ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st& 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Framing an Inviolable and De-Duplication Storage System Over Cloud Computing

M.Kaliraja, Rajalavanya Chakaravarthy

Post Graduate, Dept of Information Technology, Velammal college of Engineering and Technology, Madurai, Tamil Nadu, India.

Assistant Professor, Dept of Information Technology, Velammal college of Engineering and Technology, Madurai, Tamil Nadu, India.

Abstract— De-duplication is an approach to avoid storing data blocks with identical content. It has effectively reduced the disk space for storing large content files. However, it remains challenging to deploy de-duplication in a real system, such as a cloud platform setup. We proposed VDFS, a live de-duplication file system that enables de-duplication storage in VM. VM is an open-source cloud environment that is deployed under low-cost commodity hardware settings with limited memory footprints. We will deploy our VDFS prototype as a storage layer in a cloud platform based on Open Stack, and conduct extensive experiments. When compared to an ordinary file system without de-duplication, we show that VDFS can save at least 40% of space for storage in cloud environment. We also achieve reasonable performance in importing and retrieving files. In existing process collision may occur due to dual hashing algorithm and time latency while use of large data. Our work justifies the feasibility of deploying VDFS in an open-source cloud. This scheme not only reduces the cloud storage capacity, but also improves the speed of data de-duplication and concentrates on security to data's. Furthermore, the signature is computed for every file for verifying the integrity of files.

Keywords— De-duplication, virtual machine, open-source cloud, file system, VDFS. Introduction.

I. INTRODUCTION

Modern society is a digital universe. Almost no information or industry applications can survive without this digital universe. The size of this digital universe in 2007 is 281 Hexabytes and in 2011 [10], it becomes 10 times larger than it was in 2007. The most critical issue is that nearly half the digital universe cannot be stored properly in time. This is caused by several reasons: firstly, it is hard to find such a big data container; secondly, even if a big container can be found, it is still impossible to manage such a vast dataset; and finally, for economic reasons, building and maintaining such a huge storage system will cost a lot of money. This is particularly
challenging for non-IT sectors, for example, engineering and bio-chemistry industries. According to our experiences, a typical information management center at a city-level nuclear power generation factory needs to process hundreds of gigabytes of new data each day. Such data should also be easily accessible and used for different purposes by other information centers located in other cities in the power grid, as well as government authorities at different levels. In the area of computer aided engineering (CAE), some efforts are made to tackle challenges in the management of large quantity distributed data and knowledge. But the issue of scalability remains.

Fortunately, with the rocket-like development of cloud computing, the advantages of cloud storage have amplified significantly, and the concept of cloud storage has become vastly accepted by the community. Cloud computing consists of both applications and hardware delivered to users as a service via the Internet [13, 16].

With the rapid development of cloud computing, more and more cloud services have emerged, such as SaaS (software as a service), PaaS (platform as a service) and IaaS (infrastructure as a service). The concept of cloud storage is derived from cloud computing. It refers to a storage device accessed over the Internet via Web service application program interfaces (API). VDFS(namely Virtual Distributed File System) is a distributed file system that runs on commodity hardware; it was developed by VMWare for managing massive data. The advantage of VDFS is that it can be used in a high throughput and large dataset environment. VBase is Virtual database, which is an open-source, distributed, versioned, column-oriented database [5]. It is good at real time queries. VDFS has been used in numerous large scale engineering applications. Based on these features, in our work, we use VDFS as a storage system. We use VBase as an indexing system.

Currently cloud computing is applied more in data intensive areas such as e-commerce or scientific computations. There is little research on engineering oriented cloud system. There especially lack direct applications or experiments for cooperative work in design, where data sharing and duplication management have always been challenges.

This paper presents a deduplication cloud storage system, named "DeDu", which runs on commodity hardware. Deduplication means that the number of the replicas of data that were traditionally duplicated on the cloud should be managed and controlled to decrease the real storage space requested for such duplications. At the front end, DeDu has a deduplication application. At the back end, there are two main components, VDFS and VBase, used respectively as a mass storage system and a fast indexing system. Promising results were obtained from our simulations using VMware to simulate a cloud environment and execute the application on the cloud environment.

