IJITCE

ISSN 2347- 3657

International Journal of Information Technology & Computer Engineering

Email : ijitce.editor@gmail.com or editor@ijitce.com

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 12 , Issue 2, April 2024

DATA CONSISTENCY IN MULTI CLOUD STORAGE SYSTEM WITH PASSIVE SERVERS AND NON COMMUNICATING CLOUDS

[1]G.MAHAMMADH IDRUSH , [2]P.SRIVANI, [3]P.NAVEEN,[4] P.INDRAKARAN

[1]Assistant Professor in Department of CSE Teegala Krishna Reddy Engineering College

[2,3,4,] UG Scholars in Department of CSE Teegala Krishna Reddy Engineering College

Abstract

Multi-cloud storage systems are becoming more popular due to the ever-expanding amount of consumer data. This growth is accompanied by increasing concerns regarding security, privacy, and reliability of cloud storage solutions. Ensuring data consistency in such systems is especially challenging due to their architecture and characteristics. Furthermore, the atomicity of operations is not always guaranteed by the clouds' public APIs. In this project, we formally define data consistency in multi-cloud storage systems, identify how they can be violated, and introduce a new method that provably maintains the data consistency in these systems. The implementation and experiments show that the proposed method can maintain data consistency with a certain delay in data uploading and that it is scalable with respect to the number of used clouds as well as the number of users. Integrating this method into multi-cloud storage systems will enhance their usability and reliability

I INTRODUCTION

With the ever-growing amounts of data, users are shifting towards cloud storage services. These services provide convenience as they omit the need for maintaining local storage means and provide accessibility from anywhere and across different devices. However, outsourcing data to the cloud comes with data confidentiality and integrity critical requirements. Relying on a single cloud storage provider fails to meet these requirements due to the inevitable risks of privacy breaches, data leaks, and service outages. To tackle this issue, various multi-cloud storage systems (also known as a cloud of clouds) have been proposed in the literature. Most of these systems partition the data into multiple parts,

encode these parts using erasure codes, encrypt them, and finally, save each part on a different cloud provider. When such systems are server-less (i.e., partitioning, encoding, and encryption operations occur on clients' machines rather than on a centralized server), they can offer privacy, security, and protection from data loss. Privacy is guaranteed since individual cloud providers will have no knowledge about the content of the data as they store only an

ISSN 2347–3657
Volume 12 , Issue 2, April 2024

encrypted part. The data is also secured since its integrity is preserved (since modifications will lead to detectable data corruption). Finally, reliability is provided through erasure codes since even if part of the data is unavailable (e.g., due to an inaccessible cloud), the original data can still be reconstructed/decoded from other available parts. In general, server-less multi-cloud storage systems can provide trust in the cloud. There are several important use-cases of multi-cloud storage systems for both end-users as well as businesses. End users can store personal files that are hidden from the cloud and not subject to potential denial of access. This is becoming more important due to multiple recently reported privacy-leak incidents, which caused many cloud-end users to opt-out of using cloud services. Similarly, multiple businesses would also like to have guaranteed privacy and availability of their sensitive data. This is especially important for businesses since they are subject to different jurisdictions and potential subpoena-forced data acquisition or service denial, depending on the cloud data center location. Multi-cloud storage systems, similar to cloud storage services, should allow users to access and modify their files from anywhere. Furthermore, users can access their data from multiple independent client devices. Therefore, data should always be synchronized and consistent across all users' devices. One of the fundamental synchronization features is the ability to detect data conflicts and maintain data consistency. In general, data conflicts occur when multiple clients attempt to modify the same file at the same time. Data consistency assures that no information is lost in such a case

Multi-cloud storage systems can detect conflicts and preserve consistency through utilizing

a centralized coordination point (e.g., server) that receives and logs the modification requests from the different clients (append-log). Specialized software can parse the logs and determines the existence of a conflict. However, secure multi-cloud storage systems are server-less. Hence, there is no central controller to coordinate between clients and detect data conflicts. This is of utmost importance since users should not need to trust any third party to handle their raw data.

