ANALYSIS OF LOAD BALANCING ON DISTRIBUTED FILE SYSTEM FOR EFFECTIVE DATA RETRIVAL

[1]Salini R S, [2]Nadiya S , [3]Ramya K M, [4]Sreelakshmi V, [5]Vinodha K
[1]Student, [2]Student, [3] Student, [4] Student,[5]Assistant professor
Information Science and Engineering
The Oxford College of Engineering, Bangalore, India

Abstract– Load Balancing and Resource utilization are two important factors of DFS(Distribution File System). HDFS(Hadoop Distribution File System) plays a key role in Cloud computing environment and BigData analytics. In the present DFS, the file chunk distribution is dependent on central node which is susceptible for bottleneck and single point of failure. HDFS Uses 3 types of nodes Viz., namenode, secondary namenode and datanode to overcome single point failure. In this paper, the challenges faced in DFS are analyzed and a novel algorithm is proposed to address them. The imbalance in data access is due to the conventional parallel file system striping policies used for unevenly distribution of data among storage nodes.To overcome this HDFS stores each data unit, referred as chunk file, with several copies based on a relative random policy, which can result in an even data distribution among storage nodes . Based on the data retrieval policy in HDFS, if a storage node contains more requested data, the probability of accessing that node will be high resulting in bottleneck situation.To reduce the imbalanced access of dataOpass method is used. Opass adopts new matching-based algorithms to match processes to data so as to compute the maximum degree of data locality and balanced data access. Furthermore, to retrive the data fastly distribution algorithm and map reduce techniques are used.

IndexTerms **– Parallel data access, Distributed file system, HDFS, MapReduce.**

I.INTRODUCTION

A file system is a process that manages data storage and access on a hard disk drive(HDD). Distributed file systems such as GFS, HDFS, QFS or ceph, could be directly deployed on the disks of cluster nodes to reduce data movements. When storing a data set, distributed file systems will usually divide the data into small chunk files and randomly distribute them with several identical copies. Hadoop is an open source distributed processing framework that manages data processing and storage for big data applications running in clustered systems.The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications. It employs aNameNode and DataNode architecture to implement a distributed file system that provides high performance access to data across highly scalable Hadoop clusters. When retrieving data from HDFS, a client process will attempt to read the data from the disk that it is running on. If the required data is not on the local disk then the process will read from another node that contains the required data. The data requests from the parallel processes are referred to as parallel data requests. These data requests can be issued from Hadoop MapReduce applications, but also from MPI applications [2]. MapReduce is a programming framework that allows the user to perform distributed and parallel processing on large data sets in a distributed environment. Map Reduce consists of two distinct tasks – Map and Reduce. The message passing interface is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.

Unfortunately, the data requests from parallel processes or executors in big data processing will be served in an imbalanced fashion on the distributed storage servers and these parallel requests over the storage will compete for shared resources. Parallels Access is the simplest,fastest, and most reliable way to remotely access all your Windows/Mac applications and files on your iPhone, iPad, or Android phone or tablet. The Hadoop file system can allow parallel programs to access data by using its libHDFS library. The I/O interface, like hdfsread and hdfswrite, will be used to read/write data from/to HDFS. Another method is to use an I/O virtual translation layer to translate the parallel I/O operations. [1] Load balancing concept in a distributed system and a fully distributed load balancing algorithm mainly overcomes the load imbalance problem by using load balance Nearest Search algorithm but couldn't achieve Downtime, Limited control and

Vendor lockin.[2] DataMPI, which provided following MPI specifications: Dichotomic, Dynamic, Data-centric, and Diversified features .It provides advantages in performance and flexibility, but not able to reduce I/O response time, So SCALER is used. [3] SCALER, which allows MPI based applications to directly access HDFS without extra data movement and achieves the design goal efficiently such as Scalable parallel file write performance, Reducing I/O response time and Effective buffer for burst write workload. But this system is not suitable for small files. [4] A new methodology for managing read-write file sets across multiple file servers of a Distributed File System, thus balancing the load of file access requests across servers.It uses rule-based data mining technique and graph theory algorithms.

In section 2 represents an overview of HDFS andh MapReducing technique followed in document by section 3 that presents related work and followed by section4 presents existing models on HDFS. Various open research issues are presented in Section 5. Section 6 presents about opass technologies. A results explanation is given in Section 7 followed by concluding remarks in Section 8 .

