Scalability Analysis and Improvement of Hadoop over H2Hadoop for Big Data Analysis

Ms. Kalyani Patil[1] V.V. Dakhode[2]
[1,2]Department of Computer Engineering
[1,2]SKNCOE, Pune, India

Abstract— Cloud Computing provides different services to the users with regard to processing the Bigdata. The services such as S3 (Simple Storage Service) for storing data, EC2 (Elastic Compute Cloud) to build a private Cloud Computing environment and EMR (Elastic MapReduce) for processing the BigData. Generally EMR uses original Hadoop to process BigData. Beacuse Hadoop is a framework that allows distributed processing of large datasets across clusters of computers using simple programming models. Hence cloud computing leverages Hadoop framework to process BigData in parallel. Hadoop has certain limitations which reduces the job efficiency. These limitations are mostly because of data locality in the cluster, jobs and tasks scheduling, and resource allocations in Hadoop. The challenge remains in Cloud Computing MapReduce platforms is efficient resource allocation. Hence there is an improve system, H2Hadoop. An absolute analysis about H2Hadoop architecture and development is to store the metadata of the executed job. Common Job Blocks (CJB) tables is stored in the name node and metadata of the similar jobs are stored in Name node. Name node helps to direct future jobs to specific Data node that carry the required data sets. The CJB table gets updated with the new common features each time a new job reaches this file. The size of this table should be controlled and limited to a specific size to keep the architecture reliable and efficient. We propose the improved system, which is an enhanced Hadoop architecture that reduces the computation cost associated with BigData analysis. Comparing with H2Hadoop, this system reduces CPU time, number of read operations, and another Hadoop factors for a sequences which has common features.

Key words: Bigdata, Mapreduce, Hadoop, CJB Table

I. INTRODUCTION

Due to the heterogeneous nature and large size of data parallel processing in Cloud Computing has emerged as an main research area. [1]. In Big data, Database environment the types of data we come across are structured, semi-structured and unstructured data. Due to increasing volume of data large scale data intensive computing has become necessary for many applications. The Hadoop which is open-source implementation, has provided a cost effective solution for such data processing needs[2]. Although, some of the data intensive applications are readable by humans, it can be very complex to be understood and processed using traditional processing techniques hence Hadoop is designed to support such data intensive applications on clusters of hundreds or thousands of compute nodes. Nowadays genomic and biological data gets generated in large volume. This data is stored in geographically diverse locations. In the BigData such as genomic and biological the sequences which have common features gets generated periodically.This type of data can scale up to few terabytes. So there is need of large distributed file based processing. Hence to overcome the problem of processing such large distributed data which has some common features, the H2Hadoop is developed. The objective of H2Hadoop is to assess the feasibility of using Cloud based platform to analyze genomic big data. The H2Hadoop is an improved Hadoop architecture to reduce computation by utilizing Common Features before performing redundant computation[13]. The H2Hadoop architecture allows the jobs to share these common features among them. The common features describe the contents of data in blocks and can be used to determine DataNodes that store the required data.

II. BASICS OF H2HADOOP

H2Hadoop works on top of Hadoop to process MapReduce jobs hence it is a developed framework than Hadoop. It speed up the processing time by reducing the data input size from HDFS. By reading specific data sets that carry the target data it eliminates data redundancy. A absolute analysis about H2Hadoop architecture and development is to store the metadata of the executed job. Using Common Job Blocks (CJB) tables is stored in the name node and metadata of the similar jobs are stored in Name node. Name node helps to direct future jobs to specific Data node that carry the required data sets. CJB table is related to only one data file which is in HDFS, which means that there is only one table for each file in HDFS. CJB table gets updated everytime with the new common features (for example, common sequences in the project of human genome finding sequence alignment) each time a new job reaches this file, so this table gets resized dynamically and dramatically. Common

Features which is component of CJB table store data that is shared between jobs like DNA sequence. H2Hadoop architecture supports caching, which enables output (or part of output) to be written in CJBT after the reduce step. JobTracker that means Name Node directs any new job with the shared common features to blocks listed in CJBT.

