Hadoop Map Reduce Performance Evaluation and Improvement Using Compression Algorithms on Single Cluster

Mr. Bhavin J. Mathiya PhD (Research Scholar)
Faculty of Computer Science
C.U.Shah University
Wadhwan City, Gujarat, India.
bhavinmath@gmail.com

Dr. Vinodkumar L. Desai, Assistant Professor
Department of Computer Science
Government Science College
Chikhli, Navsari, Gujarat, India
vinodl_desai@yahoo.com

***Abstract*—** In today's scenario a word 'Big Data' used by researchers is associated with large amount of data which requires more resources likes processors, memories and storage capacity. Data can be structured and non-structured like text, images, and audio, video, social media data. Data generated by various sensor devices, mobile devices, social media. Data is stored into repository on the basis of their attributes like size, colours name. Data requires more storage space. In this paper we have evaluated performance of Hadoop MapReduce examples like TeraGen, TeraSor, TeraValidate. We have evaluated Hadoop Map Reduce performance by configuring compression related parameter and different compression algorithm like DEFLATE, Bzip2, Gzip , LZ4 on single Cluster through Word Count example. After evaluating compression algorithm through Word Count Example we found job elapsed time, I/O time and storage space requirement is reduced marginally along with increase in the CPU computation time.

Keywords*-TeraGen; TeraSort; WordCount; DEFLATE; Bzip2; Gzip; LZ4*

*****

I. INTRODUCTION

Recently growth of data is increased radically. Internet produces Terabytes of data in day. Data generated by various form like web site click, social networking site, various sensor, various scientific instruments, mobile phones. Currently databases like DBMS and RDBMS are use to store data and process it but RDBMS faces some challenges how to store huge volume data and how to process it.

Huge amount of data is called “Big Data” having attributes like volume, variety, velocity. Volume can be amount of data in size to store like Petabytes, Zotabyte, Yotta bytes. Variety can be structured data, unstructured data and semi structure data. Velocity can be growth of data generated.

Compression is used to compress data to improve overall performance like increase in response time, increase I/O performance and decrease storagerequirement.

The Apache Hadoop is open source framework for storing and processing huge amount of data across clusters [1]. Hadoop divided in two parts. HDFS and Map Reduce. HDFS means Hadoop Distributed File System. HDFS divides input in number of file with block size and store on distributed computer in data node. Map Reduce is a programming model which divides job which is processed by Mapper and Reducer function through programming. Hadoop provide various configuration file in xml format so that user can customize framework according to the requirement.

The rest of paper is organized as follows. Section I INTRODUCTION, Section II RELATED WORK, Section III ARCHITECTURE OF HADOOP YARN, Map Reduce Execution Flow, Introduction of TeraGen, TeraSort, TeraValidate, Explains various compression algorithm, Execution flow of Hadoop MapReduce WordCount with/without compression algorithm, Section IV EXPERIMENTAL SETUP, Section V EXPERIMENTS AND RESULTS, Section VI CONCLUSION AND FUTURE WORK.

II. RELATED WORK

Jeffrey Dean and Sanjay Ghemawat introduced Map Reduce programming model for processing and generating huge amount data. Map Reduce provide environment in which user can write program in Map Reduce functions which is automatically run across large clusters of machine. Map Reduce can process Petabytes of data [2].

Yanpei Chen etl developed a decision making algorithm which decide that compression is required or not based on evaluating configuration parameter of Hadoop MapReduce framework. Yanpei Chen etl. Find that some job using compression save energy up to 60% and improve datacentre energy efficiency [3].

Benjamin Welton, created a set of compression servervices for compression and decompression of dataset and evaluate performance of I/O using various data sets on a high performance computing cluster [4].

Shrinivas B. Joshi evaluated Hadoop Performance by different parameter configuration related to hardware and software and tune hadoop performance by parameter configuration [5].

Vinod Kumar Vavilapallietl designed and developed new Apache Hadoop Architecture called Hadoop Yarn. Yarn also knows as Yet another Resource *Negotiator*. Hadoop Yarn provides resource manager infrastructure, container per

 ISSN: 2321-8169
2839 – 2850

application, data node, name node, secondary name node, node manager, resource manager. Hadoop Yarn is base to MapReduce(Batch Processing), Tez (Interactive application processing),Storm(Streaming), Spark(In-Memory)[6].

Xuelian Lin etl, Introduced Predator, an experience guided configuration optimizer. Xuelian Lin classifies hadoop parameter in to different groups by tunable level [7].

Aggarwal, S etl, examined metrics generated by Hadoop Framework after job execution like no not byte read and write, job counters, job configuration parameter with value while job execution. Hadoop framework generates metrics for every MapReduce job, such as number of map and reduces tasks, number of bytes read/written to local file system and HDFS etc. Aggarwal, S etl, use these metrics and job configuration features such as format of the input/output files, type of compression used etc to find similarity among Hadoop jobs. Aggarwal, S etl, study the centroids and densities of these job clusters [8].

III. ARCHITECTURE OF HADOOP YARN

Architecture of Hadoop Yarn

Vinod Kumar Vavilapalli etl, design and developed next generation of Hadoop called YARN (Yet Another Resource Negotiator). YARN is based for Dryad, Giraph, Hoya, Hadoop MapReduce, REEF, Spark, Storm,Tez. Hadoop Yarn can be deployed on Single cluster or Multi Cluster node in distributed. DFS Means Distributed File System (Hadoop Distributed File System). Yarn has two part job tracker (Resource Manager) and task tracker (Application Manager). Figure 1 is Hadoop Yarn Architecture implemented on single Cluster environment on local host.

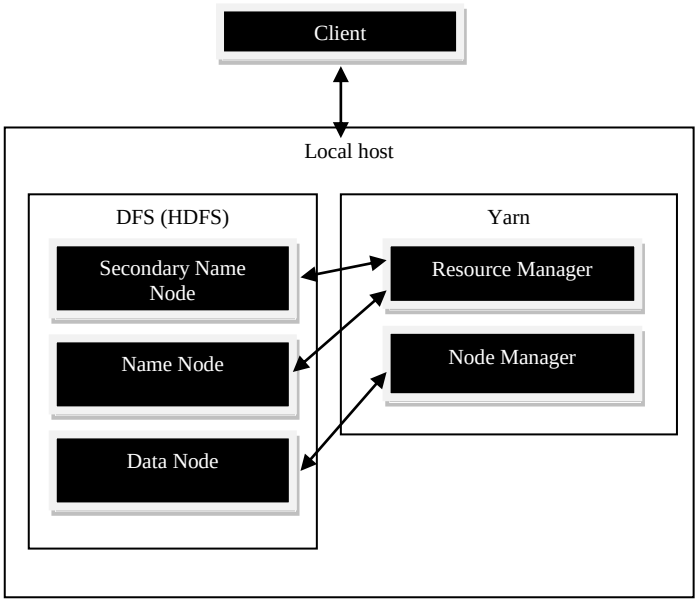

Fig. 1 Hadoop Yarn Architecture on Local host [6]

- Client

Client submits Hadoop Map Reduce to the Resource Manager. Resource pass accepted job to scheduler to run. If scheduler has enough number of resource then run the job and job is running.

- The Resource Manager

Resource Manager is called Job Tracker. The Resource Manager receives job which are submitted by client. Resource Manager has Application Manager. The application master is responsible job. Application manager manages resources for job increase and decrease resource for job and managing fault tolerance of job.

- Node Manager (NM)

Node Manager is worker for Yarn. Node Manager Manages container and monitor execution of container and provide set of service to the container.

- Name Node

Name node is manages location of file in different data node. Name node is store directory structure for all file. Job Tracker Resource Manager runs on a Name Node.

