RESEARCH ARTICLE

SAMRIDDHI Volume 14, Issue 2, 2022 Print ISSN: 2229-7111 Online ISSN: 2454-5767

Performance Tuning Of Apache Spark Framework In Big Data Processing with Respect To Block Size And Replication Factor

Brijesh Y. Joshi, Poornashankar, Deepali Sawai

Institute for Industrial and Computer Management and Research, Pradhikaran, Pune, Maharashtra, India

Abstract

Apache Spark has recently become the most popular big data analytics framework. Default configurations are provided by Spark. HDFS stands for Hadoop Distributed File System. It means the large files will be physically stored on multiple nodes in a distributed fashion. The block size determines how large files are distributed, while the replication factor determines how reliable the files are. If there is just one copy of each block for a given file and the node fails, the data in the files become unreadable. The block size and replication factor are configurable per file. The results and analysis of the experimental study to determine the efficiency of adjusting the settings of tuning Apache Spark for minimizing application execution time as compared to standard values are described in this paper. Based on a vast number of studies, we employed a trial-and-error strategy to fine-tune these values. We chose two workloads to test the Apache framework for comparative analysis: Wordcount and Terasort. We used the elapsed time to evaluate the same.

Keywords: Apache Spark, Big Data, Block size, HDFS, Replication Factor.

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology (2022); DOI: 10.18090/samriddhi.v14i02.00

Introduction

Big Data Analytics is critical to the company's overall success and market penetration. Businesses can now process massive volumes of data being consumed at exponential rates thanks to technological breakthroughs and the significant boost in computational power provided by High Performance Distributed Computing. Mobile phones, IoT devices, and other internet services are important sources of raw data that must be churned to make sense of it. Various mixes of big data To process enormous amounts of data, Hadoop and other big data technologies are commonly employed. Apache Hadoop, Cloudera, and Hortonworks are of the Hadoop varieties available. Apache Spark is also used to process large amounts of data.[12]

Spark is a data processing framework developed by Apache. It can analyze large data sets. It also can distribute data processing jobs among multiple machines. Apache Spark is an open-source cluster-computing framework. Since 2010, Spark has been an open-source project. In 2009, Zahari developed this project at UC Berkeley's AMPLab.[1] Batch processing, real-time processing, interactive processing, and graph processing are all supported by Apache Spark. It offers in-memory processing as an alternative to storing data on a hard disc drive. It's an example of a distributed computing system in action. It's based on MapReduce algorithms from Hadoop. It integrates the advantages of Hadoop MapReduce, however unlike MapReduce, the input and output are stored in memory, therefore the name Memory Computing. Its performance is improved because of this. It is best suited for iterative applications such as data mining and machine learning. It is primarily concerned with increasing speed while also ensuring reliability. It was designed specifically for computations that need to be completed quickly. In Apache Hadoop for map-reduce jobs, the centos operating system showed better performance than Ubuntu and the 256MB block size observed to be the most suitable value.[10] In Apache Hadoop, a data computing layer is MapReduce. It enables the creation of applications for large datasets. On a cluster, a MapReduce application runs. The map and reduce functions are the most important parts of a MapReduce program. The map function takes in key-value pairs as input

Corresponding Author: Brijesh Y. Joshi, Institute for Industrial and Computer Management and Research, Pradhikaran, Pune, Maharashtra, India, e-mail: brijeshjoshi1@gmail.com

How to cite this article: Joshi, B.Y., Poornashankar, Sawai, D. (2022). Performance Tuning Of Apache Spark Framework In Big Data Processing with Respect To Block Size And Replication Factor. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology,* 14(2), 1-7.

Source of support: Nil

Conflict of interest: None

and returns another set of key value pairs as output. The job execution process is managed by the job tracker. Its primary function is to coordinate all tasks that are running on the task tracker.[11]

When Spark runs on the Hadoop cluster, RDDs will be created on the HDFS in many formats supported by Hadoop, likewise text and sequence files.[2]

The RDD is at the heart of the Spark Architecture concept. It's an array of fault-tolerant components. It can execute in parallel and allows the user to store data directly on the hard disc as well as in memory. It's a distributed computing model designed for large-scale, linearly scalable fault-tolerance applications.