II. HDFS AND MAPREDUCING TECHNIQUE

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It is highly fault-tolerant and provides high throughput access to application data. It is designed to be deployed on low-cost hardware and is suitable forapplications that have large data sets. It is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time.

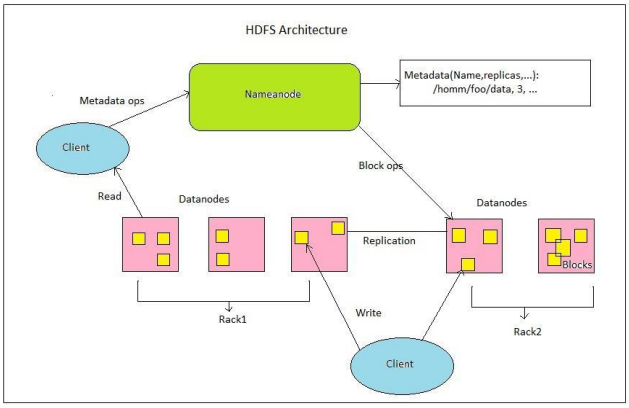

Fig 1. HDFS Architecture

HDFS has master/slave architecture.As shown in fig1, an HDFS cluster consists of a single NameNode which is a master server that manages the file system namespace and regulates access to files by clients and there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. The NameNode and DataNode are pieces of software designed to run on commodity machines.

These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software.

The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. HDFS exposes a file system namespace and allows user data to be stored in files.Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. It also performs block creation, deletion, and replication upon instruction from the NameNode.The NameNode executes file system namespace operations like opening, closing, and renaming files and directories and makes all decisions regarding replication of blocks.It also determines the mapping of blocks to DataNodes. It periodically receives a Heartbeat and a Block report from each of the DataNodes in the cluster.Receipt of a Heartbeat implies that the datanode is functioning properly.

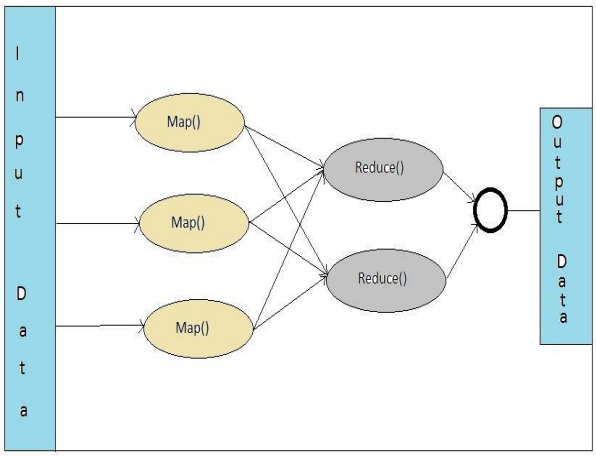

Fig2. Operation of MapReduce

Hadoop MapReduce is a software framework which is basically used for easy writing applications to process vast amounts of data in-parallel on large clusters of commodity hardware in a reliable and fault-tolerant manner. A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner as shown in fig2. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system.The framework takes care of scheduling tasks, monitors them and re-executes the failed tasks.Typically, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes.

This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very aggregate bandwidth across the cluster.The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node.The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks.The slaves execute the tasks as directed by the master.

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.The key and value classes have to be serializable by the framework and hence need to implement the Writable interface and also need to implement the WritableComparable interface to facilitate sorting by the framework.

III.RELATED WORK

Radha G. Dobale[1] discussed the load balancing concept in a distributed manner and a fully distributed load balancing algorithm is proposed to cope with the load imbalance problem. The load balance Nearest Search algorithm is used to migrate one user's whole file into any other node instead of partitioning a file into a no. of chunks. Radha G. Dobale[1] achieved Load balancing,Scalability,Availability and Maintenance, but couldn't achieve Downtime, Limited control and Vendor lockin.

Xiaoyi Lu[2] proposed DataMPI, which provided following MPI specifications: Dichotomic, Dynamic, Data-centric, and Diversified features.Performance experiments showed that DataMPI has significant advantages in performance and flexibility,while maintaining high productivity, scalability, and fault tolerance of Hadoop.

