Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 4, April 2017, pg.389 – 396

PERFORMANCE IMPROVEMENT OF MAP REDUCE THROUGH ENHANCEMENT IN HADOOP BLOCK PLACEMENT ALGORITHM

Varun Kumar, Himani Gupta

Email: me@varunkumar.in, Email: himani.b14@gmail.com

MERI College of Engineering & Technology

46th Milestone, Rohtak Road, Sampla

ABSTRACT:

In last few years, a huge volume of data has been produced from multiple sources across the globe. Dealing with such a huge volume of data has arisen the so called "Big data problem", which can be solved only with new computing paradigms and platforms which lead to Apache Hadoop to come into picture. Inspired by the Google's private cluster platform, few independent software developers developed Hadoop and, following the white papers published by Google's team, a complete set of components has been developed for big data elaboration. Two of these components are the Hadoop Distributed File System and Map reduce.

In this paper, we are going to explore the Hadoop's default data placement policy in detail and propose a modified block placement policy for the Hadoop Distributed File System (HDFS) that increases the performance of the overall system.

Finally, we will establish comparison between HDFS default blocks placement strategy and our blocks placement strategy. We discuss the test cases and compare the performance improvements in a small size cluster.

I. INTRODUCTION:

The amount of data generated and stored worldwide has increased drastically over the last two decades[A1]. About 90% of today's data in the world has been created in the past few years. According to computer giant IBM, 2.5 exabytes, that's 2.5 billion gigabytes (GB), of data was generated each day in 2012 [F1]. Today, we are surrounded by large number of sensors and

computing devices that record huge volume of information. Moreover, with the expansion of internet and mobile computing, it is even easier to copy and move data from one location to the other. In addition, since the mobile phone penetration is forecast to grow from 61% of the global population in 2013 to about 70% by 2017, those figures will grow further [F1]. Data intensive systems such as web crawling, Large Hadron collider (LHC), space satellites etc. need to access ever expanding data sets that could range from a few gigabytes to several hundred terabytes or even many petabytes.

Thus, there is an increasing demand for new tool and technologies to handle this huge volume of data. Apache Hadoop is an efficient framework to store and process large-scale data. It consists of two main layers. The MapReduce layer provides a programming paradigm to process large set of data in parallel across multiple computing nodes in a scalable cluster. MapReduce helps programmers to execute distributed programs across nodes in parallel and automatically gathers results from all the nodes and provides a single result or set as output. Moreover, MapReduce framework provide fault tolerance and transparency to programmers. Thus making MapReduce a more practical and attractive programming model for parallel data processing in a high performance cluster computing environment. Google for example leverages MapReduce model to process approximately twenty petabytes of data every day in a through parallelism[C1].

The Hadoop Distributed File System (HDFS) provides a distributed file system, which runs on low cost commodity hardware in a fault tolerant manner. It is being inspired by GFS (Google File System). HDFS divides files into fixed size blocks that are stored and replicated among multiple computing nodes with no tracking that whether the blocks are divided evenly. Whenever the computation is to be performed, each node process the data using their local resources. When a large file need to be accessed, high aggregated I/O bandwidth is achieved by accessing multiple nodes in parallel. The performance is drastically improved by Hadoop because of the multiple nodes working concurrently to provide high throughput.

II. RELATED WORK:

Priya Deshpande, Aniket Bhaise [01], In world of Information technology, the Datum (Data in Latin) increasing drastically typically multi structured data which is very important for deep dive analysis in context of data intensive application. The way of storing, processing analysing such data within stipulated time comes under Big data technique. Thus manipulation of such type of data like retrieving, storing, replicating and updating is very tedious job in data intensive application. Data cluster technique mainly deals with such big data. In the era of fast growing technology the term 'data cluster' being replaced by distributed computing to serve as one of the service as for various purpose like scientific application, data analytics, business intelligence, statistical research, & data visualization.

