AN EFFICIENT RING-BASED METADATA MANAGEMENT POLICY FOR LARGE-SCALE DISTRIBUTED SYSTEMS

A PROJECT THESIS

Submitted By

THIRUMAGAL DHIVYA S	2017506613
NITHYA S	2017506568
SANGAVI PRIYA G	2017506622

Under the supervision of

Mr. PUGHAZHENDI E

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

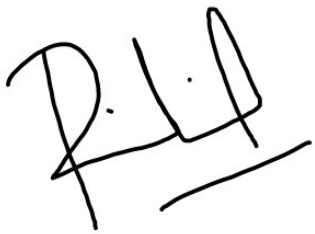
ANNA UNIVERSITY, CHENNAI-600 044

NOVEMBER-2020

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **"AN EFFICIENT RING BASED METADATA MANAGEMENT POLICY FOR LARGE-SCALE DISTRIBUTED SYSTEMS"** is the bonafide work of **THIRUMAGAL DHIVYA S** (2017506613), **NITHYA S** (2017506568) and **SANGAVI PRIYA G** (2017506622) who carried out the project under my supervision.



Signature

Dr. Dhananjay Kumar

HEAD OF THE DEPARTMENT

Professor

Department of

Information Technology

MIT Campus,

Anna University,

Chennai-600 044.

Signature

Mr. Pughazhendi E

SUPERVISOR

Teaching Fellow

Department of

Information Technology

MIT Campus,

Anna University,

Chennai-600 044.

ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement made us accomplish this project.

We express our thankfulness to our project supervisor M**r. Pughazhendi E**, Teaching Fellow, Department of Information Technology, MIT Campus, for providing invaluable support and assistance with encouragement which aided to complete this project.

We are thankful to the panel members **Dr. Radha Senthilkumar,** and D**r. P. Kola Sujatha,** Department of Information Technology, MIT Campus for their invaluable feedback in reviews.

Our sincere thanks to **Dr. Dhananjay Kumar**, Head of the Department of Information Technology, MIT Campus for catering all our needs giving out limitless support throughout the project phase.

We express our gratitude and sincere thanks to our respected Dean of MIT Campus, **Dr. T. Thyagarajan**, for providing excellent computing facilities throughout the project.

THIRUMAGAL DHIVYA S 2017506613

NITHYA S 2017506568

SANGAVI PRIYA G 2017506622

ABSTRACT

The number of files in the distributed system tends to increase constantly so, it is necessary to improve the efficiency and scalability of the metadata services. Distributed metadata management schemes use multiple metadata servers (MD's) for the storage of the metadata, providing a highly effective approach to alleviate the workload of a single server. It is quite difficult to maintain good metadata locality and load balancing among MD's at the same instance. To partitions metadata namespace trees and to serve large-scale distributed storage, a novel hashing scheme called Angle Cut is employed. At first, Angle Cut uses a locality preserving hashing function (LPH) to project the namespace tree into linear key space, i.e., Multiple chord-like rings. Then, to adjust the workload of metadata servers (MD's) dynamically, a history-based allocation strategy is designed. To reduce migration, Consistent Hashing technique is used here. Apart from that, a two-layer metadata cache mechanism is proposed, which is server-side cache and client-side cache for the purpose of providing the two stage access acceleration. Also in this proposed system the ring based metadata structure is maintained for the purpose of easily traversing the subtree and to easily perform the operation on the subtree without missing any subtrees in the metadata server. This ring based structure also help us to locate the metadata very easily in the metadata server. The partitioning of the subtree is implemented with the help of access frequency. The identification of which metadata is frequently accessed and which metadata have been mostly used and so on by using this access frequency. Also, this partitioning of the subtree protects the structure of namespace tree of the metadata node in the metadata server. The migration factor is also maintained in this system for the purpose of compatability and scalability of the metadata nodes. This migration factor help us to reduce the risk of newly adding the extra files or directories because it provide us the way for adding the new files or directories in the

metadata server. Also, this proposed system will give the most effective query mechanism. To maintain data consistency, a distributed metadata processing 2PC Protocol Based on Message Queue (2PC-MQ) is introduced. This experiment result shows that the approach preserves good metadata locality as well as maintains a high load balancing between the metadata servers.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LPH	- Locality Preserving Hashing
MDS	- Metadata Servers
DDP	- Dynamic Directory Partitioning
DROP	- Dynamic Ring Online Partitioning
2PC-MQ	- Protocol Based on Message Queue
HDFS	- Hadoop Distributed File System
XML	- Extensible Markup Language
AES	- Advanced Encryption Standard
CXFS	- Clustered Extended File System
EB	- Exabyte (Unit)

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Today the technology is growing faster and faster. As the modern file system is growing, there it becomes very essential to maintain that file system in the proper way. When less number of files are present, the searching of particular file very easily and also the processing speed of the system is very fast, but when more files are present, it will become very difficult to search the particular file and also the processing speed of the system becomes very slower. It shows that, generally it will maintain the less number of files in an easy manner but when it comes for maintaining large number of files, it becomes difficult. For that purpose some additional mechanisms is needed to maintain that large number of files effectively and very easy manner. This concept is not only for the files but also for the metadata and so on. Likewise there is a need to maintain the metadata for large scale distributed systems in a very effective manner. For this purpose, this system is going through some mechanisms which will be discussed below. The distributed system is nothing but the system with multiple computers and machines that combine their actions in order to appear as a single system for an end user. Here, this system proposes a hashing method called Angle Cut Hashing which uses the Locality Preserving Hashing for correctly balancing the workload between the metadata. The Angle Cut Hashing is completely based on the angle of the metadata. It helps us to maintain the good metadata locality. The Locality Preserving Hashing is based on the locality of the corresponding metadata in the system. For the purpose of improving the query efficiency, this system introduces the concept of two-layer metadata cache mechanism which is both client-side cache and server-side

cache. And based on the access frequency of the metadata it will partition the subtree of the metadata nodes in the metadata server.

1.2 RESEARCH CHALLENGES

The framing of effective metadata management for large file systems which operates correctly for performing all operations by the user with high processing speed becomes a great challenge for effective metadata management. And implementation of the system further has some difficulties such maintaining metadata locality, load balancing, data consistency and so on. Also maintaining the multiple server for multiple metadata management also becomes a great challenge.

1.3 OBJECTIVE

The main objective of the project is to reduce the risk of maintaining and accessing the metadata management for large distributed systems by implementing a ring based metadata management and to provide a new hashing scheme which is used to perform the partition of metadata subtree to serve large scale distributed systems. And the other objective is to maintain the data consistency and query efficiency.

1.4 SCOPE OF THE PROJECT

The proposed system of the project helps in improving the caching efficiency and the proper load balancing between the metadata since the metadata had been allocated based on the Locality Preserving Hashing function. So, the response time of the system will be improved drastically and the waiting time is completely diminished. The multiple ring design for metadata is implemented and the two level caching is also performed for two stage access simultaneously.

1.5 PROJECT CONTRIBUTION

This proposed system have been implemented by angle cut feature with access frequency based subtree partitioning. First, the n-ary tree is constructed and the angle is assigned. The angle is assigned in such a way that the node left to the current nodes will have the angle lesser than the current node. The ideal load factor is calculated and the node which has number of children nodes lesser than or equal to the ideal load factor will be cut and allocated to the metadata server. Also, access frequency based implementation is done to ensure that the hop count required to access the frequently accessed file gets minimized. By this access frequency based subtree partitioning, the locality gets improved.

1.6 ANGLE CUT HASHING

The Angle cut hashing uses a hashing called Locality Preserving Hashing for hashing the metadata namespace tree into a linear key space. This hashing projects the nodes of metadata into ring structure like chord. For performing this hashing, first it will calculate the angles of each node based on their corresponding coordinates in their namespace tree. This Angle cut hashing helps us to maintain the good metadata locality and load balancing of the entire system. Because the Locality Preserving Hashing method splits the allocation of metadata by two stages. At first stage it will project the metadata nodes into ring hash space which is the special way of this hashing function. At the second stage, with the help of constructed ring hash space in first stage, it is more easy to practice the simultaneous metadata allocation. This multiple ring like structure is mainly designed to avoid the hash collisions. Also this structure provides the reserved management choice. For example the allocation of metadata hash space to one or more metadata according to their ring number.

1.7 CACHE MANAGEMENT

For improving the query efficiency of metadata, this system is introducing the concept of two-layer metadata cache mechanism which is both client-side cache and server-side cache. The advantage of this mechanism is, it provides the two stage access. In this, the client- side cache will save the current metadata in its local storage. Actually for accessing the metadata node, this will require a permission from ancestor node that is their parent nodes for performing any operations in the metadata such as read or write operations. The Anglecut uses the mechanism to maintain the cache of this permissions and their paths. The major advantage of client cache is that the client is able to obtain the metadata directly from the available local caches which will reduces the response time. In the server-side cache, each metadata servers will replicates the available metadata nodes which will be present at near the namespace tree in its memory storage.

CHAPTER 2

LITERATURE SURVEY

2.1 METADATA DISTRIBUTION AND CONSISTENCY TECHNIQUES FOR LARGE-SCALE CLUSTER FILE SYSTEMS

In this system, they had proposed a metadata distribution policy which is DynamicDir-Grain. This policy searches the requirements of namespace locality and distribution of the load by dynamic partitioning of the namespace tree into adaptable size for equal balancing between the load. This system provides a set of solutions for basic level fundamental issues in metadata processing. There are two issues are observed in this system. They are how to equally spread the objects in the namespace tree across the metadata servers and the another issue is how to keep operations on the metadata server consistent even through in the presence of server failures. It determines the reliability and availability of the file system. This system has very poor state performance due to uneven metadata distribution. Although, this system is similar with the normal file creation and deletion. The main disadvantage of this system is if some metadata server crashes, some waiting messages may never arrive, resulting in some requests staying in waiting queue forever.