It performs memory processing by leveraging the RDD data type. RDDs are a collection of partitioned and permanent items created during the storing process. The labor in Spark is separated into two categories: Master and Slaves. Slaves are assigned tasks by the Master, who then reports back to him. Such activities are conducted on RDDs and are divided into two categories: transformations and actions. The tasks are specified in transformation groups, and the execution process is carried out as planned.

Transformations in Apache Spark are functions that take an RDD as input and return one or more RDDs as output. RDDs are immutable, which means they can't be changed. It implements the computations and generates one or more new RDDs. Transformations are treated as Lazy operations in Apache Spark. When a process occurs, these generate one or more RDDs that run. Transformation then uses the old dataset to produce a new one.

The architecture of Apache Spark is seen in Figure 1. A driver program (SparkContext), workers, often known as executors, a cluster manager, and the HDFS are all part of Apache Spark. Spark's main program is the driver program. SparkContext is an object formed throughout the execution of a Spark application and is in charge of the job's full execution. The SparkContext object links to the cluster manager, which manages the cluster's resources. Executors are provided by cluster managers and are utilized to conduct the logic and store the app data.[7]

Literature Review

Spark is a general-purpose processing engine that works with Hadoop data. It can process data in HDFS using any Hadoop Input Format and can run in Hadoop clusters using YARN or Spark's standalone mode. It's built to handle batch processing much like MapReduce as well as new workloads like streaming, interactive queries, and machine learning.[3] Apache Spark is an open-source project that builds on top of the Hadoop Distributed File System (HDFS). Spark, on the other hand, is not bound by the two-stage MapReduce paradigm.[4] It's critical to boost Spark's execution performance for various application types, which can be accomplished by optimizing a Spark job's physical execution plan, efficiently scheduling parts of a Spark job on cluster nodes, and choosing the right cluster configuration, such as the number of machines (processors) and resources available on each machine. To use Spark, programmers create a driver program that implements their application's high-level control flow and launches several processes in parallel. Spark offers two parallel programming abstractions: resilient distributed datasets and parallel operations on these datasets.[5]

Using an installed cluster in our lab, we compared the performance of Hadoop and Spark. Based on a vast number of studies, we employed a trial-and-error strategy to fine-tune these values. We chose two workloads to examine the frameworks for comparative analysis: WordCount and Terasort.[6]

All of the benchmarks have the following parameters, basic block size, number of blocks to be constructed, the number of nodes in the cluster that are truly operational; the number of cores per node in the cluster.[8]

when the data input size is equal, the FIFO scheduler performs effectively. When data input sizes are unequal and the first job is larger than the second, the Fair Scheduler performs effectively. When data input sizes are mismatched and the first job is smaller than the second, the FIFO scheduler performs well.[9]

Experimental Setup

To study the performance evaluation, we instantiated 3 nodes EC2 on AWS, each compute node was having 16GB RAM, 4 Cores, 1500 GB hard disk. Operating System was installed CentOS Linux release

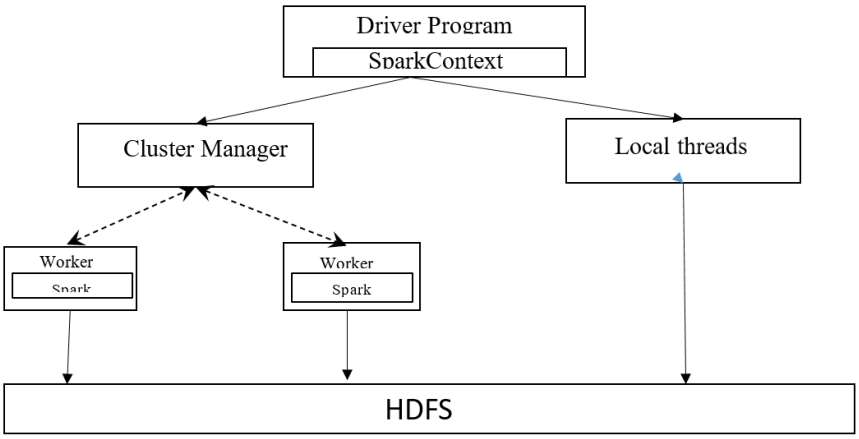

Figure 1: Spark Architecture[7]

7.8.2003, Spark 1.6.0, Java 1.8.0. Elastic IP were used to establish communication among the nodes Elapsed time the executor spent running the task. The value of elapsed time is expressed in milliseconds but we converted in seconds. Tuning parameters in Apache Spark is a challenging task. We want to find out which parameters can impact the elapsed time and we chose Block size and replication factor.

