ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

PREVENTING DATA LOSS AND INCREASING DATA SECURITY USING FILE INSTANCE SHARING (FIS)

Vijayakumar .K
Associate Professor
Department of Computer Science and Engineering
St. Joseph's Institute of Technology, Chennai, India
mkvijay@msn.com

Aravind S
Department of Computer Science and Engineering
St. Joseph's Institute of Technology, Chennai, India
aravinddbz11@gmail.com

Ullas Kumar S
Department of Computer Science and Engineering
St. Joseph's Institute of Technology, Chennai, India
ullas.kumar14@gmail.com

Abstract - Using cloud storage, users can remotely store their data and enjoy the on-demand high-quality applications and services. The users can share their file in the cloud to anyone at anytime. In this paper, the key questions of the user about the security of the shared data and data loss is answered. In the existing system, there is high possibility of data theft or loss in the cloud environment. And it is almost impossible to recover the lost data. Our proposed system put forth an efficient and feasible solution to overcome this problem. In this architecture, we develop a cloud environment that creates instances of files when the file is shared among the group of users. So the key for our system is that the instance is shared instead of the original file. This entirely prevents the data theft and loss. Even when such scenario occurs, the data can still be recovered. Also the spam users are identified and a two step verification is enabled for those users. Finally, after the file owner's approval the instances are merged to the main file and all the temporarily instances that have been created are deleted in the cloud storage. This enables the cloud users to share their data in a safe and secured way.

Index Terms – Cloud computing, Cloud Security, File sharing.

1 INTRODUCTION

Cloud computing is an emerging paradigm in the computer industry where the computing is moved to a cloud of computers. In science, cloud computing is a synonym for distributed computing over a network, and means the ability to run a program or application on many connected computers at the same time. It takes the technology, services, and applications that are similar to those on the Internet and turns them into a self-service utility. It is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility.[21]

In this architecture it is considered that the cloud user uploads the file and shares the uploaded file to a group of users. The objective of sharing the file is to make the users of the group to edit it. There are many potentials threats of sharing the file to the group. One such threat is wrong edits by different users. As soon as a file is shared to the group all the users in the group has the write to modify the entire contents of the file. Each user have their own unique way of editing. For example some may use 'organization' and some others may use 'organization'. But the content of the end file must be decided only by the user who uploaded it. This project aims at developing a cloud environment which handles the above scenario. It is achieved by creating different instances of file for different users accessing it. At the end the file owner reviews the instances and approves it so that the end file is obtained. Problems like spam users, Parallel Modification, Collaborative document centric editing are handled in a well efficient manner. Though an acknowledgement based approach can solve this issue, it is less feasible when it comes to dynamic group where there will be N number of users. The above proposed architecture ensures the data security and reduces the data loss in cloud.

2 RELATED WORK

In a secure multi-owner data sharing scheme, named Mona, for dynamic groups in the cloud. By leveraging group signature and dynamic broadcast encryption techniques, any cloud user can anonymously share data with others. Meanwhile, the storage overhead and encryption computation cost of their scheme are independent with the number of revoked users. In addition, they have also analyzed the security of their scheme with rigorous proofs, and demonstrated the efficiency of their scheme in experiments. Though the files can be accessed only with the group keys, once the file had been opened by the user he has full access to modify content of the file may not be according to the expectation of the user[1]. Several security schemes for data sharing on untrusted servers have been proposed [4], [5]. In these approaches, data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the data files because they have no knowledge of the decryption keys.

The complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the number of revoked users, respectively. By setting a group with a single attribute, Lu et al. [7] proposed a secure provenance scheme based on the cipher text-policy attribute-based encryption technique [8], which allows any member in a group to share data with others. In another model it is addressed that challenging problem of simultaneously achieving fine-grainedness, scalability, and data confidentiality of access control, on one hand, defining and enforcing access policies based on data attributes, and, on the other hand, allowing the data owner to delegate most of the computation tasks involved in fine-grained data access control to untrusted cloud servers without disclosing the underlying data contents. This goal is achieved by exploiting and uniquely combining techniques of attribute-based encryption (ABE), proxy re-encryption, and lazy re-encryption. In the proposed scheme also has salient properties of user access privilege confidentiality and user secret key accountability. Extensive analysis shows that this proposed scheme is highly efficient and provably secure under existing security models.

Once the file has been decrypted there is no restriction for the user in editing the file. First issue is without the guarantee of identity privacy, users may be unwilling to join in cloud computing because their real identities could be easily disclosed to cloud providers and attackers. Second , it should support multiple-owner manner that is any member in a group should be able to enjoy the data storing and sharing services in cloud. Last issue, groups are normally dynamic[10] in practice.

Deduplication technology tries to find the identical content and stores it once to save space requirement.[4][5]. But deduplication needs extra storage space to store index information, it needs computing resources, extra read/write operating. It also decreases the data dependability and causes security risks.

Cloud is operated by Cloud Service Providers(CSPs). It provides paid storage space on its infrastructure to store the owner's files and make them available for authorized users. However, cloud is not fully trusted by the users since CSPs are outside the cloud user's trusted domain. Cloud server do not delete or modify the data [5][6], but try to learn the content of the data. Group manager performs system parameter generation, user registration, user revocation, and revealing the real identity of the dispute owner. Group manager is fully trusted by the other parties. Group members are set of registered users. They willstore their private data into the cloud and share them with other members in the group. The group membership is dynamically changed. So, any member can join and leave the group at any time. Group Member

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

registration and revocation are managed by the Group Manager. Efficient processing of huge structured and unstructured data sets in the cloud using commodity servers has become a new paradigm for computation [3,4]. A key challenge for any large-scale computation today, whether in big data or in handling large-scale web services, has to do with the management and security of data [3]. With object storage services becoming increasingly accepted as replacements for traditional file, it is important to effectively encrypt digital data of these services.

Daisuke Tsukamoto and Takuo Nakashima proposes a distributed file sharing system over the P2P network can especially provide the quick search, while the file accesses from the remote user do not follow the uniform distribution causing the degradation of searching performance. In this paper, we propose the new framework to implement the access method of shared objects into the real environment using the MPI (Message Passing Interface). In addition, we implemented the conventional Lock/Unlock method using MPI to compare the proposed owner permission method to access shared objects. The proposed shared object management system implemented on the MPI middleware can improve the processing speed compared to the Lock/Unlock method. Chun-Hsin Wu and Pai-Hsing Wang propose a system whose approach is that a file is not completely stored in the cloud, and only authorized devices or clients have their own unique key, which is derived from erasure coding, to fulfil the file exactly. By investigating three typical erasure codes, we analyze the efficiency and trade-offs of different file splitting strategies. The results show that it is promising to share files securely for cloud storage with erasure coding.

A model CloudStash is proposed where, a multi-cloud scheme that utilizes the secret-sharing scheme [13] to directly protect data for low cost cloud storage and multithreading to address confidentiality, availability and performance in cloud storage. Specifically, our contribution is to design, implement, and evaluate a CloudStash scheme comparing it with the approach used in state of the art system wherein data is encrypted and stored in the multicloud and the standard encryption key is protected via secret sharing. Using shares to separately protect the key initially seems as if might be faster. However, as we shall show, it is often slower, as it requires multiple rounds of cloud access. Furthermore, it also leaves the data open to an attack on the key.