Xiaoyi Lu[3] was not able to reduce I/O response time. Xi Yang[3] introduced a system solution, named SCALER, which allows MPI based applications to directly access HDFS without extra data movement.SCALER supports N-1 file write at both the inter-block level andintra-block level.Experimental results confirm that SCALER achieves the design goal efficiently such as Scalable parallel file write performance,reducing I/O response time and Effective buffer for burst write workload. Here, this system is not suitable for small files.

Valeria Cardellini[4] proposed system that reviewed the state of art in load balancing techniques on distributed web server systems and analyzed the efficiencies. Popular websites canneither rely on a single powerful server nor on independent mirrored servers to support the ever increasing request load. Scalability and availability can be provided by the distributed web server architecture that schedule client requests among the multiple server nodes in a usertransparent way. The proposed system was able to eliminate server overhead and bottleneck but had limited applicability, increased latency time and dispatcher bottleneck.

Amit Gajbhiye[5] discussed about Global ServerLoad Balancing with Networked Load Balancers for Geographically Distributed Cloud Data-Centres and critically analysed the state-of-the arttechniques used for Global Server LoadBalancing and have proposed a novel model of

networked load balancers for load balancing across the datacenters in cloud computing environment. The proposed model overcomes the shortcomings of existing DNS based load balancing by considering current load status of datacenter, by making time to live of DNS server cache redundant and by finding the exact location by real IP of end user. The proposed system achieved Load balancing with networking among load balancers in datacenters in distributed environment, A novel load calculation was done and Response time was reduced to transfer the request across multiple datacenters. The system was unable to Deploy the model in real time environment.

Alexandra Glagoleva [6] presented a new methodology for managing read-write file sets across multiple file servers of a Distributed File System, thus balancing the load of file access requests across servers. The proposed methodology is based on a rule-based data mining technique and graph theory algorithms. The rule-based technique generates rules from access request data to identify present file access patterns in the system. The algorithm for fileset relocation is based on the graph coloring problem.

IV.EXISTING SYSTEM MODELS

1.MapReduce technique for parallel-automata analysis of large scale rainfall data:

MapReduce is a technique for executing exceedingly parallelizable and distributable algorithms across huge datasets utilizing countless PCs. Utilizing MapReduce with Hadoop, the large-scale rainfall could be determined without adaptability issues. Vast scale rainfall information assumes an imperative part in farming field thus early expectation of rainfall is important for the better financial development of a nation. information for accurate rainfall Big Data innovation like Hadoop have developed to fathom the difficulties and issues of huge information utilizing distributed computing.

In this model the huge scale rainfall information is anticipated by utilizing MapReduce system which plays out the capacities which are required and decrease the task to get proficient arrangements through taking the information and isolating into smaller tasks. The three Regression Automata (RA) algorithms such as Linear Regression automata, Support Vector Regression Automata and Logistic Regression Automata are utilized to forecast the future esteem of large scale rainfall data. This model also serves as a tool that takes in the rainfall information from diminished information as input and predicts the future rainfall. The outcomes obviously demonstrate that the all the three RA models can anticipate the rainfall productively in different terms, such as, error rate, coefficients and mean square error.

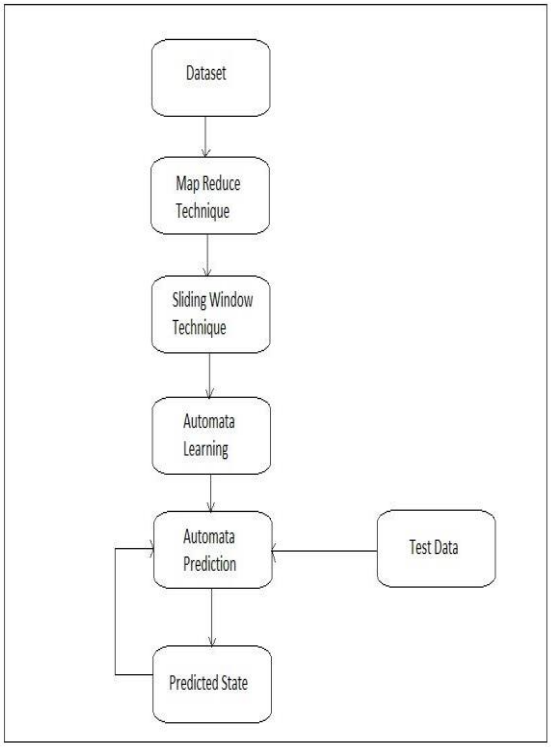

Fig 3. Map reduce technique for parallel-automata analysis.