- Data Node

Data node is on which task tracker and data node manager is run. Task tracker is responsible for execution of task.

Map Reduce Execution Flow

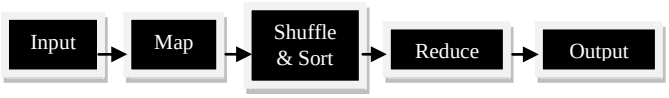

Fig.2 Map Reduce Execution Flow

User submit job to resource manager. Job is divided into no of block, map task and reduce task and distributed across the cluster.

Map

The Map receive task as a key and value pair and process key, value as per written program and generate intermediate data.

Shuffle and Sort

Shuffle and sort is intermediate data from map to reducer. Shuffle and sort is use to interchange; merge data internally to reduce computation.

Reduce

Reduce receive input as intermediate key, value pair generated by either by map or by shuffle and sort. And generate final output based on intermediate output.

Output

Output is given to the user who submitted job.

 ISSN: 2321-8169
2839 – 2850

Hadoop Yarn Map Reduce Word Count Example

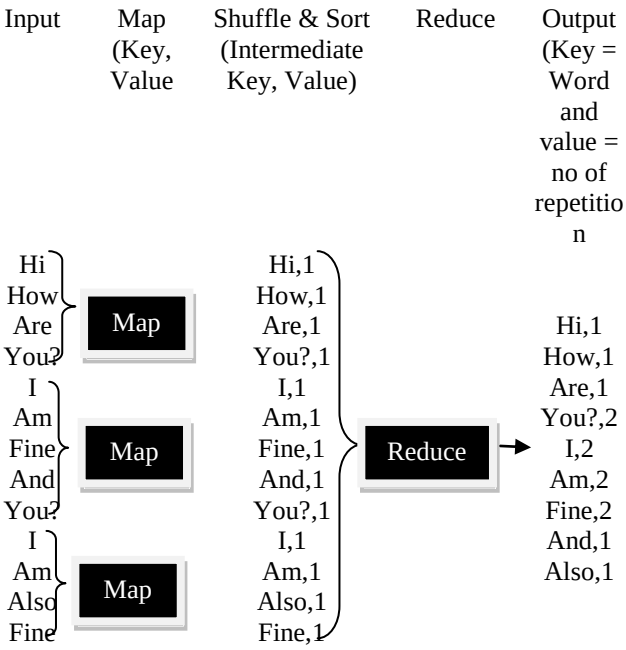

Fig. 3 Hadoop Yarn MapReduce WordCount Example

TeraSort benchmark suite

Fig. 4 TeraSort benchmark suites Example [10]

TeraSort is hadoop benchmark. TeraSort is use to sort 1TB or any amount of data. TeraSort benchmark has three steps [10].

TeraGen

TeraGen is use to generating any amount of data.

Syntax

hadoop jar hadoop-*examples*.jar teragen <number of 100-byte rows> <output dir>

Example

jar/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.4.1.jar teragen 10737418 /user/hduser/1gbteragen

Above example will generate 1GB data

TeraSort

TeraSort is use to sort data as per given input. TeraSort takes input from TeraGen and Sort data.

Syntax

hadoop jar hadoop-*examples*.jar terasort <input dir> <output dir>

Example

hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.4.1.jar terasort /user/hduser/1gbteragen /1gbsortoutput

Above example will sort 1GB data which are generated by TeraGen

TeraValidate

TeraValidate is to validate sorted data output. TeraValidate is validated output that sort output is globally acceptable.

Syntax

hadoop jar hadoop-*examples*.jar teravalidate <terasort output dir (= input data)> <teravalidate output dir>

Example

hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.4.1.jar teravalidate /1gbsortoutput /1gbteravalidate

COMPRESSION

Compression reduced physical file size and requires less storage space compare to uncompressed file. Compression increase I/O Performance, Elapsed Time. Compressions are in various compression formats, tools and algorithms with each with different features.

Below table lists some of comparisons Hadoop compression algorithm Compression formats, tool, algorithms and other attributes.

Compression format	Tool	Algorithm	Filename extension	Multiple files	Splittable
DEFLATE	N/A	DEFLATE	.deflate	No	No
gzip	Gzip	DEFLATE	.gz	No	No
ZIP	Zip	DEFLATE	.zip	Yes	Yes, at file boundaries
bzip2	bzip2	bzip2	.bz2	No	Yes
LZO	Lzop	LZO	.lzo	No	No

Table 1. A summary of compression formats [9]

Codec

Codec means compression and decompression. In Hadoop, a codec is an algorithm which is implementation of the Compression Codec Interface.

Compression format	Hadoop Compression Codec
DEFLATE	`org.apache.hadoop.io.compress.DefaultCodec`
gzip	`org.apache.hadoop.io.compress.GzipCodec`
bzip2	`org.apache.hadoop.io.compress.BZip2Codec`
LZO	`com.hadoop.compression.lzo.LzopCodec`

Table 2. Hadoop compression codecs [9]

 ISSN: 2321-8169
2839 – 2850

mapred-default.xml [1]

Parameter Name	Description	Default value	Possible value
mapreduce.output.fileoutputformat.compress	This parameter is use to decide job reduce output compressed or not?	false	true false
mapreduce.output.fileoutputformat.compress.type	If job reduce output compressed then in which format compressed one of NONE,RECORD or BLOCK	RECORD	NONE RECORD BLOCK.
mapreduce.output.fileoutputformat.compress.codec	If the job outputs are compressed then which code algorithm used.	org.apache.hadoop.io.compress.DefaultCodec	org.apache.hadoop.io.compress.DefaultCodec org.apache.hadoop.io.compress.GzipCodec org.apache.hadoop.io.compress.BZip2Codec com.hadoop.compression.lzo.LzopCodec
mapreduce.map.output.compress	This parameter is use to decide job map output compressed or not?	false	true false
mapreduce.map.output.compress.codec	If the job map outputs are compressed then which code algorithm used.	org.apache.hadoop.io.compress.DefaultCodec	org.apache.hadoop.io.compress.DefaultCodec org.apache.hadoop.io.compress.GzipCodec org.apache.hadoop.io.compress.BZip2Codec com.hadoop.compression.lzo.LzopCodec

Table 3 Compression related Configuration parameter in Hadoop [1]

Figure 5 explain flow of Hadoop MapReduce WordCount job Execution flow. Input is generated by TeraGen. User can count of word by two ways 1 Word Count Example without Compression Algorithm and 2 Word Count Example with Compression Algorithm. If user selects 1st way then WordCount takes input as TeraGen without doing any compression and count number of word. If user selects 2nd way then WordCount takes input as TeraGen with Compression algorithm and first make compression and decompression while calculating no of WordCount. User also has to select by which algorithm input data can be compress and decopressed.

Execution flow of Hadoop MapReduce WordCount with/without compression algorithm

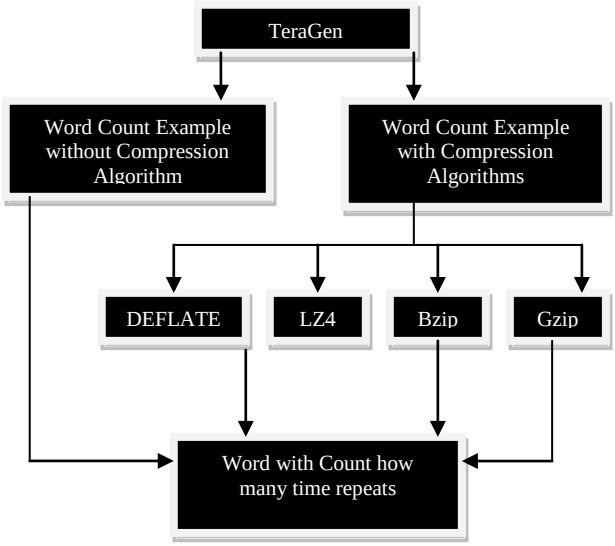

Fig. 5 Execution flow of Hadoop ReduceCount with/without compression algorithm Example

Section IV EXPERIMENTAL SETUP

In this section, we evaluated the performance of Hadoop Framework through Hadoop Map Reduce examples such as TeraGen, TeraSort, TeraValidate, and WordCount on a Single Cluster. Table 4 shows the software and hardware configuration of system on which performance is evaluated. Our focus is to evaluate performance metrics like Job Counters (Elapsed Time, Average Map Time), Map-Reduce Framework Counters(Map CPU time spent, Reduce CPU time spent, Total CPU time spent) , File System Counters(FILE: Number of bytes read/write) and Compression ratio of file through Hadoop Map Reduce Examples.