Experimental Results

Table 1 and Figure 2 represent input data size 0.6GB, 128MB, 256 MB, 512 MB, 1024MB block size, and for each block size replication factor 1,2,3 was set and WordCount program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 18sec.
- Elapsed time for combination 128MB, Replication factor 2 was observed 26sec.
- Elapsed time for combination 128MB, Replication factor 3 was observed 20 sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 13sec. Elapsed time for combination 256MB, Replication factor 2 was observed 11sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 11 sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 16sec. Elapsed time for combination 512MB, Replication factor 2 was observed 12sec.
- Elapsed time for combination 512MB, Replication factor 3 was observed 19 sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 16sec. Elapsed time for combination 1024 MB, Replication factor 2 was observed 12sec.
- Elapsed time for combination 1024 MB, Replication factor 3 was observed 19 sec.

Table 2 and Figure 3 represents for input data size 10GB, 128MB, 256 MB, 512 MB, 1024MB block size was and for each block size replication factor 1,2,3 was set and WordCount program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 90sec.

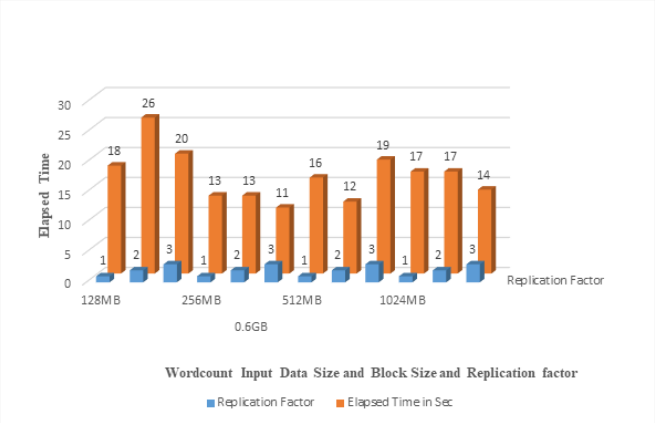

Figure 2: Wordcount Input Data Size (0.6 GB) and Block Size and Replication Factor Vs Elapsed Time

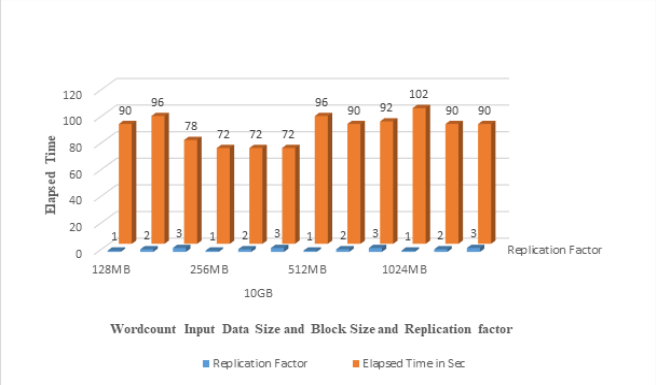

Figure 3: Wordcount Input Data Size (10GB) and Block Size and Replication factor vs Elapsed Time

Table 1: WordCount, Input Data size, Block Size, Replication factor and Elapsed time

Data size	*Block size*	*Replication factor*	*Elapsed time in Seconds*
0.6GB	128MB	1	18
		2	26
		3	20
	256MB	1	13
		2	13
		3	11
	512MB	1	16
		2	12
		3	19
	1024MB	1	17
		2	17
		3	14

Table 2: WordCount, Input Data size, Block Size, Replication Factor and Elapsed time