2.Unstructured Data Analysis on Big Data using Map Reduce:

Large amount of unstructured data needs structural arrangement for processing the data. Hadoop is binary compatible with Map reduce. Map Reduce is a shuffling strategy to perform filtering and aggregation of data analysis tasks. Map is nothing but the filtering technique used for filtering the datasets and similarly Reduce is a technique used for aggregation of data sets. In the real time scenario, the volume of data used linearly increases with time.

Social networking sites like Facebook, Twitter discovered the growth of data which will be uncontrollable in the future. In order to manage the huge volume of data, the proposed method will process the data in parallel as small chunks in distributed clusters and aggregate all the data across clusters to obtain the final processed data. In Hadoop framework, MapReduce is used to perform the task of filtering, aggregation and to maintain the efficient storage structure. The data are preferably refined using collaborative filtering, under the prediction mechanism of particular data needed by the user.Collaborative Filtering Technique is used to generate recommendations based on user data. Sentiment Analysis is a technique which uses natural

language processing and Text analysis techniques for predicting the user sentiments based on polarity.

The proposed method is enhanced by using the techniques such as sentiment analysis through natural language processing for parsing the data into tokens and emoticon based clustering. The process of data clustering is based on user emotions to get the data needed by a specific user. The results show that the proposed approach significantly increases the performance of complexity analysis.

3.Mining on Big Data Using Hadoop MapReduce Model:

The proposed hadoop model consists of five tools namely data preprocessing, data migration with sqoop, data analytics with hive, data analytics with pig and data analytics with map reduce.Data preprocessing module is used to create data set for making item-set of product.

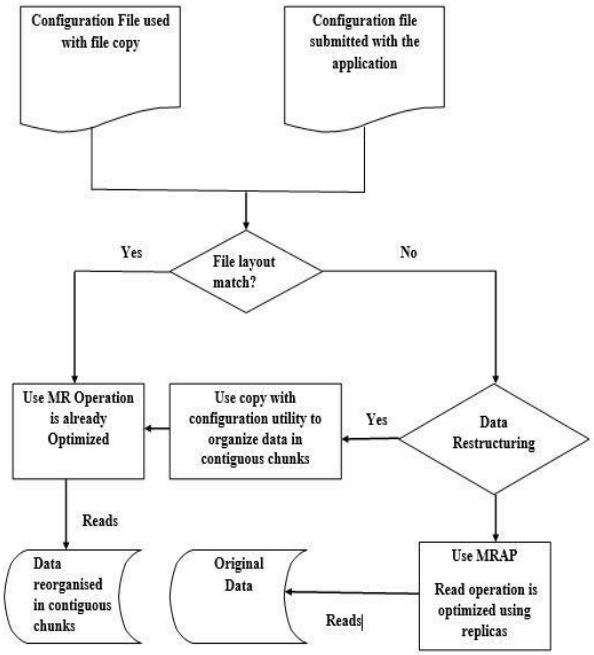

Fig 4. MRAP Technique for Date Restructuring

The data migration with sqoop module is used to transfer the dataset into Hadoop. Sqoop is a tool for transferring data between databases and Hadoop. With the help of this module the dataset is fetched into Hadoop using sqoop tool. Sqoop is used to perform many functions, such as to fetch the particular column or to fetch the dataset with specific condition that will be supported by sqoop tool and data will be stored in Hadoop.Data analytic with hive module is used to analyzestructure language.Hive is a data ware house system for Hadoop.

It runs structured query language (SQL) like queries called hive query language (HQL). The HQL is converted internally to map reduce jobs. Hive was introduced by Facebook.Hive supports functions like data definition language (DDL), Data manipulation language (DML) and user defined functions. In this module the dataset is analyzed using hive tool which will be stored in Hadoop. To analyze dataset hive is used with HQL.Using hive Partition, Bucketing can be performed.The module of data analytic with pig is also used to analyze data set.

Apache pig is a high level data flow platform to execute map reduce programs with Hadoop. Data analytic with map reduce module is also used analyze data set with map reduce. Map reduce is a processing technique using program model of java for distributed computing. The map reduce algorithm contains two important tasks such as map and reduce. The task map is used to map with chart, record, plot, drawing, plan and diagram etc. Whereas task reduce is used to minimize the dimension.