In case of distributed computing, data replication and availability plays key role to share data between nodes (data intensive applications) which automatically achieves high performance, data availability, consistency (typically eventual consistency) and low latency. In this paper, I Propose "Enhance Replication Method for Big Data with HDFS Cluster" using Apache Hadoop. Hadoop Distributed File System which provides Framework for "Big Data" storing, maintenance & manipulation which is fundamental architecture of big data technology. To achieve data availability, consistency and minimum latency we implemented the technique to replicate data on the basis of request frequency. Also we achieved the eventual consistency by updating data instances with new consistent data files. With respect to same context we will discuss the performance evaluation of this technique with

existing feature of data replication of HDFS by providing effective replication & consistency maintenance.

Sayali Ashok Shivarkar, Prof.Deepali Gatade [02], Apache's Hadoop is an open source implementation of Google Map/Reduce which is used for large data analysis and storage. Hadoop decompose a massive job into number of smaller tasks. Hadoop uses Hadoop Distributed File System to store data. HDFS stores files as number of blocks and replicated for fault tolerance. The block placement strategy does not consider data placement characteristics. It stores file as block randomly. The block size and replication factor are configurable parameters. An application can specify number of replica of file and it can be changed later. HDFS cluster has master/slave architecture with a single Name Node as master server which manages the file system namespace and regulates access to file by clients. The slaves are called to the number of Data Nodes. File is divided into number of one or more blocks and stores as set of blocks in Data Nodes. Opening, renaming and closing file and directory all operations are done by Name Node and Data Node are responsible for read and write request from Name Node. Hadoop uses Hadoop distributed File System (HDFS) which is an open source implementation Google File System for storing data. HDFS is used in Hadoop for storing data. Strategic data partitioning, processing, replication, layouts and placement of data blocks will increase the performance of Hadoop and a lot of research is going on in this area. This paper reviews and survey some of the major enhancements suggested to Hadoop especially in data storage, processing and placement.

Zhendong Cheng, Zhongzhi Luan, Alain Roy Ning Zhang, Gang Guan [03], The Hadoop Distributed File System (HDFS) is a distributed storage system that stores large-scale data sets reliably and streams those data sets to applications at high bandwidth. HDFS provides high performance, reliability and availability by replicating data, typically three copies of every data. The data in HDFS changes in popularity over time. To get better performance and higher disk utilization, the replication policy of HDFS should be elastic and adapt to data popularity. In this paper, we describe ERMS, an elastic replication management system for HDFS. ERMS provides an active/standby storage model for HDFS. It utilizes a complex event processing engine to distinguish real-time data types, and then dynamically increases extra replicas for hot data, cleans up these extra replicas when the data cool down, and uses erasure codes for cold data. ERMS also introduces a replica placement strategy for the extra replicas of hot data and erasure coding parities. The experiments show that ERMS effectively improves the reliability and performance of HDFS and reduce storage overhead.

Hnin Htet Htet Aung, and Nyein Nyein Oo [04], Cloud computing is composed of a large number of distributed computation and storage resources to facilitate the management of distributed and sharing data resources efficiently. It is a great challenge to ensure efficient access of data replication to such huge and widely distributed data in cloud computing. To address this need, we proposed an Efficient Data Access Scheme (EDAS) of data replication for Hadoop Distributed File System (HDFS) to adaptively select the replica of data file form among service nodes. HDFS is an open source cloud based storage platform and deigned to be deployed in low-cost commodity hardware. In HDFS, data are distributed and replicated in cluster of commodity nodes. EDAS supports the access nodes decision of replica data for the users to get quick access form the adaptive services nodes according to the load of nodes. Aiming to provide the high performance of replication access and achieve load balance of service nodes, the proposed EDAS Algorithm implements based on historical data access record form the metadata of HDFS and anti-blocking probability selection method.