II LITERATURE SURVEY

Various multi-cloud systems have been proposed in the literature. Cloud-RAID,NCCloud, RAIN, RAIC, Hyprid CoC, and Uni4Cloud systems leverage multiple clouds to address the aforementioned cloud trust issue. These systems do not support multiple clients and end-devices and are not prone to concurrent access issues. Thus, the data consistency issue is not considered. Other works like Spy storage and Trusty drive support multiclient access. However, the data consistency issue is either

ISSN 2347–3657

Volume 12 , Issue 2, April 2024

relayed to separate centralized service or is not addressed.Hybris is a multi-cloud storage protocol. It supports multiple writers consistency. However, inter-client and cloud communication is necessary. This required an extra layer to perform such communication, which is Apache Zoo Keeper (ZK). Depending on the configuration, ZK might form a single point of failure since all clients rely on this server to

communicate and coordinate updates among each other. Besides, such architecture still requires clients to trust a third party (Apache ZK) while using a multi-cloud storage system. SLA provides two tree and token-based distributed mutual exclusion algorithms that can be used for multi-cloud storage, but it also requires inter-client communication. Meta Sync provides a file synchronization service on top of cloud storage providers. It employs a modified version of the Paxos consensus algorithm called passive Paxos (pPaxos). This modified version allows clients to communicate passively (through files) over the clouds in order to reach consensus. This modification requires an append-only atomic list to keep track of protocol messages and eventually reach a consensus. Cloud-types proposed specialized data types that guarantee eventual consistency to all clients. A program that utilizes these data types is abstracted from synchronization complexities and can automatically synchronize data through fork-join techniques. The implementation requires communication between servers. Thus, such implementation is not suitable for a multi-cloud storage system where servers are completely passive.

Saveme is a multi-cloud storage system that proposed a mutual exclusion method for concurrent data access without the need for a central server or any communication between clouds or between clients. This method requires atomic operations that cannot be interrupted. The authors surveyed different APIs from several cloud providers to identify some atomic operations and mapped the unlock/lock operations to other atomic operations offered by the cloud providers. While the proposed system successfully addresses the concurrent access issue, it might be unreliable since these operations are not guaranteed to stay atomic by the cloud providers. Also, the implementation is more complicated since each provider offers different atomic operations. For example, placing a lock might be mapped to moving a file in cloud, whereas in cloud B, it might be mapped to adding a comment to a file. The deadlock recovery method is based on a fixed amount of time that is experimentally determined (deadline),which is not suitable for all network connections and speeds. Lastly, the method selects a cloud out of the used ones to be used for locking/unlocking operations. However, the selected cloud might

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 12 , Issue 2, April 2024

not be available to all users at the same time. In such cases, the mutual exclusion will not be sufficient. Summarizes the main approaches to achieve the data consistency feature in multi-cloud storage systems that provide such a feature. The log-parsing approach assumes a central entity that can receive and coordinate between all clients to guarantee conflict detection and resolution. Such an entity constitutes a single point of failure that jeopardizes reliability. This also creates a performance hot spot as all coordination tasks are performed on a single node. On the other hand, in the distributed systems literature, the concept of consensus has been developed and used extensively in cases where distributed entities need to agree on a single value (or plan) and is used in multi-cloud storage systems. However, utilizing consensus protocols (e.g., Paxos and its variants), mandates inter client or inter-server communication (active consensus). Nonetheless, there has been a line of work that attempts to map the consensus protocol to read/write operations (i.e., passive consensus). For example, the Paxos proposal phase is replaced with file write. Then, based on reading the written files, a proposal can be accepted or rejected. Passive consensus eliminates the need for communication but requires atomic operations to be provided by the cloud API.

III EXISTING SYSTEM

An additional and essential challenge that faces multi-cloud storage systems is the heterogeneity of consistency models followed by different providers. Having a strict consistency assumption or atomicity of operations from a cloud storage provider is an impractical assumption that should be avoided. Under such a model, any read/write sequence results cannot always be guaranteed to return the same results. Nonetheless, a reliable multi-cloud storage system should provide an application-level mechanism that ensures data consistency despite the lack of atomic operations or consistency guarantees at the individual cloud level.

