TEST Engineering & Management

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

Parametric Analysis of Resource Management in HDFS

[1]Mithun B N, [2]Viswanatha K V, [3]Manjunath T N, [4]Prabhuram

[1]Research Scholar, CMR University, Asst. Prof, CHRIST (Deemed to be University), Bengaluru, India. kanmith@gmail.com

[2]Professor, CMR University School of Engineering and Technology, Bengaluru India, viswanathakv@yahoo.com

[3]Professor and Dean, BMS Institute of Technology and Management, Bengaluru, India. manju.tn@gmail.com

[4]Principal Test Lead Consultant, Numerify, Bengaluru India, prabhuram.sridharan@numerify.com

Article Info
Volume 83
Page Number: 170 - 179
Publication Issue:
May - June 2020

Article History
Article Received*: 11August 2019*
Revised*: 18November 2019*
Accepted*: 23January 2020*
Publication*:07May2020*

Abstract:
Data has become an integral part of human life in the modern world. As there is increase in volume and different varieties of data available, big data domain has become a powerful area. Big data uses more resources for the processing. This paper focuses on resources utilized in big data environment to perform the operations. It uses various resources like memory, files and network bandwidth. Hadoop framework handles data distributed over cluster.Main concentrationof this paper is on the usage of memory space or disk space. Experimentation is carried out through map reduce programs on different versions of Hadoop by changing some of the parameter values in the configuration files.

I. INTRODUCTION

Big data is a fast growing technology in today's world. Wherever data is generated, it has its significance. Nowadays Data has become as precious as money and data is generated in every field and has great values. So big data is related to every field in one or the other way. Because of this it is widely used in many domains. On the other hand, there are many issues and research ideas available in the big data. This makes many researchers to work on it and solve the problems. Main operations related to big data are on storing data and accessing data. Storage devices are becoming less expensive, supporting to big data.

Data of any volume can be saved in memory with less cost. But time is precious to obtain any result. Size of data has direct impact on the processing time. One can work with big data operations, with the help of resource itself. Resources are part of the operation. Resources might include files, disk space, memory, network bandwidth and many others. As specified earlier, storage is the biggest parameter in big data, mainly concentrated on the memory utilization. Hadoop is a framework used to work with big data. It is an open source tool. It has become popular in today's world to work with big data problems. It follows master-slave architecture. The master and slave nodes in a Hadoop cluster communicate by exchanging messages at regular intervals. The messages are called as heart beat messages. These messages are considered as a potential scheduling opportunity for any application to run. Map reduce programs are the core part of Hadoop framework which are used to work with required problem solving. To know the efficiency of the map reduce programs, it should work on huge amount of data usually in several Giga Bytes to Terra Bytes.

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

This paper observes various parameters of Hadoop Framework that are obtained after executing any Mapreduce program.Experiments are carried out in three different versions of Hadoop framework installed in three different systems with same configurations. Data set used for the execution also remains same. Hadoop has set default values for its parameters. Study has been made with the results obtained when parameters are set to default values and modified by the authors. This paper discusses on the change in the values of results obtained for various parameters of Hadoop andgives justification for the same. It is also observed that there are many changes made between the versions of Hadoop. Apache software foundation releases release notes for every version which gives complete information about the changes made in the versionsunder various categories made by Apache Software Foundation. The changes made between the versions havean impact on the variations in the resultant values, which have been properly identified and justified with the relevant justificationsin this paper.

Generally people understand that Mapreduce programs as a framework. But it is a programming model in Hadoop framework for the faster data processing. To write a Mapreduce program, one should have sound knowledge of either Java or Python programming language. As Hadoop is developed in the Java,it is advised to write Mapreduce programs using Java. Mapreduce is a type of programming model which is used to work with large data sets for processing data and generating results. These programs can handle structured, semi-structured and unstructured data.It is combination of map and reduce functions. Programmer write a map function to process a key/value pair which generates a set of intermediate key/value pairs, and this output is given as an input to reduce function. Reduce function merges all intermediate values associated with the same intermediate key. Mapper performs map operation and reducer performs reduce operation. This is discussed in detail in next section. These mapreduce programs are parallelized in nature and mainly designed for working with large data sets. Mapreduce programs are executed with the help of Job Tracker and Task Tracker available in the Hadoop framework. Job Tracker which is available in master node is a central component which schedules tasks to run on any task tracker and also coordinates the execution of the job. It is a daemon service which would take care of submitting and tracking the mapreduce tasks. Task Tracker is available in every data node which runs tasks assigned to them and sends the report of job status to Job Tracker at regular intervals. [1][2] [3] [4] [5].

