommended.

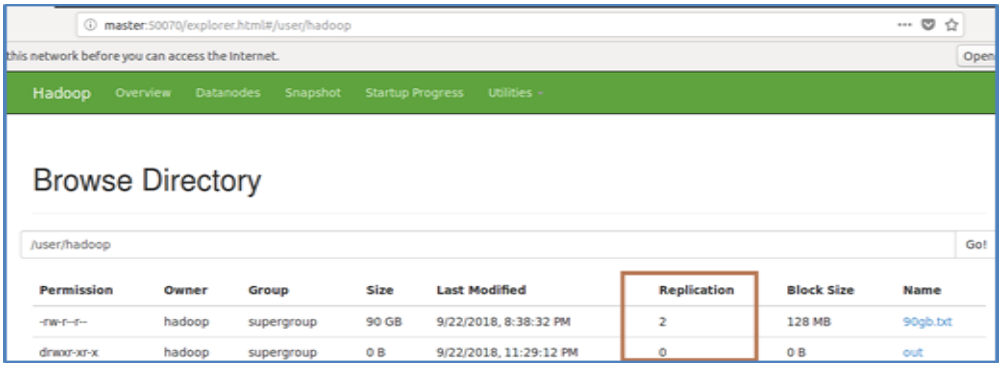


Figure 11. Replication factor 2 with 2nodes for 90GB dataset

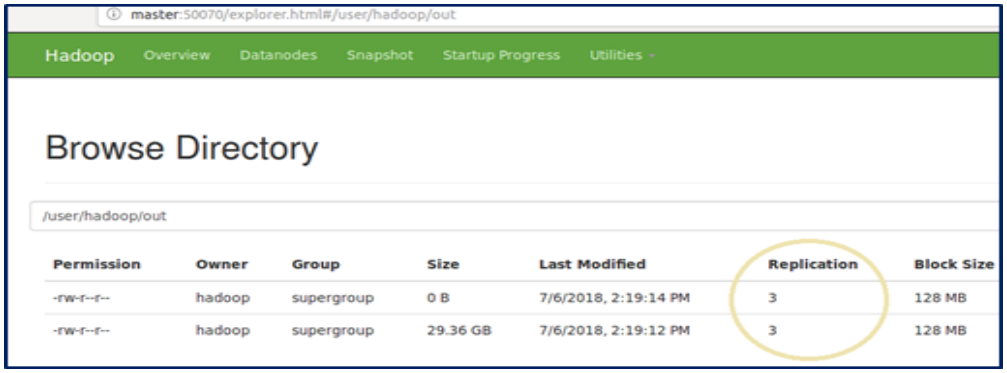
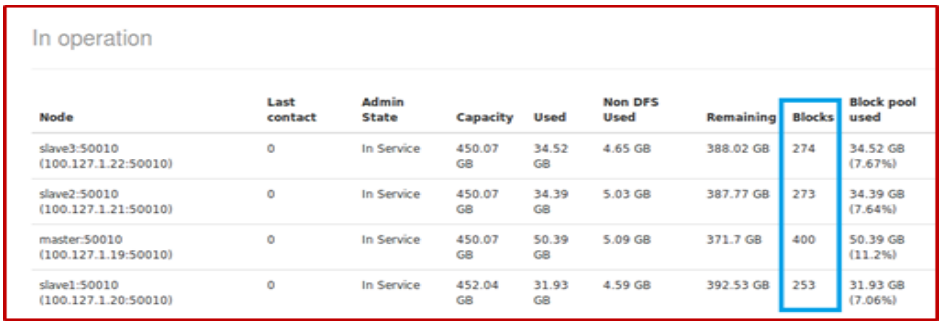


