An Innovative Secure Dynamic Access using RMDS in Cloud Computing

G Jeeva Rathanam*, V Balakumar
UG Students, Department of Information Technology,
*jrsuji@yahoo.com

***Abstract*-In the heterogeneous databases high amount of data are being stored and transferred across the globe. Data stored in the databases are exposed to threats and attacks. Data loss during transmission and corruption are also to be expected. In order to provide a sustainable storage environment handling such limitations, systems should be reliable and fault tolerant. In the proposed research work RMDS, an innovative mechanism which provides secure data storage with recovery mechanisms during faults is developed. Distributed data storage is employed using data partitioning technique. Diagnostic support is rendered by Recovery Oriented computing which recovers the failed process that may include multiple dependent processes. RMDS outperforms when compared with existing system providing fault tolerance and data security in cloud environment.**

***Keywords*- Fault tolerance, Secure partitioning, Recovery mechanisms.**

I. Introduction

High speed internet services are readily available to users nowadays. This paves way for the emergence of advanced computing technologies. Cloud computing has gained immense popularity in recent years as it enables users to utilize all services located in any remote environment. Among the services offered by cloud service providers, cloud storage is widely used by users all over the world. An overview of the cloud storage is shown in Fig [1].

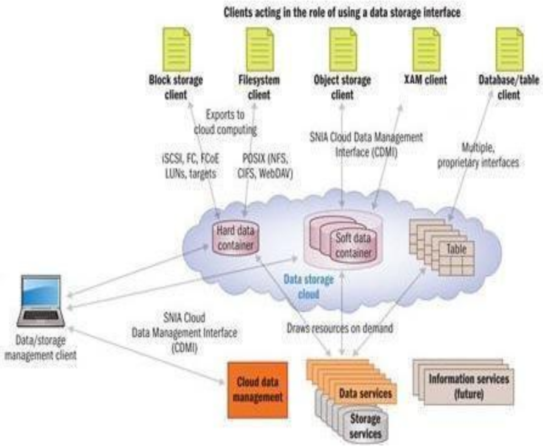
Clients acting in the role of using a data storage interface
Block storage client
Filesystem client
Object storage client
XAM client
Database/table client
Exports to cloud computing
iSCSI, FC, FCoE LUNs, targets
POSIX (NFS, CIFS, WebDAV)
SNIA Cloud Data Management Interface (CDMI)
Multiple, proprietary interfaces
Hard data container
Soft data container
Table
Data storage cloud
Data/storage management client
SNIA Cloud Data Management Interface (CDMI)
Draws resources on demand
Cloud data management
Data services
Storage services
Information services (future)

Fig.1. Overview of Cloud Storage System

In order to store the enormous amount of data, cloud storage systems use many small-scale independent storage systems. These systems together form the entire cloud storage. Using cloud storage to store the data has multiple advantages. Few of them are, data stored using an account can be synced in multiple devices using the same account. When the user is working, less storage space is needed on devices and avoiding storage of multiple conflicting replicas etc.

Data stored in the cloud therefore has to be reliable and secure. To increase reliability, remote data integrity checking protocols are used which can find corrupted data or faulty servers in the storage system. On the other hand, to increase security and scalability, fault tolerance has to be provided. In order to make a system dependable, incorporating fault tolerance is mandatory as shown in Fig [2]. For storage servers, such systems are necessary as large amount of data handling is done and at the same time a great number of end users are simultaneously seeking service.