We evaluated performance of Hadoop Framework through different input data size from 1 GB up to 5GB of data and various compression algorithms using Hadoop Map Reduce Examples.

Sr. No	Hardware /Software	Command to know Installed Hardware/Software in Computer (Ubuntu)	Installed Hardware/Software Version in Computer
1	Operating System	cat /etc/*-release	Ubuntu 12.04.4 LTS VERSION_ID="12.04"
2	Java	java -version	java version "1.7.0_55"
3	Hadoop	hadoop version	Hadoop 2.4.1
3	openssh-server	-	openssh-server
4	CPU information of Machine	cat /proc/cpuinfo	model name : Intel(R) Core(TM)2 Duo CPU T5870 @ 2.00GHz Stepping: 13 cpu MHz: 2001.000 cpu cores: 2

Table 4. Environment Setup Information already installed

 ISSN: 2321-8169
2839 – 2850

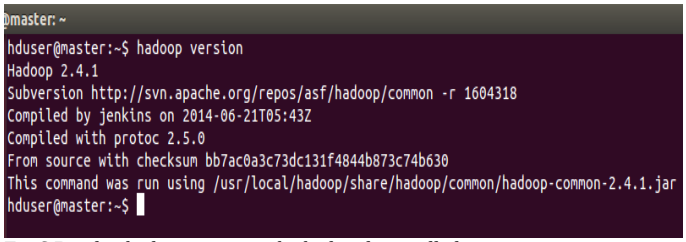

Fig.6 Display hadoop version which already installed

Running Hadoop
Format the namenode

$ hdfs namenode –format
$ start-dfs.sh
$ start-yarn.sh

$ mr-jobhistory-daemon.sh start historyserver

Check installation for localhost:
$ jps

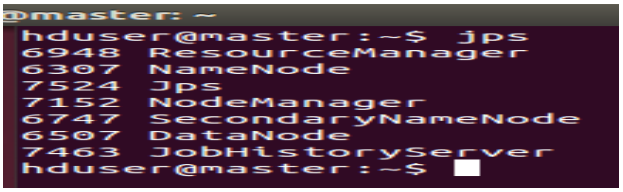

Fig. 7 Display status of hadoop started services

Check web interfaces of different services
YARN: http://localhost:8088

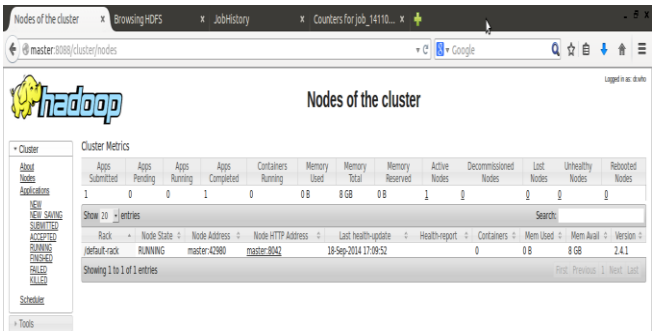

Fig. 8 Display Nodes of the cluster

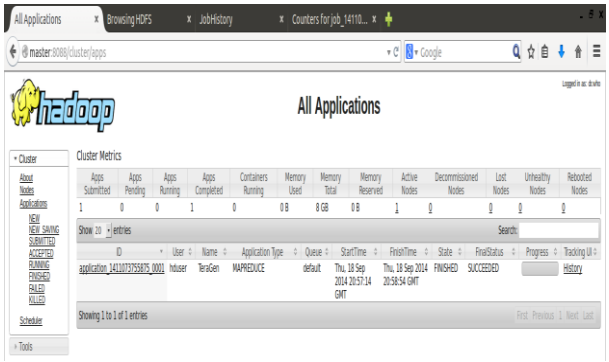

Fig. 9 Display no of applications

Namenode: http://localhost:50070

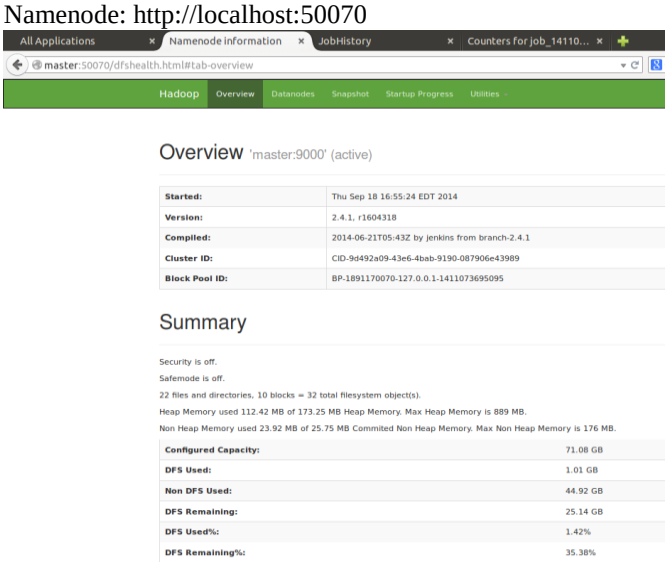

Fig. 10 Display Data Node Overview

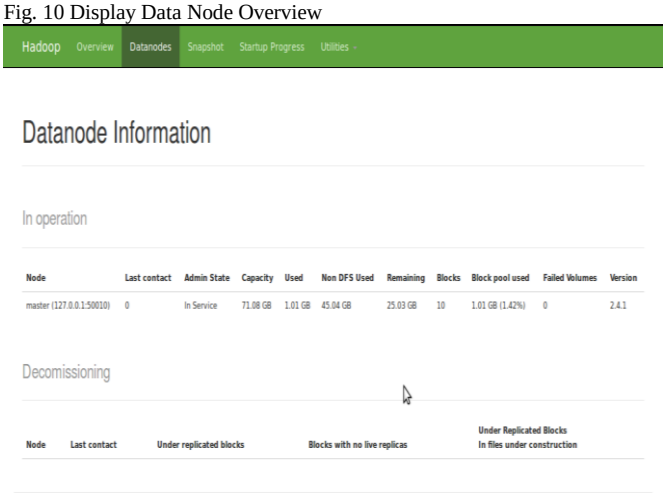

Fig. 11 Display Data Node Information

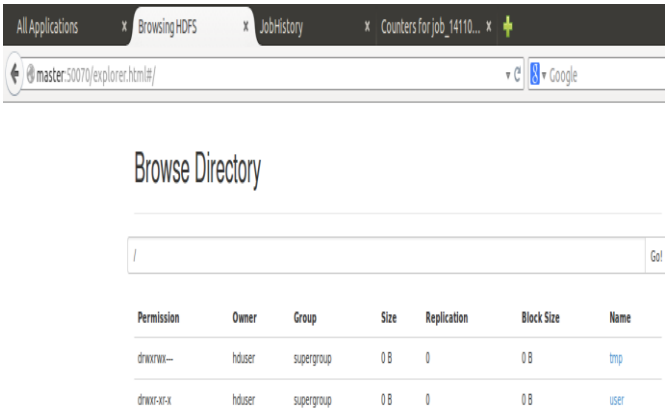

Fig. 12 Display no of file created in HDFS

 ISSN: 2321-8169
2839 – 2850

Section V EXPERIMENTS AND RESULTS

In this section, we present our analysis on the experiments performed on the Hadoop Map Reduce Framework on Single Cluster through various Hadoop Map Reduce Examples. In experiment, we compare compression algorithms namely DEFLATE, gzip, bzip2 LZO. Our focus is on studying the effects of different input data size and compression algorithms on the overall execution time and throughput of Hadoop Map Reduce Single Cluster.