In the process of execution, program takes care of handling input data, scheduling of the programs execution by partitioning the input data across the cluster of machines, handling all the processors present in various machines and handling machine failures if any. As all these operations are taken care by Mapreduce programs internally, programmers one who write programs or user one who uses these systems will not feel about these operations explicitly. They won't experience parallel execution, partitioning and scheduling of data on various machines for processing and merging the results in the specified format. The result of the program is displayed as a whole. Only this is observed by the users. There is no fixed size for the data set to work for Mapreduce programs, But to experience the features of Mapreduce programs, one has to work with large set of data. In this paper, Mapreduce programs are worked on huge data set of 70-80 GB. Authors have worked on data set provided by Wikimedia dump, which is an open source data set available for the research purpose, released by Wikimedia Foundation. There are several parameters set to work Hadoop Framework. The paper focuses

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

on the results obtained under two different categories. First category of results are obtained with default values of the Hadoop parameters and the second category of results obtained after changing those default values. The changes are made by considering many aspects which is explained in the later sections.

II. Literature Survey

Most of the research carried out in the big data domain is on the processing of domain specific data, usage of data on different applications and with data analytics for other similar usage. Study on the Hadoop architecture is a less focused area. Some of the papers reflect the same work as used by the authors. These research papers are understood and analysed by the authors in detail. Some of the papers are discussed below:

Jeffrey Dean and Sanjay Ghemawat [6] have observed the working of Mapreduce programs of big data on large data set available from the google. They have made extensive research on the same and explained about the implementation made by them in large cluster of commodity PCs connected together with switched Ethernet. They have also discussed issues related to fault tolerance, locality, task granularity and side effects during the execution of map reduce programs. They also suggested some of the improvements in the partitioning function and combiner function. They have made use of dual processor with 2-4GB of memory per machine on Linux environment. Experiments are carried out in three scenarios. The first scenariois under normal execution. The second scenario is the execution of the sort program with backup tasks disabled. The third scenario is an execution of the sort program after killing 200 processes intentionally.

MateiZaharia et al [7] have focused on implementation of Mapreduce programs in large scale data intensive applications on commodity clusters. Spark is the cluster computing working sets. They have identified that Mapreduce programs are iterative jobs which must reload the data from disk every time and it uses ad-hoc exploratory queries on large datasets. It proposes a new cluster computing framework called as spark. It improves the scalability and fault tolerance properties of Mapreduce programs. Spark is achieved by using a new abstraction called Resilient Distributed Datasets (RDDs). Experimentation is done on 29 GB dataset. It was observed that earlier iterations of spark would take more time for execution but the later one takes 6 seconds, which is very less time compared to that of Hadoop that takes around 127 seconds consistently. Time is reduced because spark reuses the cached data for the later iterations.

Mohammad Asif Khan et al [1] give highly available Hadoop HDFS architecture for name node. They have observed that Single Point of Failure (SPOF) is the major problem in the Hadoop architecture. It can be overcome by using name node replication and two phase name node commit protocol. This would increase the availability of the name node and reduces waiting time and increases the throughput.

Saurabh Gupta and Manish Pandey [3] have observed the time consumption of mappers and reducers. Always reducer operation starts after the completion of the mapper tasks. This results in larger time consumption for the execution of any tasks on Hadoop using Mapreduce program. They have proposed a new Mapreduce model which has overlapping mapper and reducer and hierarchical reduction method. This would reduce the working time of mappers and reducers.

Mithun B N et al [8] have done the comparative study on different versions of Hadoop framework on most of the parameters and concluded that later versions of Hadoop have some improvements in the resource utilization as the issues in the earlier versions of Hadoop are resolved. This impacts on the better time usage and other operations involved in the process.

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

III. Mapper And Reducers

To work with Hadoop framework, Mapreduce programs are written and executed. These programs are designed to handle huge amount of data. Programs are written in Java, Python or similar languages. These programs are the core part of Hadoop framework which specify the essence of big data in many aspects. Most of the big data analytics programs are carried out by writing Mapreduce programs. Mapreduce programs have two main operations. It is made up of mapper and reducer. Mapper and reducers are the integralparts of Hadoop Mapreduce programming model,which performs map and reduce operations.In addition, they will also take care of distributing and parallelizing the tasks across various data nodesin the cluster [8] [9] [10].Figure 1 shows the flow of operations of mapper phase and reducer phase