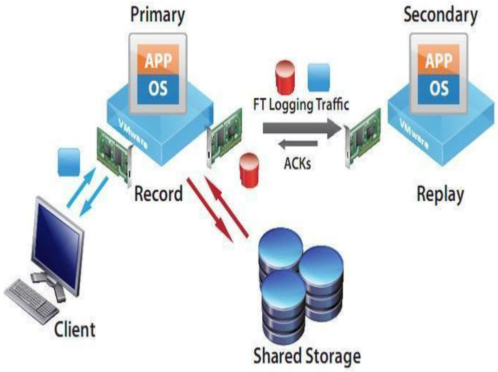

Fig.2. Fault Tolerance in Cloud

The proposed system thus intends to introduce a fault tolerant cloud storage system named RMDS. Partitioning algorithm is used to provide an effective storage system that provides high availability. Security mechanisms are also included in order to prevent unauthorized access of private data. Storage and retrieval of data is made easier since the proctor unit continuously monitors the system for inconsistencies. Replication is done as needed to produce replicas of the stored data to account for inevitable data loss. A diagnostic support is introduced which ensures that the process initiating storage and retrieval are either stopped if faulty or isolated in order to prevent other dependent processes from behaving incoherently.

As organized the rest of the paper is follow: Section II presents the related work in the respective field of research. Section III gives the detailed proposed architecture of the work. Section IV presents the implementation issues of RMDS. Section V provides the performance analysis of the proposed system in comparison to the existing work. Section VI finally provides the conclusion and the future works.

II. RELATED WORK

To accomplish the competent data dynamics, the present evidence of storage framework is enhanced by influencing classic Merkle Hash Tree construction for validation. The concept of bilinear aggregate signature has been investigated and proved that they are more securable [7]. Multiple auditing tasks were not efficiently handled by this approach. The proposed scheme design supports confident and well organized active process on outsourced data. This investigation demonstrates the future scheme is highly proficient and flexible against Byzantine failure, malicious data modification attack, and even server colluding attacks [3]. But the system takes longer time to recover.

In order to reduce the data management costs, third party can use the outsourced data [5]. To achieve this, one should offer security assurance for the outsourced data. The FADE which is the secure cover cloud storage system was implemented which accomplish fine-grained, policy-based access control and file assured deletion. The extensive experimental studies were conducted; it revealed that FADE offers security protection for outsourced data, while establishing only minimum performance and economic cost overhead. This work gives insights of how to integrate value-added security characteristics into today’s cloud storage services.

A distributed fault tolerant control technique is presented for interrelated nonlinear uncertain structure. Depending on the local state information, linearly parameterized neural networks are used to adjust the unidentified interconnections, fault functions and communicated information from other subsystems. The strength of the distributed fault tolerant control structure is recognized during a rigorous Lyapunov analysis [1].

In the recovery algorithm for Delta enabled grid services, data dependencies examines the dependent failed process by using database log files which are produced through database transaction. The algorithm guarantees contingencies among parallel processes and recovery by supporting contingency, rollback and compensation among grouped processes [6]. Support for the methods to find attacks in legacy systems are not taken into account. Partitioning and encryption methodology are not used in this system to provide more security and integrity. The fault tolerant flight control scheme plan methodology is proposed which is against the control surface impairments and represented as a polytopic linear parameter varying (LPV) system. Replication examples focus to inner impairments which is used to show the effectiveness of the proposed method.[4]

This system is enhanced in the proposed work by incorporating recovery methods. Experiment conducted response that the system can be acquired for transition faults. It also illustrates that the analytical resolution of this scheme is almost the same as that attained by monitoring all output replies.[2] Diagnostic support further makes the system more efficient.

From the survey taken, it is found out that systems with efficient and secure storage mechanisms lacked recovery operations and thus the result obtained had increased time taken to recover. The proposed work therefore takes into account all the possibilities of failure and provides techniques to make the system more reliable.

III. RECOVERY MECHANISMS IN DATA STORAGE

While considering cloud storage, there are millions of users who access it both locally and remotely. In order to provide consistent data, integrity checks have to be performed periodically. Partitioning method ensures that corruption free data is accessed by the users. At the same time, downtime in any system will render the service unavailable.

To overcome this issue, the recovery framework is used along with the storage service which provides fault tolerance components that aids in uninterrupted data transfer and storage services.

A. Partitioning data and storage in cloud

Partitioning checks the correctness of data stored using pre-computation which is done before data is storage and this increases the security. Encryption and Decryption using keys are done which further enhances security. Maintaining the integrity of data is mandatory in a cloud storage system. As per the user needs or requests, data can be retrieved from the storage system. The user has the ability to decide upon the type of data access and users of a particular data.

