Volume-3, Issue-05, May 2016 ISSN: 2349-7637 (Online)

RESEARCH HUB – International Multidisciplinary Research Journal (RHIMRJ)

Research Paper

Available online at: www.rhimrj.com

Apache Hadoop Yarn Performance Analysis and Tuning through guided Configuration Parameter Setting on Single Cluster

Mr. Bhavin J. Mathiya[1st]
Research Scholar,
C.U. Shah University
Wadhwan City, Gujarat, (India)

Dr. Dhaval S. Vyas[2nd]
Dean- M.Phil Programme
C.U. Shah University
Wadhwan City, Gujarat, (India)

Dr. Vinodkumar L. Desai[3rd]
Department of Computer Science
Government Science College
Chikhli, Navsari, Gujarat, (India)

***Abstract:** In recent year huge amount of data generated through various kinds of social networking sites, scientific devices, Sensors is called big data. It is big question how to store, process and analyze these big data. Apache Hadoop Yarn is open source framework which provides solution for big data. Apache Hadoop Yarn provides components like HDFS to store data in locally as well as distributed and MAPREDUCE programming for processing this data. When user deploys Apache Hadoop Yarn, it provides hundreds of default configuration parameters which are common for all kind of jobs which leads to under utilization of or over utilization of resources like CPU, I/O, Memory etc. But in real situation all jobs requires different parameter configuration. In this research Apache Hadoop Yarn performance analysis done through various kind of benchmark like TeraGen, TeraSort, TeraValidate, Word Count, TestDFSIO Read and TestDFSIO Write on single cluster hadoop environment and tune performance through the customization of Hadoop Configuration Parameters. Apache Hadoop Yarn performance is improved through customization of parameter configuration value as compare to default configuration parameter value. Finally provides guided configuration parameter setting value so that it helps other user for performance tuning of apache hadoop.*

***Keywords:** Apache Hadoop Yarn, HDFS, MapReduce, TeraGen, TeraSort, Tera Validate, TestDFSIO(Read), TestDFSIO(Write) WordCount.*

I. INTRODUCTION

Recently Internet Of the things (IOT) is popular. In IOT every devices connected through Internet. Each devices like mobile phone, servers, computers, scientific instruments, web sites etc and capable to generates vast amount of data rapidly, it can be structure, un structure and semi structure data connected through internet. Big Data means variety, velocity, volume of data is high. IOT and Big Data both related with each other. It is difficult for tradition system like DBMS and RDBMS for storing, processing and analysing Big Data.

Apache Hadoop Yarn is a framework developed by Apache Software Foundation to solve Big Data problem. It provides MAPREDUCE functionality for programming purpose and user can write programme logic using Mapper and Reducer based on Key Value Pairs and Hadoop Distributed File System (HDFS) functionality for Storing data locally as well as districted manner into various nodes. Apache Hadoop Yarn provides more than hundreds default parameter for each application. But in real word every application requires different type of resource consumption and requirement is different and it creates situation that resources utilization can be over or under. Apache Hadoop Yarn provides functionality that user can be customization default configuration parameter value using configuration file, writing programming logic and at run time passing value. Proper parameter configuration value can make proper resource like CPU, Memory, and I/O utilization as per application needs. In this Apache Hadoop Yarn Performance Analysis and evaluation done through various kind dataset and various kind parameter configuration setting value and tunes its performance on Single Cluster. Finally provides guided parameter configuration values to tune performance.

The rest of paper is organized as follows. Section II Related Work, Section III Apache Hadoop Yarn Architecture, Section IV Implementation and execution of different Hadoop MapReduce application using parameter configuration value on Single Cluster, Section V Result and Discussion, Section VI conclusion and future work.

