Apache Hadoop Performance Tuning through Customization of Data Block Size Parameter Configuration Setting

Dr. Bhavin Mathiya* & Dr. Vinod L Desai**
*Kadi Sarva Vishwavidyalaya University, Gandhinagar.
**Assistant Professor, Department of Computer Science, Gujarat Vidyapith, Ahmedabad. Gujarat.

Received: May 19, 2018 Accepted: July 03, 2018

ABSTRACT *Rapidly changing technology and its vast usasge generates huge amount of data through various means like web sites, mobile phones, sensors which requires storage and processing of data. Apache Hadoop is open source framework for storing, managing, processing and analyzing of big data on commodity hardware in localy as well as distributedly stored data. Apache Hadoop provides MapReduce Programing for writing customized logic using mapper and reducer for processing big data. Apache Hadoop provides HDFS (Hadoop Distributed File System) for storing data locally as well as distributely. Apache Hadoop provides common configuration parameter settings with customization facility when it is deployed. It is main problem that all configuration parameter setting are common for all application like CPU intensive, I/O intensive. In this research paper we focus on Data Block Size parameter configruation setting customization and tunning Apache Hadoop performance.*

Keywords: *Hadoop, MapReduce, HDFS, Node*

1. Introduction

Huge amount of data is called "Big Data" having attributes like volume, variety, velocity. Now a day it is challenging for storing, processing and analyzing big data. Volume of big data is huge like Terabytes, Petabytes of data generated through various kinds of social media and networking sites like facebook, twitter, Sensor devices, scientific instruments, mobile devices and mobile networks, web sites clicks and various kinds of transactions likes banking, e-commerce web site transaction. Velocity of big data generation is fast and required fast processing like online data analytics. Big data is in variety of forms like structured, semi-structured and unstructured data like text, images, pictures, numbers, audio, video, sensor data, social media data, static data, streaming data.

Hadoop was created by Doug Cutting an employee at Yahoo and Michael J. Cafarella to support distribution for the Nutch search engine project. Hadoop was inspired by papers published by Google regarding its approach in handling a huge amount of data for storing, processing and analysis on locally as well in distributed on commodity hardware. Hadoop provide fault tolerance through replication management like multiple copy on block of data on different node and scalability. Apache Hadoop is an open source framework based on implementation of MapReduce Programming model and GFS (Google File System) [1]. Hadoop's breakthrough Advantages mean that businesses and organizations can now find value in data that was recently considered useless. Apache Hadoop Yarn is a MapReduce version2.

This paper is organized as follows. Section 1 Introduction, Section 2 Literature Review, Section 3 Methodology, Section 4 Environmental Setup, Section 5 Experimental Results and Observations, Section 6 Conclusion and Future Work.

2. Literature Review

Mahesh Maurya et al. [2] discussed various MapReduce applications like pi, wordcount, grep, Terasort and shown experimental results of these applications on a Hadoop cluster.

Shrinivas B. Joshi. [3] Says that Different Hadoop workloads will have different characteristics, so it is important to experiment with different tuning options. Kewen Wang et al. [4] introduce Predator, an experience guided configuration optimizer, which does not treat the optimization problem as a pure black-box problem but utilizes useful experience, learnt from Hadoop MapReduce configuration practice to assist the optimizing process. Singh, K et al. [5] address the problem of how to place data across nodes in a way that each node has a balanced data processing load. Singh, K et al. [5] proposed data placement scheme adaptively balances the amount of data stored in each node to achieve improved data-processing performance. Yoon et al. [6] presents a new architecture for Local Caching System which adjusts content locality between local file system and Hadoop Distributed File System by using proper cache algorithm and managing content queue and location.

3. Methodology

In this research paper Experimental research method used. Apache Hadoop Performance tuning done using customization dfs.blocksize configuration parameter setting with different configuration parameter value as compared to default settings.