A. CJB Table:

CJB Table has three main components:

Common Job Name	Common Feature	Block Name		
Sequence_Alignment	GGGATTTAG	B1	B2	B3
	TTTAGA	B1	B4	
Fining_Sequence	TTTAGCC	B3	B6	
	GCCATTAA	B1	B3	B4
	AATCCAGG	B3	B5	

Fig. 1: Common Job Block Table components

1) Common Job Name (CJN):
The Common Job Name component represents of a shared name of a common job that each MapReduce job must be submitted using this common name. Because of the common job name, client gets the output for the new job which has same name as that of one which is already executed.
2) Common Feature
If job has sequence which has some specific features is executed and next time the sub sequence of that sequence is executed, then that old feature will get replaced with shortest one
3) Block Name:
Block name is the location of these common features,which means that in which block that feature is stored.This feature of table allows the NameNode to direct the job to get data only from the DataNodes that store these blocks on their HDFS. CJB table stores all blocks that are related to the results of the common feature.

III. RELATED WORK

Motivation of [6] is to develop data placements scheme that improve performance of hadoop heterogeneous clusters. So it designs and implements a data placement mechanism in HDFS. It addresses the problem of how to place data across all nodes in a way that each node has a balanced data processing load. This data placement scheme adaptively balances the amount of data stored in each node to achieve improved data-processing performance. As we know ignoring the data locality issue in heterogeneous environments can reduce the MapReduce performance. Hence the main approach is to improve performance of Hadoop heterogeneous clusters. It is a new mechanism that distributes fragments of an input file to heterogeneous nodes based on their computing capacities. The limitation of this paper is it does not handle the data redundance issue of data allocation in the cluster and designing a dynamic data distribution mechanism for multiple data intensive applications working together.

In [7] SciHadoop: a system for enhancing the performance of common analysis tasks (e.g. aggregate queries) over unmodified, array-based scientific data files using MapReduce as the execution substrate. It introduces Sci-Hadoop, that address the goals like reduce total data transfers, reduce remote reads, and reduce unnecessary reads. Duplicate reads and memory pressure caused by isolated library-based caches may occure. Also it has to expand support to other file formats such as HDF5. Also, several performance improvements has to be done to reduce storage and computational costs of routing data through the system, thereby reducing runtimes. The IO efficiency of data-intensive processing for scientific data can be improved by using various techniques such as data mining.

In [2] it present Bi-Hadoop, an efficient extension of Hadoop to better support binary-input applications. Bi-Hadoop has an easy-to-use user interface, a binary-input aware task scheduler, and a caching subsystem. Extensive experiments show that Bi-Hadoop reduces the data transfer overhead hence it improve the execution of binary-input applications. Also it outperforms existing Hadoop by up to 3.3x. The limitation of this paper is that it is not support for multiple input applications.

It is a MapReduce resource allocation system aimed at enhancing the performance of MapReduce jobs in the cloud which is presented in [8]. The limitation of this paper is, it has not develop online techniques for handling dynamic scenarios like changing job characteristics on a dataset.

In [9] it is an approach to improve the performance of the Hadoop MapReduce framework by optimizing the job and task execution mechanism. It limits the dynamic scheduling of slots for the Hadoop MapReduce execution framework.

Existing Hadoop does not co-locate the realred data. If related logs are partitioned and processed as a group, then performance of log processing operations such as indexing, grouping joins and sessionization on Hadoop can be significantly improved . To enable this, a grouping key can be used to identify the related logs. [10] represents CoHadoop, it is an extension of Hadoop, uses this key to collocate all those files which correspond to the same key. However, it selects the data nodes randomly for every new key. The limitation of this is that it does not automatically assign the grouping keys based on user requirement and the nature of data that enters into the system.

Recently, the Hadoop community is developing a new version of Hadoop Hadoop 2.0 [15]. In this version, the JobTracker in Hadoop 1.0 is replaced by the ResourceManager and per-application ApplicationMaster. The ResourceManager is responsible for computing resource allocation and the perapplication ApplicationMaster is responsible for task scheduling and coordination. MCP focuses on the taskscheduling in MapReduce Job, therefore it can be easily integrated into the ApplicationMaster of MapReduce and achieve performance improvement,

Sr.No.	Paper Title	Data Redundancy	Dynamic Data Distribution	Resource Allocation	Reduce number of read operation	Reduce computational cost
1.	Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters.	Yes	No	No	No	No
2.	SciHadoop: Arraybased Query Processing in Hadoop.	Yes	No	No	No	No
3.	Bi-Hadoop: Extending Hadoop To Improve Support For Binary-Input Applications	No	No	Yes	No	Yes
4.	Purlieus: Locality-aware	Yes	No	No	No	No