Figure 12. Replication factor 3 with 4nodes for 50GB dataset

7.4. Data Blocks

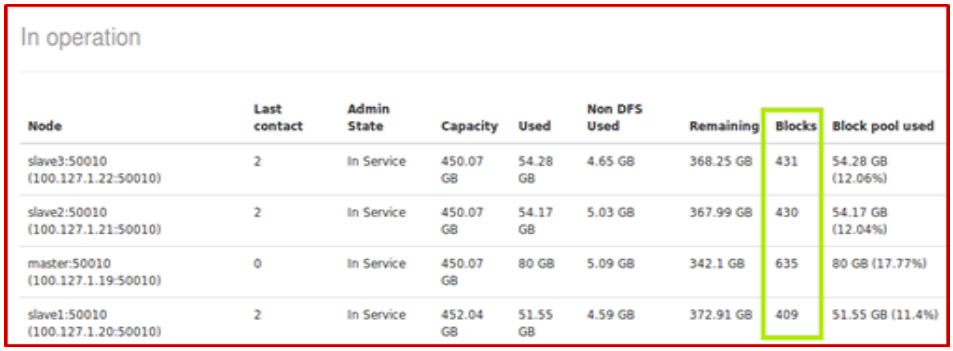
Each HDFS data block was set at 128MB throughout the experiment. Replication factors were suitably changed for 1node, 2nodes, 3nodes and 4nodes as already discussed. As per the value of replication, the input dataset was loaded by HDFS into corresponding datanodes. For example, 50GB data requires 420 data blocks to store. With replication factor 3, approximately we need 1260 number of total data blocks to store 50GB. In reality, to store 50GB dataset, 1200 data blocks were used while 1905 number of blocks were used to process WordCount of this dataset. Like manner, at the end of each dataset processing, the number of blocks utilized to process the WordCount application with varying replication factors was gathered as shown in Table 4. Some interesting details were found from this table. As a sample, the number of data blocks used for loading 50GB dataset with 4 nodes is presented in Figure 13. Corresponding to it, the actual number of data blocks used for processing are shown in Figure 14. Initially, from the Table 4, no pattern was observed in the distribution of used data blocks in processing the datasets from 10GB to 30GB. Between 40GB to 60GB, certain pattern in the distribution of used data blocks was witnessed across various nodes. Beyond 70GB, it can be clearly pointed out that the distribution of data blocks was based on a constant factor across the nodes. This constant factor was 1.58 with 1node, 3.16 with 2nodes, 4.74 with 3nodes and 4.74 again with 4nodes. It is observed that the value of this constant factor distribution is 50% over and above the value of replication factor. This pattern of data block distribution over and above the replication factor can be attributed to the Hadoop's design principle of ensuring high availability and fault-tolerance. Thus it strongly asserts the finding that beyond 70GB input, our model is quite reliable and predictable for its performance in all respects (with replication factor 3 and number of nodes are at least 3).



In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used
slave3:50010 (100.127.1.22:50010)	0	In Service	450.07 GB	34.52 GB	4.65 GB	388.02 GB	274	34.52 GB (7.67%)
slave2:50010 (100.127.1.21:50010)	0	In Service	450.07 GB	34.39 GB	5.03 GB	387.77 GB	273	34.39 GB (7.64%)
master:50010 (100.127.1.19:50010)	0	In Service	450.07 GB	50.39 GB	5.09 GB	371.7 GB	400	50.39 GB (11.2%)
slave1:50010 (100.127.1.20:50010)	0	In Service	452.04 GB	31.93 GB	4.59 GB	392.53 GB	253	31.93 GB (7.06%)

Figure 13. Data blocks to load 50GB dataset with 4nodes-1200 blocks



In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used
slave3:50010 (100.127.1.22:50010)	2	In Service	450.07 GB	54.28 GB	4.65 GB	368.25 GB	431	54.28 GB (12.06%)
slave2:50010 (100.127.1.21:50010)	2	In Service	450.07 GB	54.17 GB	5.03 GB	367.99 GB	430	54.17 GB (12.04%)
master:50010 (100.127.1.19:50010)	0	In Service	450.07 GB	80 GB	5.09 GB	342.1 GB	635	80 GB (17.77%)
slave1:50010 (100.127.1.20:50010)	2	In Service	452.04 GB	51.55 GB	4.59 GB	372.91 GB	409	51.55 GB (11.4%)