4. Environmental Setup

The experiments were carried out on the Hadoop 2.4 single nodes with operating system of Ubuntu 14.04.1 LTS, Intel(R) Core(TM)2 Duo CPU T5870 @ 2.00GHz and 3 GB of RAM. Apache Hadoop Performance Tuning through Customization of Data Block Size Parameter Configuration Setting with Different Data Size on single node Hadoop Environment. Hadoop dfs.blocksize configuration parameter used to divide file into number of block. Based on dfs.blocksize parameter setting data will stored on HDFS (Hadoop Distributed File System).The default blocks size for new files, in 128 MB. Hadoop user 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).

5. Experimental Results and Observations

Table 1 depicts the Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize (MB) on single node Hadoop Environment.

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**

Table 1 Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize (MB) on single node Hadoop Environment

Table 1 depicts Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize (MB) on single node Hadoop Environment. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance. Table 1 depicts apache Hadoop jobs performance increased in majority of customized Hadoop parameter configuration settings as compare to default Hadoop parameter configuration settings.

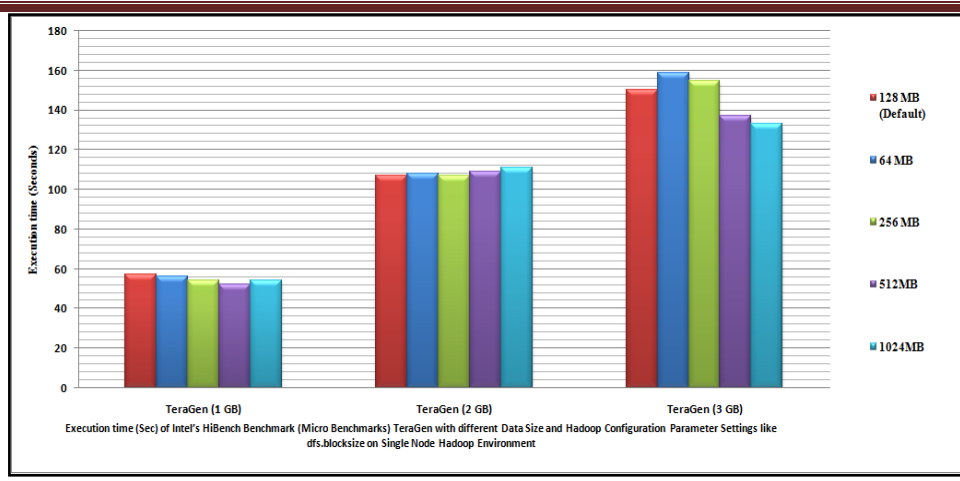

Figure 1. Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) TeraGen with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 1 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TeraGen with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Table 1 and Figure 1 showed that Apache Hadoop TeraGen (1GB) Performance increase from 2% to 9% as compare to default Hadoop parameter configuration settings. Apache Hadoop TeraGen (2 GB) default Hadoop parameter configuration settings performance is good as compare to other parameter configuration. Apache Hadoop TeraGen (3GB) Performance increase from 9% to 11% as compare to default Hadoop parameter configuration settings. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