3 THE PROPOSED SCHEME

3.1 Architecture diagram

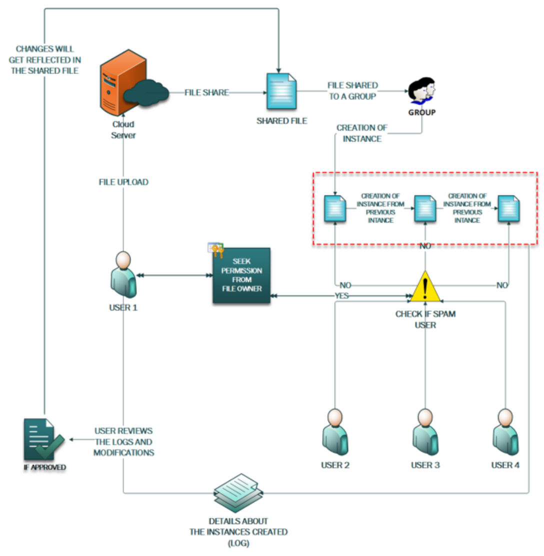

CHANGES WILL GET REFLECTED IN THE SHARED FILE
Cloud Server
FILE SHARE
SHARED FILE
FILE SHARED TO A GROUP
GROUP
CREATION OF INSTANCE
FILE UPLOAD
CREATION OF INSTANCE FROM PREVIOUS INTANCE
CREATION OF INSTANCE FROM PREVIOUS INTANCE
NO
SEEK PERMISSION FROM FILE OWNER
USER 1
NO
YES
NO
CHECK IF SPAM USER
USER REVIEWS THE LOGS AND MODIFICATIONS
IF APPROVED
USER 2
USER 3
USER 4
DETAILS ABOUT THE INSTANCES CREATED (LOG)

Fig 3.1 System Architecture

The above given architecture in fig 4.1 gives an idea about how an user uploads and shares the file to a dynamic group without the loss of data and also helps the user to identify the spam users in the group, thereby increasing the security. User 1 uploads the file to the cloud server and he/she shares the file to a group of users. Now when User 2 tries to access the file a Spam Check is performed where the spam rating of user is checked. If the rating is below the threshold value then the user is allowed to access the file. If it is above the threshold value then an auto generated mail is sent to the file owner specifying the details of the user who is not able to access the file shared in the group, so that the file owner

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

can decide whether to provide access for that particular user or not.

After the creation of instance of the shared file for User 2, whatever changes or edits the User 2 makes it gets reflected only in his instance and in the main file. The shared file gets untouched. Now when User 3 tries to access the file, an instance of previously stored instance is created so that the User 3 gets the changes made by User 2 as well.

Therefore N number of instances are created for N number of access of the file. These instances are temporarily stored in the cloud storage till the user makes his approval.

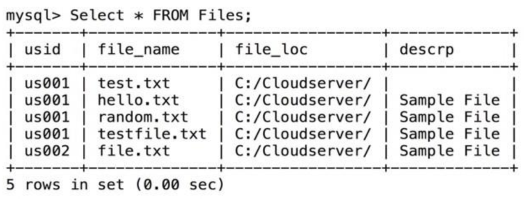
Parallel edits are handled using the Least Common Subsequence algorithm. The details about the instances are stored in the database. The file owner can select the shared file and can see all the edits made by the users of the group. He can review each edit and he can also approve it. Spam ratings are assigned to the users of the group based on the number of wrong changes they made.

3.1.1 User registration, Authentication and File Upload

In this module the cloud user can register himself to the cloud environment. After registering an id will be generated which will be intimated to him by an automated mail. After registering the user can login to the cloud environment. The authentication servlet authorizes the user. Now the user can upload the file to the cloud server. For uploading the file Commons-File Upload mechanism is used.