Figure 14. Data blocks to process 50GB dataset with 4nodes –1905 blocks

Table 4. Data blocks utilized for various datasets with varying replication factor

RF	No. of Nodes	Size	Input File Data Blocks	No. of Blocks Used	Factor	Response Time per block in sec	Size	Input File Data Blocks	No. of Blocks Used	Factor	Response Time per block in sec
1	1		80	130	1.63	4.31		480	760	1.58	4.96
2	2	10	80	260	3.25	2.18	60	480	1520	3.17	2.54
3	3	GB	80	390	4.88	1.47	GB	480	2280	4.75	1.67
3	4		80	240	3.00	2.33		480	2280	4.75	1.67
1	1		160	257	1.61	4.70		560	884	1.58	4.99
2	2	20	160	514	3.21	2.29	70	560	1768	3.16	2.51
3	3	GB	160	771	4.82	1.56	GB	560	2652	4.74	1.68
3	4		160	480	3.00	2.44		560	2652	4.74	1.67
1	1		240	383	1.60	4.72		640	1011	1.58	5.05
2	2	30	240	766	3.19	2.37	80	640	2020	3.16	2.54
3	3	GB	240	1149	4.79	1.59	GB	640	3030	4.74	1.70
3	4		240	1149	4.79	1.58		640	3030	4.74	1.68
1	1		320	495	1.55	5.04		720	1141	1.58	5.03
2	2	40	320	990	3.09	2.51	90	720	2274	3.16	2.55
3	3	GB	320	1485	4.64	1.67	GB	720	3411	4.74	1.71
3	4		320	1485	4.64	1.66		720	3411	4.74	1.70
1	1		400	635	1.59	4.93		800	1264	1.58	5.06
2	2	50	400	1270	3.18	2.46	100	800	2528	3.16	2.54
3	3	GB	400	1905	4.76	1.65	GB	800	3792	4.74	1.72
3	4		400	1905	4.76	1.63		800	3792	4.74	1.71

7.5. System Time, User Time and Total Response Time

Through this experiment, it was tried to analyze the relation between user time, system time and total response time across various nodes with varying datasets. It was observed that during the processing of datasets from 10GB to 50GB, the user time was 6 times more than the system time, on an average. From 60GB onwards, user time to system time ratio was 5.5, on an average. As such, a phenomenon can be observed between user time and system time. As the data size grows, the ratio between user time and system time decreases. In other words, when the data size grows, relatively more system time is consumed compared to user time. This phenomenon is observed across all data sizes and across different nodes. Also it was observed that 76MB of

data was processed, on an average, in one second of total response time with 3nodes and 4nodes. In Figures 15 and 16, this phenomenon is portrayed for 60GB and 100GB respectively. Initially, up to 50 GB, the user time increased proportionately while system time decreased. Beyond 60 GB, more precisely from 80 GB, it can be observed that the user time decreased while system time increased proportionately. It is assumed that this scenario might be due to the fact that Big Data application processing for greater input data sizes require more system time than user time. From the Table 4, it can also be observed that the total response time per data block of processing is substantially less with 3nodes and 4nodes in comparison to 1node and 2nodes. Also negligible difference in total response time to process each data block is observed between 3nodes and 4nodes, though the time per block is less with 4nodes. This again re-affirms that Hadoop is quite efficient for Big Data applications beyond 3nodes.