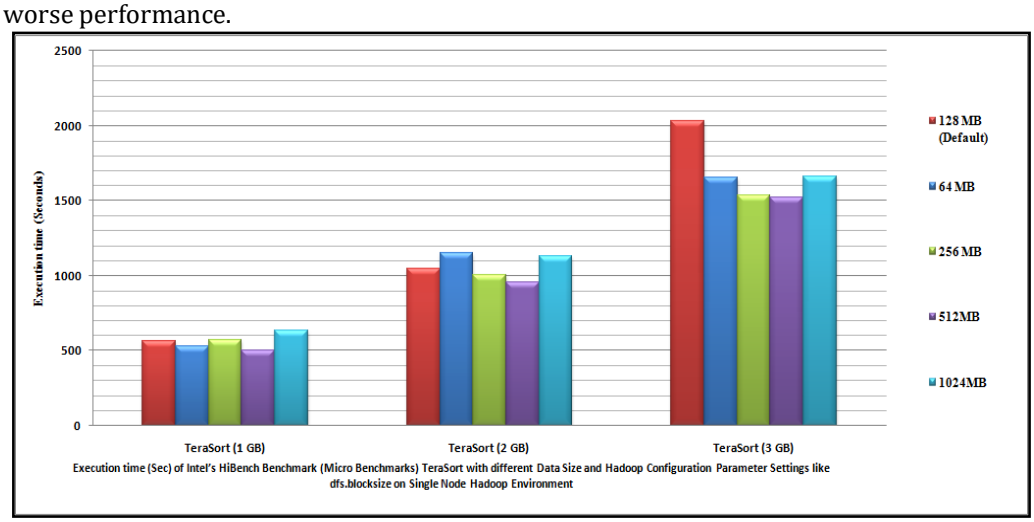

Figure 2. Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) TeraSort with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 2 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TeraSort with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Table 1 and Figure 2 showed that Apache Hadoop TeraSort (1GB) Performance increase from 7% to 11% as compare to default Hadoop parameter configuration settings. Apache Hadoop TeraSort (2 GB) Performance increase from 4% to 8% as compare to default Hadoop parameter configuration settings. Apache Hadoop TeraSort (3GB) Performance increase from 18% to 25% as compare to default Hadoop parameter configuration settings. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

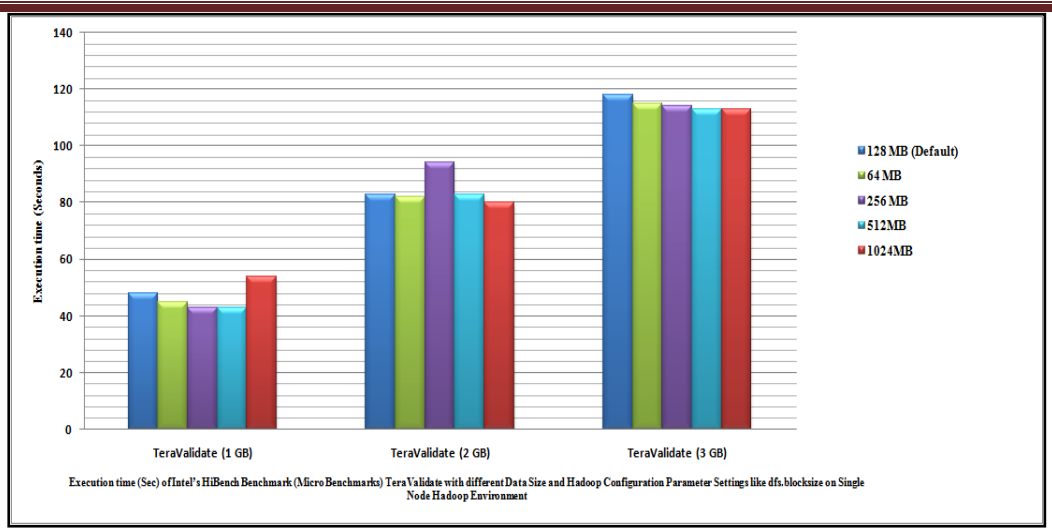

Figure 3. Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) TeraValidate with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 3 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TeraValidate with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Apache Hadoop TeraValidate (2 GB) Performance increase from 1% to 4% as compare to default Hadoop parameter configuration settings. Apache Hadoop TeraValidate (3GB) Performance increase from 3% to 4% as compare to default Hadoop parameter configuration settings. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