	Resource Allocation for MapReduce in a Cloud.					
5.	SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters	Yes	No	No	No	Yes
6.	Hadoop Based Enhanced Cloud Architecture For Bioinformatic Algorithms.	Yes	Yes	Yes	No	No

Table 1: Evaluation of literature Survey

IV. EXISTING SYSTEM

H2Hadoop eliminates data redundancy hence it is developed framework than Hadoop. The common Job Block(CJB) table performs the main role in H2Hadoop,since it is stored in Name node.This tablestores some information about the jobs and the blocks related to that job result. This architecture allows the related jobs to go and get results from these blocks without checking the entire cluster.This table is gets updated with every new job which has common feature. Feature selection should be done carefully since the response time for the jobs can increase if the common features exist in each DataNode. Using such sequences as common features can degrade the performance of the proposed architecture. CJB table in H2Hdoop is develop using HBase environment, because Hbase is column oriented database and also it is easily scaled up with respect to storage. Hence the size of CJB table increase dynamically which leads to research problem for the sequences which has common features.

V. PROPOSED SYSTEM

The propose architecture of Hadoop is same as H2Hadoop architecture in terms of hardware, network, and nodes. In proposed system we have to focus on improving the IO efficiency of data-intensive processing for large amount of data or the sequence of data which has common name and common features. As we know H2Hadoop consist of Common Job Blocks (CJB) table, the CJB table gets updated with the new common features each time a new job reaches this file, so this table gets resized dynamically and dramatically. Hence the size of this table should be controlled and limited to a specific size to keep the architecture reliable and efficient. CJB table should not exceed to be a huge table that makes the process of research for a common feature takes long time. So, the size can be limited by using Leaky Bucket Algorithm or develop the CJB table using RDBMS instead of HBase, as RDBMS has scalability but limited in storage capacity.