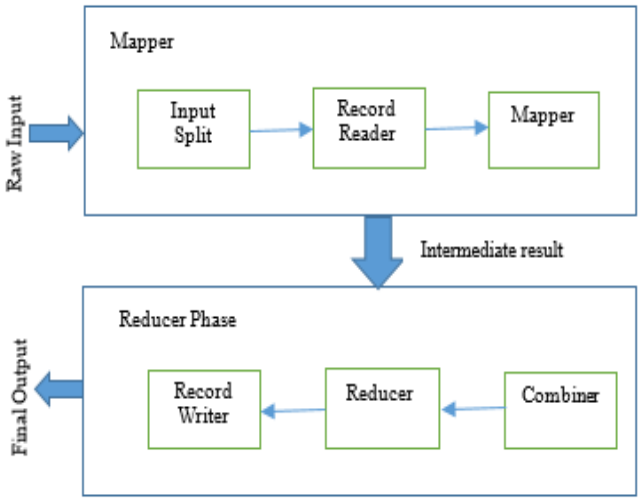

Figure 1: working process of mapper and reducer

Normally it isunderstood that the mapper performs its operation first in the process of executing map reduce programs. But, the first step is to split the input data. Since logic for input split is written by Hadoop and programmer writes code for mapper and not for input split, mapping is considered as the later operation. After input split, record reader takes input from input split and break the problem to narrower and sends it to mapper for mapping operation. So ideally speaking, mapping operation is the third operation in the process. The work of the mapper is to process each input record received from the record reader and to generate a <key, value> pair. The process of generating <key, value> pair from the input records can also be called as Tokenization. Here each token is made up of <key, value> pair. This <key, value> is different from the input given. The output generated by the mapper is called as intermediate output which is stored in the local disk. This output is given as an input to the reducer for its operation. Reducer performs reduce operation on each <key,value> pair and produces the final output. This output is saved in HDFS. Reducer performs aggregation operation or sort operation on the intermediate output produced by the mapper.In order to reduce the data transfer between mapper and the reducer, combiner is used. Combiner is also called as semi-reducer or mini-reducer.It summarizes the output received from mapper and pass it on to the reducer in the same <key, value> collection pair format as generated by the mapper. Usage of combiner is not mandatory in the process. Since combiner makes the operation of reducer easier and reduces network congestion, it is used before reducer.Usage of combiner results in the difference between the count of the output generated by mapper and the input given to the reducer. Combiner reduces the time consumption by the reducer. Figure 2 depicts the workflow of Mapreduce without use of combiner.Figure 3 depicts the workflow of Mapreduce with the use of combiner. There also exists a module called as record reader. It reads the data from the blocks as input and produces the corresponding <key, value> as output. It reads and processes only one record at a time. During shuffling stage, the transmission of data is done from mapper to reducer. Similarly Record writer writes every <key, value> generated by reducer as an output and saves in the HDFS [2]. Number of mappers depends on size of the input file. That is

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

number of mappers required can be calculated by dividing total input size and input split size. Sometimes, split size is also called as block size. For example, if input data size is 5 TB and block size (input split) is 256 MB then 20,480 mappers will be created for the operation. Similarly if the block size is 128 MB, then number of mappers created will be 40,960. Input splits usually depends on the block size. But, it can be customized by changing the value of the variable specified in the configuration file of Hadoop. The variable is: 'mapred.max.split.size'.Input splits can also be changed in the Mapreduce program by writing the instruction -'final long DEFAULT_SPLIT_SIZE'. For example, the assignment statement 'final long DEFAULT_SPLIT_SIZE = 128*1024*1024', would set default split size as 128MB. Changes can be made for setting it to 64MB, 256MB and for other values as required.

As known, Hadoop works on multiple systems in a parallel processing mode.Mapper would work in each data node parallel. Reducer takes the intermediate output from all the data nodes and combine them to develop a final output.

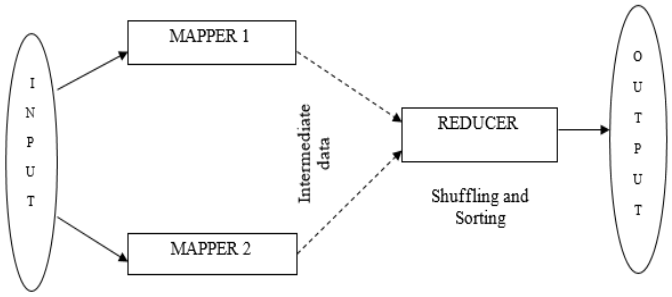

Figure 2: Mapper and Reducer without Combiner

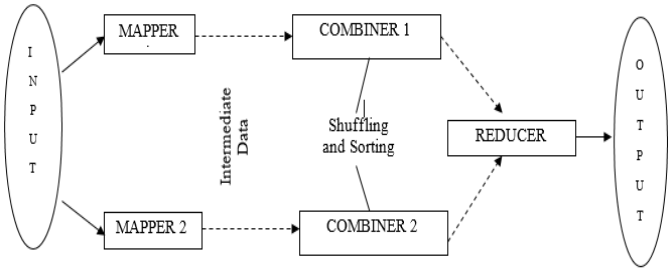

Figure 3: Mapper and Reducer with Combiner

IV. Programming Model

At the outset, one can understand that Mapreduce programs are made up of mappers and reducers. Mappers and reducers perform map and reduce functions respectively. This model is discussed in detail in this section.