II. Related Work

Jeffrey Dean and Sanjay Ghemawat introduced MapReduce programming model for processing and generating huge amount data. MapReduce provide environment in which user can write program in MapReduce functions which is automatically run across large clusters of machine. MapReduce can process Petabytes of data [2]. Rong Gu et al. [3] explore an approach to improve the performance of the Hadoop MapReduce framework by optimizing the job and task execution mechanism by analyzing the job and task execution mechanism in MapReduce framework. Rong Gu et al. [3] Experimental results show that compared to the standard Hadoop, SHadoop can achieve stable performance improvement by around 25% on average for comprehensive benchmarks without losing scalability and speedup. Zhang et al. [4] explores the efficiency and performance accuracy of the bounds-based performance model for predicting the MapReduce job completion times in heterogeneous Hadoop clusters. Yang et al. [5] propose a statistic analysis approach to identify the relationships among workload characteristics, Hadoop configurations and workload performance. Lakkimsetti et al. [6] provide a framework that optimizes MapReduce programs that run on large datasets. Babu et al. [7] make a case for techniques to automate the setting of tuning parameters for MapReduce programs.

III. Apache hadoop yarn Architecture

Apache Hadoop Yarn is open source framework developed by Apache Software Foundation. Vinod Kumar Vavilapalli et al. [8] summarize the design, development, and current state of deployment of the next generation of Hadoop’s compute platform: YARN. Apache Hadoop Yarn provides Hadoop Distributed File System for storing file as a block locally as well as distributelly and MAPREDUCE programming for writing custom program logically as per Mapper and Reducer provided based on Key Value Pair. Client submits application to Resource Manager and Resource Manager Transfers submitted job to Node Manager to storing, processing and analysis purpose. Node Manager contains Application Manager and container. After processing applications Node Manager Transfers result back to Resource Manager and Resource Manager Transfers result back to the client who submitted application. Finally client receives response from Resource Manager.

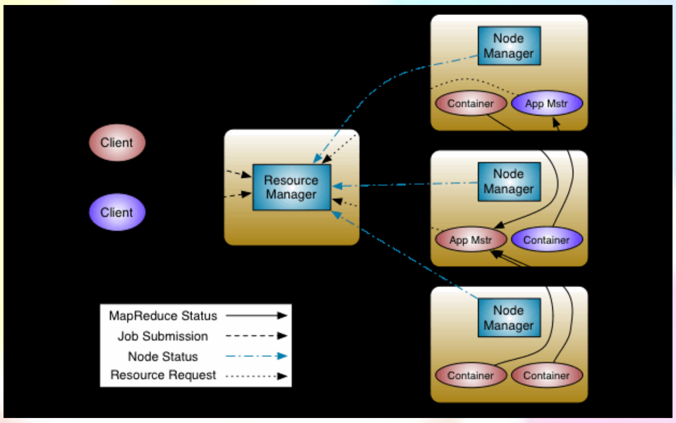

Fig. 1 Hadoop Yarn Architecture [1]

IV. Environment Setup

These experiments carried out on single cluster node and Ubntu 12.04.4 LTS Operating System and Java version 1.7, Openssh Server Apache Hadoop 2.4.1 Softwares are installed.

V. Results and Observations

Hadoop Performance tuning using Customization of Configuration Parameter

5.1 Hadoop Performance tuning through mapreduce. output.fileoutputformat.compress,mapreduce.output. fileoutputformat. compress.codec

Data Compression Related Configuration Parameter Customization

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. Table 1. A summary of compression formats.

Table 1: Hadoop Data Compression Related Configuration Parameter Customization
mapreduce.output.fileoutputformat.compress and mapreduce.output.fileoutputformat.compress.codec

Sr. No	**mapreduce.output. fileoutputformat.compress**	**mapreduce.output. fileoutputformat.compress.codec**
1	FALSE(Default)	org.apache.Hadoop.io.compress.DefaultCodec (Default)
2	TRUE	org.apache.Hadoop.io.compress.DefaultCodec
3	TRUE	org.apache.Hadoop.io.compress.BZip2Codec
4	TRUE	org.apache.Hadoop.io.compress.GzipCodec
5	TRUE	org.apache.Hadoop.io.compress.Lz4Codec

Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size, Configuration Parameter

Table 2 depicts the Intel's HiBench Benchmark (Micro Benchmarks) Execution time (Sec) for the different data sizes as well as different Hadoop's Customized Compression Parameter Configuration Settings like mapreduce.output.fileoutputformat.compress, mapreduce. output. fileoutputformat compress.codec. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Table 2: Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size, Configuration Parameter

	Execution time (Sec)				
Data Size(GB)	**mapreduce.output.fileoutputformat.compress = FALSE**	**mapreduce.output.fileoutputformat.compress=TRUE, mapreduce.output.fileoutputformat.compress.codec=org.apache.Hadoop.io.compress.DefaultCodec**	**mapreduce.output.fileoutputformat.compress=TRUE , mapreduce.output.fileoutputformat.compress.codec=org.apache.Hadoop.io.compress.BZip2Codec**	**mapreduce.output.fileoutputformat.compress=TRUE , mapreduce.output.fileoutputformat.compress.codec=org.apache.Hadoop.io.compress.GzipCodec**	**mapreduce.output.fileoutputformat.compress=TRUE, mapreduce.output.fileoutputformat.compress.codec=org.apache.Hadoop.io.compress.Lz4Codec**
TeraGen (1 GB)	**56**	60	58	58	58
TeraGen (2 GB)	107	104	**100**	**100**	102
TeraGen (3 GB)	150	150	145	150	**141**
TeraSort (1 GB)	624	534	548	562	**475**
TeraSort (2 GB)	1182	1235	1079	1183	**1039**
TeraSort (3 GB)	1755	**1560**	2045	1773	1702
TeraValidate (1 GB)	54	44	37	**36**	37
TeraValidate (2 GB)	**73**	89	86	103	98
TeraValidate (3 GB)	108	109	103	105	**102**
WordCount (1 GB)	672	**610**	991	707	664
WordCount (2 GB)	**1209**	1344	2147	1590	1224
WordCount (3 GB)	2148	2042	3148	3000	**1947**
TestDFSIO –Write (1 GB)	61	65	63	61	**60**
TestDFSIO -Write (2 GB)	99	106	**98**	102	99
TestDFSIO -Write (3 GB)	163	**135**	147	164	156
TestDFSIO -Read (1 GB)	**39**	45	43	41	52
TestDFSIO -Read (2 GB)	77	77	78	77	76
TestDFSIO -Read (3 GB)	123	109	108	**107**	108

Hadoop Performance Tuning through mapreduce. map.output.compress, mapreduce.map.output.compress. codec

Table 3: Hadoop Data Compression Related Configuration Parameter Customization mapreduce.map.output.compress and mapreduce.map.output.compress.codec

Sr. No	**mapreduce.map.output.compress,**	**mapreduce.map.output.compress.codec**
1	FALSE(Default)	org.apache.Hadoop.io.compress.DefaultCodec (Default)
2	TRUE	org.apache.Hadoop.io.compress.DefaultCodec
3	TRUE	org.apache.Hadoop.io.compress.BZip2Codec
4	TRUE	org.apache.Hadoop.io.compress.GzipCodec
5	TRUE	org.apache.Hadoop.io.compress.Lz4Codec

Table 4 depicts the Intel's HiBench Benchmark (Micro Benchmarks) Execution time (Sec) for the different data sizes as well as different Hadoop's Customized Compression Parameter Configuration Settings like mapreduce.map.output.compress. mapreduce.map.output. compress. codec. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Table 4: Execution time (Sec) of Hadoop Data Compression Related Configuration Parameter Customization mapreduce.map.output.compress and mapreduce.output.fileoutputformat.compress.codec