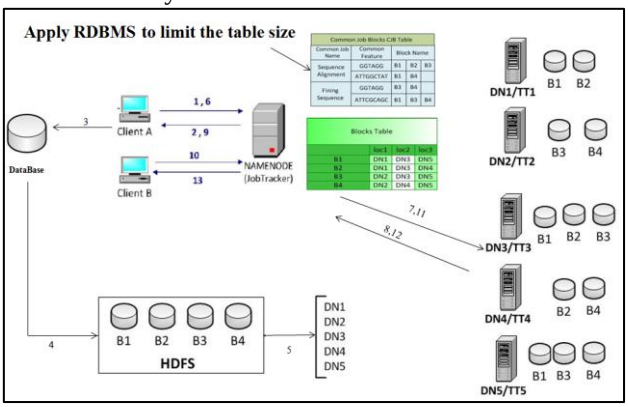

Fig. 2: Proposed System Architecture

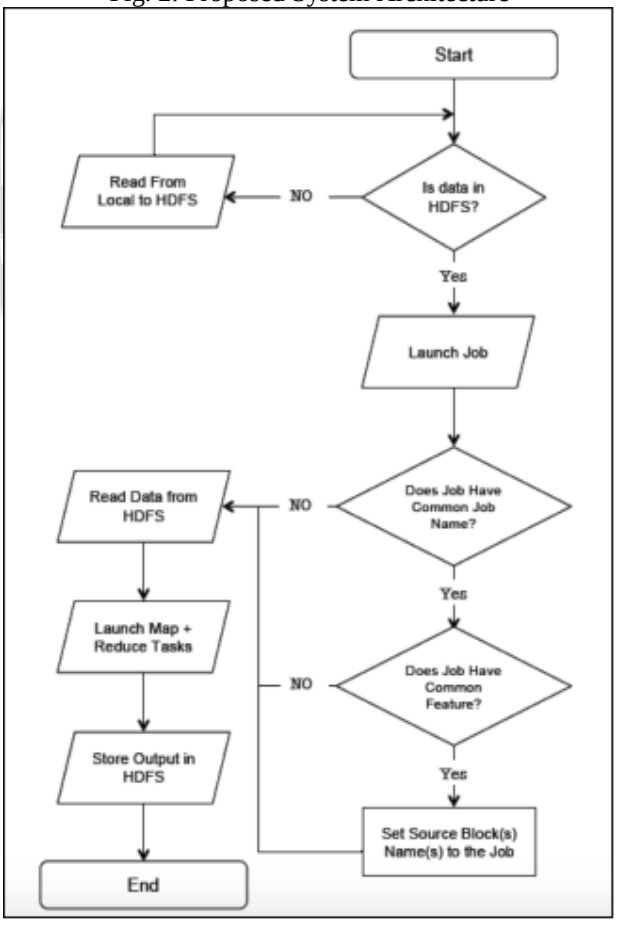

Fig. 3: Enhance Hadoop MapReduce Workflow Flowchart

VI. ALGORITHM

1) Client will send a request to NameNode.
2) If data is in HDFS format then it is ok otherwise data will be read from local cloud.
3) Then mapreduce job gets launched.

4) The mapreduce job is executed with some job name as well as features.
5) After execution, the results or part of results get stored in CJB table and expected output returns to client.
6) Next mapreduce job executed and if it has same job name or job feature that of old one then it executes immediately through that CJB table and blocks gets assign to that job.
7) If the size of CJB table gets beyond limit, it will take more time. So the size should be controlled and limited to reduce CPU time.
8) If the CJB table is develop using the RDBMS environment, then the size of CJB table may be specific or limited.

VII. MATHEMATICAL MODEL

Different parameters that jobs need to have to be executed efficiently. These parameters are:

1) Hadoop Parameters: which is a set of predefined configuration parameters that are in Hadoop setting files.
2) Profile Statistics: which are a set of user-defined properties of input data and functions like Map, Reduce, or Combine.
3) Profile Cost Factor: which are I/O, CPU, and Network cost job execution parameters.

The focus on the third category of parameters, which is the Profile Cost Factor. In this section there is explaination of the job execution cost in details. Also n the relationship between the number of blocks and the cost associated with the reading of the data from HDFS is explained here.

$$N = D / B$$

Where,
N= Number Of Blocks
D=The size of the raw data(DataSize)
B=The pre-defined size for data block (by default it is 64MB)
(BlockSize).

$$IOCR = N * HR$$

Where,
HR = The cost of reading a single data block from the HDFS(HdfsReadCost)
IOCR = The cost of reading the whole data from HDFS(IOCostRead)

$$IOCW = N* HW$$

Where,
HW = Cost of writing a single data block to HDFS(HdfsWriteCost).
IOCW = The cost of writing any data, such as MapReduce job results or raw data(IOCostWrite).

From the above equations , the total costs of reading and writing from HDFS depends on the number of blocks, which is the data size. So, by reducing the data size, one can reduce the costs of these processes, which will lead to improving the Hadoops performance. In Hadoops process the number of blocks is related to its costs. For example, the CPU cost of reading is CPUCostRead and is calculated as follows:

$$CPUCostRead = N * InUncompeCPUCost + InputMapPairs * MapCPUCost$$

Where,
InUncompeCPUCost = The compression ratio of blocks
InputMapPairs = The number of pairs for mapping process
MapCPUCost = The cost of mapping one pair.

The compression ratio that is applied to each block to have
it less in size before it is stored in the HDFS.

VIII. EXPERIMENTAL RESULTS

Lets take a DNA chromosomes example, in DNA chromosomes there are a couple of sequences that are common for searching protein process. This examples having some sequences and their locations are stored in CJBT table (i.e. it store the Chromosome Name in which chromosomes they occur).Sq1 is a common sequence in DNA chromosomes, and it is located on every single chromosome. The observations is that the longest sequence requires more CPU processing time, and that will be reduce in enhanced hadoop. We stored the CJB table in NameNode for easy access of DNA data in enhance Hadoop.

	Common Feature	Block Name/ID
sq1	GGGGCGGGG	In all Chromosomes
sq2	AAGACGGTGGTAAGG	1,8
sq3	CATTAAGATCC	1,2,3,4,6,7,9,10,11,12,13,18,19,21
sq4	GATTAGATGS	1,3,6,7,9,17,19,20,21
sq5	GATCTCAGCCAGTGTGAAA	3,7,16

Table 2: Common Job Block Table (DNA Example)

The fig3 and fig4 represents graph in the form of CPU time in seconds and number of read operations for the sequences in above table. The graphs in figures shows the CPU time and the number of read operations for selected queries in H2Hadoop and enhanced Hadoop architecture. It is observed that the proposed architecture requires less number of read operation and less CPU time than that of first one

