



IJIRCCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 6, June 2021

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 7.542



9940 572 462



6381 907 438



ijircce@gmail.com



www.ijircce.com

Secure Cloud Storage Monitoring System with Deduplication and File Access Pattern Analysis

Deepu CG, AM Prasad

PG Student, Dept. of CSE, Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India

Professor, Dept. of CSE, Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India

ABSTRACT: Cloud computing is an important technology on current demanding business requirements and it has been emerged as unavoidable technology. The usage of IaaS Service storage for Cloud Computing is being expanding exponential every year. The cloud storages are used by the cloud user due to its economy compared with other storage methods. The replications of files help users for easy access with high availability which reduces the overall access time of the files, but at the same time, it occupies more storage space and results in high storage cost. The cloud user holds multiple times of the storage than what he is actually needed. It is a dire need of system to find unwanted files in the cloud and also optimize the storage space by evaluating through file access frequency.

This paper proposes Cloud Storage Monitoring (CSM) system, which monitors the IaaS storage usage and analyzes the file access patterns by various parameters to identify the frequency of access, size, future access prediction, replication of files in the cloud storage. This allocates a ranking for each file which also predicts future access pattern. This generates a recommendation dashboard to the user who can decide on the operations such as reorganize, delete or archive the files and eliminate duplicate files in the cloud storage which can increase the space for future use. and also perform Data de-duplication.

KEYWORDS: Cloud Computing, Cloud Storage Monitoring, File access Pattern, Java Swing

I. INTRODUCTION

The data replication services of cloud storage duplicate the files in real time to increase the availability of the files which in turn increase the hardware cost. The data replication service consists of data replication, file replication, cloning infrastructure and remote storage replication. The cloud storage replication service determines of redundancy which is invaluable on main storage when backup system fails.

As the result, replication is used to reach highest availability at high cost. It is degrading the performance of the service when the cost benefits accrued from the replication. This also increases delay in request and response transaction in cloud environment. The predictive auto-scaling technique forecasts future storage workload of the cloud service and adjusts the cloud storage capacity in order to meet the future needs. The system generates the recommendation dashboard to forecast future files usability and it also eliminates the duplicate entries.

II. THEORY

A. CLOUD STORAGE MONITORING (CSM) SYSTEM

A prediction and ranking based system are proposed to handle the de-duplication in cloud storage with the following design objectives.

- Identify the frequency on access pattern
- Provide prediction on file access
- Identify the duplication of files on cloud storage
- Build storage efficient system.
- Increase efficiency of the system.
- Improve search experience
- Block duplication of files in future path and send RREP.

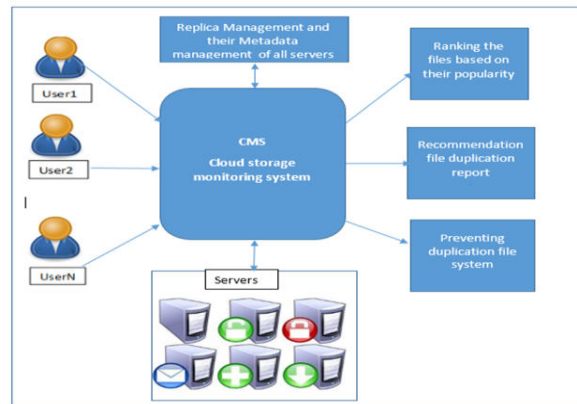


Fig. 1. Cloud Storage Monitoring Architecture

B. Challenges on Cloud Storage

Nowadays most organizations understand the benefits of migrating data to a cloud storage service but at the same time cloud services also having its own risks and drawbacks. In future cloud storage services will replace the storage network in the data centre, mostly due to highly sensitive transactional applications, data-intensive, low-response time, and deals with critical data. Most of use cases are related to organizations and companies having substantial on-premise storage requirements related to cloud storage from various vendors in a Public/Private/Hybrid model deployment. The organization is making difficult on enforcing cloud storage data management policies and best practices on storage optimization features.

III. DESIGN AND SYSTEM TESTING

In The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of Tests

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.