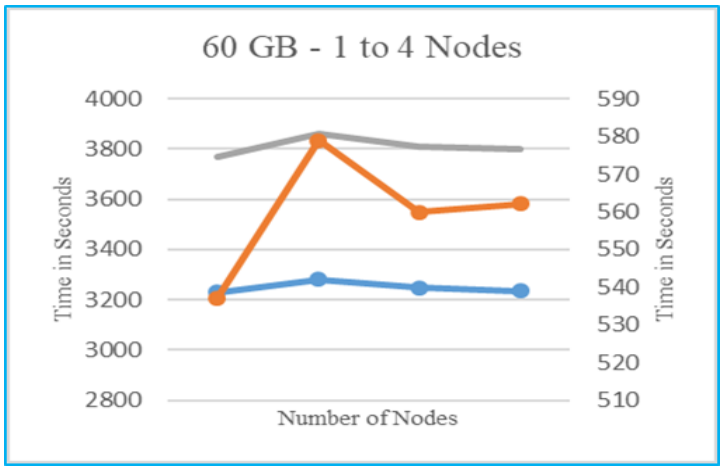


Figure 15. User vs System Time – 1node, 2nodes, 3nodes, 4 nodes – 60GB

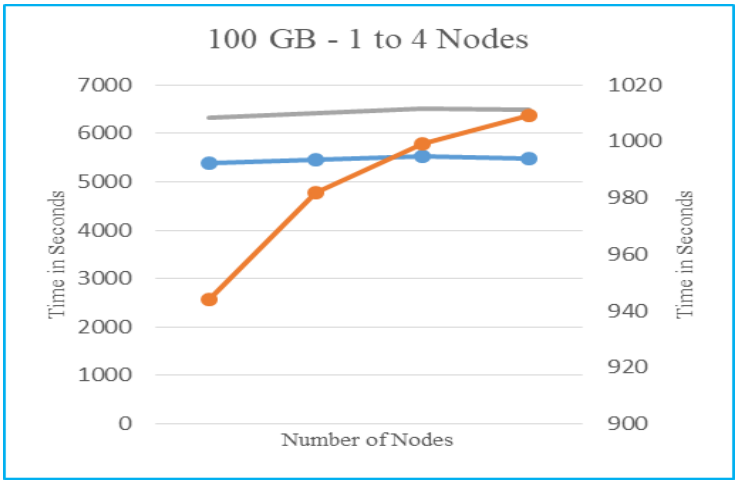


Figure 16. User vs System Time – 1node, 2nodes, 3nodes, 4 nodes – 100GB

8. CONCLUSION

Hadoop is one of the best open source MapReduce implementation to solve Big Data applications. This got confirmed through the performance analysis of an in-house 4-node cluster. Through the experiment total response time with 1node, 2nodes, 3nodes and 4nodes were observed independently for Twitter data sizes from 10GB to 100GB. It is witnessed that Hadoop is quite efficient for data sizes above 70GB. Also the replication factor plays an important role in the Hadoop performance. With replication factor 3, better performance was observed compared to replication factor 1 or 2. Another phenomenon with regard to user time and system time was observed where relatively more system time was consumed compared to user time while processing Big Data size applications. Though the number of data blocks used to store the input was as per the

value of replication factor, actual number of data blocks utilized to complete the processing was found to be 50% more than the required blocks. It was observed that the response time to process each block was significantly less with 3 or more nodes. Hence through this model it can be concluded that more the number of nodes performance was better with larger data size applications. As an extension work, we wish to analyze the performance with Spark environment and compare the results under common environment.

REFERENCES

[1] Armbrust M, Fox A, Griffith R, et al. "A view of cloud computing". *Communications of the ACM*. 2010; 53(4) : 50-58.

[2] Joseph A. Issa. "Performance Evaluation and Estimation Model Using Regression Method for Hadoop Word Count". *IEEE Access*. 2015; vol. 3: pp.2784-2793. DOI: 10.1109/ACCESS.2015.2509598

[3] Chao Shen et al. "Performance modeling of Big Data applications in the cloud centers". *Springer Journal of Super Computing.* 2017. DOI 10.1007/s11227-017-2005-y

[4] "Apache Hadoop." [Online]. http://hadoop.apache.org/ (Accessed during June 2018)

[5] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". *Communication, ACM*. 2008; 51(1) : 107–113.

[6] Tom White. Hadoop Definitive Guide. O'Reilly. 2012:pp74-97.

[7] Shvachko K, Hairong Kuang, Radia S, Chansler R. "The Hadoop Distributed File System". *In Proc. of the Mass Storage Systems and Technologies (MSST)*. IEEE 26th Symposium. 2010; pp.1 – 10.