2.2 EFFICIENT R-TREE BASED INDEXING SCHEME FOR SERVER CENTRIC CLOUD STORAGE SYSTEMS

In this system they have designed a cloud storage system. It is very useful in various applications by supporting the intensive data. The main element and the important working element in this system is Indices. Because, it will maintain the positions for where the data to be stored that is for the function of data storage. This system derived a new indexing scheme called R-tree based indexing. Also this system has designed a new index mapping techniques to

maintain the global indices. It will improve a global index allocation and also it maintains efficient load balancing system. The Query processing algorithms present in this system will support the effective query tasks. This indexing scheme is very effective for storing and retrieval of data. Although, this newly introduced indexing scheme is efficient than traditional approach, this system also meets the disadvantage such as execution of a point location query in this tree based indexing scheme.

2.3 SUPPORTING SCALABLE AND ADAPTIVE METADATA MANAGEMENT IN ULTRA LARGE-SCALE FILE SYSTEMS

The metadata servers present in this system itself will organizes that metadata servers into several layered query hierarchy. Also it will exploits bloom filters which are grouply available for effectively routing the metadata requests to the desired metadata servers through query hierarchy. This metadata scheme can be executed at the network or memory speed. This system should not be bounded by the performance of slow disks. An effective workload balance method is also developed in this system for server reconfigurations in case of any failure occur in the system. The major advantage of this system is it requires a predefined maximum size which will not be easily predictable at all the time.

2.4 A HIGHLY RELIABLE METADATA SERVICE FOR LARGE-SCALE DISTRIBUTED FILE SYSTEMS

This system introduces a highly comfortable metadata service for the main purpose of addressing the issues that is present in large scale file systems. It is very different from the traditional methods. That is, the flexible metadata service present in this system will adopts new architecture for the purpose of fault tolerance. It uses the approach called holistic approach for improving availability of the file system. Also, a storage pool is designed for the usage of

transparenting the synchronization of metadata and replication of multiplies of data between the active servers. Based on this method, a new policy known as multiple actives multiple standbys is presented for performing the metadata service recovery in case of failures that occurs in the metadata servers. A intelligent fault tolerance mechanism is developed by this system for maintaining the continuity service of the metadata. Also, this system had implemented the flexible and scalable and reliable metadata service in a prototype file system. Also it had conducted the various tests to evaluate this functions in the system. The policy named MAMS will affect the operation performance of this system, but simultaneously it will enhances the metadata service reliability. The disadvantage of this system is it will decrease the efficiency of the time.

2.5 A NOVEL DYNAMIC METADATA MANAGEMENT SCHEME FOR LARGE SCALE DISTRIBUTED SYSTEMS

In this system, like in all the large distributed systems, the present metadata is usually managed by a metadata server cluster separately. Among all the servers, the splitting of the metadata is of very important for maintaining the efficient metadata server operations and a load distribution throughout the cluster. In this system, they had propose a dynamic directory partitioning which is known as DDP. Here, in this system, the management of the metadata, directory metadata and file metadata are managed in very different ways by various methods. The workload which is maintaining change dynamically can be balanced by adjusting the distribution of metadata servers. When they had compared this system with other metadata management schemes or policies, it had proved that this system has the features of scalability and adaptability and also the improvement in the performance. The disadvantage present in this system, is the security function of the system is very poor due to partitioning of the data. Also, it has the advantage that, the security mechanism available in this

system should be achieved only through by adding the another security layer with this security layer.

2.6 EFFICIENT AND SCALABLE METADATA MANAGEMENT IN EB-SCALE FILE SYSTEMS

In this system, they had proposed metadata server cluster architecture and they named it as Dynamic Ring Online Partitioning(DROP). In DROP, they have used the hashing technique of Locality-Preserving Hashing(LPH) for improving the namespace locality. In working with LPH function, first they will propose a hashing technique, and then secondly they present an dynamic load balancing algorithm which is known as HDLB for balancing the storage system. Next they gave a consistency mechanism with the help of Zookeeper for maintaining the consistency of the metadata present in the system. Finally, they will evaluate the DROP. The disadvantage of the system, the key distribution of this system is no longer uniform in DROP because of using the locality preserving hashing function. Also, it causes the load balancing in the system.

2.7 SCALING FILESYSTEM METADATA PERFORMANCE WITH STATELESS CACHING AND BULK INSERTION

In this system, they introduced a system called INDEXFS. This system is a middleware design. It adds support to existing file systems such as PVFS, LUSTRE, and HDFS for high performance operations on both metadata and small files. This designing system present in the model will preserve the server and disk locality for small directories. Because of using the architecture known as table based architecture. This architecture equally splits the namespace on a tree on the basis of the directory available. Also, they had proposed the two client based storm free caching techniques. Namely they are Bulk namespace insertion and stateless consistent metadata caching. The Bulk namespace insertion is used to create the intensive workloads. The stateless consistent

caching is used for hotspot mitigation. With the help of these two techniques they proposed the INDEXFS to 1128 metadata servers. This proposed model implements the most common POSIX file system operation except hardlink and xattr operations. But they have not implemented the some failuring mechanisms such as replaying write-ahead logs.

2.8 SCALE AND CONCURRENCY OF GiGa+: FILE SYSTEM DIRECTORIES WITH MILLIONS OF FILES

In this system, they examined the problem of scalable file system directories where all the applications requiring millions and billions of small files to be integrated into a single directory. They also introduced the POSIX which is the complaint scalable directory design, GiGa+. It equally distributes the directory entries over a large number of cluster of nodes. It normally uses two internal implementations. Namely asynchrony and eventual consistency. The asynchrony partitions an index among all servers without synchronization or serialization. And the eventual consistency tolerated the staleindex at the clients. GiGa+ has been influenced by the scalable and concurrency limitation of several distributed indices and their implementation.

2.9 CALVIN FS: CONSISTENT WAN REPLICATION AND SCALABLE METADATA MANAGEMENT FOR DISTRIBUTED FILE SYSTEMS

This system describes a different approach among all the traditional methods by the process of replicated design and scalable and flexible file systems which are investing a high throughput throughout the distributed database system for managing the database. It will result in improving the scalability of the metadata layer present in the file system. Because the metadata present in the file system will be able to partitioning across the large number of clusters independent servers, and the different operations performing in the on metadata files can be transformed into distributed transactions. In addition to

this, it also has the advantage of adding and editing their file system in order to support the various standard file system semantics. They also demonstrated their file system design approach to various servers that can scale upto more than billions of files. This system is optimized for operations only on a single file. Multiple file operations require distributed transactions. This is one of the major disadvantage present in this system. Also, it requires the support from many other operations in the system.

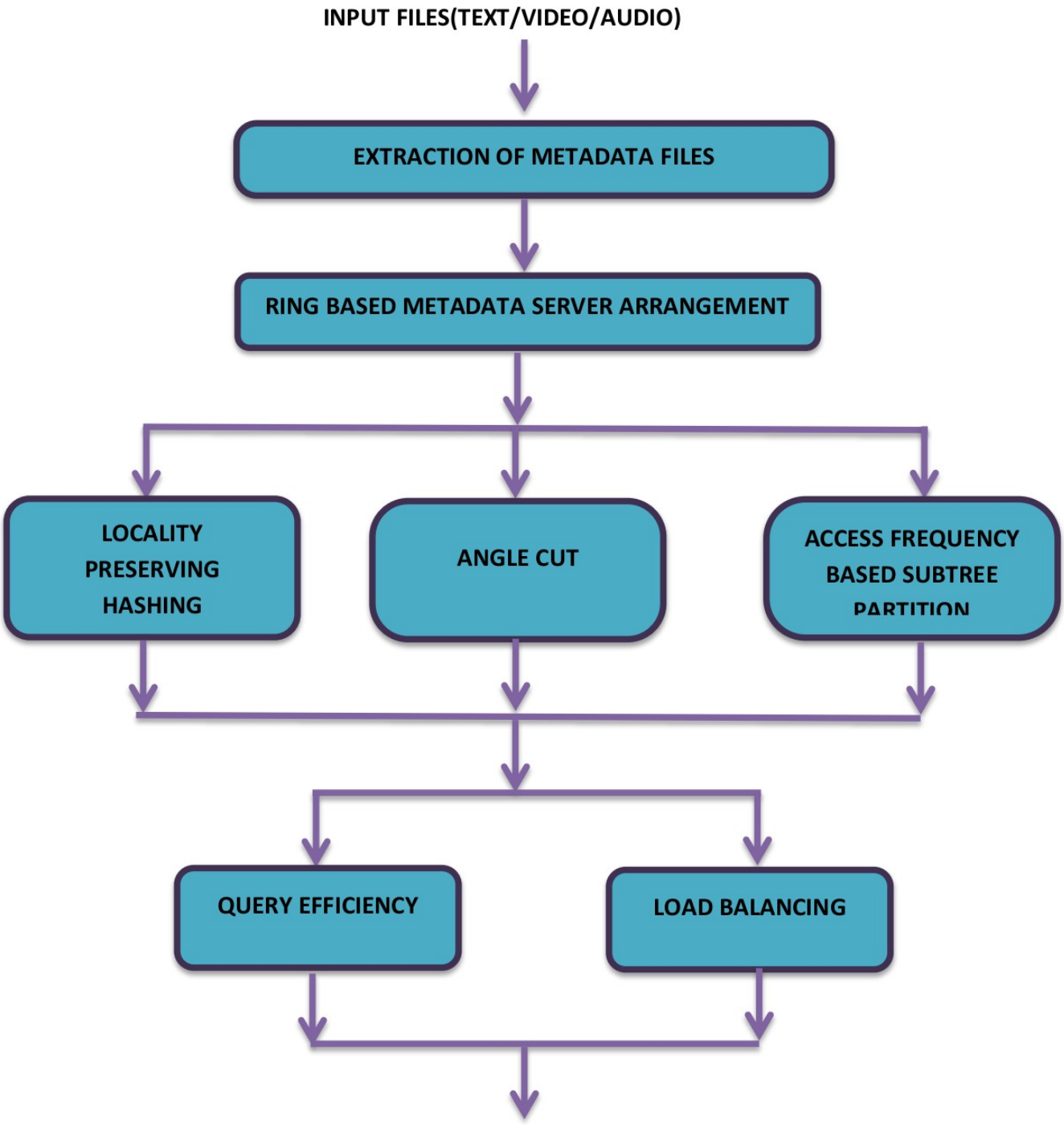
2.10 SUMMARY OF LITERATURE SURVEY

In file systems such as Lustre, CODA, CXFS the locality and MDS independence is good compared to hash based mapping but the workload is not equally distributed which results in 'hotspot' easily. In ceph, the load measurement method should be accurate and load information should be exchanged by the servers periodically. The workload balancing method is developed by the scheme of logically organizing the metadata servers into a multilayered query hierarchy and it filters the efficient route to metadata requests to the desired metadata servers through the hierarchy. The R tree based indexing is very efficient in terms of storage and retrieval of data. It maintains the global indices. In dynamic directory partitioning metadata management scheme, the file metadata and directory metadata are arranged in different ways, and changing workload can be balanced by adjusting the metadata distribution on metadata servers. In the summary of the existing models, all are failed to consider the problems of overloading, not taking into consideration of the multiple files and dynamic files, security and consistency. So, in order to overcome the issues in existing models, an efficient ring based metadata management is proposed here. The Locality Preserving Hashing technique will improve the namespace locality. And the dynamic load balancing algorithm will balance the storage.