A. Effect of Different Bytes Written using Hadoop Map Reduce TeraGen Example

Job Counters			Map-Reduce Framework Counters
Bytes Written	Elapsed Time (Min .Sec)	Average Map Time (Min .Sec)	CPU time spent (ms)
1(GB)	1.15	1.11	39100
2(GB)	2.13	2.9	68520
3(GB)	3.2	3.11	99810

Table 5 Job Counters of TeraGen Bytes Written

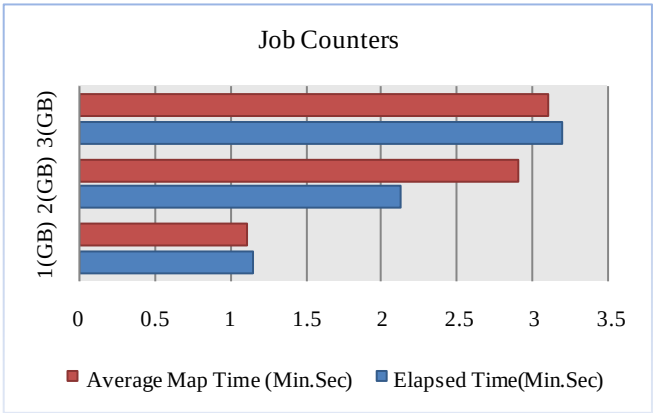

Fig. 13 Job Counters of TeraGen Bytes Written

When Byte Written Size increased respectively Elapsed Time , Average Map increased as shown in table 5 and Figure 13.

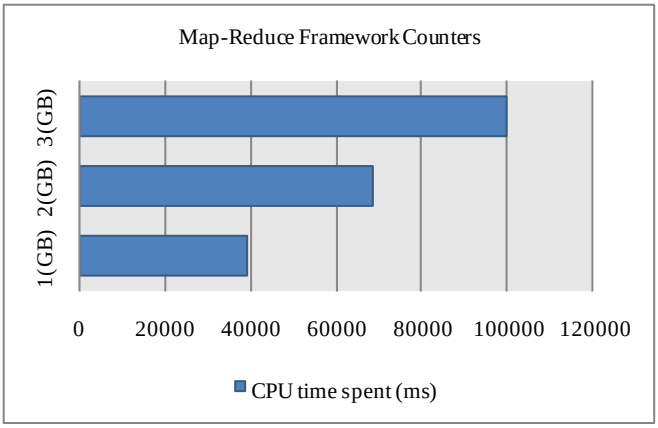

Fig. 14 CPU Time Spent of TeraGen Bytes Written

When Byte Written Size increased respectively CPU time spent increased as shown in table 5 and Figure 14.

B. Effect of Different Bytes Sort using Hadoop Map Reduce TeraSort Example

Job Counters					
Bytes Sort	Elapsed (Min.Sec)	Average Map Time (Min.Sec)	Average Reduce Time (Min.Sec)	Average Shuffle Time (Min.Sec)	Average Merge Time (Min.Sec)
1(GB)	8.45	2.54	1.39	3.18	0
2(GB)	18.2	3.37	3.45	9.26	1. 17
3(GB)	26.2	3.48	4.45	16.24	0.48

Table 6 Job Counters by TeraSort

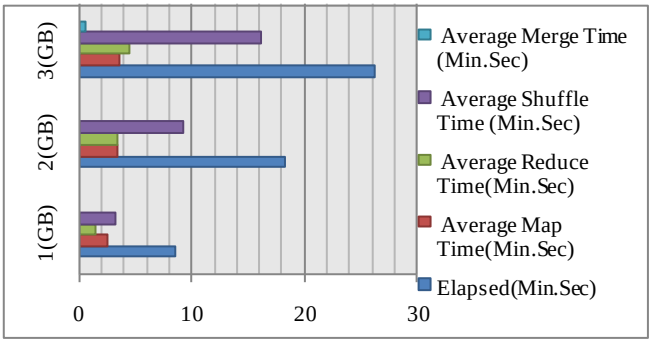

Figure 15 Job Counters of TeraSort

When Bytes Sort increased respectively Elapsed, Average Map Time, Average Reduce Time, Average Shuffle increased as shown in table 6 and Figure 15.

Map-Reduce Framework Counters			
Bytes Sorts	Map CPU time spent (ms)	Reduce CPU time spent (ms)	Total CPU time spent (ms)
1(GB)	127590	50780	178370
2(GB)	259550	123860	383410
3(GB)	375960	196390	572350

Table 7 Map-Reduce Framework Counters of TeraSort

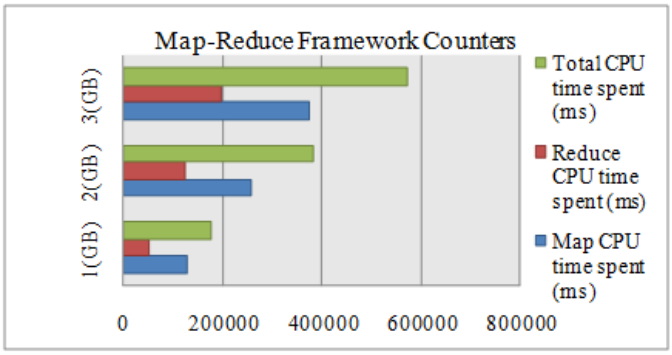

Fig. 16 Map-Reduce Framework Counters of TeraSort

When Bytes Sort increased respectively Map CPU time spent, Reduce CPU time spent, Total CPU time spent increased as shown in table 7 and Figure 16.

 ISSN: 2321-8169
2839 – 2850

C. Effect of Different Bytes Validated Sorted output using Hadoop Map Reduce TeraValidate Example

In this, we evaluated performance of Hadoop Map Reduce Framework Different Bytes Validated Sorted output using Hadoop Map Reduce TeraValidate Example.

Job Counters					
Validate Sorted output	Elapsed Time (Min. Sec)	Average Map Time (Min .Sec)	Average Reduce Time (Min .Sec)	Average Shuffle Time (Min .Sec)	Average Merge Time (Min. Sec)
1(GB)	1.23	0.48	0.1	0.24	0sec
2(GB)	1.57	1.36	0.1	0.13	0sec
3(GB)	2.8	1.52	0.1	0.8	0sec

Table 8 Job Counters of TeraValidate

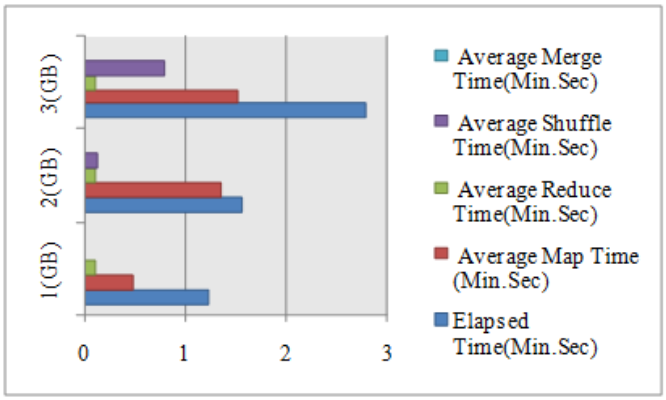

Fig.17 Job Counters of TeraValidate

When TeraSort output Validate size increased respectively Elapsed Time, Average Map Time increased as shown in table 8 and Figure 17.