Dr. D Rajya Lakshmi, Mr. R Praveen Kumar, Mr. N K Sumanth [06], There is a lot of hype around Hadoop now a day`s, this mistakenly leads to a fact that Hadoop can do anything but it's not. Hadoop is meant for certain kind of applications and not suitable for real time applications. But still we go for Hadoop in enterprises for its extreme horse power in processing huge data in less time. Hadoop is an open-source, framework of tools that is used to support the running of applications on Big Data. The simple Hadoop Architecture consists of mainly two components Hadoop Distributed File system and Map-Reduce. HDFS breaks the data/file into fixed size blocks and spreads over the cluster. Map-Reduce will take care about Computation by moving the code or piece of code to the data/blocks where they are present in a cluster of commodity hardware. Hadoop overcomes the Limitations of Distributed File System by providing the features like stores large data, Offers Failure Protection and provides fast access but Hadoop is built for specific class of applications. HDFS blocks are large, the default block size is 64MB compared to the block size in other structured-file system 4KB-8KB. The major contribution of this work is Hadoop performance evaluation on writing/reading with variable block sizes that results low performance and full understanding of Map Reduce concept as well and the distribution of jobs. At any stage Hadoop clusters can run with a default block sizes, which is fixed (multiples of 64MB). Here reading/writing of such huge filecan vary from one node to other depends on commodity hardware. We are taking only one parameter that is Disks Speed. Traditions Hard Disks speed could be 60-100MB/sec, now we have come up with Solid state Disk that could run with a speed of 250-500MB/sec. That means the hard disk which have high performance will read the entire file and computes fast and others will compute slow. The Job Tracker should wait for all computations to return final result to the application. This would cause the performance issue. So by altering the block sizes i.e. for the high performance node have large block size and low performance node have small block size. By tuning up the block sizes, all the nodes will have equal performance of read/writes. So that master node will not wait for other nodes for collecting the computations and it leads to better performance. There are some other contributions includes Changing the Hadoop Configuration Files, mapper – reducer coordination (reducers need not to wait until all mappers finish their task), maximum tasks for a task tracker, networking in the rack architecture or core switch configuration in a cluster and many more works can be done to improve the performance of Hadoop framework. Note: I have given only one example parameter that is Disks Speed. Traditional Hard Disks speed could be 60-100MB/sec, with Solid state Disk that could run with a speed of 250-500MB/sec.

Sudhakar Singh, Rakhi Garg, P. K. Mishra [06], Designing fast and scalable algorithm for mining frequent itemsets is always being a most eminent and promising problem of data mining. Apriori is one of the most broadly used and popular algorithm of frequent itemset mining. Designing efficient algorithms on MapReduce framework to process and analyze big datasets is contemporary research nowadays. In this paper, we have focused on the performance of MapReduce based Apriori on homogeneous as well as on heterogeneous Hadoop cluster. We have investigated a number of factors that significantly affects the execution time of MapReduce based Apriori running on homogeneous and heterogeneous Hadoop Cluster. Factors are specific to both algorithmic and non-algorithmic improvements. Considered factors specific to algorithmic improvements are filtered transactions and data structures. Experimental results show that how an appropriate data structure and filtered transactions technique drastically reduce the execution time. The non-algorithmic factors include speculative execution, nodes with poor performance, data locality & distribution of data blocks, and parallelism control with input split size. We have applied strategies against these factors and fine-tuned the relevant parameters in our particular application. Experimental results show that if cluster specific parameters are taken care of then there is a significant reduction in execution time. Also we

have discussed the issues regarding MapReduce implementation of Apriori which may significantly influence the performance.

Billel Arres, Nadia Kabachi, Omar Boussaid [07], In the recent past, we have witnessed dramatic increases in the volume of data literally in every area: business, science, and daily life to name a few. The Hadoop framework an open source project based on the MapReduce paradigm is a popular choice for big data analytics. However, the performance gained from Hadoop's features is currently limited by its default block placement policy, which does not take any data characteristics into account. Indeed, the efficiency of many operations can be improved by a careful data placement, including indexing, grouping, aggregation and joins. In our work we propose a data warehouse partitioning strategy to improve query gain performances. We investigate the performance gain for OLAP cube construction with and without data organization on a Hadoop cluster. And this, by varying the number of nodes and data warehouse size.

Our experiments suggest that a good data placement on a cluster during the implementation of the data warehouse can significantly increase the OLAP cube construction and querying performances. In the next step, we will extend the experiments to study the effects of other configuration parameters on collocation data in the context of parallel data warehousing, such as partitions size, replication factor and OLAP query complexity. We plan also to study an intelligent system for warehouses data placement on clusters by integrating Multi-Agent System (MAS) and Intelligent Agents to the process.