CHAPTER 3

PROPOSED SYSTEM ARCHITECTURE AND DESIGN

3.1 ENHANCED METADATA MANAGEMENT ARCHITECTURE



For designing this proposed system, first this system will take a large number of input files in the form of text files, video files and audio files. In this input files(n = 1554 files) are taken for preprocessing. The next step of the designing the system is to extract the metadata files available in the taken input files. Next, it will construct the ring based metadata server based on the extracted metadata information available in the input files. Next, based on the metadata information and the ring based metadata structure, Locality Preserving Hashing is performed for the purpose of easily searching the nearest neighbour nodes. This hashing will provide us the easiest way of searching the metadata nodes. Next, this system, will perform the Angle Cut method. This Angle Cut is completely based on the corresponding angle of their corresponding metadata nodes. It is used for partitioning the subtree and to group the same nodes into same folder. Next, it will perform the access frequency based subtree partitioning. It is used for partitioning the subtree of the metadata nodes. This proposed system will provide us the query efficiency and the load balancing. The architecture of the proposed system is shown above.

3.2 EXTRACTION OF METADATA

In this system, the first step is extraction of metadata files. The metadata files is nothing but the file which contains information about the file such as file size and directory contents and so on. In this step it will extract that metadata files. These extracted metadata files are collected and stored in the different servers. Through this stored metadata this will perform all other operations in this system. Here this extraction of metadata files achieved with the help of dropbox cloud. It is a cloud storage provider and sometimes it is referred as an online backup service that is frequently used as a file-sharing service. The information everything in the dropbox is securely encrypted with 256-bit AES. Also for this extraction of metadata, the system will use the hadoop. A hadoop cluster is created with a multiple namenode and a multiple datanode. The

datanode is stored among the HDFS file system. The namenode contains the information about the metadata in the format of FSimage. The metadata information is extracted and retrieves the data XML file.

3.3 RING-BASED METADATA

Next, this system will maintain the ring based metadata server. It is a ring like structure. Therefore, it is essential to maintain the ring based structure for the efficiency of the system. In ring based structure every metadata node will be atleast of two metadata nodes. Generally, in the ring based structure will have the single pathway for traversing between the nodes. Similarly, the ring based metadata structure also will have a single pathway for travelling between the metadata nodes. With this speciality of ring based structure it is essential to maintain the ring based structure to traverse and perform operations on all the subtrees without missing any some trees because of some traffic which will be occurred in other structures.

3.4 LOCALITY PRESERVING HASHING

The Locality Preserving Hashing hashes the same files into same folder. So, the data of similar files will be available in the same folder. For this speciality, this hashing can be effectively used for data searching and data clustering. So, with this help the searching and assessing the nearest neighbour nodes becomes very easy. For that purpose,it uses a Locality Preserving Hashing technique in this system. Approximately, the searching algorithm generally use two hashing methods. That is either data independent method (For.Ex: Locality-Sensitive Hashing) or data dependent method (For.Ex:Locality- Preserving Hashing). Through this statement it is understood that Locality Preserving Hashing is the data dependent method. These LPH techniques are sometimes adopted in Distributed Hash Table which is a

distributed data structure to ensure that they have connected to the neighbour dataset.

3.5 ANGLE CUT

Angle cut is one of the hashing scheme which is used to partition the metadata namespace tree for easy purpose of serving the large distributed systems. Generally, in the hashing function the nodes will be mapped to the unique hash value and the unique hash function. Likewise, in this Angle cut Hashing scheme, it will map each metadata node based on the unique hash value of angle. This Angle cut hashing is completely based on the angle. In this system the angle of the metadata node will be calculated based on their metadata node coordinates. Each metadata node will have the unique angle. In this system, the metadata node is allocated to the nearest metadata based on the hash value of angle in the clockwise direction. It will help to maintain the balancing of load between the metadata nodes. This system, will maintain many metadata servers to reduce the risk of single server. In such a case, this system is needed to maintain the proper and effective balancing between these metadata servers. For this purpose, the Angle cut hashing is implemented in this system. This hashing will cut the each metadata node based on their particular angle. The subtree containing the metadata situated in the metadata server is divided using by this method of Anglecut. First, this Angle cut will perform the Locality Preserving Hashing function to projects the metadata namespace tree into a linear keyspace which is the special function of this hashing which is ring based structure. To perform this, first it will calculate angle of each metadata node and then ring based structure is constructed. This angle assign to the each metadata node will keep and maintain the locality of the namespace tree. Such that, this Angle cut helps us to maintain the good metadata locality and good load balancing between the metadata because of using the hashing technique of Locality Preserving Hashing . This hashing also helps us to assign the metadata

nodes in two ways. The first way is assigning the metadata nodes in ring hash space in unique LPH way which helps to maintain the metadata locality. In the second way, with the help of the ring hash space obtained in the first way, it is very suitable for us to simultaneous assigning of the metadata. Next, after performing the LPH hashing technique, a allocation policy based on the history for the main purpose of adjusting the workload between the metadata server is proposed. For the feature of maintaining the load balance, this system is need to allot the each metadata nodes uniformly to the metadata server. This uniform allotment of metadata will effectively maintain the balancing between the metadata nodes. Once the file details have changed from time to time, to obtain that changing information and the changed information it is needed to maintain the information of the metadata nodes in the metadata server. For this purpose this system proposed this allocation policy based on the history.

3.6 ACCESS FREQUENCY BASED SUBTREE PARTITION

In this system, the partitioning of the metadata subtree is implemented by using the hashing of Angle cut with the help of using the access frequency. Generally, the frequency is defined as the number of access of something. Here, this system will use the access frequency for the metadata to divide the subtree metadata nodes in the metadata servers. Apart from hash mapping, the partitioning of subtree protects the structure of namespace tree of metadata node in the metadata server. Generally, there are two types of subtree partitioning. Namely, static subtree partitioning and dynamic subtree partitioning. The static subtree partitioning is which separates the entire namespace tree into different subtrees. This partitioning is adopted by some systems such as Lustre and CXFS. Also this partitioning, have the advantages of maturity, simplicity and popularity , this partitioning is not able to balance the load correctly. The dynamic subtree partitioning is repartitioning the hotspots of subtree using vigorous subtree partitioning. This type of partitioning is adopted by some

systems such as CepH. This dynamic subtree partitioning overcomes the drawback of the static subtree partitioning. Also, this dynamic subtree partitioning has increase the complexity of the system. The key concept behind this dynamic subtree partitioning is the workload of metadata can be evenly distributed across the each metadata servers. This distribution will help us to maintain the balancing between the loads of metadata server. This partitioning is evenly distributed approach. So, this system will use the dynamic subtree partitioning for mainly balancing the load between the metadata server. Through this it will achieve the balancing load in the metadata server management which is the key feature of the proposed system. This subtree partitioning is one of the main important part for implementation of this proposed system in an effective manner.

CHAPTER 4

METADATA MANAGEMENT SYSTEM

4.1 METADATA LOCALITY

This proposed system consists of many metadata files in the form of subtrees. Generally, the locality is defined as the place where the locating of something or somebody is very easy. Here, in this system the locality refers to the place of metadata files. This system provides a good metadata locality with the help of some hashing functions. It is needed for the system easily to locate the metadata nodes.

4.2 LOAD BALANCING DEGREE

This system should maintain the proper load balance between the metadata nodes to avoid any unneeded crashes or some confusions or some break of the metadata nodes. Commonly, the proper balance for everything is needed to work everything properly. To achieve this correct load balance between the metadata nodes, the correct balance should be done by calculating the degree for every metadata nodes. In other words it may be tell as angle of the metadata node. The proper load balancing is needed for the system to work properly. By using this degree of the metadata node the Angle cut hashing is performed.

4.3 MIGRATION FACTOR

The migration factor is maintained for this system for compatability and scalability purpose of metadata nodes. In some system when some extra files or directories to the existing system is added, it may cause some problems such as over crashing between the files, collisions, slow processing of the system, not enough space to adopt some more files and so on. To avoid this existing problem,the migration factor is maintained in this system, that is if any

metadata nodes or more metadata nodes is need to add in existing system this migration factor will easily allot the metadata nodes to this existing system. By adding this migration factor to this proposed system, the system will not experience any crashes between the metadata nodes. Also this will not affect the processing speed of the system at any cost. This is one of the feature of this proposed system.