Limitations

- Client-side only.
- Lack of communication between clients.
- Lack of processing resources on passive storage servers

IV PROPOSED SYSTEM

we investigate multi-cloud storage systems and propose an application-level client-centric consistency method that provably detects data conflicts and resolves them. Such a consistency feature will enhance the usability of multi-cloud storage systems and hence contribute to the establishment of private, secure, and reliable storage to end users.The contributions of this project are summarized as follows:

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

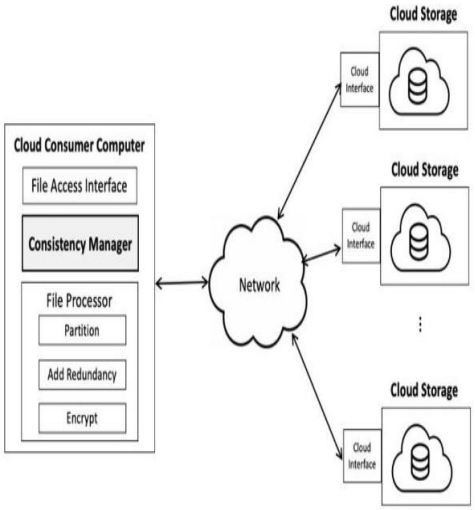
Volume 12 , Issue 2, April 2024

➢ Formally defining the data consistency problem in the context of multi-cloud storage systems

➢ proposing a novel method that guarantees eventual data consistency in multi-cloud storage systems with passive servers and non-communicating clients

➢ implementing a multi-cloud storage system that utilizes the proposed method to demonstrate its performance empirically.

V ARCHITECTURE



System Architecture

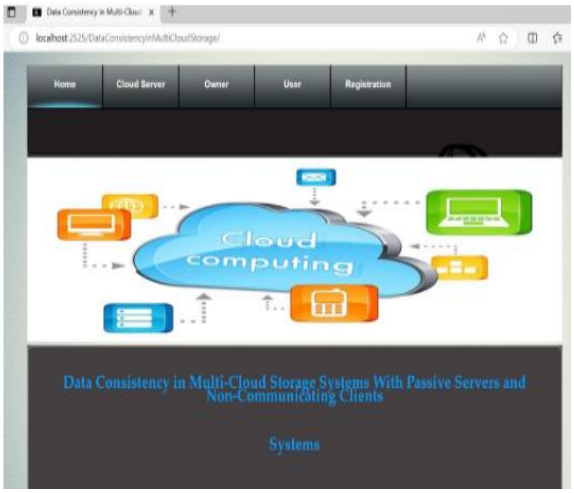
VI IMPLEMENTATION

User Module: User can register, Key processing and File search. He user will decrypt the Data file and Download the file process will be done.

Cloud Module: In this module, Cloud will be maintaining the Data in multiple clouds and that data is Encrypted.

Data Owner Module: In this module, Data Owner will register, file upload and modify. Data Owner will be create Encrypted key for data file and if owner have that file decrypt and download the data file.