Data size	*Block size*	*Replication factor*	*Elapsed time in Seconds*
10GB	128MB	1	90
		2	96
		3	78
	256MB	1	72
		2	72
		3	72
	512MB	1	96
		2	90
		3	92
	1024MB	1	102
		2	90
		3	90

- Elapsed time for combination 128MB, Replication factor 2 was observed 96sec.
- Elapsed time for combination 128MB, Replication factor 3 was observed 78 sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 72sec.
- Elapsed time for combination 256MB, Replication factor 2 was observed 72sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 72sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 96sec.
- Elapsed time for combination 512MB, Replication factor 2 was observed 90sec.
- Elapsed time for combination 512MB, Replication factor 3 was observed 92sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 102sec.
- Elapsed time for combination 1024 MB, Replication factor 2 was observed 90sec.
- Elapsed time for combination 1024 MB, Replication factor 3 was observed 90sec.

Table 3 and Figure 4 represents for input data size 100GB, 128MB, 256 MB, 512 MB, 1024MB block size was and for each block size replication factor 1,2,3 was set and WordCount program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 458sec.
- Elapsed time for combination 128MB, Replication factor 2 was observed 438sec.
- Elapsed time for combination 128MB, Replication factor 3 was observed 414 sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 456sec.
- Elapsed time for combination 256MB, Replication factor 2 was observed 360sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 348sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 474sec.
- Elapsed time for combination 512MB, Replication factor 2 was observed 582sec.
- Elapsed time for combination 512MB, Replication factor 3 was observed 366sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 462sec.
- Elapsed time for combination 1024 MB, Replication factor 2 was observed 366sec.
- Elapsed time for combination 1024 MB, Replication factor 3 was observed 372sec.

Table 4 and Figure 5 represents for input data size 0.6GB, 128MB, 256 MB, 512 MB, 1024MB block size was and for each block size replication factor 1,2,3 was set and Terasort program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 97sec.
- Elapsed time for combination 128MB, Replication factor 2 was observed 101sec.

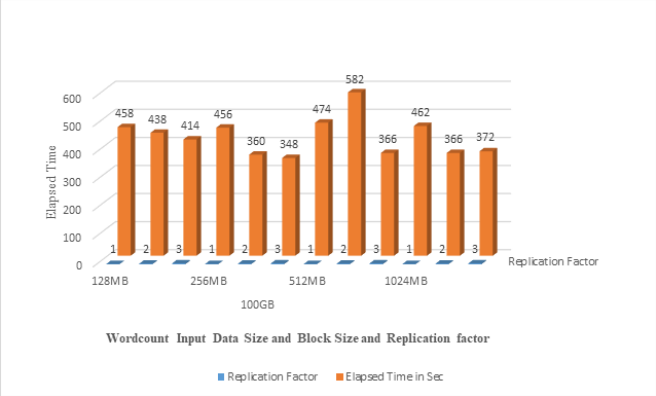

Figure 4: Wordcount Input Data Size (100GB) and Block Size and Replication Factor Vs Elapsed Time

Table 3: WordCount, Input Data size, Block Size, Replication Factor and Elapsed time

Data size	*Block size*	*Replication factor*	*Elapsed time in Seconds*
100GB	128MB	1	458
		2	438
		3	414
	256MB	1	456
		2	360
		3	348
	512MB	1	474
		2	582
		3	366
	1024MB	1	462
		2	366
		3	372

Table 4: Terasort, Input Data size, Block Size, Replication factor and Elapsed time

Data Size	*Block Size*	*Replication factor*	*Elapsed Time in Sec*
0.6 GB	128MB	1	97
		2	101
		3	99
	256MB	1	89
		2	89
		3	87
	512MB	1	108
		2	106
		3	103
	1024MB	1	99
		2	102
		3	101

- Elapsed time for combination 128MB, Replication factor 3 was observed 99sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 89sec.
- Elapsed time for combination 256MB, Replication factor 2 was observed 89sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 87sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 108sec.
- Elapsed time for combination 512MB, Replication factor 2 was observed 106sec.
- Elapsed time for combination 512MB, Replication factor 3 was observed 106sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 99sec.
- Elapsed time for combination 1024 MB, Replication factor 2 was observed 102sec.
- Elapsed time for combination 1024 MB, Replication factor 3 was observed 101sec.