4.4 HASHING SCHEME

Angle cut is a hashing scheme proposed in this system. It is mainly used to divide the namespace tree. The namespace tree is defined as the tree which consists of many subtrees. To perform this angle cut hashing, first the Locality Preserving Hashing is proposed and then a allocation policy is proposed based on the history of the metadata nodes. By proposing this two designs, the system will maintain the good metadata locality and good load balancing between the metadata nodes. The good locality refers to the metadata nodes that are closer to the another metadata nodes to the same metadata server. This good locality will reduce the late response of the metadata nodes and it will increase the query efficiency of the metadata nodes. And the good balancing refers to the equal balancing of the metadata nodes on every metadata server. In this Angle cut hashing the metadata node will have the unique hash value of angle. This angle is calculated based on the coordinates of the each and every metadata nodes. This coordinates are the coordinates of the metadata nodes in the namespace tree. This angle cut will displays this namespace tree into ring like structure. In calculating angle for metadata node this will perform in the level order traversal. In the first layer of the tree that is the root node will equally share the degree of 360. And this 360 degree is shared by dividing this degree by 2,4,8 and so on based on their coordinates for calculating the angle on further layers and later it will assign the angle to the metadata nodes entirely based on their coordinates of the metadata nodes. This helps the Angle cut to maintain the

locality between the metadata nodes. So, this tree will be well balanced. Because of the ring like structure the permissions for read or write operations on the namespace tree can be very easy. In the next step, a allocation policy based on their history is proposed for the metadata nodes. In each and every update of the system, if any information about the metadata nodes is missed, by using this allocation policy the data can be recollected very easily. This Angle cut is mainly based on the angle of the metadata nodes. The LPH techniques are sometimes adopted in the Distributed Hash table also. This LPH are most commonly used for searching the nearest or neighbour nodes. By using this LPH, the searching of metadata nodes is very easy and performance of some operations is also very easy on this metadata nodes. The proposed allocation policy will adjust the work load.

4.5 PROPOSED ALGORITHM

INPUT: Namespace Tree

OUTPUT: The angle keyspace

1.for i €[0,Number of metadata nodes] do

2. set i^{th} metadata node to valid;

3. if i>0, then get the coordinate of i^{th} metadata node;

4. Current load of i^{th} metadata node=Number of childrens of i^{th} metadata node;

5.Θ=360/Current load 0^{th} metadata node;

6.for i € [1,Number of metadata nodes] do

i^{th}metadatanode.Θ=

$$\sum_{j=1}^{depth\ of\ ith\ metadata\ node}[j\ th\ component\ of\ i\ th\ metadata \frac{node\Theta}{(maximum\ capacity\ of\ i\ th\ metadata\ node.ancestor-1)2j-1}]$$

7. if i^{th} metadata node.Θ conflicts with existing rings then

8. i^{th} metadata node.i^{th} node information.i^{th} node Θ=radius.order of metadata node

9.add i^{th} metadata node to the rings

This algorithm is used to construct the ring like structure. That is it is used to convert tree like structure into ring like structure. It is constructed with the help of their corresponding angle of the each and every metadata nodes in the metadata server system. In this algorithm, first this will set if metadata node is present from 0 to number of metadata nodes available, that metadata node is set to valid. And then it will get the coordinates of the metadata node to calculate the angle of that metadata node. Then, using that angle it will calculate the load of the metadata node for maintaining the load balancing. Then it will perform some mathematical operations to check whether the ring is conflicting with existing node, if not it will add that metadata node to the ring like structure, otherwise it will calculate some other additional mathematical notations to place this metadata nodes without conflicting the other rings. It will continue this steps until this allocate all the metadata nodes in the ring like structure.

4.6 CONSISTENT HASHING

The consistent hashing is defined as the distributed hashing scheme which will operate independently of the number of servers available in the system in the distributed hash values by assigning them in a ring like structure that is like a circle or hash ring. Mostly, in this consistent hashing performance is operated in the clockwise direction. By using this hashing, the location of metadata server in all systems is very easy. This wonderful consistent hashing will solve the problem of rehashing. It is a strategy for dividing the data between the multiple metadata servers. This way of dividing the data will

enable the uniform or even distribution of metadata nodes which will relieves the hotspots. This consistent hashing allow us to perform very easily to add or remove the metadata nodes in the metadata servers. When the system perform this add or remove operations, it is needed to move some metadata nodes between the metadata servers to achieve the even balance between the metadata nodes. This help us to maintain the load balance.

4.7 SUBTREE PARTITIONING

The subtree partitioning is very different with available mappings based on the hash value. In order to divide the metadata nodes and to allocate the metadata nodes in the metadata server, the most ultimate step is to get the scatter the access frequency of the metadata nodes. And based on this access frequency the partitioning of the subtree of the metadata nodes in the metadata server is performed. The subtree partitioning is very important in the large amounts of data that is metadata nodes in the metadata server for controlling its allocation of metadata nodes in the appropriate metadata servers. This subtree partitioning will help us to achieve the better performance of the entire system. As any operations between the metadata nodes, the access frequencies of the metadata nodes will keep changing by time to time. This will cause the metadata server to become unbalanced in the system. When this change happens, the updation of access frequency is needed to balance the metadata server. In this proposed system, it will maintain the allocation policy based on history. By using this allocation policy, the access frequency of the metadata nodes in the metadata servers can be easily updated over the changes by time to time in the future. This subtree partition enables us to traverse between all the metadata nodes and helps us to perform all the operations on each and every metadata nodes in the metadata server.

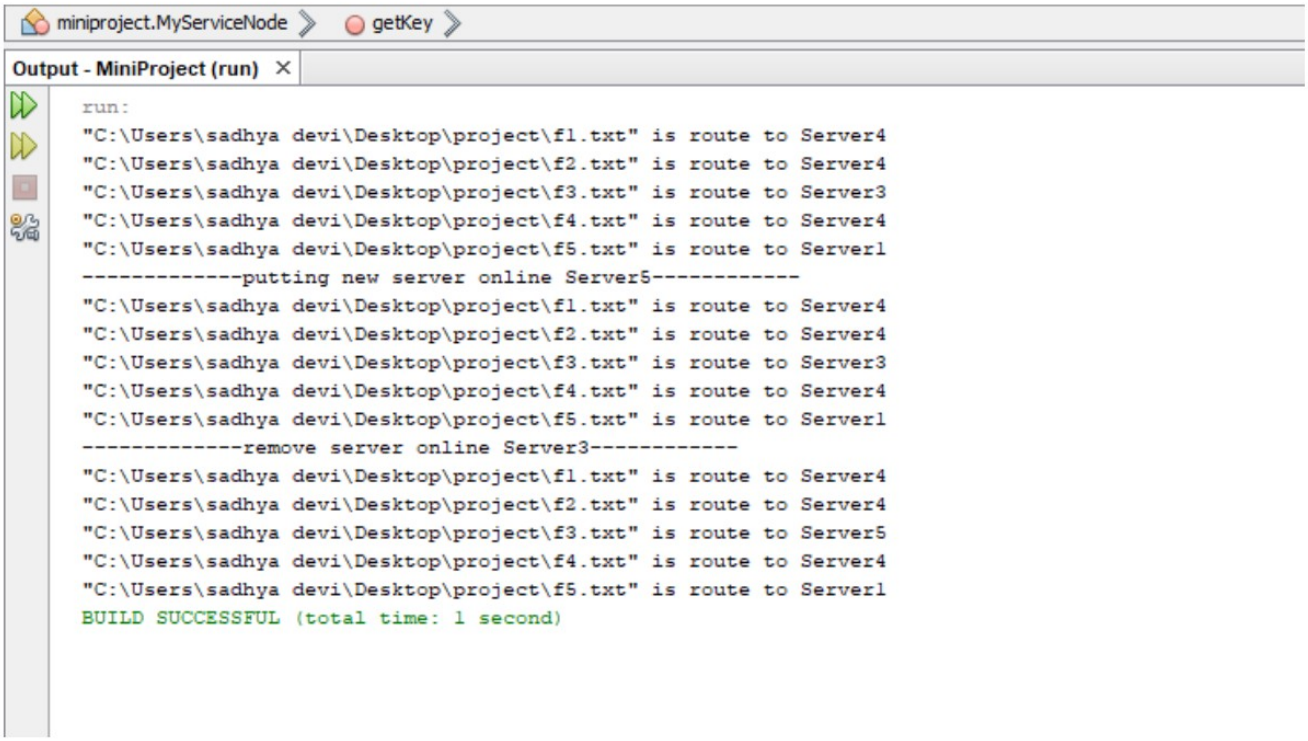
CHAPTER 5

RESULTS AND DISCUSSION

5.1 LOCALITY PRESERVING HASHING

This is a distributed hashing scheme that operates independently of the number of servers or objects in a distributed hash table by assigning them a position on an abstract circle, or hash ring. This allows servers and objects to scale without affecting the overall system. In general, k/N keys need to be remapped when k is the number of keys and N is the number of servers (more specifically the maximum of the initial and final number of servers). It enables elastic scaling of cluster of database or cache servers. It is a hashing technique which hashes the same files into the same folder. Actually ,number of input files is taken as a input for this system. Through, this Locality Preserving Hashing, the input files which are the same files are put into the same folder. This hashing is mainly used for the data searching. By using this hashing such that, the searching of particular metadata nodes in this distributed systems is very easy. So, with this help the searching and assessing the nearest neighbour nodes is very easy. This is one of the speciality of this Locality Preserving Hashing technique.