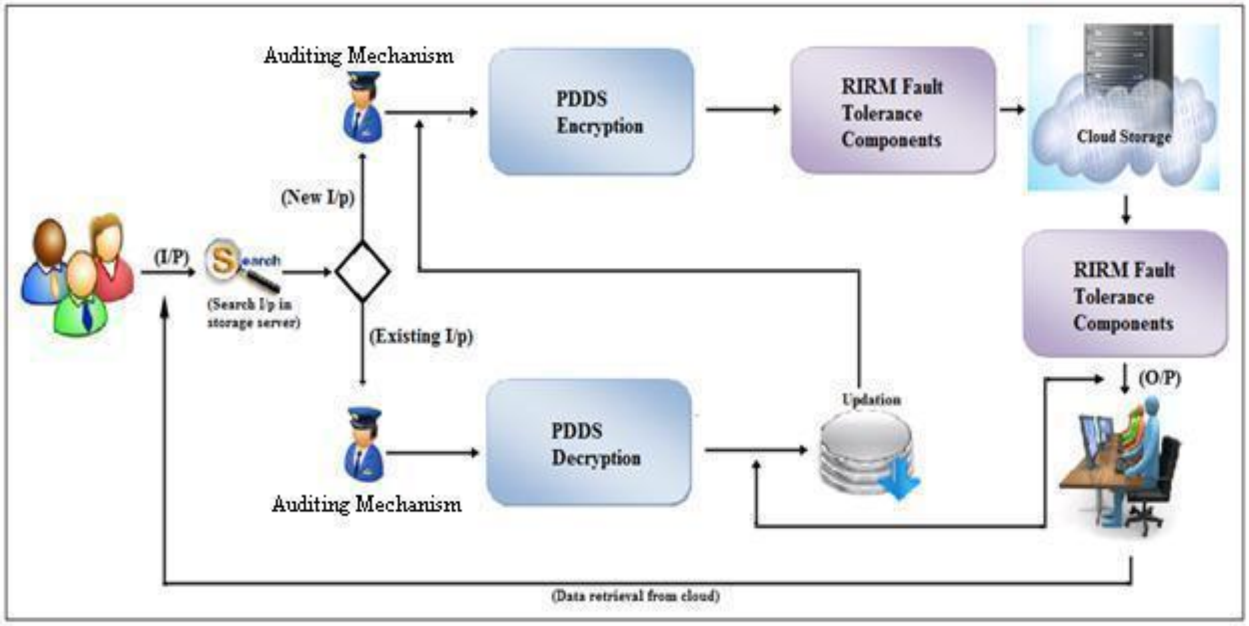

Fig.3. Architecture of RMDS

Partitioning system enhances the performance of storage service by providing security during storage and retrieval. Data is first partitioned into smaller sized blocks and encrypted during storage. The partitioning function reduces the overhead of maintaining complex data since they are split up into blocks for easier storage. During data retrieval, the data is decrypted and the smaller blocks are now merged to form the original data which is then accessed by the user. Threats which may occur during this process are avoided using remote data integrity checking. Storage costs are decreased and performance is increased by employing dynamic operations. Thus data retrieval and storage is done optimally using the partitioning technique and necessary security mechanisms are employed. But failure of any one component may result in loss of data. Therefore fault tolerant mechanisms are introduced in this storage service which effectively takes care of recovering back the failed transaction.

B. Ranking decisions

Ranking of processes involved in cloud storage is done. This is carried out by first analyzing the dependent processes and generating a connectivity graph. Thus the rank is decided after evaluating the graph. The decisions made help is providing ranks to various read/write, insert/delete/update processes taking place within the storage system. Furthermore, the processes are executed in parallel which reduces the time taken and enhances system utilization. Failure of one process is thus handled effectively by uninterrupted execution using other processes.