Commons-File Upload

A file upload request comprises an ordered list of items that are encoded according to RFC 1867, "Form-based File Upload in HTML". FileUpload can parse such a request and provide the application with a list of the individual uploaded items. Each such item implements the FileItem interface, regardless of its underlying implementation. Every item has a name and a content type, and can provide an InputStream to access its data. On the other hand, the items must be processed differently, depending upon whether the item is a regular form field - that is, the data came from an ordinary text box or similar HTML field - or an uploaded file. The FileItem interface provides the methods to make such a determination, and to access the data in the most appropriate manner. FileUpload creates new file items using a FileItemFactory. This is what gives FileUpload most of its flexibility. The factory has ultimate control over how each item is created. The factory implementation that currently ships with FileUpload stores the item's data in memory or on disk, depending on the size of the item (i.e. bytes of data). The user registers himself and his data gets stored in the database. Once the user has been registered he is allowed to login and upload the file to the cloud server using the Apache Common File Upload. The details of the file uploaded by the user gets stored in the database table Files.



```
mysql> Select * FROM Files;
+-------+---------------+-----------------+-------------+
| usid  | file_name     | file_loc        | descrp      |
+-------+---------------+-----------------+-------------+
| us001 | test.txt      | C:/Cloudserver/ |             |
| us001 | hello.txt     | C:/Cloudserver/ | Sample File |
| us001 | random.txt    | C:/Cloudserver/ | Sample File |
| us001 | testfile.txt  | C:/Cloudserver/ | Sample File |
| us002 | file.txt      | C:/Cloudserver/ | Sample File |
+-------+---------------+-----------------+-------------+
5 rows in set (0.00 sec)
```

Fig 3.2.1 Details of various files in the database

3.2.2 Sharing files to dynamic group

This module explains about how an uploaded file is shared by the user in the dynamic group. The user is allowed to select the file that he/she wishes to share. Once the user has selected the file, he is allowed to share the file to the group of users. Now all the users in the group can access the file shared by the user. Whenever an user access the file an instance of the file is been created.

Whatever the changes made by the user gets reflected only in that instance so that the original file remains unchanged. Therefore the original file shared by the user is stored separately in the cloud being undisturbed. The details of the file shared by the user gets stored in the database. In further modules the file instances are handled accordingly. The details of each instances are stored in the database.

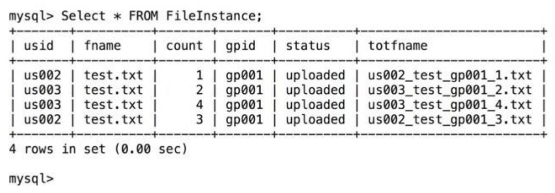
3.1.3. File instance management

This module explains about how the different file instances created in the cloud server are managed. These different file instances created in the cloud server are managed in such a way that there is no collision between them. Each instance of a file has a unique identifier as their file name which helps to identify the file of a particular user who has made the changes and uploaded the it.

The format of the filename will be as follows.
<Username>_<Original File Name>_<Group Name>_<Instance Number>

The above format ensures that no two instances with same filename is created in the cloud server.

After the instance has been created for the user, he/she works on that file. The user can either download the instance and work on it and then upload it to the cloud server. The other way is to work in the default online editor of the cloud environment. The existing CSP'S doesn't support the synchronization of files that are worked offline which is a big drawback in Cloud Computing. Once the user has made the changes in the file, saved it or uploaded, a conflict check is undergone between the recently changed file and the saved or uploaded file. The purpose of this conflict check is to ensure that the user has worked on the latest version of the file. If a newer version of the file is available, then the user is redirected to a page where the user can merge his file with the newer version. This ensures proper synchronization of files. The details of the various instances of the files are stored and managed in the database.



```
mysql> Select * FROM FileInstance;
+-------+----------+-------+-------+----------+-----------------------+
| usid  | fname    | count | gpid  | status   | totfname              |
+-------+----------+-------+-------+----------+-----------------------+
| us002 | test.txt |     1 | gp001 | uploaded | us002_test_gp001_1.txt |
| us003 | test.txt |     2 | gp001 | uploaded | us003_test_gp001_2.txt |
| us003 | test.txt |     4 | gp001 | uploaded | us003_test_gp001_4.txt |
| us002 | test.txt |     3 | gp001 | uploaded | us002_test_gp001_3.txt |
+-------+----------+-------+-------+----------+-----------------------+
4 rows in set (0.00 sec)

mysql>
```

Fig 4.2.3 Details of various instances of files in the database

The latest version of instance available is checked by using the status field. When the user downloads the instance status column is updated as created and when the user tries to save or upload his/her instance it is checked that the last uploaded instance is the instance on which he/she started to work. This process easily identifies the conflict or mismatch of instances. Once the user has saved online or uploaded the file the status column gets updated as uploaded.

