Secured Load Re-Deployment of File Chunks in Distributed File System

Roopadevi D S, Mrs. Nandini G B.E,M.Tech(Ph.D)
PG Student, Assistant Professor ,Department of Computer Science,
RajaRajeswari College of Engineering, Bangalore-74
roopadevids@gmail.com, Email: nanduamma@gmail.com

ABSTRACT-In cloud computing environment, the Distributed File Systems are the most important building blocks. The DFS is used to store nodes and they perform many functions of computing applications. Files can be created, deleted and appended dynamically. It results in load imbalance in a distributed file system, that is, file chunks are not distributed as uniformly as possible among the nodes. The distributed file system in production system powerfully depend on the central node for chunk re allocation. This dependency is clearly inadequate in a large scale; failure-prone environment because the load balancer in the center is put under significant workload that is linearly scaled with the system size and because of that, it becomes the performance bottleneck and the single point failure. A fully distributed file load balancing algorithm is proposed to handle the load imbalance problem. Load is transferred from heavily loaded node to physically closed lightly loaded node. Encryption is done for the raw file before dividing it into chunks and later it will be decrypted.

Keywords - Distributed File System, Name node, Data Node, Cloud Computing, Map Reduce.

1. INTRODUCTION

Distributed file system is mainly based on the client-server architecture, where the client obtains the data stored on the server as if it is locally present on their own system. The distributed file system is mainly developed for extensive data storage and data access.

The main characteristic is that the file contents can be stored in various nodes which shares and transfer the data. There are various types of distributed file system, one among them is the HDFS (Hadoop Distributed File System).

HDFS is very similar to the Google File System. HDFS is built using java language. HDFS is designed to store and process large data sets. It is designed in such a way that it can run on the computer hardware which is affordable and easy to obtain. HDFS takes care of storing files of huge volume of data. HDFS component is used by the administrator. The main features of the HDFS are:

- It is fault tolerant and can be developed on the low cost commodity hardware.
- It provides high throughput admission to application data.
- Quick and automatic recovery from the faults.

HDFS applications are based on the notion of ―write one read many‖. HDFS architecture is like master-slave architecture. The HDFS consists of two main components namely, A single name node and the multiple instances of data node. The name node is also called as master server or admin node. Remaining all other nodes are called data nodes. These Data nodes are also called as slave node or worker node. The name node and data nodes are software‘s designed to run on the low cost hardware machines.

A file is split or divided into a number of small pieces called chunks. These chunks of files are assigned to the different data nodes. The data nodes create or delete the file chunks in accordance with the instruction of the central node or main server.

The functionalities of the name node are:

- It manages the metadata information of the file system.
- It controls the access to files by client.
- It allows mapping of data blocks to data nodes.
- It performs operation like opening, closing and renaming of files.
- It provides the list of HDFS files that belong to each data block, the current location of the block and the state of the file.

The functionalities of data node are:

- Information contained in the files is stored in the data nodes, that is, the file contents are stored in the data nodes.
- Data nodes serve the read and write requests from the clients.
- It performs creation, deletion and replication of the blocks based on the instructions from the name node.

In, HDFS, the file is divided or split into number of small pieces called chunks. These chunks are assigned to different data nodes. The load possessed by the nodes is proportional to the number of file chunks the data node holds. A HDFS cluster is said to be in a balanced state if there are no overloaded or under loaded based on the percentage of the DFS space used by the data nodes.

A data node is said to be overloaded data node if the DFS space used by the data node is greater than the predefined threshold. A data node is said to be under loaded if the DFS space used by the data node is lesser than the predefined threshold. The threshold can be

configured by the user that is, the threshold can be changed by the user.

The imbalanced state is created in the HDFS as and when the new data nodes joins the system. Whenever the file is created and also when the file chunks possessed by the data nodes exceed the threshold capacity or when the file chunks possessed by the data nodes are below the threshold capacity. The distributed file system depends on a central node for chunk redistribution. This dependency puts the central node under the considerable workload and becomes the performance bottleneck that causes the entire process to slow down or stop and single point of failure.