C. Monitoring and recovery support

Recovery method uses a monitoring unit called proctor for continuous monitoring of all the processes. It checks if there are any failures occurring and decides whether to isolate that failure or continue execution using replicas. Isolation occurs if the faulty process will cause undesirable changes in the system whereas recovery is initiated if the degree of fault that occurred is less. Diagnostic support is also provided in addition for the recovery of a process. The root cause of failure is analyzed in detail to protect other connected processes and to recover the failed process.

IV. IMPLEMENTATION ISSUES

In this section we present the implementation issues of RMDS. The partition is done before the encryption of a file to transform the large file into smaller partitioned segments. Recovery method is applied to enhance the fault tolerance. The fault tolerant execution is done by analyzing the Process Connectivity Graph which is generated by examining dependencies between the processes based on the log files in delta based cloud services. After executing recovery mechanisms the file is stored in cloud storage. It provides flexible and efficient way of storing data with high security by employing encryption and decryption methodology and partitioning technique from RMDS. It also emphasizes the secure approach of a file by dynamic operation of storing and integrity checking for remote data to guarantee that the file is transmitted in an unharmed fashion .

Algorithm 1: Partitioning and merging files:

This algorithm partitions the larger file into smaller portions and merges in order to ensure easy and efficient processing of files. It finally merges the partitioned portions of a file into original file and integrity checking is done after merging to ensure whether the file is properly processed.

1. Load the Input file and size.
2. Partitioning files: Count size <= s then split file in to n blocks with extension and index value.

   Return files, otherwise declare as Invalid size.
3. Encrypt all partition files and store in cloud.
4. Merging files: check if(file!) then file is missing.

   Otherwise count the index value and merge files.

   Return file.
5. Decrypt the merge file for access.

The main aim of this algorithm is to provide better secure access of a file using partitioning method. It first gets the file from the disk and checks the size of the file is within the count / threshold value. If it is not, then the entire file is detached into several segments and encrypted. After encryption, the corresponding divisions of the file are merged and transmitted through the untrusted medium of communication. The file is then stored into delta oriented cloud systems in which fault tolerance execution is enabled. Finally, the file will be decrypted whenever any request demanding the data file appears.

Algorithm 2: Encryption:

Encryption algorithm focuses on encrypting the segments which are divided by partitioning algorithm. It uses RSA algorithm with random public key generation techniques. This algorithm stores the public and private key rings in the delta oriented cloud. It first creates cipher and key generator modules. The key generated consists of 2048 bits. It uses private key and public key using symmetric and asymmetric encryption techniques respectively. It produces the partitioned and sealed Objects.

1. Create a Cipher object and Key Generator object.
2. Create a Secret (session) key using cipher object.
3. Initialize it with session key. And encrypt the files.
4. Get recipient's public key and Create Cipher and initialize it for encryption with recipient's public key.
5. Create Sealed Object to seal session key using asymmetric Cipher and Serialize Sealed Object.
6. Return the encrypted files and serialized Sealed Object to recipient.

Algorithm 3: Decryption:

The main focus of this algorithm is to unseal the object and finally decrypt the segments of data and to form the original file. It ensures file access control by using asymmetric cryptosystem in which each individual user will have separate private key to access the data from the delta oriented cloud systems. It re-serializes the encrypted sealed message object and finally unseals the segments to decrypt using private key which is stored in the delta enabled cloud.

After the partitioning is completed, recovery methods are applied in the Delta enabled cloud services in order to bestow high fault tolerant processing in the cloud. It incorporates the concept of ROC (Recovery Oriented Computing) along with replicating techniques to provide effective fault tolerance execution of a process. It examines the transactions carried out in the cloud service and finally uses the decision making module to find whether to apply ROC techniques or isolation along with redundancy scheme to furnish the best fault tolerant system.

1. Get encrypted message and serialized Sealed Object. Re-serialize Sealed Object.

3. Create Cipher object, and initialize it for decryption and generate private key.

4. Unseal the key using the asymmetric Cipher.

5. Create Cipher object and Initialize it with the recovered session key for decryption.

Algorithm 4: Process Connectivity:

1. Initialize a variable N as the total no of processes.
2. Using Loops check and obtain all the processes' connectivity.
3. Outcomes written in the database.