Execution time (Sec)					
Data Size (GB)	mapreduce.map.output.compress = false	mapreduce.map.output.compress = true, mapreduce.output.fileoutputformat.compress.codec = **org.apache.Hadoop.io.compress.DefaultCodec**	mapreduce.map.output.compress = true, mapreduce.output.fileoutputformat.compress.codec = **org.apache.Hadoop.io.compress.BZip2Codec**	mapreduce.map.output. compress = true, mapreduce.output.fileoutputformat.compress.codec = **org.apache.Hadoop.io.compress.GzipCodec**	mapreduce.map.output. compress = true, mapreduce.output.fileoutputformat.compress.codec = **org.apache.Hadoop.io.compress.Lz4Codec**
TeraGen (1 GB)	56	59	57	**52**	**52**
TeraGen (2 GB)	107	105	94	**93**	96
TeraGen (3 GB)	150	146	**138**	140	141
TeraSort (1 GB)	624	**343**	1066	478	346
TeraSort (2 GB)	1182	**800**	2361	942	879
TeraSort (3 GB)	1755	1237	3900	1436	**1163**
TeraValidate (1 GB)	54	46	48	**43**	44
TeraValidate (2 GB)	**73**	95	90	91	90
TeraValidate (3 GB)	108	**103**	105	**103**	**103**
WordCount (1 GB)	672	495	1288	**452**	474
WordCount (2 GB)	1209	831	2643	803	**649**
WordCount (3 GB)	2148	1904	4604	1463	**1361**
TestDFSIO -Write (1 GB)	61	62	61	**60**	**60**
TestDFSIO -Write (2 GB)	99	102	107	**95**	97
TestDFSIO -Write (3 GB)	163	**136**	140	138	144
TestDFSIO -Read (1 GB)	61	51	52	**50**	51
TestDFSIO -Read (2 GB)	77	**76**	**76**	**76**	**76**
TestDFSIO -Read (3 GB)	123	**108**	112	110	**108**

Hadoop performance tuning through dfs.blocksize Parameter Customization

The default blocks size for new files, in bytes. You can use the following suffix (case insensitive): k(kilo), m(mega), g(giga), t(tera), p(peta), e(exa) to specify the size (such as 128k, 512m, 1g, etc.), Or provide complete size in bytes (such as 134217728 for 128 MB). Table 5 depicts dfs.blocksize configuration parameter with different values.

Table 5: dfs.blocksize (MB)

Sr. No	**dfs.blocksize (MB)**	**dfs.blocksize (Bytes)**
1	**128 MB (Default)**	**134217728**
2	**64 MB**	**67108864**
3	**256 MB**	**268435456**
4	**512MB**	**536870912**
5	**1024MB**	**1073741824**

Table 6 depicts the Intel's HiBench Benchmark (Micro Benchmarks) Execution time (Sec) for the different data sizes as well as different Hadoop's Customized dfs.blocksize Parameter Configuration Settings like 128 MB (Default), 64 MB, 256 MB, 512MB. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Table 6: Execution time (Sec) with Configuration Parameter: dfs.blocksize (MB)

Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size					
Configuration Parameter : dfs.blocksize					
Execution time (Sec)					
Data Size (GB)	**128 MB (Default)**	**64 MB**	**256 MB**	**512 MB**	**1024 MB**
TeraGen (1 GB)	57	56	54	**52**	54
TeraGen (2 GB)	**107**	108	**107**	109	111
TeraGen (3 GB)	150	159	155	137	**133**
TeraSort (1 GB)	564	527	571	**503**	630
TeraSort (2 GB)	1044	1149	1007	**958**	1127
TeraSort (3 GB)	2031	1650	**1536**	1523	1659
TeraValidate(1 GB)	48	45	**43**	**43**	54
TeraValidate(2 GB)	83	82	94	83	**80**
TeraValidate(3 GB)	118	115	114	**113**	**113**
WordCount (1 GB)	642	635	**526**	669	671
WordCount (2 GB)	1343	**1271**	1451	1421	1538
WordCount (3 GB)	2148	1904	4604	1463	**1361**
TestDFSIO -Write (1 GB)	59	59	58	**57**	**57**
TestDFSIO –Write (2 GB)	**95**	98	102	98	100
TestDFSIO -Write (3 GB)	163	142	**139**	142	146
TestDFSIO -Read (1 GB)	46	48	44	**38**	39
TestDFSIO –Read (2 GB)	**76**	**76**	77	79	82
TestDFSIO –Read (3 GB)	**106**	107	107	108	**106**

5.4 Hadoop performance tuning through mapreduce.task.io.sort.factor Parameter Customization

Table 35 depicts mapreduce.task.io.sort.factor configuration parameter value.