Regarding contributions of this paper, there are two issues to be addressed. Firstly, how does the system identify the data duplications? Secondly, how does the system manage and manipulate the data to reduce the duplications, in other words, to deduplicate them? For the first issue, we use hashing algorithm to make a unique hash code for each file or data block, and set up a fast index to identify the duplications. For the second problem, we set up a distribution file system to store data and develop 'link files' to manage files in a distributed file system.

II LITERATURE SURVEY

The HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. POSIX imposes many hard requirements that are not needed for applications that are targeted for HDFS. POSIX semantics in a few key areas has been traded to increase data throughput rates. Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance.

An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. 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 Data Nodes. The Name Node executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to Data Nodes. The Data Nodes are responsible for serving read and write requests from the file system's clients. The Data Nodes also perform block creation, deletion, and replication upon instruction from the Name Node.HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. 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 writing-once and have strictly one writer at any time. The Name Node makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Block report from each of the Data Nodes in the cluster. Receipt of a Heartbeat implies that the Data Node is functioning properly. A Block report contains a list of all blocks on a Data Node.

IIi. DE-DUPLICATEDVIRTUALCLOUDENVIRONMENT

The architecture of the proposed de-duplicated based cloud system is illustrated in Fig. 1.

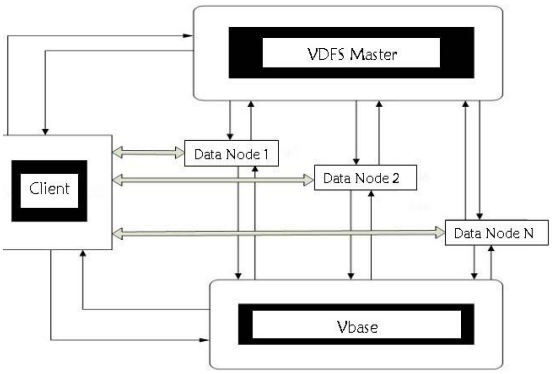

Fig.1. Architectural design for De-duplication

When a user uploads a file for the first time, the system records this file as source data in Vbase, and the user will receive a link file for this user himself, and the same for other potential users, to access the source data. When the source data has been stored in the VDFS, if the same data

is uploaded by other users, the system will not accept the same data as new, but rather, the new user, who is uploading data, will receive a link file to the original source data. Users are allowed to read the source data but not to write. Once the initial user changes the original data, the system will set the changed data as a new one, and a new link file will be given to the user. The other users who connect to the original file will not be impacted. Under these conditions, the more users share the same data, the more storage space will be saved.

IV. IMPLEMENTATION OF CLOUD COMPUTING

In this system, HDFS and HBase must collaborate to guarantee that the system is working well. There are two types of files saved in HDFS, one is source file, and the other one is link file. We separate source files and link files into different folders (see Fig. 2).

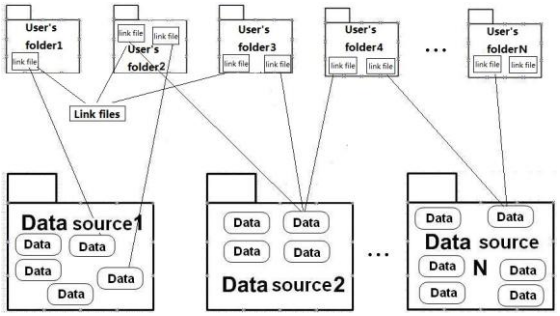

Fig.2. Data organization for De-duplication

After the combination hash value and saved in a folder which is named by date. When the source file is over 64MB, it will be divided into 64MB chunks and saved in the system, but these chunks will be distributed in different data nodes. As for the link file, the filename is in the form "***.lnk", where "***" is the original name/extension of the source file. Every link file records one hash value for each source file and the logical path to the source file, and it uses approximately 320 bytes to store the essential information. Both link file and the folder created by the user are saved in the distribution file system. HBase records all the hash values for each file, the number of links, and the logical path to the source file. There is only one table in HBase, which is named "dedu". There are three columns in the table, which have the headings: hash_value, count, and path. Hash_value is the primary key. Count is used to calculate the number of links for each source file. Path is used for recording the logical path to the source file.