3.1.4 Merging changes after file owner's approval

In this module, the file owner reviews all the instances and he approves or selects the file instance that he wants, so that it gets reflected in the main file. Now the file which he/she gets is the end file. The file owner can select the file instance based on the three criteria. First one is based on Time, the second one is based on the number of Correct changes made and the last one is based on the number of Wrong changes. This helps the file owner to select the right file he/she wants to be updated in the main file. The changes made by each user are highlighted to the file owner so it becomes easy to identify the edits. This is achieved by using Longest Common Subsequence(LCS)algorithm.

If there are more instances present then the user have to review all the instances. Since it is not feasible, an option has been introduced with which the user can sort the instances based on the instances that has more probability of wrong edits. This probability of wrong edits is computed by the modified version of Levenshtein distance algorithm. This makes the user to easily identify the user who has made most number of wrong edits.

As soon as the owners approves the instances all the temporarily stored instances are deleted. This ensures that no storage overhead occurs in the cloud server.

3.1.5 Spam Identification

In this module, the process of spam identification is implemented. When an user tries to access the shared file, his/her spam rating is checked. If his spam rating is below a threshold value, then an automated mail is generated to the file owner regarding the access permission of that user. Only if the file

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

owner grants permission to that user, he/she can access the file. Otherwise it is impossible for the user to make any edits in the shared file. Therefore this increases the security by allowing the user to share the file in a trustworthy group. For automatic mail generation JAVA MAIL BY ORACLE is used.

Java Mail by Oracle

The JavaMail defines classes such as Message, Store, and Transport. It can be extended and can be subclassed to provide new protocols and to add functionality when necessary. In addition, it provides concrete subclasses of the abstract classes. These subclasses, including MimeMessage and MimeBodyPart, implement widely used Internet mail protocols and conform to the RFC822 and RFC2045 specifications. The JavaMail includes support for the IMAP4, POP3, and SMTP protocols.[22]

The JavaMail architectural components are as follows:

- ü The abstract layer declares classes, interfaces, and abstract methods intended to support mail handling functions that all mail systems support.
- ü The internet implementation layer implements part of the abstract layer using the RFC822 and MIME internet standards.
- ü JavaMail uses the JavaBeans Activation Framework (JAF) to encapsulate message data and to handle commands intended to interact with that data.

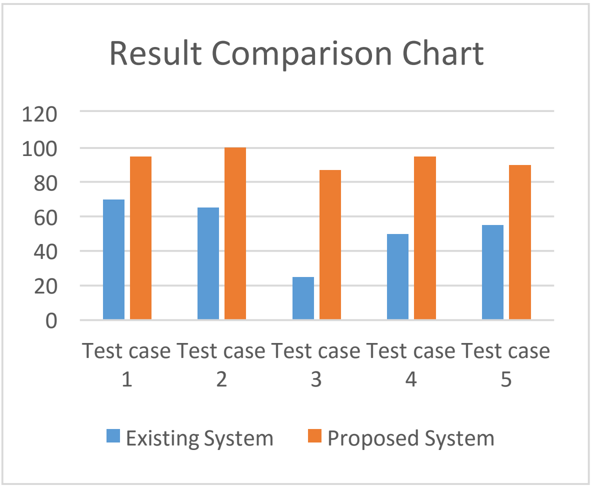
4 PERFORMANCE EVALUATION

A group of cloud users are made to share their file in the existing system and our proposed system. The output of these two systems are compared with their expected file. The percentage of similarity is calculated and tabulated below. Here each user is termed as a test case.