Table 7: mapreduce.task.io.sort.factor

Property	**Default Value**	**Description**
mapreduce.task .io.sort.factor	10	The maximum number of streams to merge at once when sorting files. The default value is 10. It is recommended that you to tune this value, usually upwards, in conjunction with your cluster hardware and workloads, with the caution that larger values can stress your cluster more, leading to cluster instability.

These parameters are usually tuned for each individual Hadoop job, but, depending on your cluster environment, you may also want to change the default values for all jobs by updating these parameters in the mapred-site.xml file.

Table 8 depicts the Intel's HiBench Benchmark (Micro Benchmarks) Execution time (Sec) for the different data sizes as well as different Hadoop's Customized preduce.task.io.sort.factor Parameter Configuration Settings like 10 (Default), 30, 50 ,70, 100.

The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Table 8 : Execution time (Sec) with Configuration Parameter: different mapreduce.task.io.sort.factor settings

Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size					
Execution time (Sec)					
Data Size (GB)	**mapreduce.task.io.sort.factor = 10**	**mapreduce.task.io.sort.factor = 30**	**mapreduce.task.io.sort.factor = 50**	**mapreduce.task.io.sort.factor = 80**	**mapreduce.task.io.sort.factor = 100**
TeraGen (1 GB)	56	57	54	**53**	**53**
TeraGen (2 GB)	107	96	**94**	100	98
TeraGen (3 GB)	150	148	**132**	133	136
TeraSort (1 GB)	624	534	548	562	**475**
TeraSort (2 GB)	1182	1235	1079	1183	**1039**
TeraSort (3 GB)	1755	**1560**	2045	1773	1702
TeraValidate (1 GB)	54	46	**45**	**45**	46
TeraValidate (2 GB)	**73**	92	91	89	88
TeraValidate (3 GB)	108	**101**	**101**	**101**	102
WordCount (1 GB)	672	689	811	665	**606**
WordCount (2 GB)	1209	**1023**	1041	1072	1097
WordCount (3 GB)	2148	1904	4604	1463	**1361**
TestDFSIO –Write (1 GB)	61	62	69	63	**60**
TestDFSIO -Write (2 GB)	99	98	99	102	**94**
TestDFSIO -Write (3 GB)	163	143	**138**	140	140
TestDFSIO -Read (1 GB)	61	51	**50**	51	**50**
TestDFSIO -Read (2 GB)	77	77	82	79	**76**
TestDFSIO -Read (3 GB)	123	**108**	110	110	110

5.5 Hadoop performance tuning through mapreduce.reduce.speculative Speculative Execution Parameter Customization

When the tasks running sustained Hadoop doesn't try to identify and fix slow-running tasks; instead, it tries to identify when a task is running slower than expected and launches another, equivalent, task as a sponsorship. This is termed speculative execution of tasks.

The objective of speculative execution is reducing job execution time, However the cluster effectiveness touching due to replica tasks. Table 9 depicts mapreduce.task.io.sort.factor configuration parameter value.

Table 9: mapreduce.reduce.speculative Configuration Parameter

Property	**Default**	**Description**
mapreduce.reduce. speculative	true	If true, then multiple instances of some reduce tasks may be executed in parallel.

These parameters are usually tuned for each individual Hadoop job, but, depending on your cluster environment, you may also want to change the default values for all jobs by updating these parameters in the mapred-site.xml file.