The objective is to allocate the chunks of the files as equally as possible among the storage nodes such that no nodes manage an excessive number of chunks and also movement cost is reduced. To achieve the objective, a load redistribution algorithm is used to solve the load imbalance problem among all the chunk servers that is the data nodes in the HDFS. Data nodes performs the load rebalancing task spontaneously without name node due to this the performance of the utilization of the system is improved.

2. LITERATURE SURVEY

Paper 1: Hsueh-Yi Chung, Che-Wei Chanz Hung-Chang and Hsiao Yu-Chang Cho [1] proposed the Load Rebalancing Problem in Distributed File Systems where the Distributed File Systems are principal and the crucial building blocks for cloud computing applications. In such a file systems, nodes concurrently assist storage and computing functions. A file is divided into a number of small pieces called as file chunks and are assigned in evident nodes so that venture can be performed in analogous over the nodes. In a cloud computing environment, failure is normal and nodes maybe enhanced, restored and added in the system. Files can also be dynamically deleted, created and added. The outcome of this is load imbalance state, that is, the file chunks are not distributed as evenly as possible among the nodes. Despite of distributed load balancing algorithm exist in the literature to deal with the load imbalance problem. Upcoming distributed file system in production as systems completely depend on the central node for chunk re assignment. This dependence is clearly insufficient in a large scale failure and vulnerable environment because the central load balancer is placed under considerable workload, that is, it grows linearly with the system size and thus becomes the staging bottleneck and the single point of failure.

A file is divided into a number of small pieces called as file chunks and are assigned in evident nodes so that venture can be performed in analogous over the nodes. In a cloud computing environment, failure is normal and nodes maybe enhanced, restored and added in the system. Files can also be dynamically deleted, created and added. The outcome of this is load imbalance state, that is, the file chunks are not distributed as evenly as possible among the nodes. Despite of distributed load balancing algorithm exist in the literature to deal with the load imbalance problem. Upcoming distributed file system in production as systems completely depend on the central node for chunk re assignment. This dependence is clearly insufficient in a large scale failure and vulnerable environment because the central load balancer is placed under considerable workload, that is, it grows linearly with the system size and thus becomes the staging bottleneck and the single point of failure.

Paper 2: Prasanna Ganesan, Mayank Bawa and Hector Garcia [2] proposed Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems where the range partitioning is used to solve the problem of horizontally partitioning a dynamic relation over a large number of disks of nodes. Such a kind of segregation is usually advisable in large scale parallel databases, as well as in peer-to-peer systems. As and when the tuples are deleted and inserted, the segregation may need to be modified and data migrated in order to achieve storage balance across the participant disks or nodes. Efficient asymptotically optimal algorithms that confirm storage balance at all times are used, even against a characterized deletion and insertion of tuples. The above algorithm is consolidated with distributed routing structure to architect a peer-to-peer system that supports efficient range queries, while concurrently assuring storing balance.

Paper 3: H.-C. Hsiao, H.Liao and S.-S. Chen [3] proposed Load Balance with imperfect Information in Structured Peer-to-Peer System with the conviction of virtual server, peers involved in a heterogeneous, structured peer-to-peer network may host different numbers of virtual servers, peers can balance their loads corresponding to their loads corresponding to their capacities. The already present decentralized load balance algorithms designed for the heterogeneous, structured peer-to-peer networks either construct additional networks to exploit global information or demand peer-to-peer nodes to be organized in a hierarchical fashion. Without depending on any backup networks and independent of the geometry of the peer-to-peer substrates. Based on the partial knowledge of the system, A novel load balancing algorithm that is distinct estimates the probability of distribution of the capacities of peers and loads of virtual servers, causing in imperfect system condition, peers can determine their anticipated loads and reassign their loads in parallel. The comparison is done through notable simulations; it is differentiated with prior load balancing algorithm.