X.Z. Lu, K.K. Phang [08], In this paper, we proposed a novel mechanism, namely Enhanced Dynamic Data Placement (EDDP). There are two components in EDDP: data partitioning and virtual machines (VMs) optimization. The first component is adapted from [Lee et al, 2014] whereby data placement and their size at the computing nodes must be proportional with their computation capability. In the second component, the configurations of the virtual machines created to handle the incoming jobs are optimized based on benchmarking. Experimental results show that EDDP managed to shorten job completion time.

III. PROPOSED METHODOLOGY:

We proposed a novel approach which will diminish the shortcomings mentioned above by suggesting an improvement in the Hadoop default Data Placement Policy. The proposed block placement policy consists of two major objectives which has been incorporated into a single algorithm. First, the new placement algorithm distributes blocks across all the DataNodes in the cluster evenly. Second, nodes having higher I/O efficiency would handle more of data in order to improve the overall performance of the cluster.

Implementing the proposed technique for data placement instead of the default data placement policy of Hadoop is expected to increase the overall performance of the cluster.

Proposed Algorithm

In order to ensure that DataNodes with higher I/O speeds cater to relatively more data blocks and read/write requests as compared to those with relatively slow I/O speeds, a concept of weight and Quantum is introduced. Weight is the factor that denotes the relative processing power for a generation of hardware. Quantum is the maximum number of data blocks that can be placed on a node pertaining to a single replica of a file. Let r be the replication factor of file in the cluster. Assuming

that the file has been split into n blocks, the Quantum of each node can be calculated as follows:

$$Quantum_{node} = ceil\ (n * weight_{node} / weight_{total})$$

IV. Result Analysis/Implementation:

The proposed algorithm described in previous chapter has been implemented in Hadoop-2.3 by making necessary amendments to the existing source code of Block Placement class file written in Java. The cluster consisting of 4 nodes with the below mentioned configuration is used for testing.

	Node 1 (N1) Master	Node 2 (N2) Slave	Node 2 (N3) Slave	Node 4 (N4) Slave
CPU	4 x Intel(R) Core(TM) i7 CPU @ 2.3 GHz	4 x intel(R) Core(TM) i3-2120 CPU @ 3.30GHz	4 x intel(R) Core(TM) i3-2120 CPU @ 3.30GHz	4x intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
RAM	16 GB	4 GB	2 GB	2 GB
Hard Disk	500 GB SSD	1024 GB SATA @ 3.0 Gbps	512 GB SATA @ 1.5 Gbps	512 GB SATA @ 1.5 Gbps
OS	Mac OS X 10.9	Ubuntu 14.04.1 LTS	Ubuntu 14.04.1 LTS	Ubuntu 14.04.1 LTS
Java JDK Version	1.7.0_79	1.7.0_79	1.7.0_79	1.7.0_79
Ethernet	1 Gbps	1 Gbps	1 Gbps	1 Gbps
Weight	2	2	1	1

Table 4.1: Node Configurations on the Cluster

Two benchmark cases have been used in order to prove the performance gain using the proposed approach,. First case is the conventional TeraSort Algorithm that is part of Hadoop Distribution package. Second is a simple write-read benchmark.

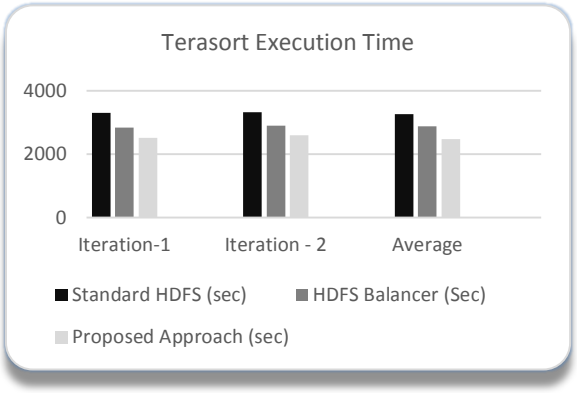

Figure 4.1: The time taken by the Terasort command to execute

As it can be seen from the results, there is a significant reduction in the execution time taken while using the proposed placement policy. The approximate percentage reduction as compared to standard HDFS is 24.17% and to HDFS balancer is 13.88%.