System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Integration testing for Server Synchronization:

- Testing the IP Address for to communicate with the other Nodes.
- Check the request is sent from client to CMS.
- Check the request is sent from CMS to storage nodes.
- Check the file is sent from storage node to CMS.
- Check the file is sent from CMS to client.

Testing results are as mentioned in the table below:

Table 1. Testing result

| MODULE | GIVEN INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT | RESULT |
|--------|----------------------------|---|---|--------|
| Client | File Upload | Upload request should be sent to CMS | File upload request sent to CMS | OK |
| Client | File Download | File download request should be sent to CMS | File Download request should be sent to CMS and file downloaded | OK |
| CMS | Upload request from client | File should be checked for duplication and encrypted, compressed and sent to storage node | File successfully checked for duplication, encrypted, compressed and sent to storage node | OK |



| | | | | |
|--------------|------------------|--|---|----|
| Storage node | Upload request | File should be saved and acknowledgement should be sent to CMS | File successfully saved and ACK sent to CMS | OK |
| Storage node | Download request | File content should be sent to CMS | File content successfully sent to CMS | OK |
| Storage node | Delete request | File should be deleted | File successfully deleted | OK |

A. AWT and Swing class hierarchy

The Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system. Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing (JApplet, JDialog, JFrame, and JWindow) extend an AWT top-level container. However, the use of both lightweight and heavyweight components within the same window is generally discouraged due to Z-order incompatibilities

ServerSocket(int port, int maxQueue, InetAddress localAddress)-Creates a server socket on the specified port with a maximum queue length of maxQueue. On a multihomed host, localAddress specifies the IP address to which this socket binds.

In this training dataset it contains the primary dataset. The model is trained on the training dataset employing a supervised learning method. The training dataset often consists of pairs of an input vector and thus the corresponding output vector.

IV. IMPLEMENTATION AND METHODOLOGY

Our project implementation consists of three major modules i.e. Client, Cloud Managements Service (CMS) and Storage Nodes. The sub modules present in each module are as described below:

1. CLIENT

- **Register:** In this module a new client can enter his username and password and register with CMS.
- **Login:** The registered clients can enter their username and password for login. These details will be verified by CMS and allowed to login on providing valid details.
- **File Upload:** A client can browse a file and upload it to storage node using this module. Upload request is sent to CMS and a response is sent back to client.
- **File Download:** In this module client will get list of files which are uploaded by him. The files are displayed based on rank and also last access details are shown. Client can select a file and download it.
- **Delete:** If client want to delete a file from storage node, he can select the file from list of available files and delete it.