Mapreduce follow <key, value> pair as a data structure of the operation. Data types of keys and values may be primitive numeric and alphabet data types including raw bytes. But, most of the times programmers define their own data types depending on the program need anddesign. The signature of mapper and reducer is as shown below:

map: (k1,v1) → [(k2,v2)]

reduce: (k2, [v2]) → [(k3, v3)]

Any Mapreduce program usually contains three main classes, namely Driver class, Map class and Reduce class. These classes can be written in different files or in a single file. Changes have to be made accordingly. Java has rich set of packages available that supports the working of MapReduce programs. Programmer has to import those packages in their program. Map class inherits the public class,MapReduceBase and implements an interface Mapper with input and output variables in key value format used for the mapping operations. The Map class statement is as shown below:

public class WordCountMap extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>

The public method map() needs to be overridden in the implemented class. This method takes key value pair and status of operation as parameters. The signature of the map() method is as shown below:

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

The complete code for map operation as per programming logic is written in the map() method. Similarly, Reduce class extends MapReduceBase and implements Reducer interface. This implemented class has to override public method reduce(). This methodtakes key value pair as parameters for reduce operation. At the end of the reducing process, it converts the key value pair output in the required format depending on the program. The signature of method reduce() is as shown below:

public void reduce(Text key, Iterator<IntWritable> value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

The methods map() and reduce() throws IOException. Driver class is written to initiate the process of map reducing. This carries some of the background job required for the process. The statement - *JobConfconf = new JobConf(WordCountDriver.class);* creates the job configuration event for the specific class. Input and output path has to be set in this class for the object '*conf*'. Input and output paths are given as parameters to the methods *setInputPath()* and *setOutputPath()* respectively. This would read the input from the path where the actual data set is loaded. After setting input and output paths, map and reduce classes are set to initiate the process. There is also a need to specify the data types required for the key and the value in <key, value> pair for the processing. The instruction *JobClient.runJob(conf);* is used to run the job on the given input data set. The complete discussion is made on single node cluster. Necessary changes are made to the code if the Mapreduce programs need to work on multi node cluster. Hadoop Framework takes care of job scheduling and load sharing processes during execution. Authors have written a simple Mapreduce program (word count program) in java for the experimentation purpose. It identifies and counts each word in the given input data set.

V. Experimentation

In order to study and understand the various features of Hadoop Framework, analysis is made on different versions of Hadoop. This is carried out by installing different versions of Hadoop in different systems. In the first system Hadoop version 2.6.5 is installed and Hadoop version 2.7.7 is installed in second system and the Hadoop version 3.0.3 is installed in the third system. A simple word count Mapreduce program is written and executed on the data set with default values of various Hadoop parameters. The same program is executed by changing some of the parameter values defined in the configuration files of Hadoop framework.Both results are compared and analysed. Data set is essential component to work with big data. Huge size of data is required in order to observe the working of the Mapreduce programs. Since authors are concentrating on several parameters of Hadoop architecture, there is no need of specific data set. For the experimentation, Wikimedia data set is used. Wikimedia provides open data set (freely available on internet) for the research purposes. These data sets size varies from some Megabytes to several Gigabytes. Downloaded data set has to be pre-processed before using it.

VI. Results And Performance Evaluation

During experimentation, split size and block size of the HDFS are changed. Observations and analysis are made for 64MB, 128MB and 256MB values. For the results obtained from the experimentation, bar graphs are drawn for various parameters and analysed as shown from figure 4 to figure 16.Some of the parameter values remains same and some of the parameter values are different. This section concentrates on specifying the reasons behind change in those values. Bar graphs are plotted to understand, compare and analyse the results. In the bar graph, X-axis indicates the different Hadoop version and the Y-axis indicates the values of parameterssuch as Mega Bytes,Milli seconds or number of operations depending on the parameters.

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

The bar graph in the figure 4 outlinesthe number of megabytes read and the bar graph in figure 5 outlinesthe number of megabytes written under the file systems category ofHadoop Framework.