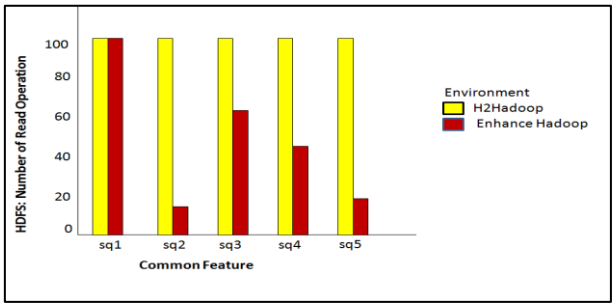

Fig. 3: Number of read operations in H2Hadoop and Enhance Hadoop for the same jobs.

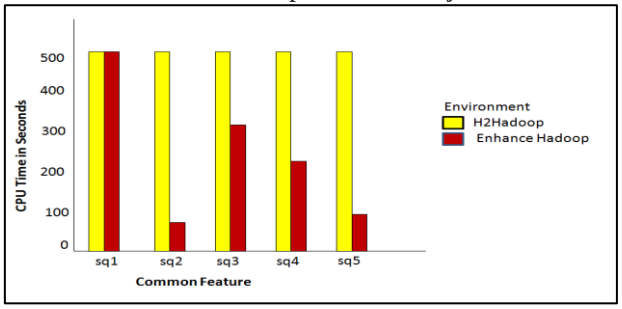

Fig. 4: CPU processing time in H2Hadoop and Enhance Hadoop for the same jobs.

IX. CONCLUSION

There are some limitations in H2Hadoop for the sequences which have common name and common features. The size of CJB table may increase dynamically, but the size of table should be limited and controlled to a specific size. If the size of table is controlled then it is efficient and reliable to execute sequence which have common features. So that the throughput of system will increase by reducing CPU time, number of read operations and another Hadoop factors.

REFERENCES

[1] Hamoud Alshammari, Jeongkyu Lee and Hassan Bajwa "H2Hadoop: Improving Hadoop Performance using the Metadata of Related Jobs IEEE TRANSACTIONS ON Cloud Computing, manuscript ID TCC-2015-11- 0399.
[2] Xiao Yu and Bo Hong "Bi-Hadoop: Extending Hadoop To Improve Support For Binary-Input Applications 2013 IEEE.
[3] Lohr, S., The age of big data. New York Times, 2012. 11.
[4] Changqing, J., et al. Big Data Processing in Cloud Computing Environments. In Pervasive Systems, Algorithms and Networks(ISPAN), 2012 12th International Symposium on. 2012.
[5] Chen, M., S. Mao, and Y. Liu, Big Data: A Survey. Mobile Networks and Applications, 2014. 19(2): p. 171-209.
[6] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin"Improving MapReducePerformance through Data Placement in Heterogeneous Hadoop Clusters 2010 IEEE.
[7] Joe B. Buck Noah Watkins Jeff LeFevre Kleoni Ioannidou"SciHadoop: Array-based Query Processing in Hadoop SC11 November 1218, Seattle, WA, USA.
[8] Balaji Palanisamy, Aameek Singh, Ling Liu" Purlieus: Locality-aware Resource Allocation for MapReduce in a Cloud SC 11, November 12-18,2011, Seattle, Washington, USA
[9] Rong Gua, Xiaoliang Yanga, Jinshuang Yana" SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters.
[10] Nishanth S, Radhikaa B, Ragavendar T J, Chitra Babu, and Prabavathy B"CoRadoop++: A Load Balanced Data Colocation in Radoop Distributed File System 2013 Fifth International Conference on Advanced Computing (ICoAC).
[11] Mehul Nalin Vora"Hadoop-HBase for Large-Scale Data
[12] Xuhui Liu, Jizhong Han, Yunqin Zhong, Chengde Han" Implementing WebGIS on Hadoop: A Case Study of Improving Small File I/O Performance on HDFS 2009 IEEE.
[13] Hamoud Alshammari, Hassan Bajwa and Jeongkyu Lee "Hadoop Based Enhanced Cloud Architecture For Bioinformatic Algorithms.
[14] Marx, V., Biology: The big challenges of big data. Nature, 2013.498(7453): p. 255-260.
[15] Apache hadoop 2.0, http://hadoop.apache.org/docs/r2.0.0-alpha/.