Map-Reduce Framework Counters			
Validate Sorted output	Map CPU time spent (ms)	Reduce CPU time spent (ms)	Total CPU time spent (ms)
1(GB)	20030	1280	21310
2(GB)	37560	1270	38830
3(GB)	56300	1310	57610

Table 9 Map-Reduce Framework Counters by TeraValidate

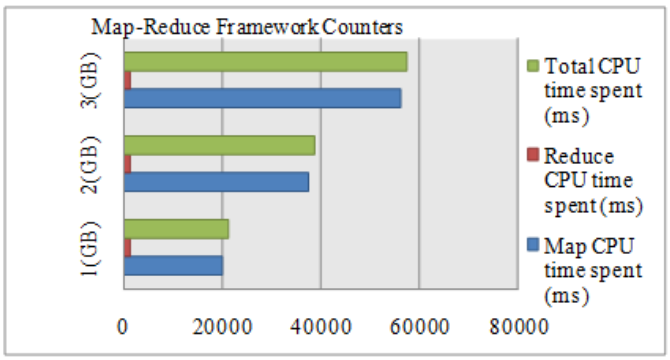

Fig.18 Map-Reduce Framework Counters by TeraValidate

When TeraSort output Validate size increased respectively Map CPU time spent, Total CPU time spent increased as shown in table 9 and Figure 18.

D. Effect of compression algorithm using Hadoop MapReduce Word Count Example

In this, we evaluated performance of Hadoop Map Reduce Framework through with/without various Compression algorithms using Hadoop Map Reduce Word Count Example.

File System Counters(FILE: Number of bytes read)			
WordCount of 1GB Data Generated By TeraGen (Without Compression/With Compression Map output)	Map Number of bytes read	Reduce Number of bytes read	Total Number of bytes read
Without Compression	1349851854	1349777949	2699629803
DEFLATE	301863933	302070640	603934573
Bzip2	200756866	201072316	401829182
Gzip	301864221	302070736	603934957
LZ4	641458168	641577529	1283035697

Table 10 File System Counters (FILE: Number of bytes read)

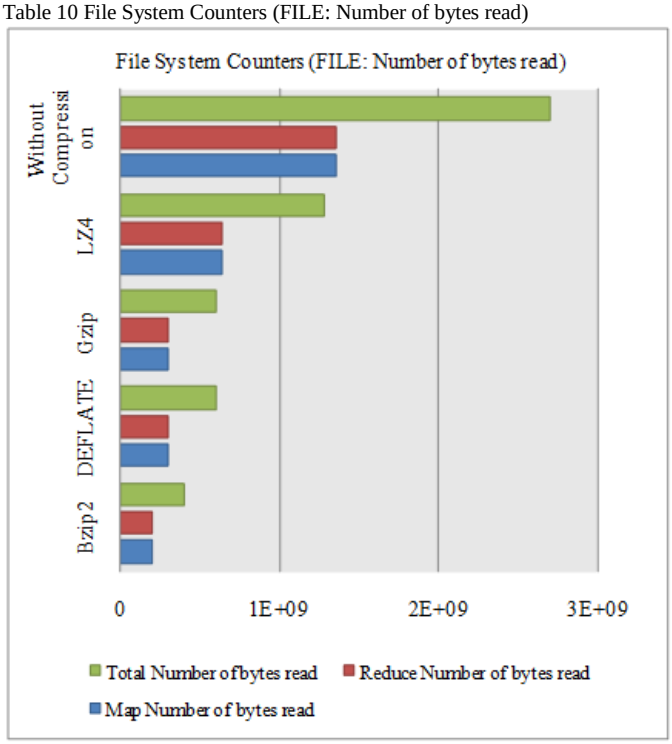

Fig. 19 File System Counters (FILE: Number of bytes read) by WordCount of 1GB Data Generated by TeraGen (Without Compression/With Compression Map output)

Without compress data required more data to read as compared to compressed data as showed in table 10 and Figure 19. Bzip2 algorithm requires less data to read as compared to other compression algorithm as shown in table 10 and Figure 19.

 ISSN: 2321-8169
2839 – 2850

File System Counters (FILE: Number of bytes written)			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)	Map Number of bytes written	Reduce Number of bytes written	Total Number of bytes written
Bzip2	402574222	202736347	605310569
DEFLATE	604679629	304523725	909203354
Gzip	604679997	304523819	909203816
LZ4	1283780729	646683007	1930463736
Without Compression	2700374867	1360416251	4060791118

Table 11 File System Counters (FILE: Number of bytes written) by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)

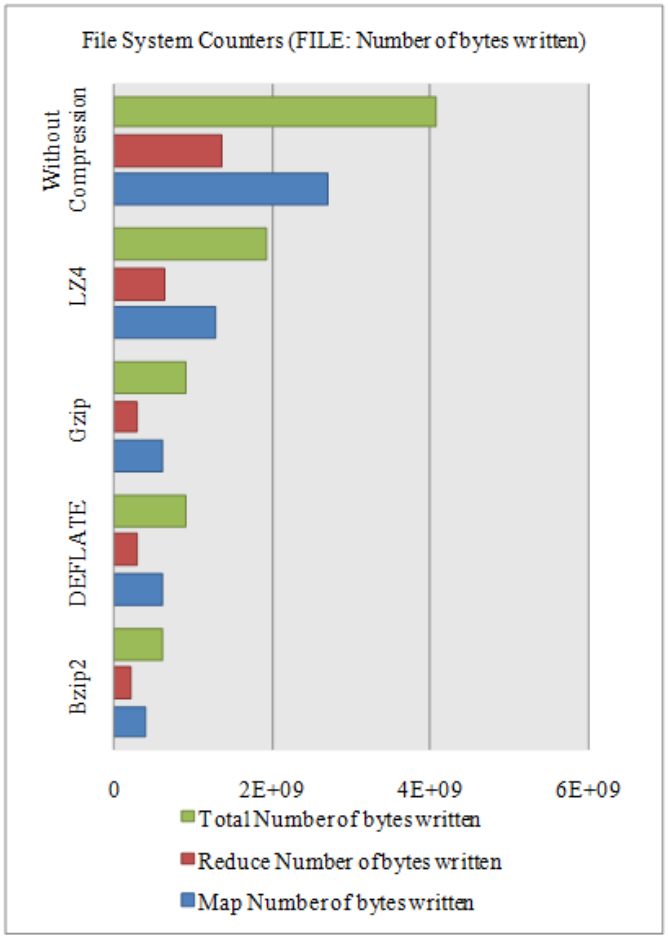

Fig. 20 File System Counters (FILE: Number of bytes written) by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map output)

Without compress data required more data to written as compared to compressed data as showed in table 11 and Figure 20. Bzip2 algorithm requires less data to written as compared to other compression algorithm as shown in table 11 and Figure 20.

Job Counters					
WordCount of 1GB Data Generated By TeraGen (Without Compression / With Compression Map output)	Elapsed (Min .Sec)	Average MapTime (Min. sec)	Average Reduce Time (Min .Sec)	Average Shuffle Time (Min .Sec)	Average Merge Time (Min. Sec)
LZ4	9.26	3.2	1.5	2.54	0
DEFLATE	9.31	4.2	1.39	3.17	0
Gzip	10.25	4.31	1.42	3.2	0.2
Without Compression	12.53	4.19	2.12	5.36	0
Bzip2	26.16	11.4	4.23	9.4	0.1

Table 12 Job Counters by WordCount of 1GB Data Generated By TeraGen (Without Compression / With Compression Map output)

Bzip2 Compression data required more Elapsed for WordCount as Compared to others as shown table 12 and Figure 21. Bzip2 Compression requires more computation as compared others.