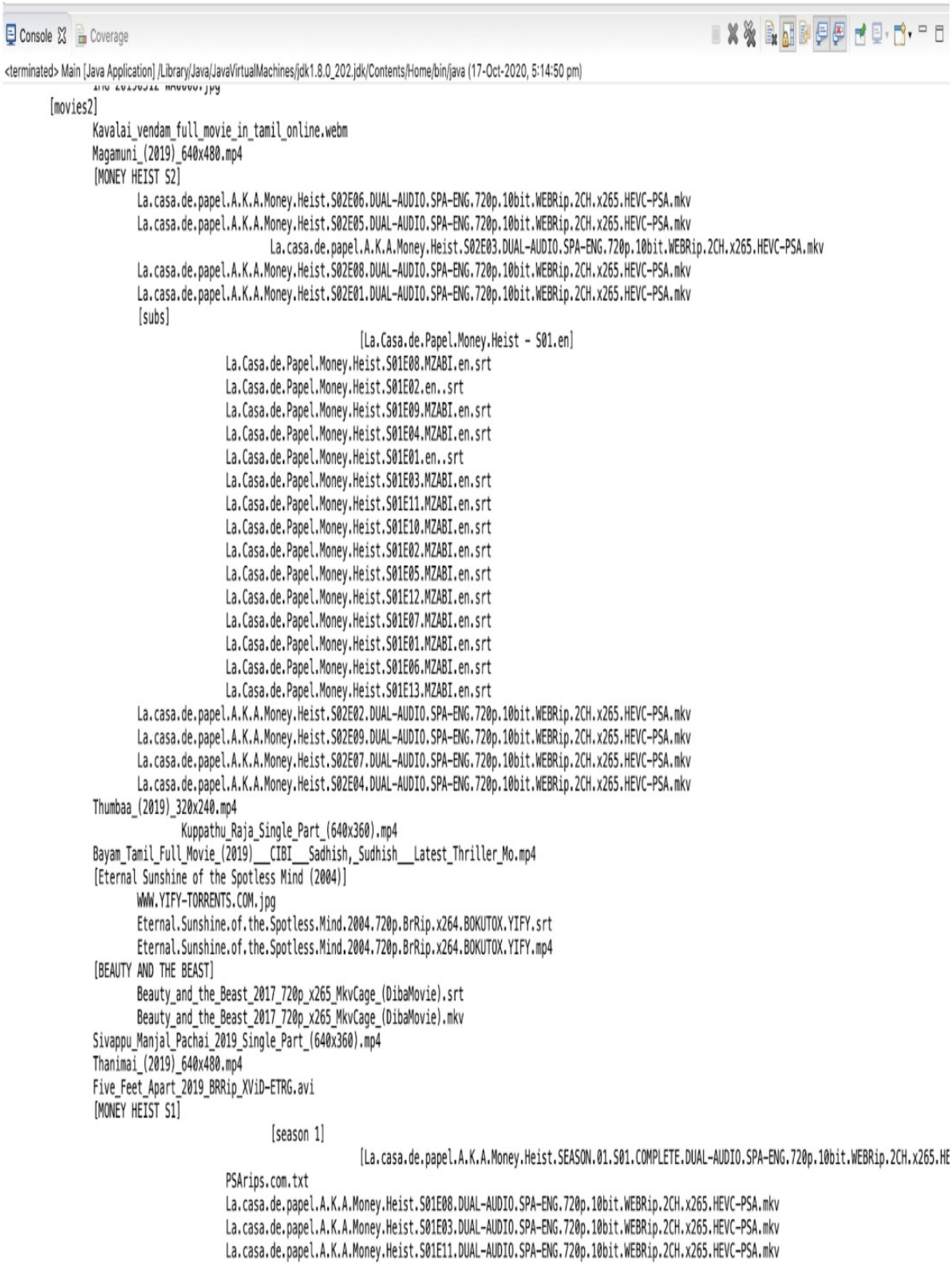
LOCALITY PRESERVING HASHING



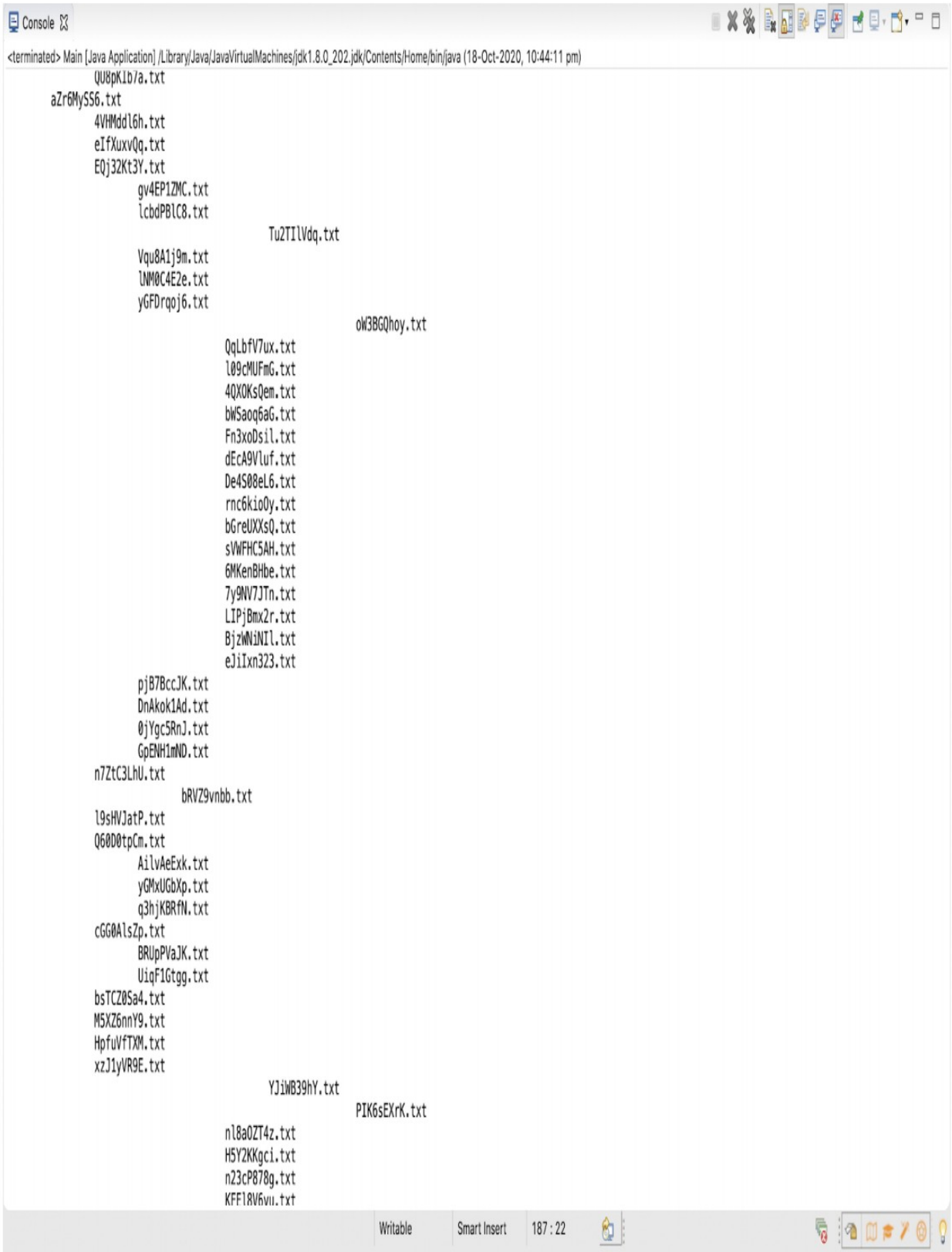
5.2 CONSTRUCTION OF NAMESPACE TREE

For constructing the namespace tree, this system,had taken the input files of 1554 files in the form of directory structure. This input is actually a local file system path. This input files will be available in the form of text files and audio files and video files. This system will extract the metadata information available in this input files.This input file system is iterated and metadata information present in this input files are extracted and this extracted information is written into separate folder. By, fully traversing this separated folder, this system will get the metadata information available in this folder. By using this information, it will construct the metadata text files into separate folder in a separate location.Using this, separated folder which contains the metadata text files, N-ary tree is constructed which is nothing but the namespace tree. By extracting the metadata information, the constructed metadata namespace tree structure is shown below.