[8] T Lakshmi Siva Rama Krishna, Dr T Ragunathan, Sudheer Kumar Battula. "Performance Evaluation of Read and Write Operations in Hadoop Distributed File System". *IEEE Sixth International Symposium on Parallel Architectures, Algorithms and Programming*. 2014. DOI 10.1109/PAAP.2014.49

[9] T. White. Running Hadoop MapReduce on Amazon EC2 and Amazon S3. 2007. [Online]. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=873&categoryID=112 (Accessed during June 2018)

[10] E. Vianna et al."Modeling the performance of the Hadoop online prototype". *Proc. 23rd Int. Symp. Comput. Archit. High Perform. Comput. (SBACPAD)*. 2011; pp.152-159. DOI**:** 10.1109/SBAC-PAD.2011.24

[11] P. Gohil, D. Garg, and B. Panchal. "A performance analysis of MapReduce applications on Big Data in cloud based Hadoop". *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*. 2014; pp. 1-6.

[12] Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang. "Hadoop Performance Modeling for Job Estimation and Resource Provisioning". *IEEE Transactions on Parallel and Distributed Systems*. 2015. DOI 10.1109/TPDS.2015.2405552

[13] Vladyslav Taran, Oleg Alienin, Sergii Stirenko, and Yuri Gordienko. "Performance Evaluation of Distributed Computing Environments with Hadoop and Spark Frameworks". *IEEE International Young Scientists Forum on Applied Physics and Engineering*. 2017; pp 80-83.

[14] Wenhui Lin and Jun Liu. "Performance Analysis of MapReduce Program in Heterogeneous Cloud Computing". *Journal of Networks*. 2013; 8 (8):1734-1741. DOI:10.4304/jnw.8.8

[15] Ankita Jain, Monika Choudhary. "Analyzing and Optimizing Hadoop Performance". *IEEE International Conference On Big Data Analytics and computational Intelligence (ICBDACI)*. 2017

[16] Satish Londhe, Smita Mahajan. "Effective and Efficient Way of Reduce Dependency on Dataset with the Help of Mapreduce on Big Data". *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2015; 15 (1):171–176. DOI: 10.11591/telkomnika.v15i1.8080

[17] Sachin Arun Thanekar, K. Subrahmanyam, A.B. Bagwan. "A Study on MapReduce: Challenges and Trends". *Indonesian Journal of Electrical Engineering and Computer Science*. 2016; 4(1):176-183. DOI: 10.11591/ijeecs.v4.i1.pp176-183

[18] Archana RA, Ravindra S Hegadi, Manjunath TN. "A Big Data Security using Data Masking Methods". *Indonesian Journal of Electrical Engineering and Computer Science*. 2017; 7(2):449-456. DOI: 10.11591/ijeecs.v7.i2.pp449-456

[19] Bart Baesens. "Analytics in a Big Data World: The Essential Guide to Data Science and Its Applications". Wiley. 2004

[20] Ambiga Dhiraj, Michael Minelli and Michele Chambers. "Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses". Wiley CIO Series. 2013

[21] http://www.gartner.com (Accessed on 18th June 2018)

[22] E. Dede, B. Sendir, P. Kuzlu, J. Weachock, M. Govindaraju, and L. Ramakrishnan. "Processing Cassandra Datasets with Hadoop-Streaming Based Approaches". *IEEE Transactions on Services Computing*. 2016; 9 (1).

[23] Dali Ismail, Steven Harris. "Performance Comparison of Big Data Analysis using Hadoop in Physical and Virtual Servers". 2013. (Online) www.cse.wustl.edu/~jain/cse570-13/ftp /bigdatap/ index.html

[24] www.coursera.org/learn/hadoop (Online)) (Accessed during 18th to 24th June 2018)

[25] https://dzone.com/articles/word-count-hello-word-program-in-mapreduce (Online) (Accessed during 21st to 27th June 2018)