Procedures to store the files

In DeDu, there are three main steps to save a file. Firstly, it is necessary to make a hash value at the client; secondly, the system identifies any duplication; thirdly, the system saves the file.

Firstly, users select the files or folders which are going to be uploaded and stored by using a DeDu application. The application uses the MD5 and SHA- 1 hash functions to calculate the file's hash value, and then pass the value toHBase.

Secondly, the table 'dedu' in HBase keeps all file hash values. HBase is operated in the HDFS environment. It will compare the new hash value with the existing values. If it does not exist, a new hash value will be recorded in the table, and then HDFS will ask clients to upload the files and record the logical path; if it does exist, HDFS will check the number of links, and if the number is not zero, the counter will be incremented by one. In this case, HDFS will tell the clients that the file has been saved previously. If the number is zero, HDFS will ask the client to upload the file and update the logical path.

Thirdly, HDFS will store source files, which are uploaded by users, and corresponding link files, which are automatically generated by DeDu. Link files record the source file's hash value and the logical path of the source file.

V.VIRTUAL CLOUD-BASED DE-DUPLICATED SYSTEM MECHANISM

For setting up virtual cloud we need two mechanisms to set up our storage system. One is used to store mass data, and the other is used to keep the sparse index. On the one hand, there are several secondary storage systems, like Ceph [24], Petal [9], being used as mass data storage systems. On the other hand, there are several database systems such as SQL, Oracle, HBase, and BigTable [8] that can be used as index systems. All these systems have their own features, but we need two systems combined together to achieve our data access requirements. With regard to our requirements, in order to store masses of information, the file system must be stable, scalable, and fault-tolerant; for the sparse index, the system must perform nicely on real time queries. Considering these requirements, we employ HDFS and HBase to structure our storage mechanisms.

A.Vbase and VDFS Conception.

In this process the cloud setup is created by using the Virtual Machine Workstation. Workstation is desktop software that allows you to run multiple x86-compatible desktop and server operating systems simultaneously on a single PC, in fully networked, portable virtual machines with no rebooting or hard drive partitioning required.

B.Hash Value generation and Verification

MD5 Hash code is a unique string of characters that can be generated for a given character string. This Hash code is very important when you compare string values. It is not technically correct to compare two string directly.

When a file is selected the hash value generator, generate the relevant hash value for the file in the basis of file content. There it is required to read a file as byte stream and generate Hash code for that stream.

To generate MD5 Hash for any object is convert that object into a byte array. For this take MemoryStream and BinaryFormatter objects.

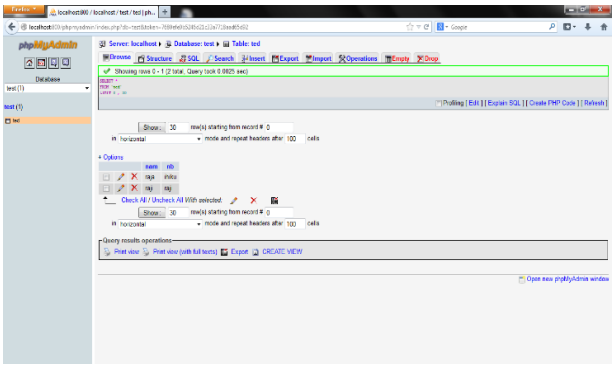

Fig.3. Database Evaluation on server side

MemoryStream fs = new MemoryStream();
BinaryFormatter formatter = new BinaryFormatter();
Serialize() method is used to directly convert object into MemoryStream. Before this operation to be thread-safe we should lock the object.

C.File Uploading

In DeDu, there are three main steps to save a file. Firstly, users select the files or folders which are going to be uploaded and stored by using a DeDu application. The application uses the MD5 hash functions to calculate the file's hash value, and then pass the value to VBase.