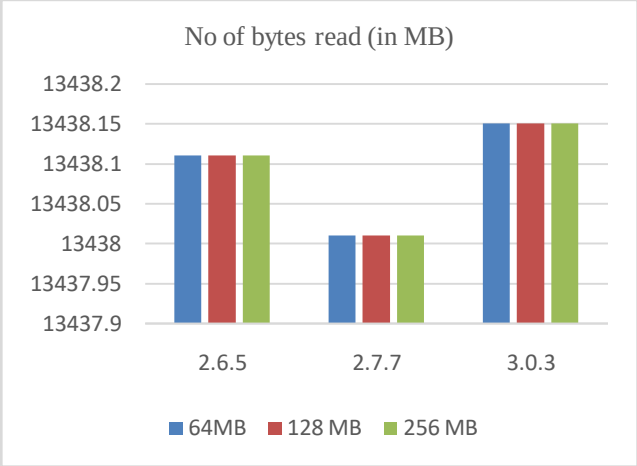

Figure 4: Bar graph indicating number of bytes bytesreadunder file systems category

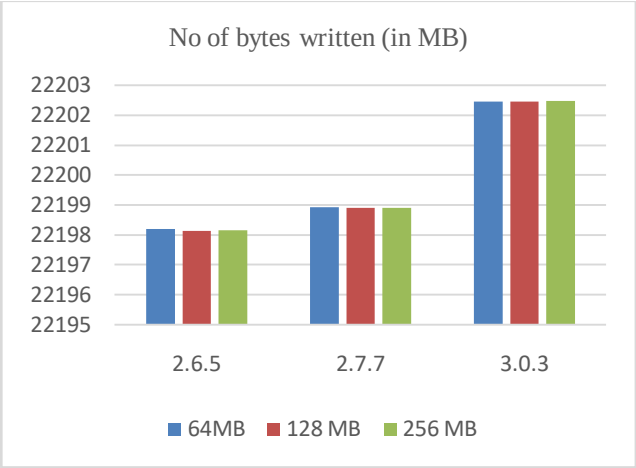

Figure 5: Bar graph indicating number of written under file systems category

The bar graphs in the figures 6, 7, 8 and 9 represents the resultant parameters related to HDFS category. Figures 6 and 7 show bar graphs which indicate the number of megabytes read and written with respect to HDFS. Total number of megabytes read and written are the same. Data read and written in HDFS purely depends on the size of the input data set used for the execution,because of this, there is no impact of change in parameter valuesof block size in all three versions of Hadoop.

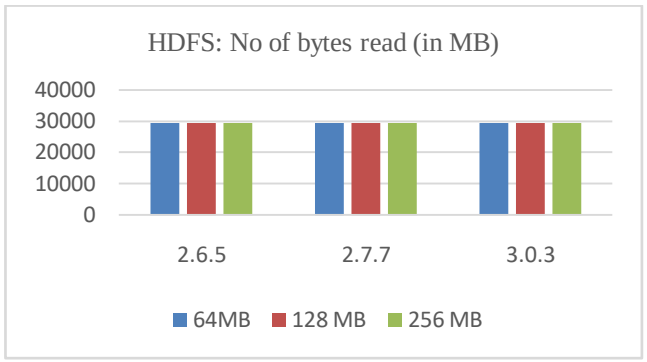

Figure 6: Bar graph indicating number of bytes read under HDFS

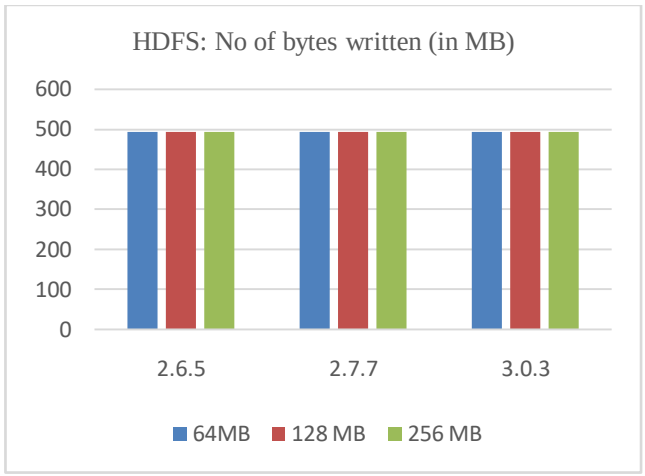

Figure 7: Bar graph indicating number of bytes written under HDFS

Figures 8 and 9 show bar graphs that depicts the total number of read and write operations performed during execution of the Mapreduce program. They fall under the HDFS category. There is a change in the number of operations performed between the versions of Hadoop,but no change within the same version irrespective of the block size. There are more read operations performed in the earlier version of Hadoop and more write operations in the later version of the Hadoop. This is because, Hadoop version 2.6.5 uses POSIX_STYLE of file system [8] and this is not seen in the version 2.7.7 and version 3.0.3.