INPUT FILE SYSTEM



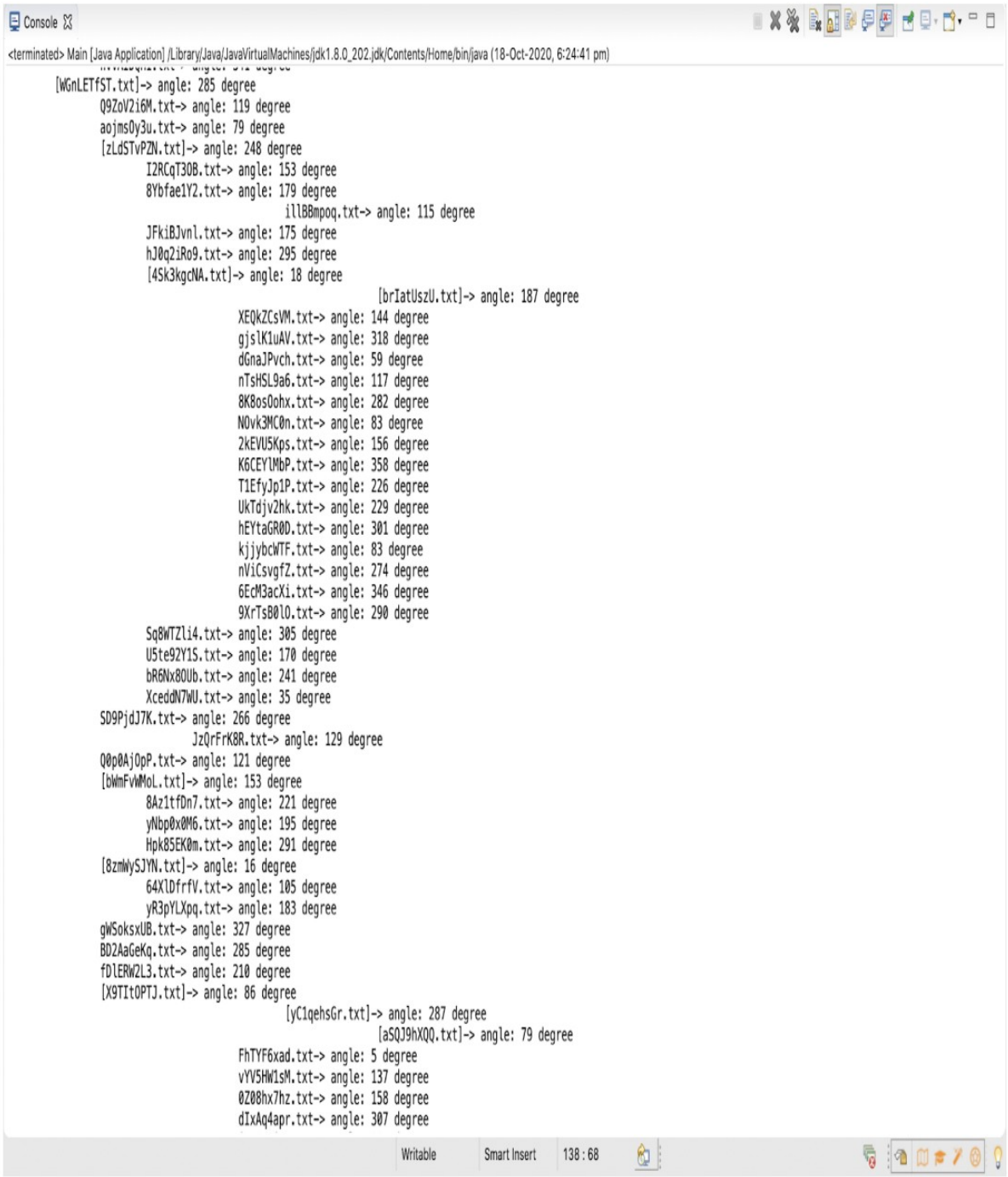
METADATA NAMESPACE TREE



5.3 ANGLE CUT IMPLEMENTATION

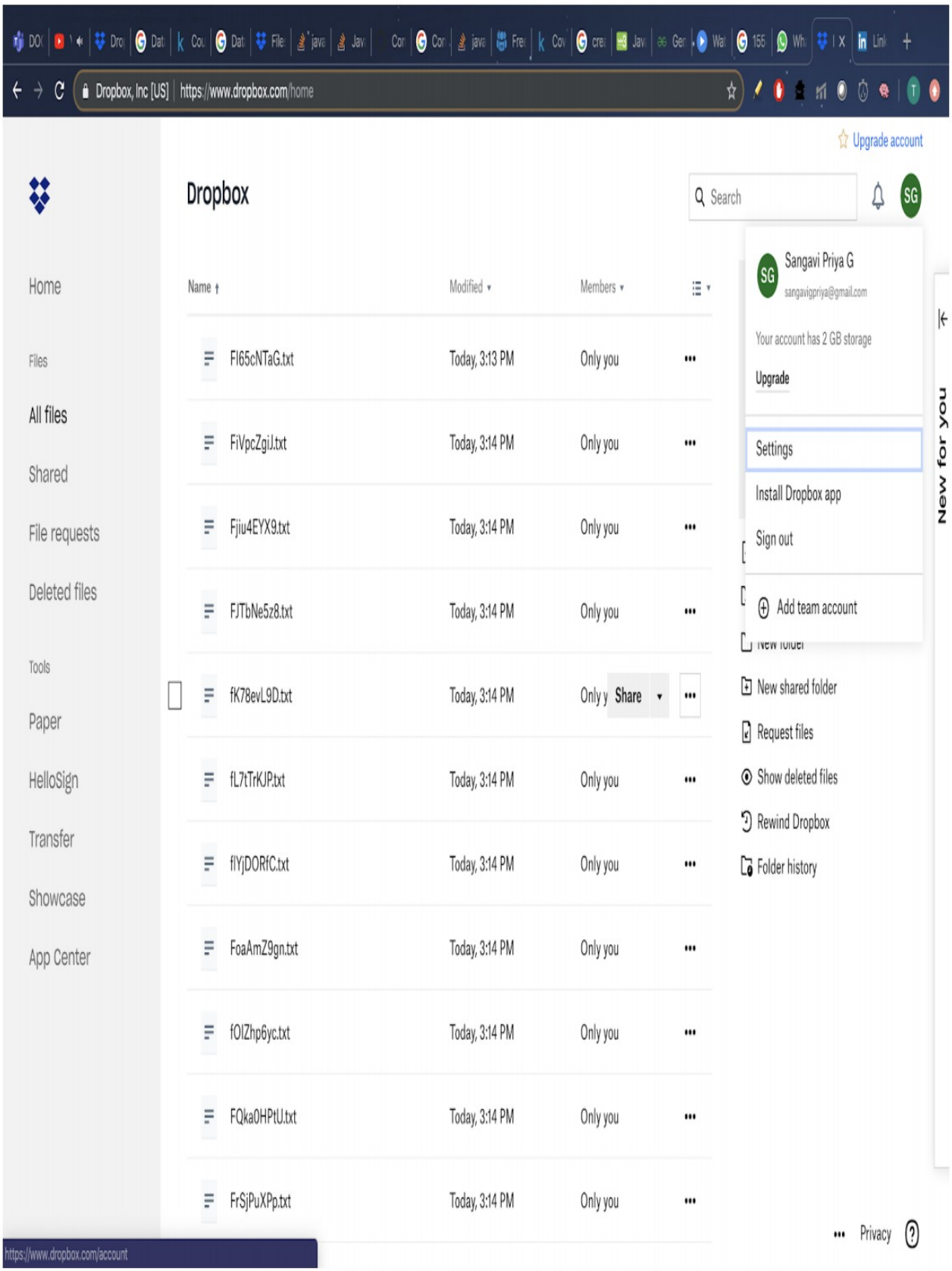
This angle cut implementation is mainly used to divide the namespace tree. For performing this implementation, first it will assign the angle to the corresponding metadata nodes. And after that, based on their angle, it will divide the subtree. The namespace tree will consist of many subtrees and root nodes. It will divide the subtrees based on their corresponding angle of that metadata node. By using this Angle Cut, the metadata node is checked whether it is valid or invalid. In this Angle cut hashing it will have the unique hash value of angle of the metadata nodes. This angle is calculated based on the coordinates of the each and every metadata nodes. This coordinates are the coordinates of the metadata nodes in the namespace tree. This angle cut will displays this namespace tree into ring like structure. In calculating angle for metadata node it will perform in the level order traversal. The nodes in the first layer of the namespace tree will share 360 degree angle uniformly. The second and third layer of the namespace tree will shares the difference of the angle computed in the parent and their neighbouring nodes as theta/2 and theta/4. Also, this Angle Cut implementation will compute the ideal load factor. By, using this ideal load factor computed based on the number of metadata servers the angle assigned subtree partitioning is performed.