Table 5 and Figure 6 represents for input data size 10GB, 128MB, 256 MB, 512 MB, 1024MB block size was and for each block size replication factor 1,2,3 was set and Terasort program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 201sec.
- Elapsed time for combination 128MB, Replication factor 2 was observed 203sec.
- Elapsed time for combination 128MB, Replication factor 3 was observed 199sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 190sec.
- Elapsed time for combination 256MB, Replication factor 2 was observed 190sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 189sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 208sec.
- Elapsed time for combination 512MB, Replication factor 2 was observed 210sec.

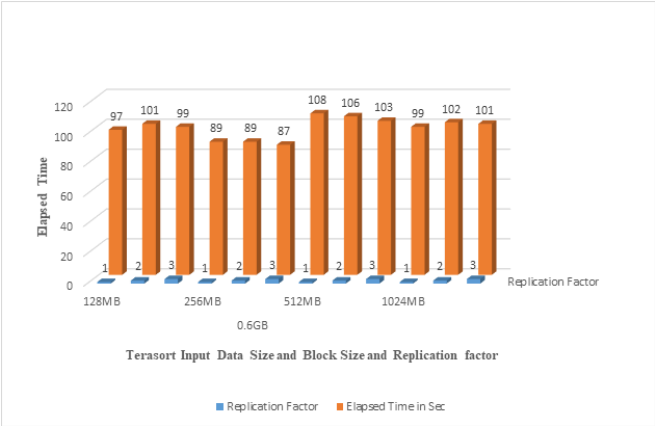

Figure 5: Terasort Input Data Size (0.6 GB) and Block Size and Replication Factor Vs Elapsed Time

- Elapsed time for combination 512MB, Replication factor 3 was observed 212sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 219sec.
- Elapsed time for combination 1024 MB, Replication factor 2 was observed 217sec.
- Elapsed time for combination 1024 MB, Replication factor 3 was observed 215sec.

Table 6 and Figure 7 represents that for input data size 100GB, 128MB, 256MB, 512MB, 1024MB block size was and for each block size replication factor 1,2, 3 was set and Terasort program was executed on Apache Spark.

- Elapsed time for combination 128MB, Replication factor 1 was observed 758sec.
- Elapsed time for combination 128MB, Replication factor 2 was observed 738sec.
- Elapsed time for combination 128MB, Replication factor 3 was observed 714sec.
- Elapsed time for combination 256MB, Replication factor 1 was observed 656sec.

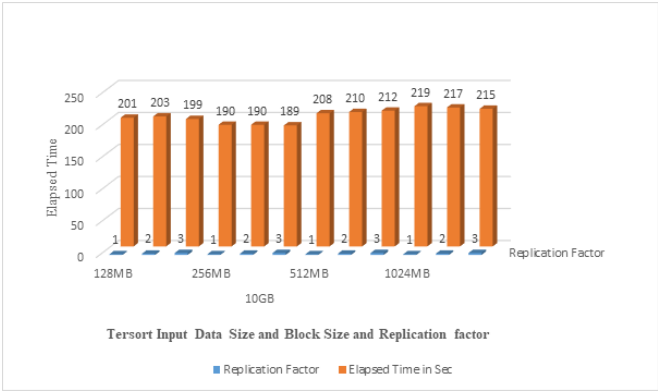

Figure 6: Terasort input data size (10 GB) and block size and replication factor vs elapsed time

Table 5: Terasort, Input Data size, Block Size, Replication factor and Elapsed time

Data size	*Block size*	*Replication factor*	*Elapsed time in Seconds*
10GB	128MB	1	201
		2	203
		3	199
	256MB	1	190
		2	190
		3	189
	512MB	1	208
		2	210
		3	212
	1024MB	1	219
		2	217
		3	215

Table 6: Terasort, Input Data size, Block Size, Replication Factor and Elapsed time