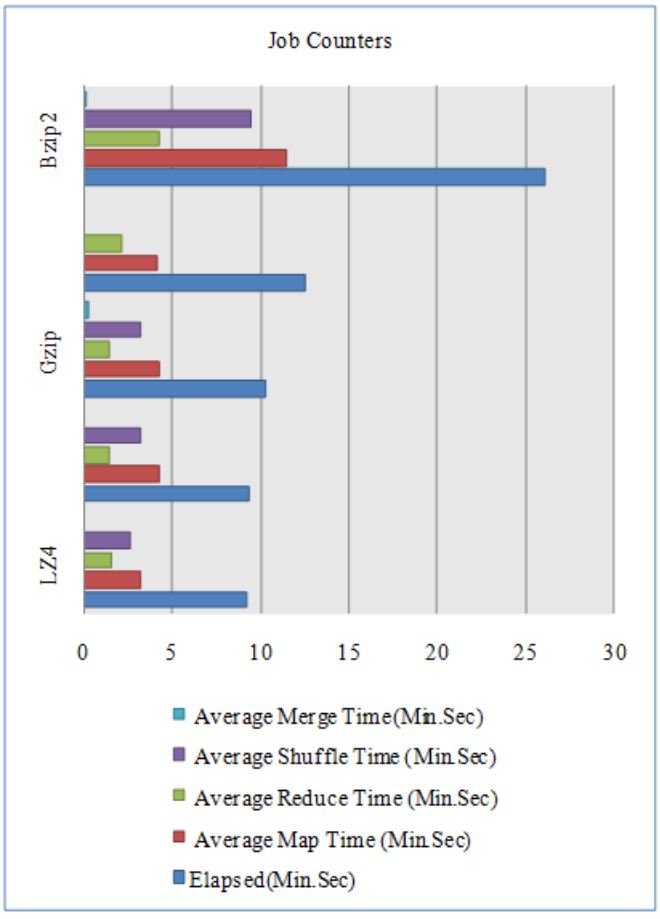

Fig. 21 Job Counters by WordCount of 1GB Data Generated By TeraGen (Without Compression / With Compression Map output)

 ISSN: 2321-8169
2839 – 2850

Map-Reduce Framework Counters			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)	Map CPU time spent (ms)	Reduce CPU time spent (ms)	Total CPU time spent (ms)
LZ4	183080	33790	216870
Without Compression	185110	38860	223970
Gzip	443960	152230	596190
DEFLATE	444190	153780	597970
Bzip2	1533160	792520	2325680

Table 13 Map-Reduce Framework Counters by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)

Bzip2 Compression data Map CPU time spent, Reduce CPU time spent, Total CPU time spent are more as compared to others as shown table 13 and Figure 22. Bzip2 Compression requires more computation as compared others.

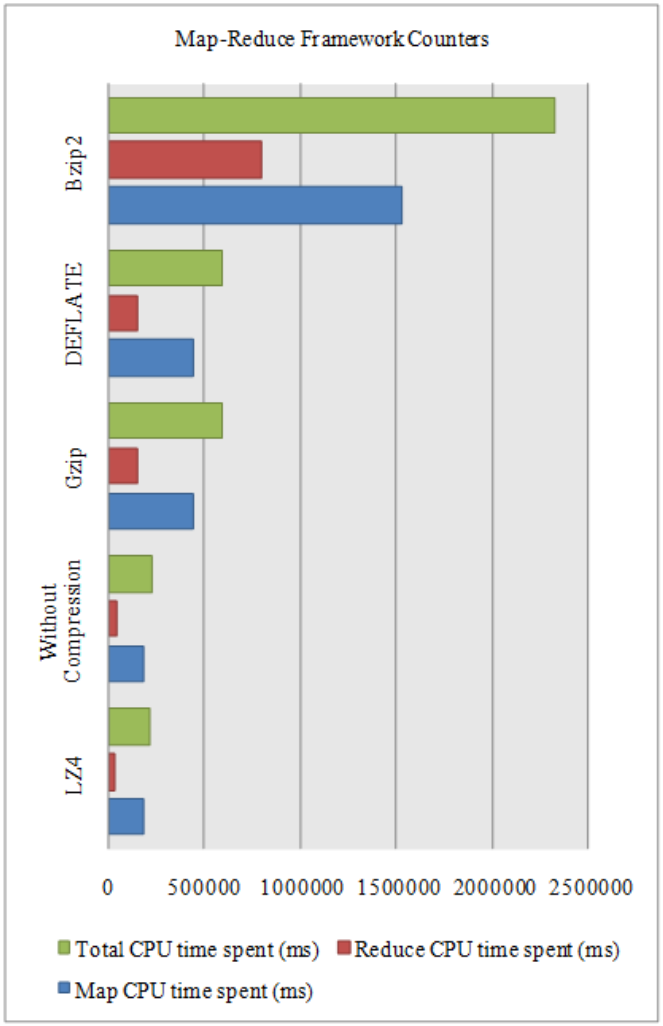

Fig. 22 Map-Reduce Framework Counters by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map output)

Job Counters				
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map +Reduce output)	Elapsed (Min.Sec)	Average Map Time (Min.Sec)	Average Reduce Time (Min.Sec)	Average Shuffle Time (Min.Sec)
LZ4	6.33	2.37	1	2.29
Gzip	9.52	3.5	2.39	2.59
DEFLATE	10.42	4.13	2.39	3.15
Without Compression	12.53	4.19	2.12	5.36
Bzip2	35.17	11.34	13.39	8.58

Table 14 Job Counters by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map + Reduce output)

Bzip2 Compression data required more Elapsed for WordCount as Compared to others as shown table 14 and Figure 23. Bzip2 Compression requires more computation as compared others.

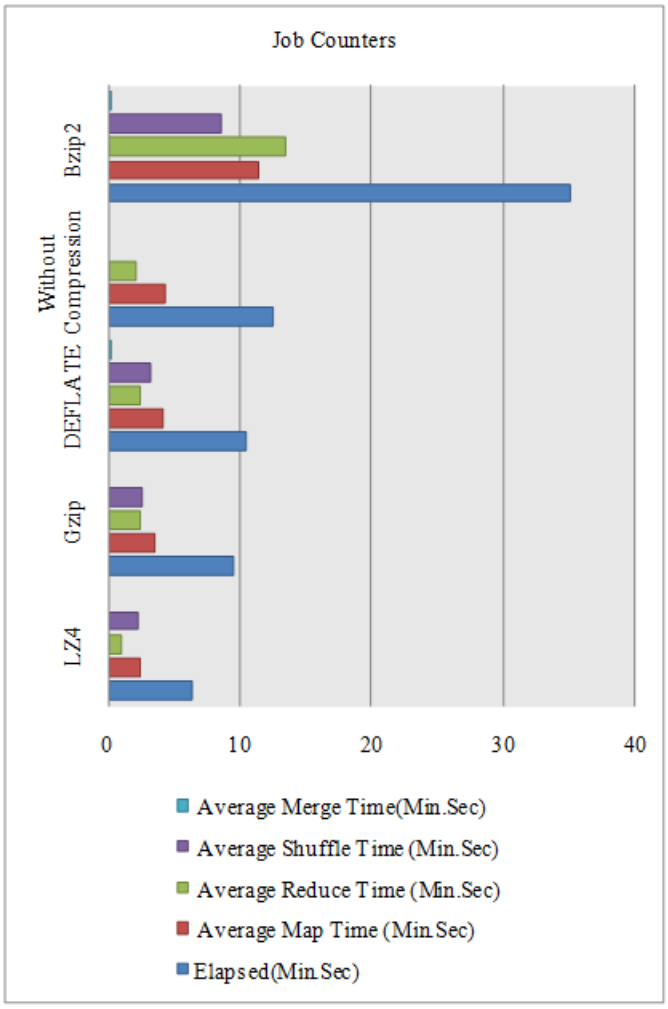

Fig. 23 Job Counters by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map + Reduce output)