ANGLE ASSIGNMENT

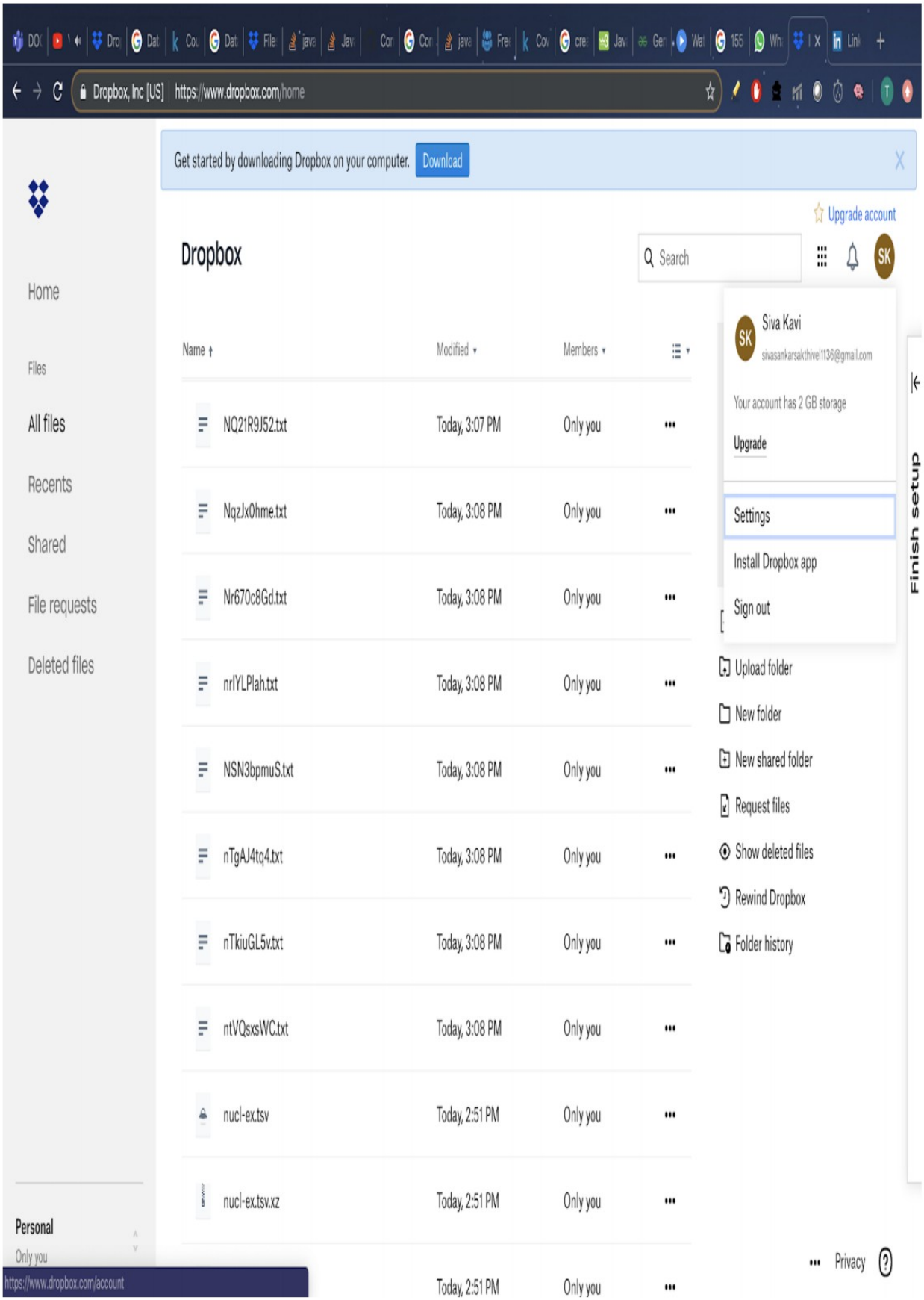


ANGLE CUT IMPLEMENTATION

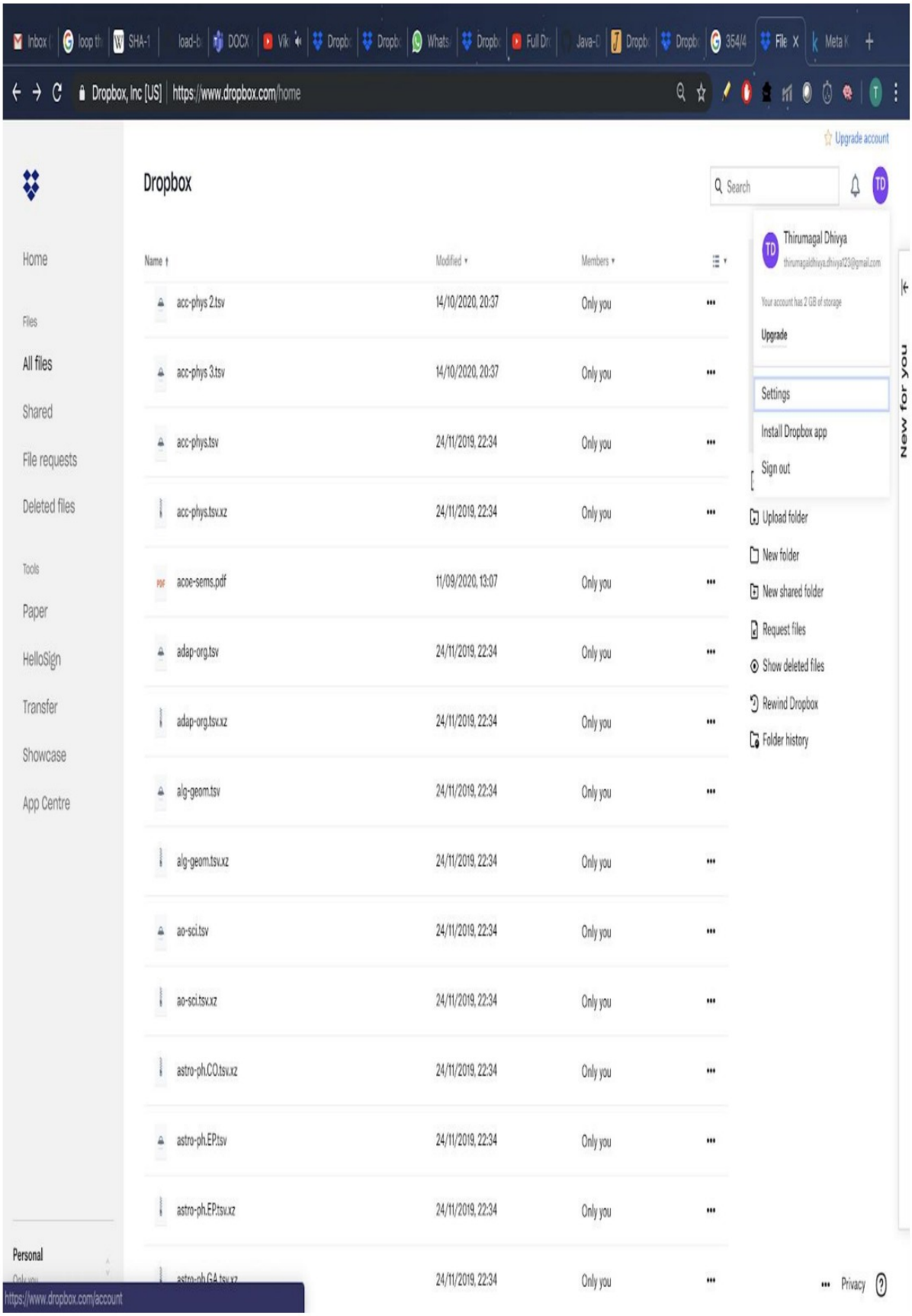
ANGLE CUT IMPLEMENTATION



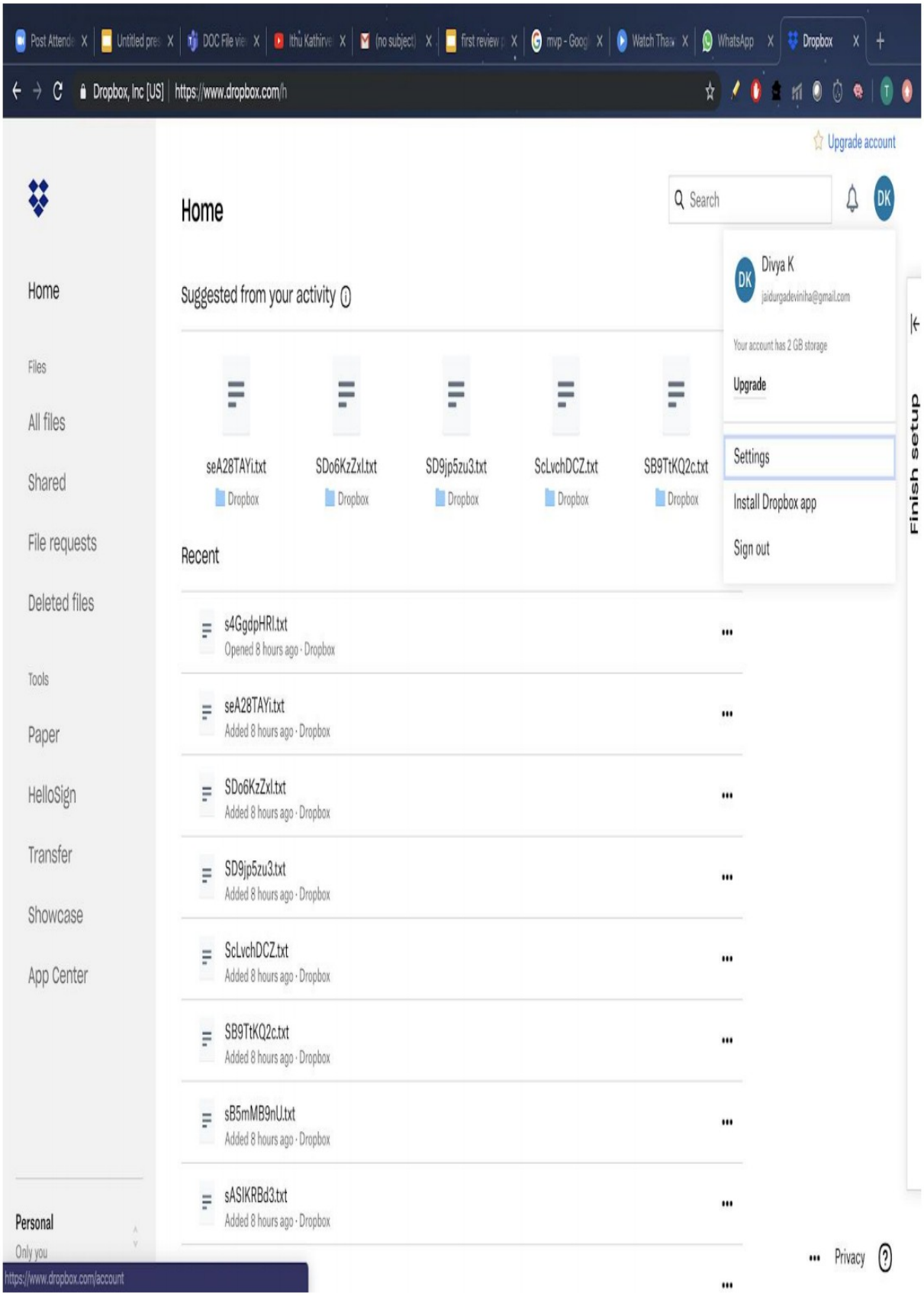
ANGLE CUT IMPLEMENTATION



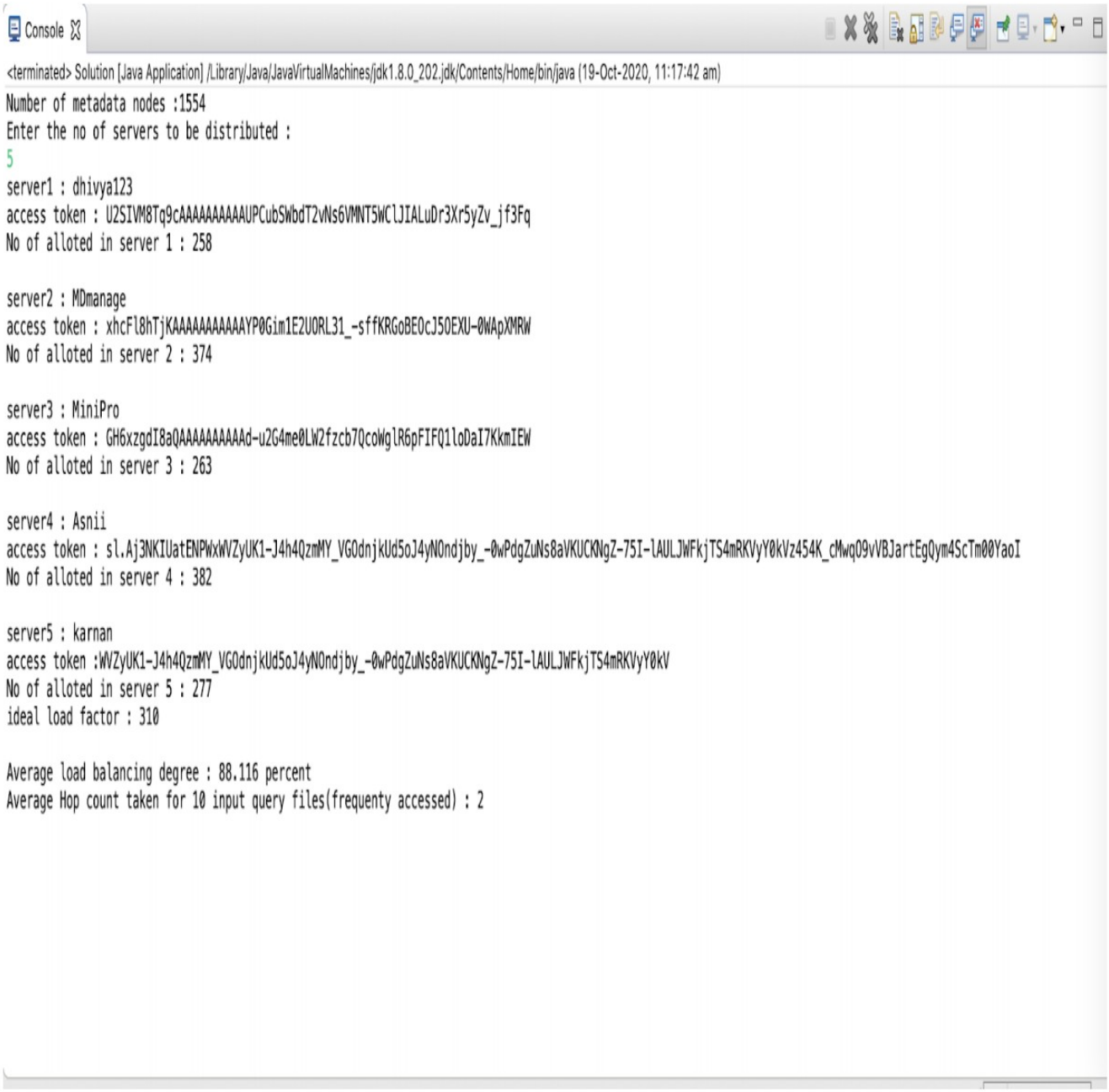
ANGLE CUT IMPLEMENTATION



ANGLE CUT IMPLEMENTATION



5.4 ACCESS FREQUENCY BASED SUBTREE PARTITIONING



Using an API, the frequency of the file is calculated. When the user requests for the file in this JSP page, the frequency of the file is incremented. Based on the frequency of the metadata file, the subtree partitioning is done. More frequently accessed file is allocated to the metadata server along with the parent hierarchy. So, hop count is lesser for the frequently accessed files and therefore the locality of the metadata is improved.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This proposed system have taken a large number of input files and constructed the namespace tree by fully traversing that input files and also by obtaining the metadata information present in that input files. The Angle Cut Hashing technique is designed which is completely based on the angle of the metadata files in the distributed systems with the help of Locality Preserving Hashing. The Locality Preserving Hashing is independent of the number of servers and it is mainly used for the data searching. And, the consistent hashing is independent of the number of servers and this hashing will assign the hash values in the ring like structure. This hashing will solve the problem of rehashing. This angle cut method will have a unique hash value of the angle. This angle cut method effectively used for dividing the namespace tree that is used for dividing the metadata files based on their specification. The migration factor is designed for easy adaptation of new metadata files such that the collapsing of files and no place for files will not happen. Also, the subtree partitioning is improved with the help of access frequency. This proposed system will give effective load balancing and the query efficiency.

FUTURE WORK

The proposed system is designed with construction of namespace tree, angle cut, locality preserving hashing, migration factor and access frequency based subtree partitioning but this system also can extend with designing the two level layer cache mechanism which is both client side cache and the server side cache. This mechanism will further improve the metadata query efficiency. Because, in the client side cache the client saves the currently available

metadata in its local storage and in the server side cache the metadata servers multiplies the metadata nodes. In a large distributed systems, the communications between every metadata files should be proper, if any one of the metadata files crashes, the entire system will be blocked such that the system will not work properly. It is not good for high applications. This problem in this system can be solved by Two Phase Commit Message Queue (2PC-MQ). This will improve the performance of the applications and also the consistency of the data. This dirty read problem will be enhanced or avoided in the future.

REFERENCES

1. Y.Gao,X.Gao,G.Chen "DeepHash:An End-to-End Learning Approach for Metadata Management in Distributed File Systems" Proceedings of the 48th International Conference on Parallel Processing, 2019.

2. Z. Chen et al., "A Multi-database Hybrid Storage Method for Big Data of Power Dispatching and Control," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, pp. 502-507, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00127, 2019.

3. "IEEE Draft Standard for Learning Object Metadata," in IEEE P1484.12.1/D7, January 2020 , vol., no., pp.1-48, 30 April 2020.

4. Supriadi, A. Wajiansyah, H. purwadi, R. Malani, A. Yunianta and A. Pratomo, "Secured Data Transmission using Metadata Logger Manipulation Approach," 2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT), Makassar, Indonesia, 2018, pp. 340-344, doi: 10.1109/EIConCIT.2018.8878601.

5. R. Patgiri, S. Nayak and S. K. Borgohain, "Impact of Metadata Server on a Large Scale File System," 2018 IEEE Colombian Conference on Communications and Computing (COLCOM), Medellin, 2018, pp. 1-6, doi: 10.1109/ColComCon.2018.8466729.

6. J. Xiong, Y.Hu, G.Li, R.Tang, and Z.Fan, "Metadata Distribution and consistency techniques for large-scale cluster file systems," IEEE Trans. Parallel Distrib. Syst., vol. 22, no.5, pp.803-816, May 2011.

7. Y. Zhang, "Application of Metadata Management System in Banking Industry," Financial Times, vol. 20, no. 12, pp. 49–51, 2012.

8. X. Zhang, "Design and implementation of metadata management system based on CWM in e-government," M.S. thesis, Shanghai Jiaotong University, Shanghai, China, 2012.

9. A. Thomson and D. J. Abadi, "CalvinFS: Consistent wan replication and scalable metadata management for distributed file systems," in Proc. USENIX Conf. File Storage Technol., 2015, pp. 1-14.