Figures 10, 11, 12, 13, and 14 show bar graphs which show the results of various parameters under Mapreduce category. Bar graph in the figure 10

May – June 2020
ISSN: 0193-4120 Page No. 170 - 179

representing the input splits in different versions with change in split size. Input splits are logical division of data which pertain to a single mapper job. Each input split can span across various physical blocks. As the data is spread across various nodes in the distributed environment, input splits helps to identify the data belonging to a single mapper. Usually input splits and number of mappers are same. Since there is a change in the input splits during the experimentation, there are changes accordingly in the bar graph. Less split size will yield more splits. This is reflected in the bar graph.

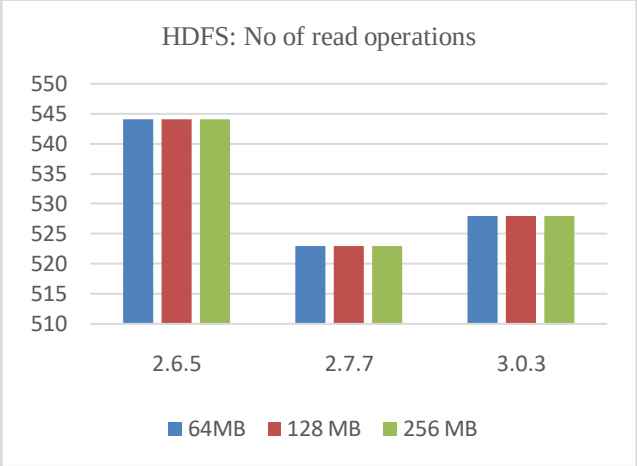

Figure 8: Bar graph indicating number of read operations under HDFS

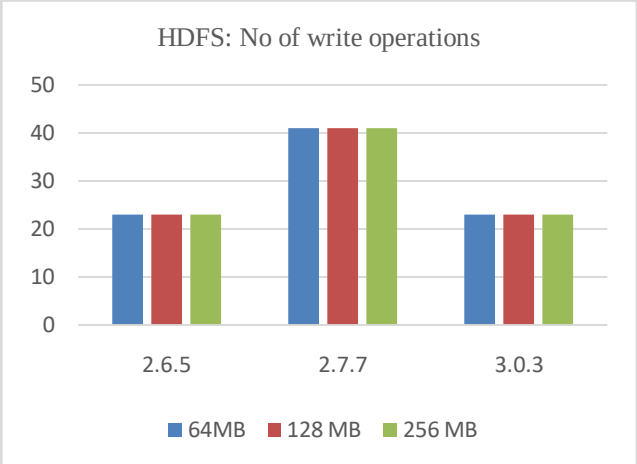

Figure 9: Bar graph indicating number of write operations under HDFS category

The bar graph in the figure 11 depicts the results of combined input records and the bar graph in the figure 12 depicts the results of combined output records of the Mapreduce category. These are related to combiner operations. As discussed in the section 3, combiner performs its operations by taking the output generated by the mapper. This results in reduction in the input records compared to that of output records generated by the mapper. Similarly combiner output records results less compared to the combiner input records. This is the input for the reducer and is impact on the result of reduce records. Combined input and output records may also vary as combiner operation is overlapped between mapping and reducing. The values of combiner input and output records change depending on the Hadoop versions. But change in parameter values has no much impact on the operations of the combiner.