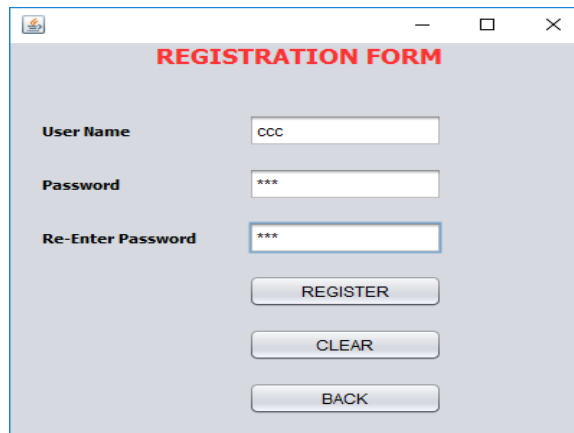


Fig. 2. User Registration form

2. CMS

- **Deduplication:** When a file upload request is sent from client to CMS, this module will check for duplication of file based on extension and size of file. If any file which is already available in storage nodes matches with extension and size of file to be uploaded, that file will be matched for duplication. If the file size is less than 100 bytes, whole content will be matched else first and last 50 bytes will be matched. If the file already exist, upload is cancelled and a link will be created to client for downloading existing file.
- **Find Rank:** When a new file is uploaded by client or existing file is downloaded, frequency for that file is updated and rank will be generated for the file based on the frequency of access. Whichever file has more frequency will have 1st rank and vice versa.
- **Update Frequency:** Whenever a client uploads a file or downloads existing file, count will be updated for the file. This count is called as frequency. It is used for finding rank for file.
- **Encryption:** All the files which will be uploaded to storage nodes are encrypted for security purpose. We use DES encryption algorithm for encryption files. It uses 64 bit key.
- **Decryption:** When a client downloads file or a file has to be matched for duplication, that file will be decrypted.
- **Compression:** To reduce the space to be used by file in storage nodes, the files will be compresses and stored. We use zip compression scheme for compressing files.
- **Decompression:** When a client downloads file or a file has to be matched for duplication, that file will be decompressed and decrypted.
- **Upload:** When a client sends upload request to CMS, the file will be checked for duplication, then encrypted and compressed and finally it is uploaded in storage node. The storage node which has max memory will be selected by CMS for file upload.
- **Download:** When a client sends download request, CMS finds the storage node which contains file and downloads file from it. Then the file is decompressed and decrypted and sent to client.
- **Delete:** When client sends delete request for a file, CMS checks if the file is used by more than one client. If so, the link for the file which is created for the client will be reused. Hence file will be available for other users. If file is not used by other clients, then the file will be deleted from storage node.

Fig. 3. After user registration

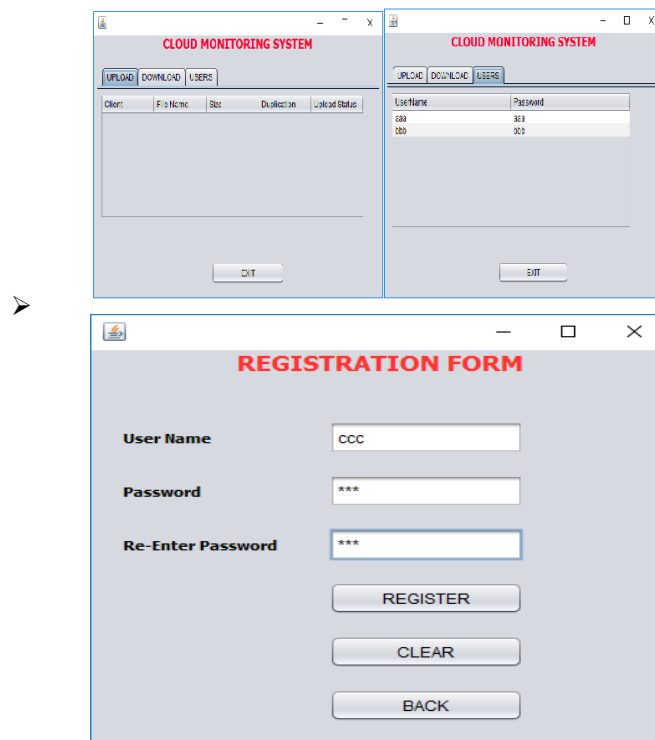
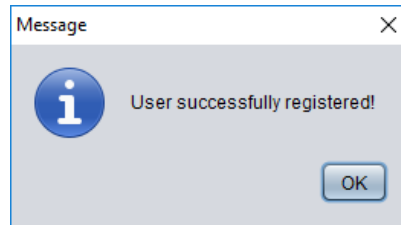


Fig. 4. User Registration form

3. CMS

- **Deduplication:** When a file upload request is sent from client to CMS, this module will check for duplication of file based on extension and size of file. If any file which is already available in storage nodes matches with extension and size of file to be uploaded, that file will be matched for duplication. If the file size is less than 100 bytes, whole content will be matched else first and last 50 bytes will be matched. If the file already exist, upload is cancelled and a link will be created to client for downloading existing file.
- **Find Rank:** When a new file is uploaded by client or existing file is downloaded, frequency for that file is updated and rank will be generated for the file based on the frequency of access. Whichever file has more frequency will have 1st rank and vice versa.
- **Update Frequency:** Whenever a client uploads a file or downloads existing file, count will be updated for the file. This count is called as frequency. It is used for finding rank for file.
- **Encryption:** All the files which will be uploaded to storage nodes are encrypted for security purpose. We use DES encryption algorithm for encryption files. It uses 64 bit key.
- **Decryption:** When a client downloads file or a file has to be matched for duplication, that file will be decrypted.