Table 10 depicts the Intel's HiBench Benchmark (Micro Benchmarks) Execution time (Sec) for the different data sizes as well as different Hadoop's Customized mapreduce.reduce.speculative Parameter Configuration Settings like TRUE (Default), FALSE. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Table 10: Execution time (Sec) Of Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size with mapreduce.reduce.speculative Configuration Parameter

Intel's HiBench Benchmark (Micro Benchmarks) With Different Data Size, With mapreduce.reduce.speculative Configuration Parameter		
Execution time (Sec)		
mapreduce.reduce.speculative		
Data Size(GB)	**TRUE(Default)**	**FALSE**
TeraGen (1 GB)	**56**	61
TeraGen (2 GB)	**107**	108
TeraGen (3 GB)	**150**	163
TeraSort (1 GB)	624	**573**
TeraSort (2 GB)	1182	**978**
TeraSort (3 GB)	2031	**1913**
TeraValidate (1 GB)	**48**	53
TeraValidate (2 GB)	83	**76**
TeraValidate (3 GB)	118	**115**
WordCount (1 GB)	696	**642**
WordCount (2 GB)	1343	**1254**
WordCount (3 GB)	2148	**1904**
TestDFSIO -Write (1 GB)	59	**58**
TestDFSIO -Write (2 GB)	**95**	96
TestDFSIO -Write (3 GB)	163	**136**
TestDFSIO -Read (1 GB)	**46**	48
TestDFSIO -Read (2 GB)	76	**75**
TestDFSIO -Read (3 GB)	**106**	**106**

Guided Apache Hadoop Parameter Configuration values for Performance Tuning

In this section guided customized Apache Hadoop Parameter configuration value suggested which help user to tuning performance and maximum utilization of available resources compare to default parameter configuration setting. This is guide list of Apache Hadoop parameter configuration value derived from various kind experiments carried out with different type of parameter configuration combination and suggested best among them. This guided Apache Hadoop parameter configuratation help those user whose have lack of Apache Hadoop framework configuration, hardware and software knowledge. Table 11, depicts Guided Apache Hadoop Parameter Configuration values for Performance Tuning for the different data sizes and different kind of applications for better performance and maximum utilization of available resource.

Table 11: Guided Apache Hadoop Parameter Configuration values for Performance Tuning

Guided Apache Hadoop Parameter Configuration values for Performance Tuning								
Hadoop Jobs	**Data Size (GB)**	**mapreduce. output. fileoutputformat. Compress (Default = False)**	**mapreduce. output. fileoutputformat. compress.codec (Default = org.apache. Hadoop.io. Compress = DefaultCodec)**	**mapreduce. map.output. compress (Default = False)**	**mapreduce. output. fileoutputformat. compress.Codec (Default = org.apache. Hadoop.io. Compress = DefaultCodec)**	**dfs.blocksize (MB) (Default = 128)**	**mapreduce. task.io. sort.factor (Default = 10)**	**mapreduce. reduce. speculative (Default = True)**
TeraGen	1	false	DefaultCodec	True	GzipCodec	512	100	True
TeraGen	2	true	BZip2Codec	True	GzipCodec	256	50	True
TeraGen	3	true	Lz4Codec	True	BZip2Codec	1024	50	True
TeraSort	1	true	Lz4Codec	True	DefaultCodec	512	100	False