Data Size	Block Size	Replication Factor	Elapsed Time in Sec
100GB	128MB	1	758
		2	738
		3	714
	256MB	1	656
		2	660
		3	648
	512MB	1	774
		2	782
		3	766
	1024MB	1	762
		2	766
		3	772

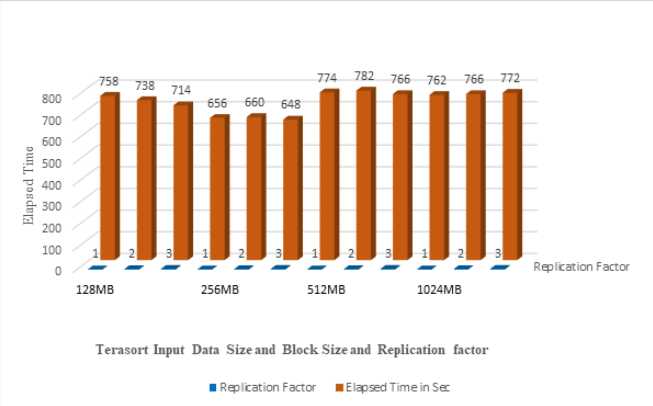

Figure 7: Terasort input data size (100 gb) and block size and replication factor vs elapsed time

- Elapsed time for combination 256MB, Replication factor 2 was observed 660sec.
- Elapsed time for combination 256MB, Replication factor 3 was observed 648sec.
- Elapsed time for combination 512MB, Replication factor 1 was observed 774sec.
- Elapsed time for combination 512MB, Replication factor 2 was observed 782sec.
- Elapsed time for combination 512MB, Replication factor 3 was observed 766sec.
- Elapsed time for combination 1024MB, Replication factor 1 was observed 762sec.
- Elapsed time for combination 1024MB, Replication factor 2 was observed 766sec.
- Elapsed time for combination 1024MB, Replication factor 3 was observed 772sec.

Observations and Findings

From Table 1 and Figure 2 it is observed that minimum Elapsed time for WordCount job with input data size 0.6MB was 11 seconds and it was for combination of block size 256MB and Replication Factor 3, Followed by that second minimum Elapsed time was 12 secs for the combination of block size 512MB and Replication factor 2. Maximum elapsed time for WordCount job with input data size 0.6GB is 26 for combination of block size 128MB and Replication factor 2.

From Table 2 and Figure 3 it is observed that minimum Elapsed time for WordCount job with input data size 10GB was 72 seconds and it was for combination of block size 256MB and Replication Factor 1, 256MB and Replication Factor 2, 256MB and Replication Factor 3. Maximum elapsed time for WordCount job with input data size 10 GB is 102sec for combination of block size 1024MB and Replication factor 1.

From Table 3 and Figure 4 it is observed that minimum Elapsed time for WordCount job with input data size 100GB was 348 seconds and it was for combination of block size 256MB and Replication Factor 3. Maximum elapsed time for WordCount job with input data size 100 GB is 582sec for combination of block size 512MB and Replication factor 2.

From Table 4 and Figure 5 it is observed that minimum Elapsed time for Terasort job with input data size 0.6GB was 87 seconds and it was for combination of block size 256MB and Replication Factor 3, Maximum elapsed time for WordCount job with input data size 0.6GB is 108 for combination of block size 512MB and Replication factor 1.

From Table 5 and Figure 6 it is observed that minimum Elapsed time for Terasort job with input data size 10GB was 189 seconds and it was for combination of block size 256MB and Replication Factor 3 followed by that second minimum elapsed time 190 sec for 256 MB and Replication factor 1 and Replication 2, Maximum elapsed time for WordCount job with input data size 0.6GB is 219 for combination of block size 512MB and Replication factor 1.

From Table 6 and Figure 7 it is observed that minimum Elapsed time for Terasort job with input data size 100GB was 648seconds and it was for combination of block size 256MB and Replication Factor 3, Maximum elapsed time for WordCount job with input data size 100GB is 782 for combination of block size 512MB and Replication factor 2.