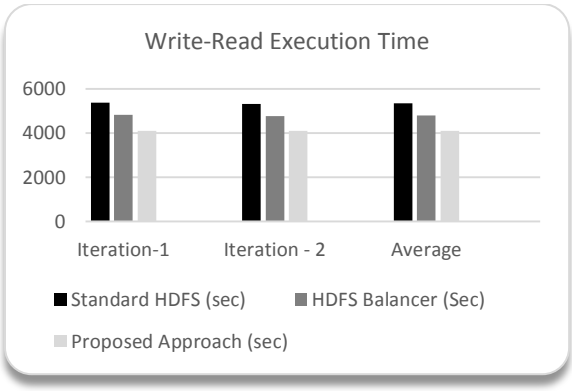

Figure 4.2: Time taken using the various data placement strategies

The proposed approach has shown an improvement of 23.27% over the standard HDFS and 14.62% over HDFS Balancer.

V. CONCLUSION:

As it can be concluded from the results above, the proposed approach provides a significant performance enhancement over the HDFS default data placement strategy and Hadoop balancer as well. The enhancement is better visible in the write-read benchmark as compared to TeraSort benchmark.

The reason for this is that the TeraSort benchmark:

1. It involves more computation as compared to simple write-read benchmark. Since this paper is focussed on data placement policy pertaining to HDFS and not the performance of MapReduce, hence it is expected.

2. It leverages MapReduce framework to distribute data over network whereas the write-read benchmark writes data into an HDFS directory from a single node there creating unevenness.

The results could be seen more prominently if the size of the cluster and the total data stored in HDFS is increased.

REFERENCES

[01] Priya Deshpande, Aniket Bhaise, **"Enhanced Replication Method for Big Data with HDFS Cluster"**, Proceedings of 43rd IRF International Conference, 29th May, 2016, Chennai, India, ISBN: 978-93-86083-23-4,

[02] Sayali Ashok Shivarkar, Prof.Deepali Gatade, **"Speed- Up Extension To Hadoop System- A Survey Of HDFS Data Placement"**, International Journal of Advance Foundation And Research In Science & Engineering (IJAFRSE) Volume 2, Issue 2 , July 2015. Impact Factor: 1.036, Science Central Value: 26.54,

[03] Zhendong Cheng, Zhongzhi Luan, Alain Roy Ning Zhang, Gang Guan, **"ERMS: An Elastic Replication Management System for HDFS",** 2012 IEEE International Conference on Cluster Computing Workshops, DOI 10.1109/ClusterW.2012.25,

[04] Hnin Htet Htet Aung, and Nyein Nyein Oo, **"EDAS: Efficient Data Access Scheme of Data Replication for Hadoop Distributed File System (HDFS)",** ISBN 978-93-84468-20-0, Proceedings of 2015 International Conference on Future Computational Technologies (ICFCT'2015), Singapore, March 29-30, 2015, pp. 177-183,

[05] Dr. D Rajya Lakshmi, Mr. R Praveen Kumar, Mr. N K Sumanth, **"Tuning the Performance of Hadoop Map Reduce Jobs by Altering Various Parameters",** International Journal of Science, Technology and Management, Vol No 5, February 2016

[06] Sudhakar Singh, Rakhi Garg, P. K. Mishra, **"Observations on Factors Affecting Performance of MapReduce based Apriori on Hadoop Cluster",** International Conference on Computing, Communication and Automation (ICCCA2016), doi: 10.1109/CCAA.2016.7813695,

[07] Billel Arres, Nadia Kabachi, Omar Boussaid, **"Intentional Data Placement Policy for Improving OLAP Cube Construction on Hadoop Clusters",** Proceedings of the BDA 2014 Conference (October 14, 2014, Grenoble-Autrans, France),

[08] X.Z. Lu, K.K. Phang, **"Enhanced Dynamic Data Placement and Virtual Machine Creation for MapReduce",** international Conference on Research & Innovation in Computer, Electronics and Manufacturing Engineering (RICEME-17) Feb. 2-3, 2017 Bali (Indonesia), https://doi.org/10.17758/EIRAI.F0217115