- *Compression:* To reduce the space to be used by file in storage nodes, the files will be compressed and stored. We use zip compression scheme for compressing files.
- *Decompression:* When a client downloads file or a file has to be matched for duplication, that file will be decompressed and decrypted.
- *Upload:* When a client sends upload request to CMS, the file will be checked for duplication, then encrypted and compressed and finally it is uploaded in storage node. The storage node which has max memory will be selected by CMS for file upload.
- *Download:* When a client sends download request, CMS finds the storage node which contains file and downloads file from it. Then the file is decompressed and decrypted and sent to client.
- *Delete:* When client sends delete request for a file, CMS checks if the file is used by more than one client. If so, the link for the file which is created for the client will be reused. Hence file will be available for other users. If file is not used by other clients, then the file will be deleted from storage node.

Fig. 5. After user registration

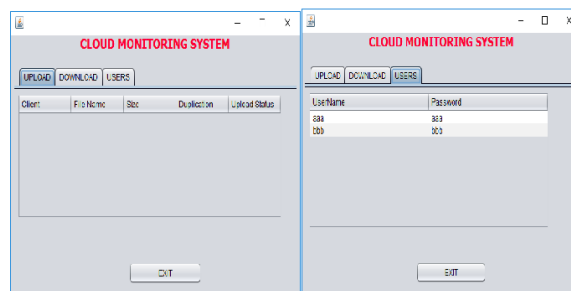
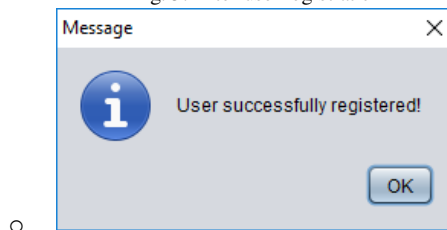


Fig.6. CMS Before and after Registration

4. Storage Node

- *Upload:* When storage node gets upload request from CMS, it saves the file and calculates available memory and used memory and sends reply back to CMS.
- *Download:* When a download request is sent to storage node, it sends the file content to CMS.
- *Delete:* In this module storage node deletes the file and updates its available memory and used memory.

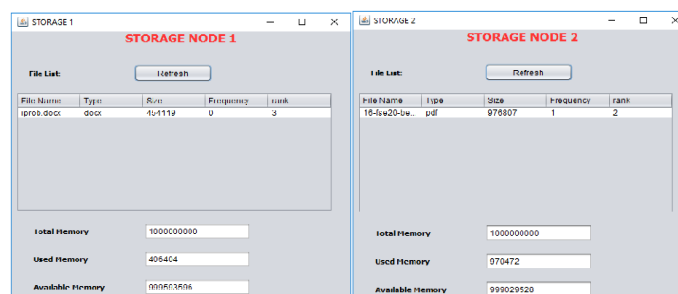


Fig.7. File storage

Updating the frequency and rank correctly for the various varieties of files and also displaying the available memory.

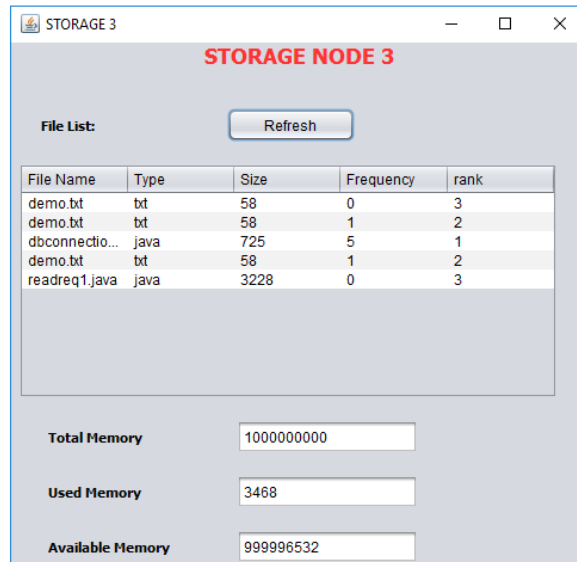


Fig. 6. Updating Frequency and Rank

V. RESULT ANALYSIS

When the client sends upload request to CMS, it check the duplication of file and also allocates a storage node having highest memory. Every time when the file is accessed by user, frequency for that file is updated and rank is updated for all files



Fig.8. Deduplication of files.