TeraSort	2	true	Lz4Codec	True	DefaultCodec	512	100	False
TeraSort	3	true	DefaultCodec	True	Lz4Codec	256	30	False
Tera Validate	1	true	GzipCodec	True	GzipCodec	256	50	True
Tera Validate	2	false	DefaultCodec	FALSE	DefaultCodec	1024	10	False
Tera Validate	3	True	Lz4Codec	True	DefaultCodec	512	30	False
Word Count	1	true	DefaultCodec	True	GzipCodec	256	100	False
Word Count	2	false	DefaultCodec	True	Lz4Codec	64	30	False
Word Count	3	true	Lz4Codec	True	Lz4Codec	1024	100	False
TestDFSIO -write	1	true	Lz4Codec	True	GzipCodec	512	100	False
TestDFSIO -write	2	true	BZip2Codec	True	GzipCodec	128	100	True
TestDFSIO –write	3	true	DefaultCodec	True	DefaultCodec	256	50	False
TestDFSIO –Read	1	false	DefaultCodec	True	GzipCodec	512	100	True
TestDFSIO –Read	2	true	Lz4Codec	True	DefaultCodec	64	100	False
TestDFSIO –Read	3	true	GzipCodec	True	DefaultCodec	128	30	False

VI. CONCLUSION AND FUTURE WORK

In this research identify Hadoop Job Parameter Configuration Open Issue and studied various research paper related to Hadoop Job Parameter Configuration to find out current Hadoop Job Parameter configuration issue and solutions. Default Hadoop parameter configuration setting is not suitable for all kind of application. Proper parameter configuration setting can tune performance. Apache Hadoop Parameter configuration setting is still an open issue for researcher. User of Hadoop can get better performance from system resource through fine tuning parameter configuration setting. User can develop framework which contain good parameter configuration setting combination through empirical research by executing various Apache Hadoop job with different parameter configuration setting. Based on Literature Review proposed Customized Parameter Configuration Framework for Performance Tuning in Apache Hadoop and explain each components of propose framework.

Apache Hadoop Framework Performance improved through customizing parameter configuration values as compared to default configuration. So that is suggested to use customized configuration parameter. Customized Apache Hadoop parameter configuration framework not only save valuable resource like CPU , I/O, Memory, Network but also save users lots of time as compare to default configuration parameter value.

Guided customized Apache Hadoop Parameter configuration value suggested which help user to tuning performance and maximum utilization of available resources compare to default parameter configuration setting. This is guide line of Apache Hadoop parameter configuration value derived from various kind experiments carried out with different type of parameter configuration combination and suggested best among them. This guided Apache Hadoop parameter configuratation help those user whose have lack of Apache Hadoop framework configuration, hardware and software knowledge.

Proposed Customized Framework for Performance Tuning in Hadoop implemented and experiment done through built in Hadoop Example. In the future work Proposed Customized Framework for Performance Tuning in Hadoop implemented and experiments will be done in real time example like Weather Data, Sensor Data Analysis, and Traffic Data Analysis. In present work considered configuration parameter are data compression, block size like mapreduce.output.fileoutputformat.compress, mapreduce. output. fileoutputformat.compress.codec, mapreduce.map.output.compress.mapreduce. map.output.compress.codec, dfs.blocksize, mapreduce.task.io.sort.factor, mapreduce.reduce.speculative Speculative Execution.

In future work experimental work will be done through more configuration parameter like mapred.map.tasks ,mapred.reduce.tasks, mapreduce.map.memory.mb, mapreduce.map.java.opts, dfs.replication.

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. Rong Gu, Xiaoliang Yang, Jinshuang Yan, Yuanhao Sun, Bing Wang, Chunfeng Yuan, Yihua Huang, SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters, Journal of Parallel and Distributed Computing, Volume 74, Issue 3, March 2014, Pages 2166-2179, ISSN 0743-7315, http://dx.doi.org/10.1016/j.jpdc.2013.10.003
4. Zhang, Zhuoyao, Ludmila Cherkasova, and Boon Thau Loo. "Performance modeling of mapreduce jobs in heterogeneous cloud environments." Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing. IEEE Computer Society, 2013.

5. Yang, Hailong, et al. "Statistics-based Workload Modeling for MapReduce." Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. IEEE, 2012.
6. Rao, B. Thirumala, et al. "Performance issues of heterogeneous Hadoop clusters in cloud computing." arXiv preprint arXiv:1207.0894 (2012).
7. Babu, Shivnath. "Towards automatic optimization of MapReduce programs." Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010.
8. 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