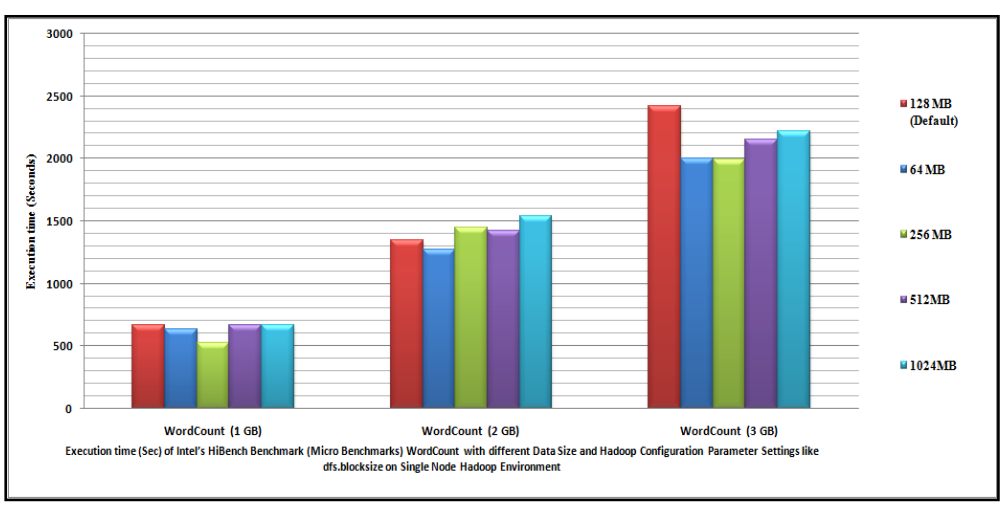

Figure 4. Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) WordCount with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 4 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) WordCount with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Table 1 and Figure 4 showed that Apache Hadoop WordCount (1GB) Performance increase from 1% to 18% as compare to default Hadoop parameter configuration settings. Apache Hadoop WordCount (2 GB) Performance increase 5% as compare to default Hadoop parameter configuration settings. Apache Hadoop WordCount (3GB) Performance increase from 11% to 37% as compare to default Hadoop parameter configuration settings. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

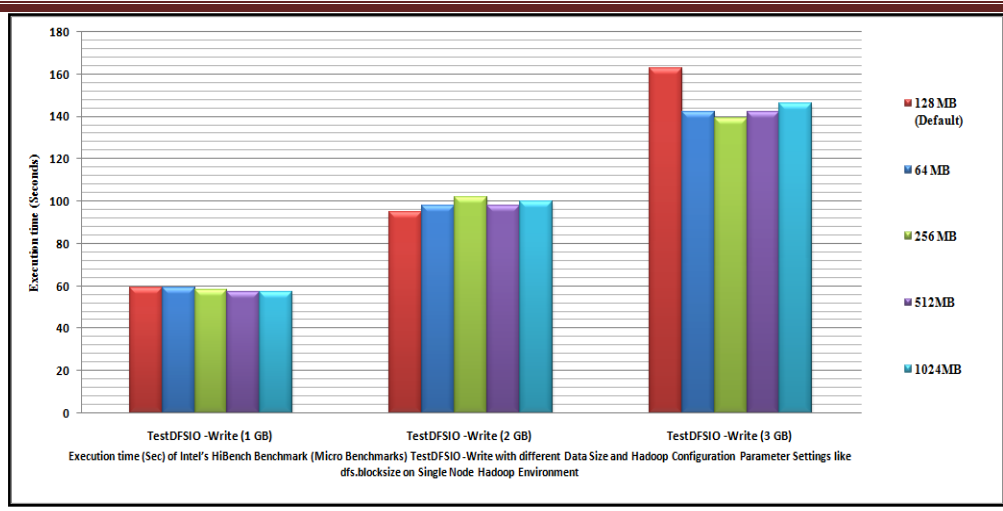

Figure 5. Execution time (Sec) of Intel's HiBench Benchmark (Micro Benchmarks) TestDFSIO -Write with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 5 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TestDFSIO -Write with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Table 1 and Figure 5 showed that Apache Hadoop TestDFSIO -Write (1GB) Performance increase from 2% to 3% as compare to default Hadoop parameter configuration settings. Apache Hadoop TestDFSIO -Write (2 GB) Performance default Hadoop parameter configuration settings performance is good as compare to other parameter configuration. Apache Hadoop TestDFSIO –Write (3GB) Performance increase from 13% to 15% as compare to default Hadoop parameter configuration settings. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