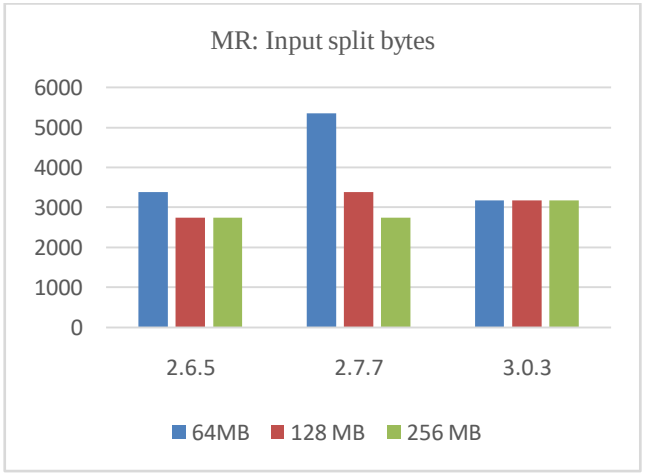

Figure 10: Bar graph indicating input split bytes under Mapreduce category

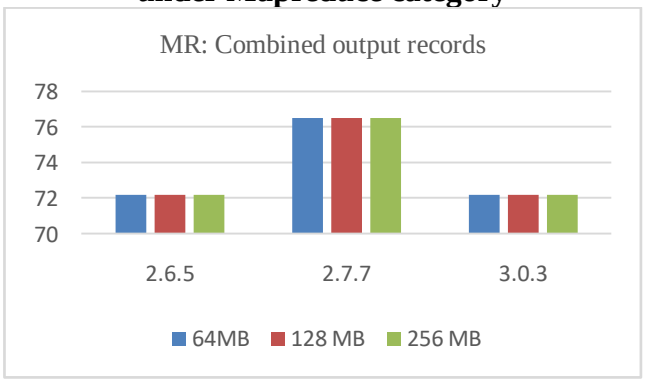

Figure 11: Bar graph indicating combined input records under Mapreduce category

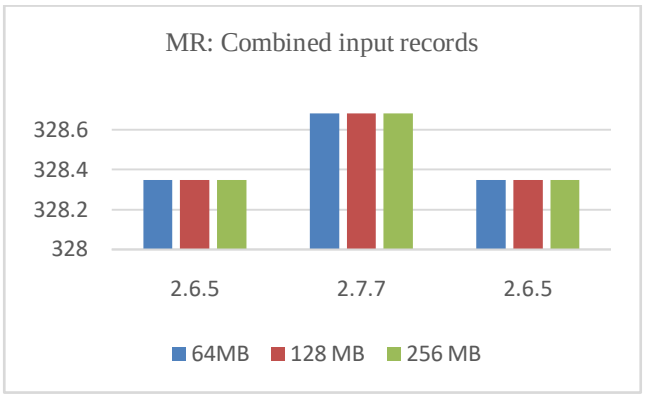

Figure 12: Bar graph indicating combined output records under Mapreduce category

A Spilled Record is the total number of records that were spilled to disk during execution of the job. Spilling happens only when buffer reaches some threshold value, during map and reduce phases [11].The bar graph in the figure 13 represents the resultant values of spilled records under Mapreduce category in all three versions of Hadoop with change in the split size and block size. Since spilling and split size are not dependent on each other, there is no change in count of spilled records with the changesmade to the values of parameters.Shuffling is the process of transferring data from mapper to reducer. Shuffled map values are generated during the process of data transfer. The bar graph in the figure 14 depicts the values of shuffled maps. Shuffling happens between mapper and reducer and has no impact of changes in the parameter values.

Figure 15 has the bar graph which shows the Garbage Collection (GC) elapsed time during the execution of the Mapreduce program. As this activity is dependent on the architecture of the Hadoop, there is a change in the time consumption with respect to the parameter values. There is a variation in the time usage depending on the block size. Earlier version of the Hadoop has less time consumption. In the later version, garbage collection is consuming more time, varying with the block size. While performing the execution of the Mapreduce program, heap memory plays an important role. The bar graph in the figure 16 shows the total heap usage during the execution in megabytes. Changes in split size values and block size affect the memory usage. Allocation and utilization of heap memory resulted in different values as there is change in the parameters.