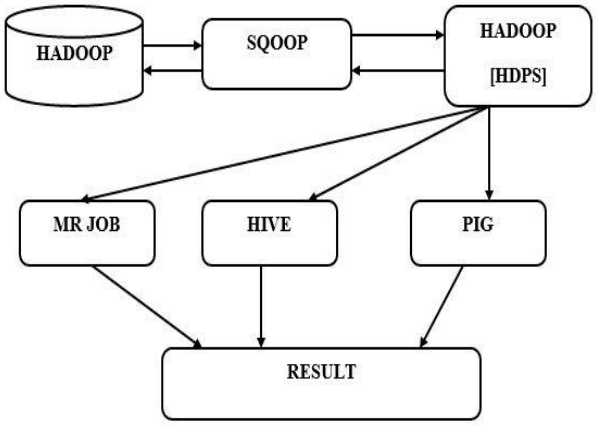

Fig 5. Hadoop MapReduce Technique on mining

V.ISSUES IN HADOOP MAPREDUCE

Challenge 1: lack of performance and scalability The hadoop mapreduce programming model do not provide a fast, scalable distributed resource management solution. Organisations require a distributed Mapreduce solution that can deliver competitive advantage by solving wider range of data intensive analyticproblems faster. The mapreduce implementation should help organizations run complex datasimulation with submillisecond latency,high data throughput and Thousands of mapreduce tasks completed per seconds depending on complexity.

*Challenge2:*Lack of flexible resource management. The hadoop MapReduce programming model are not able to react quickly to real time changes in application or user demands. Based on the volume of tasks, the priority of the job and time varying resource allocation policies, mapreduce jobs should be able to quickly grow or shrink the number of

concurrently executing tasks to maximize throughput,performance and cluster utilization while respecting resource ownership and sharing policies.

Challenge3: Lack of application deployment support. The hadoop MapReduce programming model do not make it easy to manage multiple application integrations on production-scale distributed system with automated application service deployment capability.

Challenge4: Lack of quality of service assurance.The hadoop Mapreduce programming model are not optimized to take full advantage of modern multicore servers.The implementation shouls allow for both multithreaded an single threaded tasks and be able to schedule them with a view to maximize cache effectiveness and data locality into consideration.

Challenge 5: Lack of multiple data source support
The hadoop Mapreduce programming model only support a single distributed file system, the most common being HDFS. A complete implementation of the MapReduce programming model should be flexible enough to provide data access across multiple distributed file systems. In this way,existing data does not need to be moved or translated before it can be processed. MapReduce services need visibility to data regardless of where it resides.

Challenge 6: Privacy and security challenges
There are issues in auditing, access control. Authentication and privacy when performing mapper and reducer jobs.
To solve the mentioned challenges in this section,the following opass methodologies are used.

VI. METHODOLOGY

1. Encoding Process:

The parallel data read requests can be served in a balanced way through maximizing the degree of data locality read. To achieve this, we retrieve data distribution information from storage and build the locality relationship between processes and chunk files, where the chunk files will be associated with data processing operators/ tasks according to different parallel applications[16].

Map reduce background: MapReduce is a programming model suitable for processing of huge data. Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.
MapReduce consists of several components, including:

- Job Tracker -- the master node that manages all jobs and resources in a cluster
- Task Trackers -- agents deployed to each machine in the cluster to run the map and reduce tasks.

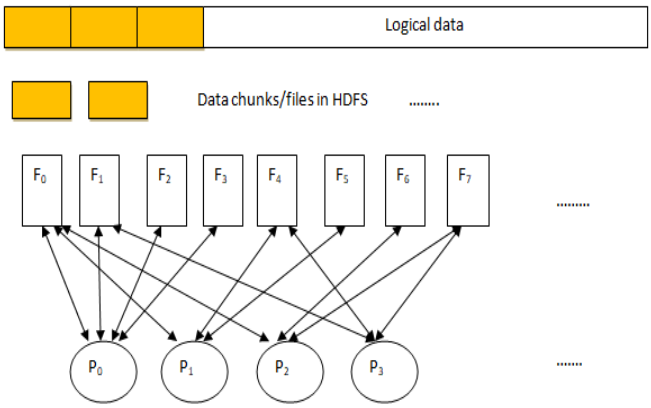

Fig 6. A bipartite matching example of processes and chunk files[16].