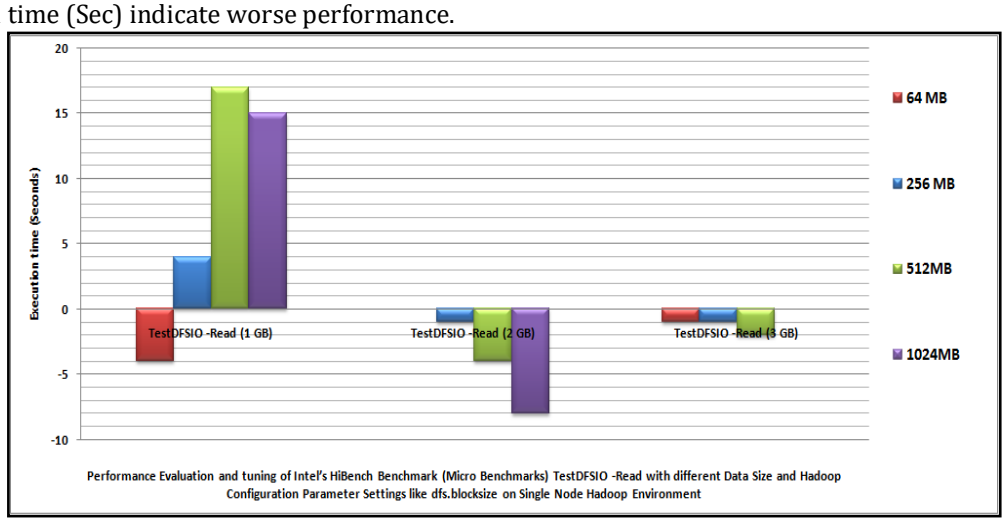

Figure 6. Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TestDFSIO -Read with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment

Table 1 and Figure 6 describe Performance Evaluation and tuning of Intel's HiBench Benchmark (Micro Benchmarks) TestDFSIO -Read with different Data Size and Hadoop Configuration Parameter Settings like dfs.blocksize on Single Node Hadoop Environment. Table 1 and Figure 6 showed that Apache Hadoop TestDFSIO -Read (1GB) Performance increase from 4% to 15% as compare to default Hadoop parameter configuration settings. Apache Hadoop TestDFSIO -Read (2 GB) Performance default Hadoop parameter configuration settings performance is good as compare to other parameter configuration. Apache Hadoop TestDFSIO -Read (3GB) Performance default Hadoop parameter configuration settings performance is good as compare to other parameter configuration. The shorter Execution time (Sec) indicates better performance and respectively the Execution time (Sec) indicate worse performance.

Conclusion and Future Work

In this research paper, Apache Hadoop performance improved through the customization of dfs.blocksize parameter configuration settings as compare to default configuration setting. In the future more configuration parameter can be customized like dfs.replication, mapred.output.compress, mapred.output.compression.type,mapred.output.compression. codec, mapred.compress.map.output,mapred.map.output.compression.codec.

References

[1] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.

[2] Maurya, M.; Mahajan, S., "Performance analysis of MapReduce programs on Hadoop cluster," Information and Communication Technologies (WICT), 2012 World Congress on , vol., no., pp.505,510, Oct. 30 2012-Nov. 2 2012
doi: 10.1109/WICT.2012.6409130

[3] 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

[4] Kewen Wang; Xuelian Lin; Wenzhong Tang, "Predator — An experience guided configuration optimizer for Hadoop MapReduce," Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on , vol., no., pp.419,426, 3-6 Dec. 2012

[5] Singh, K.; Kaur, R., "Hadoop: Addressing challenges of Big Data," Advance Computing Conference (IACC), 2014 IEEE International , vol., no., pp.686,689, 21-22 Feb. 2014

[6] Yoon, Sang-Deok, et al. "Improving HDFS performance using local caching system." Future Generation Communication Technology (FGCT), 2013 Second International Conference on. IEEE, 2013.