Conclusion

We focused mainly on the two hdfs parameters such as block size and replication factor in this research work. In this empirical study we executed Wordcount and Terasort jobs with varying input data for combination of 128MB and Replication factor 1, Replication factor 2, Replication factor, 256MB and Replication factor 1, Replication factor 2, Replication factor, 512MB and Replication factor 1, Replication factor 2, Replication factor, 1024MB and Replication factor 1, Replication factor 3, Replication factor. We conclude that for Wordcount job for input data size 0.6GB minimum elapsed time for block size 256 MB and replication factor 3, for input data size 10GB minimum elapsed time for block size 256MB and replication factor 1,2,3, for input data size 100GB minimum elapsed time for block size 256MB and replication factor 3. For Terasort job input data size 0.6GB minimum elapsed time for block size 256 MB and replication factor 3. For Terasort job input data size 10GB minimum elapsed time for block size 256MB and replication factor 3 and for Terasort job 100GB minimum elapsed time for block size 256MB

and replication factor 3. Hence, the final conclusion drawn that the optimum combination for block size and replication factor to execute WordCount and Terasort jobs at minimum elapsed time is 256MB and replication factor 3 in turn, it can be concluded that minimum elapsed time represents the better performance.

Scope of Future Work

In this research work we executed the WordCount and Terasort benchmarking programs on AWS Cluster and Input Data Size was taken into consideration was 0.6GB, 10GB,100GB in short we did empirical study on Input data size in terms of MB and GB, further empirical study can be by giving input in terms of GB, TB, PB and check the optimum values of the considered parameters and further the elapsed time prediction model can be developed.

References

[1] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, Mccauley M, Franklin M, Shenker S, Stoica I.(2012), *Fast and interactive analytics over hadoop data with spark*, Usenix Login.; 37:45–51.

[2] Kannan P. (2015) , *Beyond hadoop mapreduce apache tez and apache spark*, San Jose State University, http://www.sjsu.edu/people/robert.chun/courses/CS259Fall2013/s3/F.pdf.

[3] *Spark.apache.org*, 2021. [Online]. Available: https://spark.apache.org/

[4] *Apache Spark with Hadoop – Why it Matters?* | Edureka.co, *Edureka*, (2021) [Online].

[5] Zaharia, M., Chowdhury M., Franklin M. J., Shenker S. J., & Stoica I.,(2010), *Spark: Cluster Computing with working Sets*, In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud), pp. 1-7, https://www.icsi.berkeley.edu/icsi/node/5074

[6] N. Ahmed, A. Barczak, T. Susnjak and M. Rashid,(2020), *A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench*, Journal of Big Data, vol. 7, no. 1.

[7] *Apache Spark™ - Unified Engine for large-scale data analytics*, *Spark.apache.org*, (2021) [Online]. Available: http://spark.apache.org/

[8] G. Thiruvathukal, C. Christensen, X. Jin, F. Tessier and V. Vishwanath,(2021) , *A Benchmarking Study to Evaluate Apache Spark on Large-Scale Supercomputers Arxiv.org*, [Online]. Available: https://arxiv.org/pdf/1904.11812.pdf

[9] Brijesh Y. Joshi, Dr. Poornashankar,(2021), *Performance Evaluation of Apache Spark Framework in Big Data Processing with Respect to Scheduling Algorithms in Multiuser Environment*, Journal of the Maharaja Sayajirao University of Baroda, ISSN: 0025-0422, Volume-55, No.2

[10] M. Y. J. Dr. Poornashankar,(2021), *Performance Tuning of Apache Hadoop Framework in Big Data Processing with Respect to Block Size, Operating System Clusters and Map Reduce Techniques*, *DE*, pp. 5766- 5778

[11] Joshi B., Dr. Poornashankar,(2020),*HADOOP Performance Evaluation with respect to Scheduling Algorithms in Multiuser Environment*, Vol50issue11 – Pensee.Penseeresearch.com. http://penseeresearch.com/index.php/volume50issue11/.

[12] Brijesh Y. Joshi, Dr. Poornashankar, (2018), *A Study of usage of Big Data Processing technologies and sectorial analysis in Industries of Pune Region, Journal of Applied Science and Computations,* ISSN NO 1076-5131.