- JobHistoryServer -- a component that tracks completed jobs, and is typically deployed as a separate function or with Job Tracker.

The basic architecture of map reduce is as shown in fig 2. The term MapReduce actually refers to two distinct jobs that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. Both these functions can be prototyped as follows:
Map(K1,V1)→[(K2,V2)].
Reduce(K2,{V2})→[(K3,V3)].
When a file is divided into equal sized blocks (64MB-128MB) and each block is assigned to a cluster, the job tracker starts a map task for each data block and typically assigns it to the task tracker on the machine. Each data block will have a mapper. Each mapper will have a corresponding reducer. The mapper performs map function and we get the intermediate results. This intermediate results enters the reduce phase. The reducer performs reduce function. The results of all task tracker are combined to get the output.

Matching-Based Algorithm for Tasks with Multi-Data Inputs is used for encoding process. This algorithm is accessed through port number 50070.

2. Optimizing Parallel Single Data Accessin DFS:

The overall execution time for parallel data analysis will be decided by longest running process.For a single task with multiple data inputs as shown in fig 7, the required inputs may be placed on a multiple data nodes, which imply that some of the data associated with given task may be local to the process assigned to that task and some might be remote so, tasks with multiple inputs will complicate the matching of processes to data. we propose a novel matching based algorithm for this type of parallel data accesses. Our algorithm aims to associate each process with data processing tasks, such that a large amount of data can be read locally. We assign each process with the equal number of tasks for parallel execution. Our algorithm achieves the optimal matching value from the perspective of each process. To begin, we compute the amount of co-located data associated with each task and each process and encode these values as the matching values between them and we will find a task .

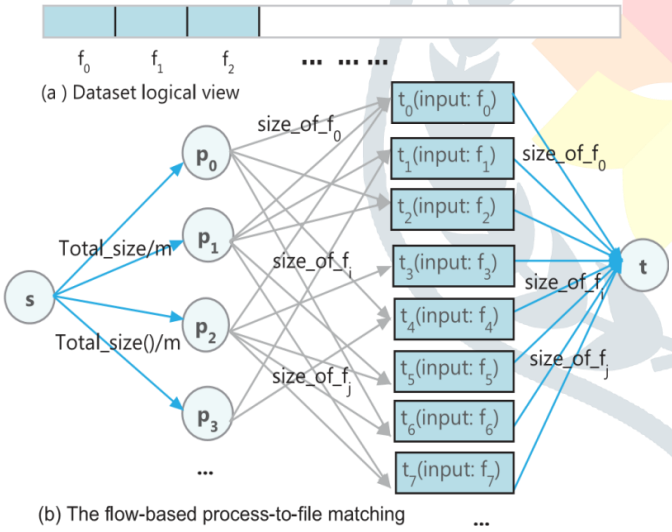

Fig 7. The matching-based process-to-file configuration for single-data access[16].

3. Optimizing Parallel Multi-Data Read Access in DFS:

In previous method we are accessing single data but here we are accessing multiple data.We propose a novel matching based algorithm for this type of parallel multi-data read access. Our algorithm aims to associate each process with data processing tasks such that a large amount of data can be read locally.

4.Opass for Parallel Data Read Access in DFS:

HDFS uses an r-way replication to providehigh availability. Files in HDFS are referred as chunks

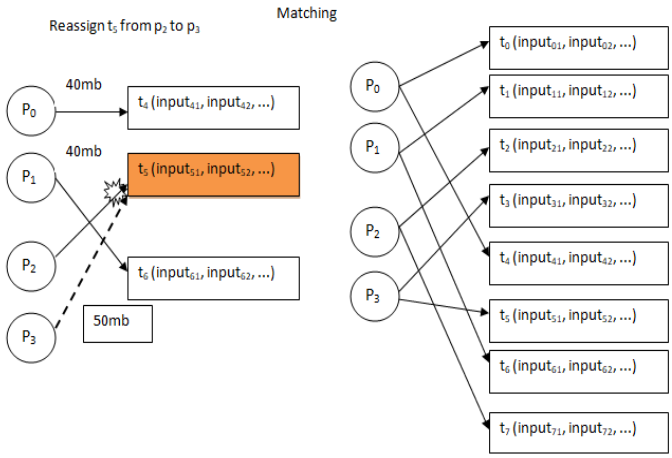

Fig 8 The process-to-data matching example for multiple data assignment[16].

and each chunk will be copied to r DataNodes. When a read request is initiated for a chunk, by default, the NameNode will return a sorted list of DataNodes that holds the requested chunk. The nearest Data Node will be picked to serve the read request. When the degree of imbalance crosses a pre-defined threshold, we will replace the default locality driven read strategy with our matching based method.