File details

| fname | ftype | fsize | frequency | rank | storage | lastaccess |
|-------------------|-------|---------|-----------|------|----------|---------------------|
| dbconnection.java | java | 725 | 3 | 2 | STORAGE1 | 19-05-2021 18:17:55 |
| a.doc | doc | 8 | 4 | 1 | STORAGE2 | 19-05-2021 18:35:53 |
| certificate.pdf | pdf | 323,403 | 2 | 3 | STORAGE3 | 29-06-2021 12:01:06 |

Share details

| uname | fname |
|-------|-------------------|
| aaa | dbconnection.java |
| bbb | dbconnection.java |
| aaa | a.doc |
| aaa | certificate.pdf |
| bbb | certificate.pdf |

VI. CONCLUSION

Hence based on file access pattern files will be displayed to user based on rank. User can also see last access time details to know whether user requires the file for future or not. If file is not required, user can delete the file and free up memory in storage nodes. For file security, every file is encrypted by CMS before storing in storage nodes. And also file compression is used to save storage memory.

When user downloads a file, CMS encrypts the file using a OTP and OTP is sent to mailed of user. Hence user can get his OTP and decrypt downloaded file. When user downloads a file, CMS encrypts the file using a OTP and OTP is sent to mailed of user. Hence user can get his OTP and decrypt downloaded file.

REFERENCES

[1] A. Rajalakshmi, D. Vijayakumar, Dr. K. G. Srinivasagan, An Improved Dynamic Data Replica Selection and Placement in Hybrid Cloud, International Journal of Innovative Research in Science, Engineering and Technology, Volume 3, Special Issue 3, March 2014.

[2] Ali Yadavar Nikravesh, Samuel A. Ajila* and Chung-Horng Lung "An autonomic prediction suite for cloud resource provisioning" Nikravesh et al. Journal of Cloud Computing Advances, systems and Applications, December 2017.

[3] A. Augustus Devarajan and Dr. T. Sudalai Muthu, "Cloud Storage Monitoring System analysing through File Access Pattern", International Journal of Engineering & Technology (IJET), Vol 7 No 3.32 (2018): Special Issue 32, October 2018.

[4] Jonathan L. Krein, Lutz Prechelt "Multi-Site Joint Replication of a Design Patterns Experiment using Moderator Variables to Generalize across Contexts" IEEE Transactions On Software Engineering, Vol. X, No. X, April 2016.

[5] M. Du and F. Li, "ATOM: Efficient Tracking, Monitoring, and Orchestration of Cloud Resources", IEEE Transactions on Parallel & Distributed Systems, Vol. 28, No. 8, pp. 2172-2189, April 2018.

[6] Masoud Saeida, Ardekani, Douglas B. Terry, A Self-Configurable Geo-Replicated Cloud Storage Systems, 11th USENIX Symposium on Operating System Design and Implementation (OSDI' 14), pp367-381, October 2014.

[7] Navneet Kaur Gill and Sarbjeet Singh, Dynamic Cost-Aware Rereplication and Rebalancing Strategy in Cloud System, © Springer International Publishing Switzerland 2015 S.C. Satapathy et al. (eds.), Proc. of the 3rd Int. Conf. on Front. of Intell. Computer. (FICTA) 2014 – Vol. 2, Advances in Intelligent Systems and Computing



INNO  **SPACE**
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542



ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

 **9940 572 462**  **6381 907 438**  **ijircce@gmail.com**



www.ijircce.com

Scan to save the contact details