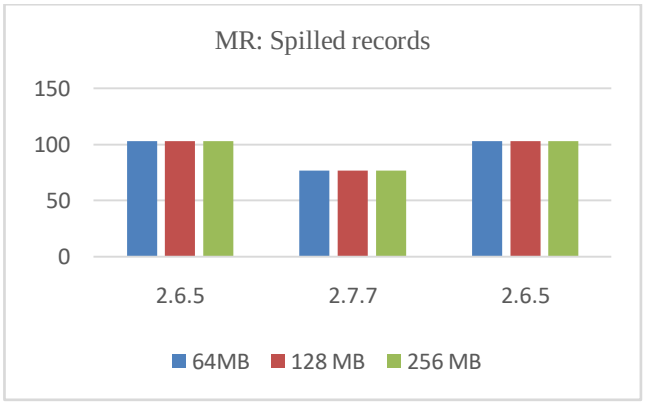

Figure 13: Bar graph indicating spilled records under Mapreduce category

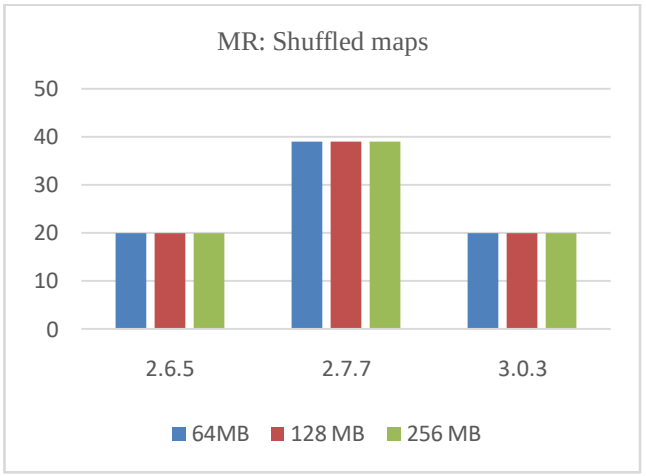

Figure 14: Bar graph indicating shuffled maps under Mapreduce category

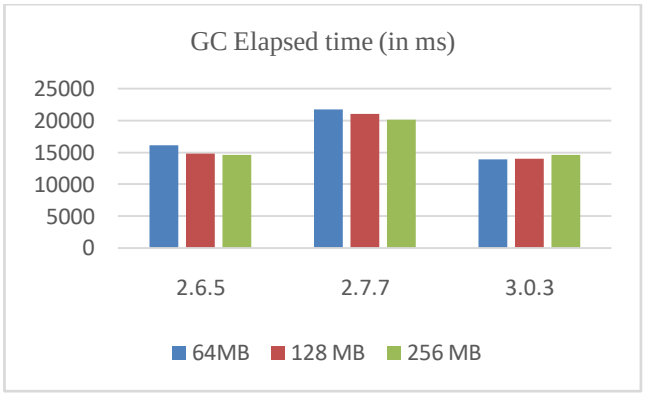

Figure 15: Bar graph indicating elapsed time in millisecond

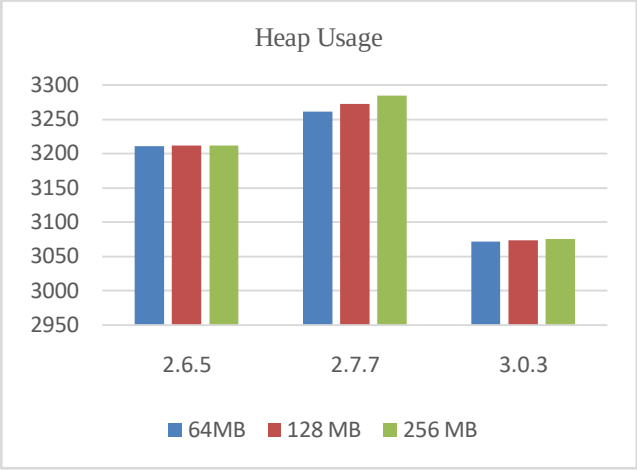

Figure 16: Bar graph indicating heap usage

VII. CONCLUSION

Mapreduce programs are the central part of the Hadoop framework. The Study was made on the different versions of the framework by changing the parameter values in the configuration files and in the Mapreduce program. It can be concluded that changes in split size and block size have not directly affected the parameters under HDFS category and have an impact on the parameters under Mapreduce category. Newer versions of the Hadoop are more efficient with many updates. But, execution time with garbage collection is still more and need to address the issue in order to reduce the time consumption for faster results.

REFERENCES

1. Mohammad Asif Khan, Zulfiqar A Memon, Sajid Khan, Highly Available Hadoop Namenode Architecture, International Conference on Advanced Computer Science Applications and Techniques, 2012.
2. Saurabh Gupta, Manish Pandey, Performance improvement in MapReduce via overlapping of Mapper and Reducer, International Journal of Computer Science and Information Security, 14(7), July 2016
3. MadhaviVaidhya, Shriniwas Deshpande, Critical study of Hadoop Implementation and Performance issues,
4. Uma Patel, Rakesh Patel, Nimita Patel, Study of Apache Hadoop, International Journal of Engineering Sciences and Research Technology, 3(12), December 2014
5. YogineeSurendraPethe, Inception of Big Data with Hadoop and Map Reduce, IJSART, 3(3), March 2017
6. Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of ACM, 51(1) January 2008, pages 107-113
7. MateiZaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, Ion Stoica, Spark: Cluster Computing with working sets, Hotcloud, 10(10), June 2010
8. Mithun B N, Viswanatha K V, Manjunath T N, A Performance Analysis of different Hadoop Versions on a Commodity Hardware, International Journal of Management, Technology and Engineering, 9(5), May 2019, Pages 590-598.
9. LatikaKakkar, Gaurav Mehta, A Review: Hadoop Storage and Clustering Algorithms, IOSR Journal of Computer Engineering, 18(1), January – February 2016, pages 23-29
10. VedulaVenkateswara Rao, VKSK Sai Vadapalli, Big Data Analysis – Hadoop Performance Analysis, International Journal of Computer Science and Information Security, 14(9), September 2016
11. https://data-flair.training/forums/topic/explain-the-process-of-spilling-in-mapreduce/