This algorithm analyzes transaction logs of cloud services to find the connectivity or dependency of a process with another. After gathering all the values, it saves the outcomes in the database as a graph of 2D array. Since, it uses partition method; it ensures the high level security of large data since it is stored by partitioned method. The recovery oriented algorithm, isolation and redundancy methods ensures the fault tolerance execution of tasks in the delta enabled cloud services. Hence, the performance and security is maintained in the above approach effectively.

Algorithm 5: Find Faulty Process:

1. Find the Total number of processes.
2. Obtain the message values of processes from systems using loops.
3. If message < 0 then Increment count1 and store the message in array (P).

   Else Increment count2 and store the message in array (Q).

   If count1 < count 2 then print array P

   Else print array Q.

It checks the nature of the message from one process to another. It also maintains the counter value to increment if the matched type is found again. It differentiates each and every message from one process to another by upholding the identity values of each process.

V. PERFORMANCE ANALYSIS

The performance of RMDS is shown in Fig [4]. It is analyzed that the RMDS method is coupled with fault tolerance techniques provides a better storage facility which outperforms other existing systems. The processing time and the recovery process are improved in the proposed system.

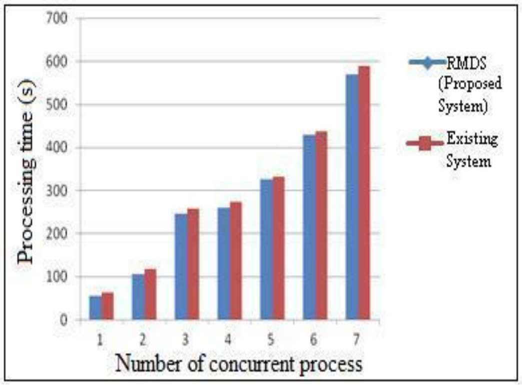

Fig.4. Performance analysis of RMDS

VI. CONCLUSION

The data partitioning method along with diagnostic support and isolation techniques helps in better storage and fault tolerant process execution in the delta oriented cloud systems. Effective system utilization of resources is obtained by Recovery oriented computing and it also incorporates partitioning method inorder to provide high integrity and secured data transfer from client to server. RMDS reduces the average processing time taken for execution and increases the system throughput by proper utilization of the system using proctor module. Proctor module examines the entire system scenario and takes decision in an intelligent manner providing highly fault tolerant system.

REFERENCES

[1] Panagiotis Panagi and Marios M. Polycarpou ,"A Coordinated Communication Scheme for Distributed Fault Tolerant Control", 2013 ,IEEE Transactions on industrial informatics , pp 386-394.

[2] Wei-Cheng Lien, Kuen-Jong Lee, Tong-Yu Hsieh, Krishnendu Chakrabarty, and Yu-Hua Wu , "Counter-Based Output Selection for Test Response Compaction ",2013 , IEEE Transactions on computer-aided design of integrated circuits and systems , pp 152-164.

[3] Cong Wang, Qian Wang, Kui Ren, Ning Cao, and Wenjing Lou ,"Toward Secure and Dependable Storage Services in Cloud Computing" 2012 , IEEE Transaction on services computing , pp 220-232.

[4] XIANG YU, JIN JIANG, "Fault-Tolerant Flight Control System Design Against Control Surface Impairments",2012 , IEEE Transactions on aerospace and electronic systems , pp 1031-1050.

[5] Yang Tang, Patrick P.C. Lee, John C.S. Lui, and Radia Perlman,"Secure Overlay Cloud Storage with Access Control and Assured Deletion", 2012 ,IEEE Transactions on dependable and secure computing , pp 903-916.

[6] Yang Xiao and Susan D , "Using Rules and Data Dependencies for the Recovery of Concurrent Processes in a Service-Oriented Environment", 2012 ,IEEE Transactions on services computing , pp 59-71.

[7] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li,"Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", 2011 IEEE Transaction s on parallel and distributed systems, pp 847-859.