 ISSN: 2321-8169
2839 – 2850

Map-Reduce Framework Counters			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map +Reduce output)	Map CPU time spent (ms)	Reduce CPU time spent (ms)	Total CPU time spent (ms)
LZ4	182500	32930	215430
Without Compression	185110	38860	223970
DEFLATE	439510	38530	478040
Gzip	441930	38160	480090
Bzip2	1527480	232560	1760040

Table 15 Map-Reduce Framework Counters by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

Bzip2 Compression data Map CPU time spent, Reduce CPU time spent, Total CPU time spent are more as compared to others as shown table 15 and Figure 24. Bzip2 Compression requires more computation as compared others.

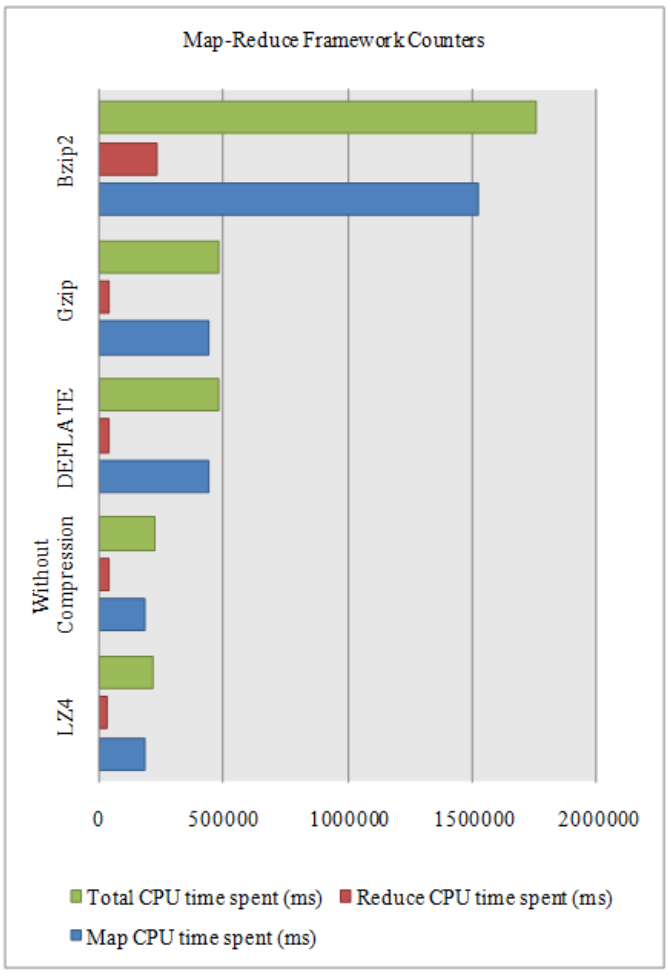

Fig. 24 Map-Reduce Framework Counters by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

File System Counters (FILE: Number of bytes read)			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map +Reduce output)	Map Number of bytes read	Reduce Number of bytes read	Total Number of bytes read
Bzip2	200756866	201072316	402574214
DEFLATE	301863933	302070640	603934573
Gzip	301864221	302070736	603934957
LZ4	641458168	641577529	1283035697
Without Compression	1349851854	1349777949	2699629803

Table 16 File System Counters (FILE: Number of bytes read) By WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

Without compress data required more data to read as compared to compressed data as showed in table 16 and Figure 25. Bzip2 algorithm requires less data to read as compared to other compression algorithm as shown in table 16 and Figure 25.

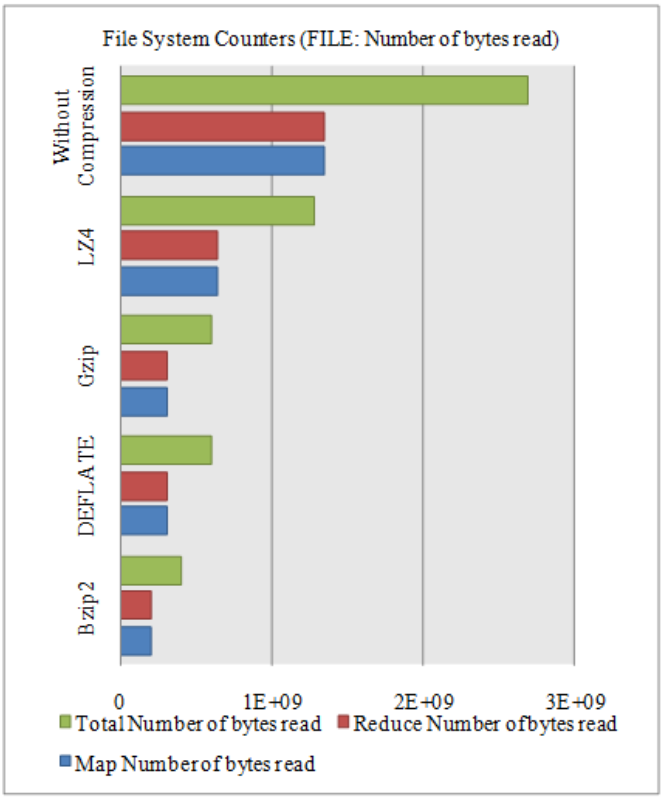

Fig. 25 File System Counters (FILE: Number of bytes read) by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

File System Counters (FILE: Number of bytes Written)			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map +Reduce output)	Map Number of bytes Written	Reduce Number of bytes Written	Total Number of bytes Written
Bzip2	402574214	202736346	605310560
DEFLATE	604679613	304523723	909203336
Gzip	604680005	304523820	909203825
LZ4	1283780681	646683001	1930463682
Without Compression	2700374867	1360416251	4060791118

Table 17 File System Counters (FILE: Number of bytes Written) by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

 ISSN: 2321-8169
2839 – 2850

Without compress data required more data to Written as compared to compressed data as showed in table 17 and Figure 26. Bzip2 algorithm requires less data to read as compared to other compression algorithm as shown in table 17 and Figure 26.

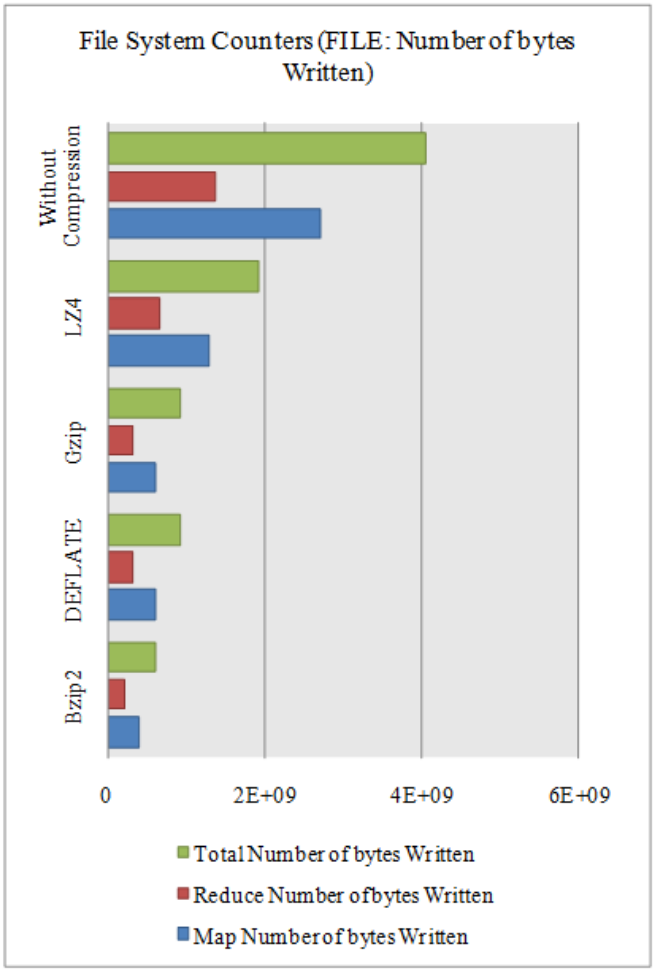

Fig.26 File System Counters (FILE: Number of bytes Written) by WordCount of 1GB Data Generated by TeraGen (Without Compression /With Compression Map + Reduce output)