10. S. Patil and G. A. Gibson, "Scale and concurrency of GIGA+: File system directories with billions of files," in Proc. USENIX Conf. File Storage Technol., 2011, vol. 11, pp. 177-190.

11. Y.Fu, N. Xiao, and E. Zhou, " A novel dynamic metadata management scheme for large distributed systems," in Proc. IEEE Int. Conf. High Perform. Comput. Commun., 2008, pp.177-190.

12. K. Ren, Q. Zheng, S. Patil, and G. Gibson, "IndexFS: Scaling File system metadata performance with stateless caching and bulk insertion," in Proc. IEEE Int. Conf. High Perform. Comput. Netw. Storage Anal., 2014, pp. 237-248.

13. Q. Xu, R.V. Arumugam, K.L. Yong, and S. Mahadevan, “Efficient and Scalable metadata management in EB-scale file systems,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 11, pp. 2840-2850 Nov. 2014.

14. Y. Hong, Q. Tang, X. Gao, B. Yao, G. Chen, and S. Tang, “Efficient R-Tree based indexing scheme for server-centric cloud storage system,” IEEE Trans. Knowl. Data Eng., vol. 28, no. 6, pp. 1503-1517, Jun. 2016.

15. G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, Distributed Systems: Concepts and Design (5th Edition). Boston, MA, USA: Addison Wesley, 2011.

16. Hsiao, H. C., Chang, C. W., “A Symmetric Load Balancing Algorithm with Performance Guarantees for Distributed Hash Tables,” IEEE Transactions on Computers, 62(4), 2013, pp. 662-675.

17. D. R. Karger and M. Ruhl, “Simple efficient and load balancing algorithms for peer-to-peer systems,” in SPAA, 2004, pp. 36-43.

18. J. Xiong, Y. Hu, G. Li, R. Tang, and Z. Fan, ‘‘Metadata Distribution and Consistency Techniques for Large-Scale Cluster File Systems,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 803-816, May 2011.

19. Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, ‘‘Supporting Scalable and Adaptive Metadata Management in UltralargeScale File Systems,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 4, pp. 580-593, Apr. 2011.

20. H. Wei, L. Shi, Y. Haoyun and S. Chuanjie, "Metadata-based management for educational resources," 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, Xi'an, 2013, pp. 253-255, doi: 10.1109/ICIII.2013.6702922.

21. A. R. Bharambe, M. Agrawal, and S. Seshan, “Mercury: supporting scalable multi-attribute range queries,” in SIGCOMM, 2004, pp.353-366.

22. D. Qin, L. Huang and Y. Wang, "Construction of Railway Metadata Management System Based on Metadata Content Model and CWM Exchange Mechanism," 2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS), Toronto, ON, 2018, pp. 1-6, doi: 10.1109/LISS.2018.8593216.

23. M. Cha, S. Lee, D. Kim, H. Kim and Y. Kim, "High Performance Metadata Management Engine for Large-Scale Distributed File Systems," 2015 9th International Conference on Future Generation Communication and Networking (FGCN), Jeju, 2015, pp. 29-32, doi: 10.1109/FGCN.2015.12.

24. A. Thomson, T. Diamond, S. Chun Weng, K. Ren, P. Shao, and D. J. Abadi. Calvin: Fast distributed transactions for partitioned database systems. In SIGMOD, 2012.

25. J. Zhou, Y. Chen, W. Wang, S. He and D. Meng, "A Highly Reliable Metadata Service for Large-Scale Distributed File Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 2, pp. 374-392, 1 Feb. 2020, doi: 10.1109/TPDS.2019.2937492.

26. M. Balakrishnan, D. Malkhi, T. Wobber, M. Wu, V. Prabhakaran, M. Wei, J. D. Davis, S. Rao, T. Zou, and A. Zuck. Tango: Distributed data structures over a shared log. In SOSP, 2013.

27. K. Aikoh, Y. Isoda and K. Sugimoto, "Data Profiling Method for Metadata Management," 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), sydney, Australia, 2020, pp. 779-780, doi: 10.1109/DSAA49011.2020.00113.

28. J. Baker, C. Bond, J. C. Corbett, et al. Megastore: Providing scalable, highly available storage for interactive services. In CIDR, volume 11, pages 223-234, 2011.

29. C. Guo, H. Wu, K. Tan, et al. Dcell: a scalable and fault-tolerant network structure for data centres. ACM SIGCOMM Computer Communication Review, 38:75-86, 2008.

30. Kai Ren, Garth Gibson, TableFS: Enhancing metadata efficiency in the local file system. USENIX annual technical conference (ATC), 2013.

31. Saranya, D., and L. Sankara Maheswari. "Load balancing algorithms in cloud computing: a review." International Journal of Advanced Research in Computer Science and Software Engineering Research Paper (2015).

32. J. Liu, R. Wang, X. Gao, X. Yang, and G. Chen, "AngleCut: A ring-based hashing scheme for distributed metadata management," in Proc. Int. Conf. Database Syst. Adv. Appl.,2010, pp.19-33.

33. Q. Xu, R. V. Arumugam, K. L. Yang, and S. Mahadevan, "Drop: Facilitating distributed metadata management in eb-scale storage systems," in MSST, 2013, pp. 1-10.

34. Palak Shrivastava1 Sudheer Kumar Arya2 , Dr. Priyanka Tripathi "Various Issues & Challenges of Load Balancing Over Cloud: A Survey" International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume 5 Issues 8 Aug 2016.