Distribution algorithm is used to evenly distribute the read requests for file's chunks.In fig 9,Capacity is determined by each Data Node's current number of chunks served and the number of chunks of newly requested file. Expected matching is denoted in red doted line.

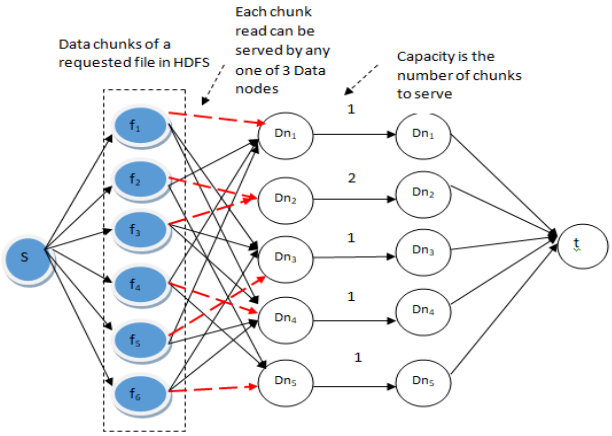

Fig 9. Balanced file read requests matching using network flow[16].

Algorithm1. Algorithm to read chunks in distributed file system.

Let the file f consists of m chunks stored on n DataNodes.

Let the set NP ={np_0,np_1,…,np_{n-1}} represent the current number of chunk data requests served by data nodes.
Let the set P ={p_0,p_1,…,p_{n-1}}represent the number of chunk data requests for file F that the data nodes will serve.
Let the set l={l_0,l_1,….l_{n-1}} represent the number of chunks of F that each Data nodes holds.
Input: m,n,P,NP; **Output:** P
Steps:
Find the maximum number of read requests served by current data bodes: np_j=Max(nP)
* assign m chunks to n data nodes */
For np_i in NP and np_i<np_j**do**
if np_j- np_i>l_i**then**
p_i = l_i ; m=m-p_i;
else
p_i = np_j- np_i ; m=m-p_i;
end if
while m>0 **do**
for np_i in PC and m>0 **do**
if p_i< l_i **then**
p_i= p_i+1; m=m-1
end if
end for
end while

Algorithm2. Dijkstra's algorithm

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbours and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbour *B* has length 2, then the distance to *B* through *A* will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

VII. RESULT:

To test OPASS on parallel processing applications, we record the I/O time taken to read each chunk files by comparing with three matrics, the average sI/O time taken to read all chunk files, the maximum I/O time and the minimum I/O time as shown in fig 10,11. Fig 10. represents reading data from HDFS without implementing opass . This shows that the I/O time become more variant has the cluster size increases. the maximum I/O time increases drastically while the minimum I/O time remains constant.For instance, on the 16 node to 80 node cluster the maximum I/O time to read a chunk file reaches from 7 X to 18X. This is not suitable for parallel programs which has longest operation for execution.

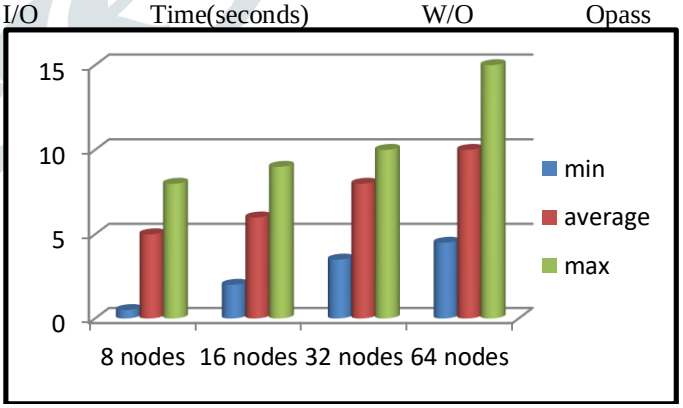

Fig 10.Read data from HDFS without Opass.