E. Compression Ratio

In this, we calculate compression ratio of file using various compression algorithms.

Formula to calculate compression ratio:
Compression ratio: (Uncompressed data size - compressed data size)/uncompressed data size percentage. For example: (500 -400)/500 = 0.2, or 20%

Compression ratio			
WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map	Compression ratio of 1 GB Byte Read (%)	Compression ratio of 1 GB Byte Write (%)	Compression ratio of 1 GB Byte Read + Write
Without Compression	0	0	0.00
LZ4	52.47	52.46	52.47
Gzip	77.63	77.61	77.62
DEFLATE	77.63	77.61	77.62
Bzip2	85.12	85.09	85.10

Table 18 File Compression ratio by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)

When Compression is applied using specified algorithms, compression ratio of 1 GB Byte in read, write and read + write is increased as shown in table 18 and Figure 27. Compression ratio of LZ4, Gzip, DEFLATE, Bzip2 are 52.47 %, 77.62%, 77.62%, 85.10% respectively as shown in table 18 and Figure 27.Compression ratio of Bzip2 algorithm is 85.10% as compared to others. Bzip2 algorithm data require less data storage as compared to others. Compression reduces file size as compared to uncompressed data. Compressed data requires less storage as compared to uncompressed data.

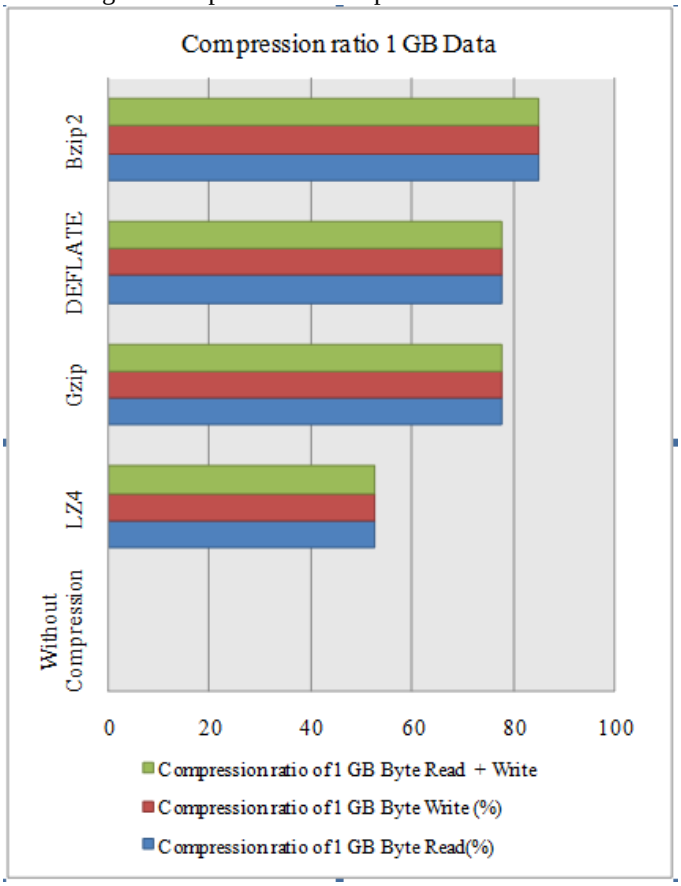

Fig. 27 File Compression ratio by WordCount of 1GB Data Generated By TeraGen (Without Compression /With Compression Map output)

VI. CONCLUSIONS AND FUTURE WORK

When size of data increased respectively storage, CPU time increased. Uncompressed data required more storage; data I/O (read/write) decreased and required more CPU Computational. Compression algorithm compressed data so it requires less size for storage as compare to uncompressed data. Bzip algorithm compression ratio is 85.10% as shown in table 18 and Figure 27 so that it required less storage space to store data; amount of I/O (Read/Write) is low as compared to others. Bzip algorithm required more computation as compared to others.

Bzip2 algorithms Map CPU time spent, Reduce CPU time spent, Total CPU time spent, Elapsed time, Average MapTime, Average Reduce Time, Average Shuffle Time is more as compared to other algorithms as shown in table 12, 13 and figure 21, 22.

In this paper only 1GB data and Hadoop MapReduce Yarn WordCount example is used for performance evaluation of compression algorithms. We are in process with other examples having high volume of data and computation to evaluate performance.

REFERENCES

[1] http://hadoop.apache.org/
[2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107–113, 2008.
[3] Yanpei Chen, Archana Ganapathi, and Randy H. Katz. 2010. To compress or not to compress - compute vs. IO tradeoffs for mapreduce energy efficiency. In Proceedings of the first ACM SIGCOMM workshop on Green networking (Green Networking '10). ACM, New York, NY, USA, 23-28. DOI=10.1145/1851290.1851296 http://doi.acm.org/10.1145/1851290.1851296
[4] Benjamin Welton, Dries Kimpe, Jason Cope, Christina M. Patrick, Kamil Iskra, and Robert Ross. 2011. Improving I/O Forwarding Throughput with Data Compression. In Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER '11). IEEE Computer Society, Washington, DC, USA, 438-445. DOI=10.1109/CLUSTER.2011.80 http://dx.doi.org/10.1109/CLUSTER.2011.80
[5] Shrinivas B. Joshi. 2012. Apache hadoop performance-tuning methodologies and best practices. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering* (ICPE '12). ACM, New York, NY, USA, 241-242. DOI=10.1145/2188286.2188323 http://doi.acm.org/10.1145/2188286.2188323
[6] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. 2013. Apache Hadoop YARN: yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing* (SOCC '13). ACM, New York, NY, USA, Article 5, 16 pages. DOI=10.1145/2523616.2523633 http://doi.acm.org/10.1145/2523616.2523633
[7] Xuelian Lin, Wenzhong Tang, and Kewen Wang. 2012. Predator — An experience guided configuration optimizer for Hadoop MapReduce. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)* (CLOUDCOM '12). IEEE Computer Society, Washington, DC, USA, 419-426. DOI=10.1109/CloudCom.2012.6427486 http://dx.doi.org/10.1109/CloudCom.2012.6427486
[8] Aggarwal, S.; Phadke, S.; Bhandarkar, M., "Characterization of Hadoop Jobs Using Unsupervised Learning," Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, vol., no., pp.748,753, Nov. 30 2010-Dec. 3 2010 doi:10.1109/CloudCom.2010.20 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5708526&isnumber=5708426
[9] Tom White. Hadoop: The Definitive Guide. O'Reilly, 3rd Edition, 2012.
[10] http://hadoop.apache.org/common/docs/r0.20.2/api/org/apache/hadoop/examples/terasort/TeraGen.html