Paper 4: Wenqiu Zeng, Ying Li and Jian Wu [4] proposed load rebalancing in Large-Scale distributed File System with the advancement of data on the internet has shown the massive growth. Some researchers have paid their attention to find an effective way to keep and govern these data. A load rebalancing algorithm to solve the load balancing problem all compute nodes in distributed file

system is used. It also guarantees that one block of file and its two replicates are stored in three different chunk servers simultaneously at the same time. This algorithm fulfills the load balancing while guarantying the reliability of the system.

Paper 5: Qingqing Zhong [5] proposed a load-balancing approach for DHT-based P2P Networks the nitpicking issue in the efficient operation of peer-to-peer network is load balancing. There are various number of suggestions exist for load balancing for structured peer-to-peer network. The adaptive load balancing for structured peer-to-peer system is used. The technique aims to balance the request and routing load of the peer under unfairly request of workloads. These procedures outstandingly improve the distribution of the load and provide inevitably better scalability. With more workloads the false positives rate is reasonably low through experiment results.

3. PROPOSED SYSTEM

In this paper, in the proposed system a load rebalancing algorithm is used to tackle the load imbalance problem. This algorithm is made to run in the rebalancing server which makes the rebalancing decision. It is invoked with the command as and when the imbalance state is created in the system. This rebalancing algorithm makes each and every data node present in the HDFS to communicate with each other and estimate the load of the neighboring data nodes. Based on the estimated load the data nodes are classified as overloaded data node and under loaded data node. These data nodes are sorted based on the number of file chunks each data nodes possesses. The data nodes are sorted from least lightly loaded node to heavily loaded node. The least lightly loaded node migrates the chunk of file it possess to the next lightly node to transfer the file chunks which is lesser than or equal to the threshold. This process repeats unless and until all the data nodes in the system becomes normal loaded. The Raw data file is encrypted before dividing the file into chunks and it will be decrypted again later.

Benefits of Proposed System

- The chunks are assigned as equally as possible among the data nodes.
- The movement cost is reduced.
- It solves the performance bottleneck problem and a single point of failure as it no longer depends on the central nodes for chunk reallocation.
- The data nodes balance their loads spontaneously by eliminating the dependency on the central node.
- The performance of the system is improved.
- The resource availability is improved and resource utilization also increases.

In the Proposed system four different module are implemented. They are as follows,

- Chunk Distribution
- Identifying Nodes
- Load Rebalancing Algorithm
- Security

Load Rebalancing Algorithm Module

In the rebalancing algorithm, every chink storage node initially estimates whether or not it is lightly loaded or heavily loaded node. A node is lightweight if the amount of chunks it hosts is smaller than the predefined threshold. Every node contacts variety of other nodes within the system and builds a vector denoted by V. this vector consists of entries and every entry contains the IDs of the nodes. This light weighted node selects one of the heavy weighted node for the reallocation of the chunks that heavy weighted node possesses.

Chunk creation

```
Begin
    Select a file to split
    For the selected file
        Split the file into chunks
    For end
    Store the file chunks into the servers
End
```

Chunk Servers Module

```
Begin
Interact with each server to gather information regarding
light and heavy load, including locations of chunks.
    If under loaded then send request to migrate the
    chunks
        Repeat till no progress
End
```

Stepwise Implementation of Rebalancing Algorithm:

1. To Estimate and find the lightly loaded node in the set of sample data nodes

Step 1: Get the live data nodes report from name node.
Step 2: Calculate the average utilization of DFS space used.
Step 3: Sort the data nodes according to the DFS space used in ascending order.
Step 4: Choose the least lightly loaded node as source node i in sample data nodes.
Step 5: Choose the heavily loaded node as destination node j in sample data nodes.

2. To migrate file chunks across the data nodes

Step 6: i moves the file chunks it possesses to i+1.
Step 7: i requests j to migrate the chunks it possesses below or equal to the threshold.
Step 8: i becomes j's successor that is i←j+1.
Step 9: j removes the file chunks assigned to i.
Step 10: Estimate the load of the sample data nodes.
Step 11: If imbalanced goto step-3 repeat the process unless and until all the sample nodes are balanced.