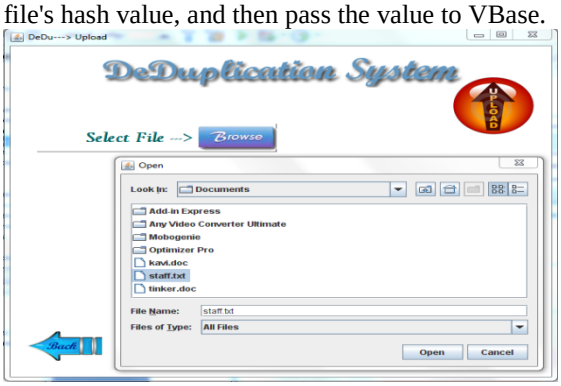

Fig.4. Upload Screen for Virtual cloud

Secondly, the table ‘dedu’ in VBase keeps all file hash values. VBase is operated in the VDFS environment. It will compare the new hash value with the existing values. If it does not exist, a new hash value will be recorded in the table, and then VDFS will ask clients to upload the files and record the logical path; if it does exist, VDFS will check the number of links, and if the number is not zero, the counter will be incremented by one. If the number is zero, VDFS will ask the client to upload the file and update the logical path. Thirdly, VDFS will store source files, which are uploaded by users, and corresponding link files, which are automatically generated by DeDu. Link files record the source file's hash value and the logical path of the source file.

VI. Results and Discussion

In our experiment, our cloud storage platform was set up on a VM ware workstation. The configuration of the host machine is that the CPU is 2.32 GHz; RAM is 2 GB; Hard disk is 320GB.

We uploaded some massive amount of files, amounting to 15 to 20 MB, into DeDu. In a traditional storage system, they should occupy approximately 50 MB, both the physical storage space and the number of files should be three times larger than actual size. In a perfect de-duplication distribution file system, the results should take up and shown that the storage capacity are minimized effectively. The extra consumption of storage is occupied by link files and the 'dedu' table, which is saved in VBase. By using the distribution hashing, an exact de-duplication result is achieved. In DeDu storage system , each file could only be kept in 3 copies at different data nodes, as backup in case some data nodes are dead. This means that if a file is saved into this system less than three times, the efficiency of de-duplication is low. When a file is put into the system more than three times, the efficiency of de-duplication will become high. Thus, the exact de-duplication efficiency depends on both the original data duplication ratio and how many times the original data has been saved. The higher the duplication ratio the original data has, the greater the de-duplication efficiency. It is also true that the greater the number of times that the original data is saved, the greater the de-duplication efficiency that can be achieved.

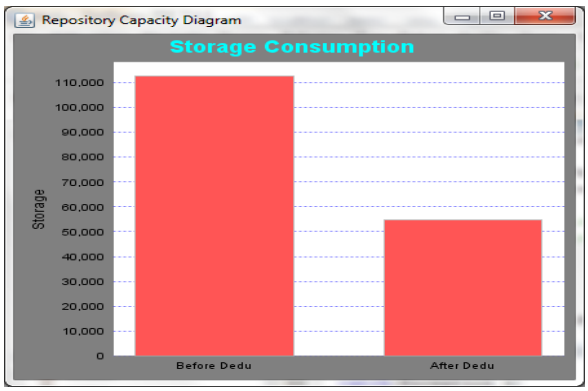

Fig.5. Result analysis

On resulting the writing efficiency the fewer the data nodes it has, the higher the writing efficiency , but the lower the reading efficiency. The more data nodes there are, the lower the writing efficiency it will be, but the higher the reading efficiency. When a single file is big, the time to calculate hash values becomes higher, but the time of transmission cost is low. When a single file is small, the time to calculate hash values becomes lower, but the transmission cost is high in real cloud environment.

VII. Conclusions

In conclusion, we have introduced an inviolable approach to data de-duplication over the engineering oriented cloud systems, which we have named as DeDu. DeDu is not only useful for IT enterprises or engineering industry to backup data, but also for common users who want to store data. This project approach exploits a file’s hash

value as an index saved in VBase to attain high lookup performance, and it exploits 'link files' to manage mass data in a Virtual distributed file system. In DeDu, the higher configuration of a single node will not impact the performance of the whole system too much. To get better performance from such a cloud-type system, we need to consider how to tune up critical nodes' configurations. Furthermore, data compression is another potential hurdle in DeDu. It sacrifices too much overall data transmission efficiency to save the storage space in a de-duplication application. We will look into these issues in future work, to circumvent the design objective of DeDu.