TEST CASES	EXISTING SYSTEM	PROPOSED SYSTEM
	PERCENTAGE OF SIMILARITY REPORTED BY USER	PERCENTAGE OF SIMILARITY REPORTED BY USER
Test case 1	70	95
Test case 2	65	100
Test case 3	25	87
Test case 4	50	95
Test case 5	55	90

Table 5.7 Result Comparison Table

The above values are represented in graph as shown below.



5 CONCLUSION

Data loss and data security are one of the critical problems in Cloud Computing. With the concept of File Instance Sharing (FIS) the above two problems are addressed and solved in an efficient manner. With modified LCS and Levenshtein distance algorithm a cloud environment has been developed which implements the proposed system. The results show that it gives a greater flexibility to the file owner who is sharing his file in the cloud environment.

This experiment showed that our solution completely prevents data loss and increases data security in the cloud environment when data is shared by creating different instances of the file for different users.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G.Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[2]. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine Grained Data Access Control in Cloud Computing," Proc. IEEE INFOCOM, pp. 534-542, 2010.

[3]. B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," Proc. Int'l Conf. Practice and Theory in Public Key Cryptography Conf. Public Key Cryptography, http://eprint.iacr.org/2008/290.pdf, 2008.

[4]. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 89-98, 2006.

[5]. D. Boneh, X. Boyen, and H. Shacham, "Short Group Signature," Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-55, 2004.

[6]. Shraddha B. Toney and Sandeep U.Kadam," Cloud Information Accountability Frameworks for Data Sharing in Cloud - A Review" International Journal of Computer Trends and Technology volume4 Issue3-2013

[7]. Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya "A Taxonomy and Survey of Energy-Efficient Data Centres and Cloud Computing Systems"

[8]. J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks," Computer Communications, 2002.

[9]. H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun, and B.-Y. Yang, "A Study of User- Friendly Hash Comparison Schemes," Proc. Ann. Computer Security Applications Conf. (ACSAC), pp. 105-114, 2009.

[10]. A.R. Yumerefendi and J.S. Chase, "Strong Accountability for Network Storage," Proc. Sixth USENIX Conf. File and Storage Technologies (FAST), pp. 77-92, 2007.

[11]. Y. Tang, P. Lee, J. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," IEEE Trans. Dependable Secure Comput., vol. 9, no. 6, pp. 903–916, Nov./Dec. 2012.

[12]. K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," IEEE Internet Comput., vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[13]. C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," IEEE-ACM Trans. Netw., vol. 8, no. 1, pp. 16–30, Feb. 2000.

[14]. T. Jung, X. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in Proc. IEEE Conf. Comput. Commun., 2013, pp. 2625–2633.

[15]. I. Lam, S. Szebeni, and L. Buttyan, "Invitation-oriented TGDH: Key management for dynamic groups in an asynchronous communication model," in Proc. 41st Int. Conf. Parallel Process. Workshops, 2012, pp. 269–276

[16] Kan Yang, Xiaohua Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing", IEEE Transactions on Parallel and Distributed Systems.

[17] Daniel W.K. TSE, "Challenges on Privacy and Reliability in Cloud Computing Security", Department of Information

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

Systems City University of Hong Kong, Hong Kong SAR, 2014 IEEE.

[18]https://www.usenix.org/system/files/login/articles/105470-Darcy.pdf

[19]http://www.oracle.com/us/products/database/cloud-file system/overview/index.html

[20]http://searchcloudstorage.techtarget.com/podcast/Cloud-data-access Transfer-retrieval-not-so-simple

[21]http://dspace.cusat.ac.in/xmlui/bitstream/handle/123456789/2625/Cloud%20Computing.pdf?sequence=1

[22]http://www.oracle.com/technetwork/java/javamail/index.html