Step 12:update the chunk location information to name node.

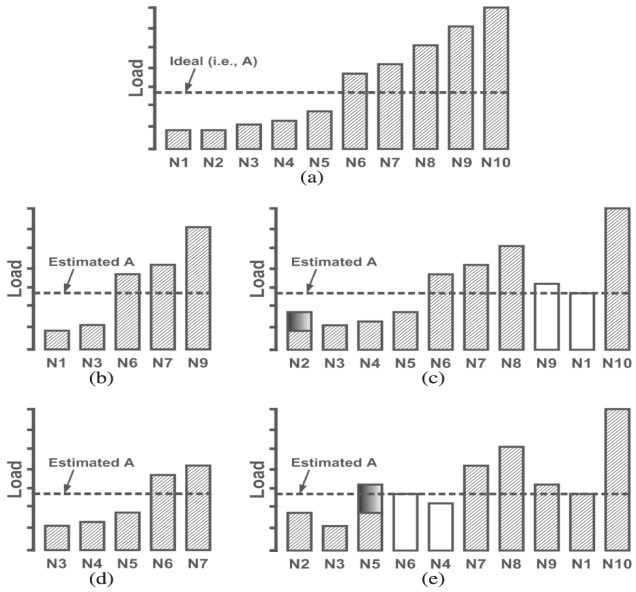

Fig 1. An example illustrating our algorithm where (a) the initial loads of chunk servers N1,N2,N3,…..N10, (b) N1 samples the load of N1,N3,N6,N7 andN9 in order to perform the load rebalancing algorithm, (c) N1 leaves and sheds its loads to its successor N2 and then rejoins as N9'ssuccessor

We have to make sure that the data that has to be stored in the cloud has to be secured. In order to provide the security, the raw data is encrypted. In this proposed system, an algorithm is used to encrypt the data. Later this encrypted file is divided into chunks and stored in various nodes.