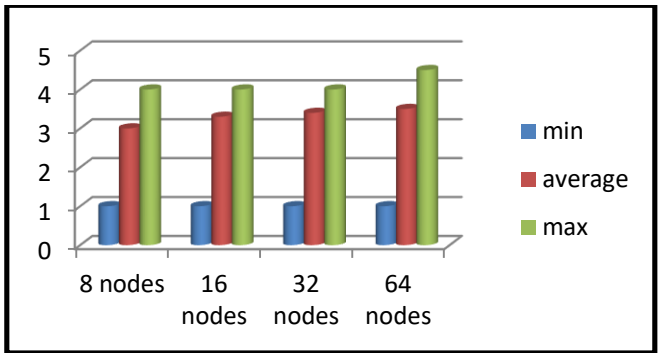

Fig. 11. Read data from HDFS with Opass.

With the use of Opass, as shown in Fig. 11, the I/O performance remains constant as the cluster sizeincreases, with an average I/O time of around0.9 seconds. With the use of Opass, the I/O time during the entire execution is approximately oneto two seconds. In all, the average I/O operation time with the use of Opass is a quarter of that without opass.

VIII. CONCLUSION

In this paper the problems of parallel data reads/writes on distributed file systems is analysed due to the lack of consideration of data distribution, parallel data requests are often servedin an imbalanced and remote fashion.

To overcome the imbalance fashion in distributed file system ,opass methods are used.The data retrieve is improved by 10% compared to existing system.

In future, on hadoop for dynamic and iterative processing we are trying to identify the data changes whenever there is a dynamic updation using FP algorithm.

ACKNOWLEDGMENT

We would like to express our gratitude to Dr. Praveena Gowda, Principal, The Oxford College of Engineering for providing us a congenial environment and surrounding to work in. Our hearty thanks to Dr. R. Kanagavalli, Professor & Head, Department of Information Science and Engineering, The Oxford College of Engineering for her encouragement and support. We convey our gratitude to Dr. Vinodha K, Asst. Professor, Department of Information Science and Engineering for having constantly monitored the completion of the Project Report and setting up precise deadlines.

References:

[1] Review of Load Balancing for Distributed Systems in Cloud

[2] DataMPI: Extending MPI to Hadoop-like Big Data Computing

[3] SCALER: Scalable Parallel File Write in HDFS

[4] Title Dynamic Load Balancing on Web-server Systems

[5] Amit Gajbhiye discussed about Global Server Load Balancing with Networked Load Balancers for Geographically Distributed Cloud Data-Centres and critically analysed the state-of-the art techniques used for Global Server L

[6]Alexander Glagoleva presented a new methodology for managing read-write file sets across multiple file servers of a distributed file system, thus balancing the load of file access requests.

[7] Apache Hadoop YARN: Yet Another Resource Negotiator

[8] G. King, "Big Data is Not About the Data!" Presentation *(*Harvard University USA, 19 November 2013*)*.

[9] Samee U. Khan Enabling Big Data Complex Event Processing for Quantitative Finance through a Data- Driven Execution.

[10] T. H. Davenport and J. Dyché, "Big data in big companies," International Institute for Analytics.

[11] B. Fang, P. Zhang, in BigData in Finance. Big Data Concepts, Theories, andApplications,ed. by S.Yu, S. Guo (Springer International Publishing, Cham, 2016).

[12] T.-C. Dao and S. Chiba, “HPC-reuse: Efficient process creation for running MPI and Hadoop MapReduce on supercomputers,” in Proc. 16th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput., 2016, pp. 342–345.

[13] A. Darling, L. Carey, and W.-C. Feng, “Thedesign, implementation, and evaluation of mpiblast,” presented at the ClusterWorld, San Jose, CA, USA, 2003.

[14] J. Dean and S. Ghemawat,“MapReduce: Simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[15] L. R. Ford Jr and D. R. Fulkerson, “A suggested computation for maximal multi-commodity network flows,” Manag. Sci., vol. 5, no. 1, pp. 97–101, 1958.

[16] “Achieving Load Balance for Parallel Data Access on Distributed File Systems”,
Dan Huang, Dezhi Han, Jun Wang, Jiangling Yin, Xunchao Chen , Xuhong Zhang, Jian Zhou, and Mao Ye, vol.67,no.3, 2018