VII RESULTS



HomeScreen

Registration Page

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

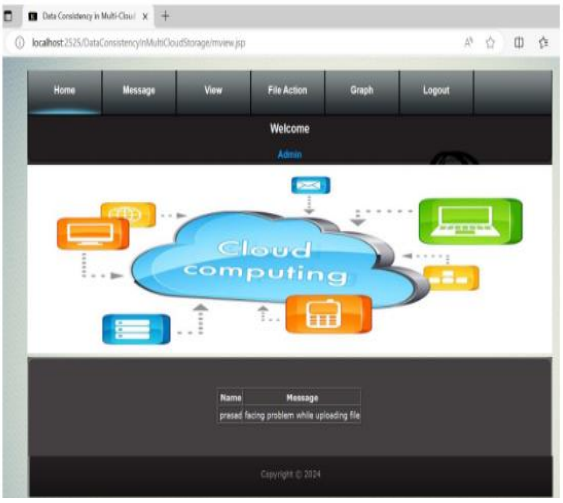
Volume 12 , Issue 2, April 2024

Owner Login Page

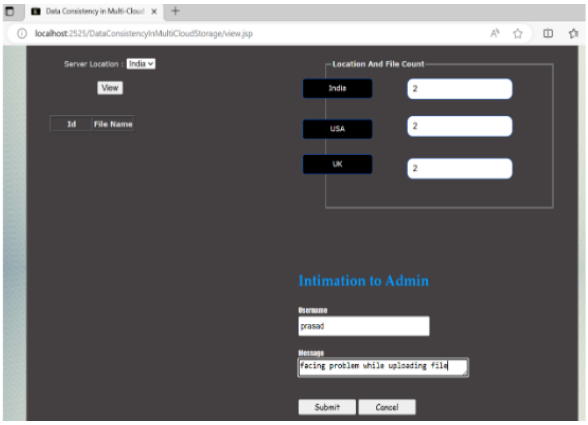
View page

Download page

Upload Page



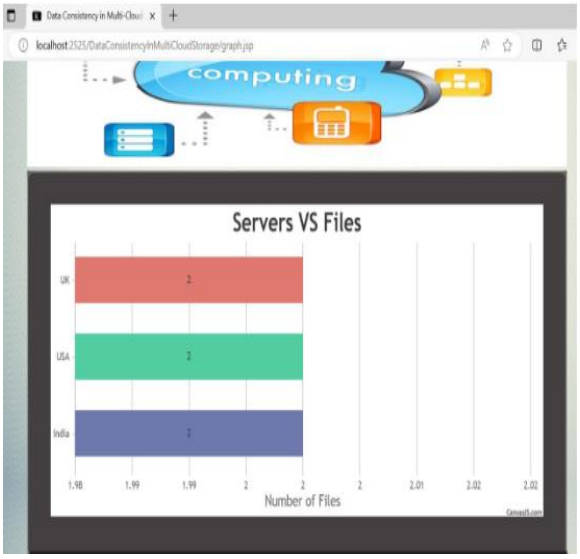
Message Page



International Journal of Information Technology & Computer Engineering

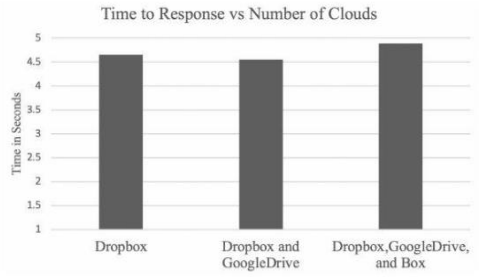
ISSN 2347–3657

Volume 12 , Issue 2, April 2024



Server and Files

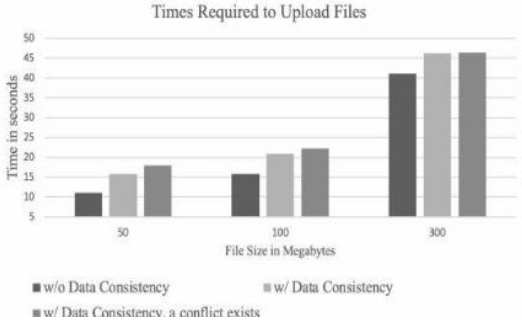
VIII PERFORMANCE EVALUATION AND ANALYSIS



(a) TTR vs. number of used cloud storage providers

Comparison between the case when the data consistency module is not activated versus the case when it is activated. Fig. 6 shows actual times for uploading files in the designed multi-cloud storage system with and without the proposed data consistency method. These times include the partitioning, encoding, and encryption of the data parts. As shown in the figure, the overhead added by the data consistency algorithm is roughly 5 seconds, including the TTR and the time required for temporary file renaming (third phase). The additional 1 second that exists in the case of conflict is mainly due to the conflict handling mechanism (copying data to the new conflicted copy) of our system, and not due to the algorithm itself since, as shown earlier, the TTR is the same whether there is a conflict or not.

Reference	**Upload overhead** (seconds)
MetaSync [25]	≈ 5.1
SaveMe [28]	$\approx 1 - 20$ (Depending on the deadlock deadline)
This work	≈ 4.7



IX CONCLUSION

we discussed multi-cloud storage systems and their significant advantages in establishing trust in the cloud. The paper focuses on addressing the concurrent data access issue and reasons why it is especially challenging in such systems compared to conventional storage systems. Various previous solutions to this issue were discussed. The paper offered a useful formal definition of data consistency and data conflicts in a multi-cloud storage system and proposed a novel method to detect data conflicts and

ISSN 2347–3657

Volume 12 , Issue 2, April 2024

maintain data consistency. The method: Does not require inter-client or inter-server communication. Avoids the reliability and security issues associated with a single point of failure as it does not require the help of any server (a server-less system that runs fully on the client's machine). Utilizes passive cloud storage services. Is scalable with respect to the number of clouds. Is scalable with respect to the number of users. Works as long as users of Multi-cloud storage system overlap in using at least one cloud. Experimental results on real cloud systems show an API calls-delay of approximately seconds before uploading data to the multiple clouds. The proposed algorithm is best suited for a multi-cloud storage system that requires data consistency while also being scalable and tolerant to different cloud failures. Future work might consider utilizing local caching or API call optimization to minimize the delay