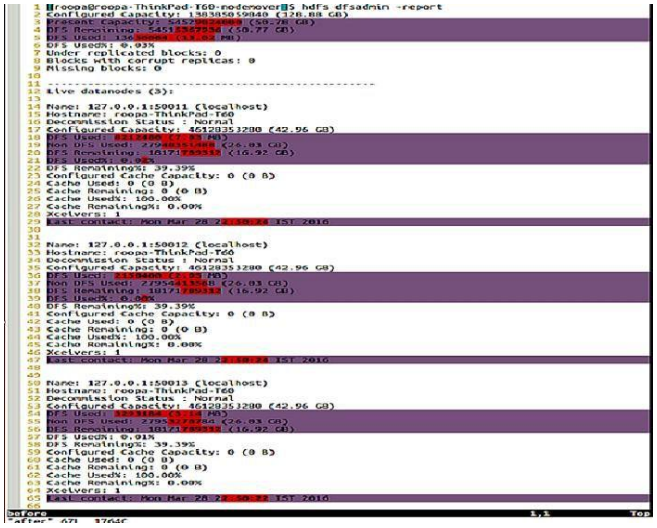

Fig. 2. Before: Imbalanced loads on Data Nodes

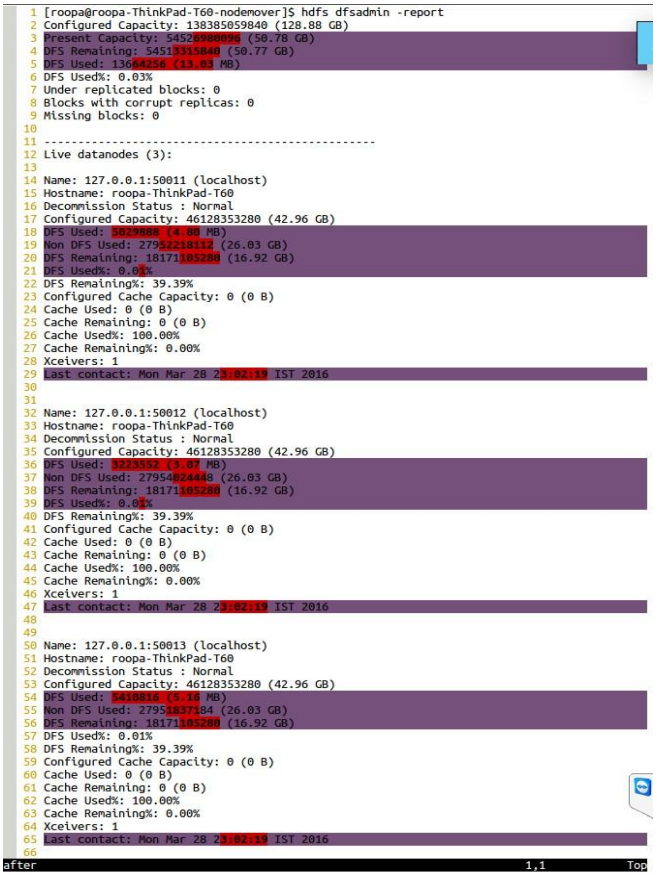

Fig. 3. After- Balanced loads on DataNodes

CONCLUSION

This paper focuses the load rebalancing algorithm that solves the load imbalance problem in distributed file system. It also tries to balance the loads of nodes and minimizes the migration cost by taking the information of the node locality. It solves the performance bottleneck problem by self modifying the loads of the nodes without depending on the central node to balance the load. It also improves the outcome of the system meanwhile improves the resource availability and resource utilization.

The algorithm plays vital role in the performance of the system. The algorithm attempts to balance the loads of the file chunks as much as feasible, since the data nodes spontaneously does the rebalancing task without the interference of the name node. It not only accomplishes the load balancing but it also makes sure that the system is reliable.

ACKNOWLEDGEMENT

I am very much thankful to **Dr. Balakrishna**, Principal, RajaRajeshwari college of Engineering, Bangalore for giving this prestigious opportunity of being a member of this institution.

I am very proud and thankful to **Dr. Usha S,** HOD, Department of Computer Science of Engineering for her valuable advice and ideas at various stages of this work.

I render my whole hearted thanks to my project guide **Mrs. Nandini G** Assistant Professor, Department of Computer Science and Engineering, who has been guiding staff to achieve the goal.

REFERENCES

[1]. Hsueh-Yi Chung Che-Wei Chanz Hung-Chang Hsiao Yu-Chang Chao, –*The Load Rebalancing Problem in Distributes File System*‖, 2013.

[2]. Prasanna Ganesan, Mayank Bawa and Hector Garcia-Molina, ― *Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems*‖, 2012. [3]. H.-C. Hsiao,H. Liao, s.-s Chen , –*Load Balance with Imperfect Information in Structured Peer-to-Peer Systems*‖, April. 2011.

[4]. Wenqiu Zeng, Ying Li, Jian Wu, –*Load rebalancing in Large-Scale Distributed File System*‖ , 2009.

[5]. Qingqing Zhong, ― *A Load-Balancing Approach for DHT-Based P2P Networks*‖, 2009.

[6]. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, ―*Dynamo: Amazon's Highly Available Key-Value Store*‖, Oct.2007.

[7]. Wang Yue, Cai Wangdong, Duan Qi, –*An Adaptive Dynamic Load Balancing Algorithm*‖, 2006.

[8]. Yingwu Zhu and Yiming Hu,‖ *Efficient Proximity-Aware Load Balancing for DHT-Based P2P System*‖ , April 2005.

[9]. D. KArger and M. Ruhl, ― *Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems*,‖ June 2004.

[10]. J.dean and S.Ghemawat,‖ *MapReduce: Simplified Data Processing on Large Clusters*‖, Proc.Sixth Symp.Operating System Design and Implementation(OSDI '04),pp.137-150,Dec 2004.

[11]. P.Jamuna and R. Anand Kumar –*Optimized Cloud Computing Technique to Simplify Load Balancing*‖ International Jornal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 11, November 2013.