ACKNOWLEDGEMENT

I would like to thank our head of the department Mr.S.RajPandian and my guide Mrs.Rajalavanya Chakaravarthyfor motivating me in doing this projects and thanks to all my friends and the web sources.

REFERENCES

[1] Zhe Sun, Jun Shen, Jianming Young , A novel approach to data de-duplication over the engineering-oriented cloud systems, Faculty of Engineering and Information Sciences,University of Wollongong, 2013,14.

[2] S. A. Weil. S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: a scalable, high-performance distributed file system," in Proceedings of the 7th symposium on Operating systems design and implementation, Seattle, Washington. 2006, pp. 307-320.

[3] A. Atul, J.B. William, C. Miguel, C. Gerald, C. Ronnie, R.D. John, H. Jon, R.L. Jacob, T. Marvin and P.W. Roger, Farsite: Federated, available, and reliable storage for an incompletely trusted environment, SIGOPS Oper. Syst. Rev., 2002, 36, pp. 1-14

[4] K. L. Edward and A. T. Chandramohan, "PetaJ: distributed virtuaJ disks," in Proceedings of the seventh international conference on Architectural support for programming languages and operating systems, Cambridge, Massachusetts, US, 1996, pp. 84-92.

[5] A.-E.-M. Michael, William V. Courtright, C. Chuck, R. G. Gregory, H. James, J. K. Andrew, M. Michael, P. Manish, S. Brandon, R. S. Raja, et al., "Ursa minor: versatile cluster-based storage," in Proceedings of the 4th conference on USENfX Conference on File and Storage Technologies, San Francisco, CA, 2005, pp. 59-72.

[6] T. C. Austin, A. Irfan, V. MuraJi, and L. Jinyuan, "Decentralized de-duplication in SAN cluster file systems," in Proceedings of the 2009 conference on USENIX Annual technical conference, San Diego, CaJifornia, 2009, pp. 101-114.

[7] A. W. Sage, W. L. Andrew, A. B. Scott, and M. Carlos, "RADOS: a scaJable, reliable storage service for petabyte-scaJe storage clusters," in Proceedings of the 2nd international workshop on Petascale data 355 storage: held in conjunction with Supercomputing, Reno, Nevada, 2007, pp. 35-44.

[8] J.F. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting and A. Toncheva, The Diverse and Exploding Digital Universe,March2008.URL:http://www.emc.com/collateral/analystreports/diverseexplodin g-digital-universe.pdf, accessed in Oct 2011.

[9] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse Indexing: Large Scale, Inline De-duplication Using Sampling and LocaJity," in 7th USENIX Conference on File and Storage Technologies, San Francisco, California 2009, pp. 111-123

[10] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain de-duplication file system," in Proceedings of the 6th Usenix Conference on File and Storage Technologies (Fast '08), 2008, pp. 269-282

[11] D. Bhagwat, K. Eshghi, D.D.E. Long and M. Lillibridge, Extreme Binning: Scalable, Parallel De-duplication for Chunkbased File Backup, in 2009 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems Mascots, 2009, pp. 237-245.

[12] B. Hong and D.D.E. Long, Duplicate data elimination in a san file system, In Proceedings of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST), 2004, pp. 301-314.

[13] J.O. Gutierrez-Garcia and K.M. Sim, Agent-based cloud workflow execution, Integrated Computer-Aided Engineering, 2012, 19:1, pp. 39-56.

[14] F. Zhang, Z.M. Ma and L. Yan, Construction of Ontology from Object-oriented Database Model, Integrated Computer- Aided Engineering, 2011, 18:4, in press.

[15] J. Wei, H. Jiang, K. Zhou and D. Feng, MAD2: A scalable high-throughput exact de-duplication approach for network backup services, in Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on Incline Village, NV, USA 2010 pp. 1-14

[16] Z. Sun, J. Shen and G. Beydoun, P2CP: A New Cloud Storage Model to Enhance Performance of Cloud Services, in Proceedings of International Conference on Information Resources Management, Conf-IRM, 2011, on CD-ROM