

## **A Review on Comprehensive File Synchronization Algorithms for Effortless Local and Cloud File Management with Conflict Resolution**

<sup>1</sup>Mr. Sandip Dabre, <sup>2</sup>Prof. Sachin Vyawahare, <sup>3</sup>Prof. P. P. Rane

<sup>1</sup>Student, CSE Department, Rajarshi Shahu College of Engineering, Buldhana

<sup>2</sup>Assistant Professor, CSE Department, Sanmati College of Engineering, Washim

<sup>3</sup>HOD, CSE Department, Rajarshi Shahu College of Engineering, Buldhana

<sup>1</sup>sandip.dabre87@gmail.com, <sup>2</sup>vyawahare01@gmail.com, <sup>3</sup>koltepallavi200@gmail.com

### **Abstract:**

The majority of application data and software in the context of cloud computing are moved to the cloud computing data center and network service provider; all application and information management and maintenance tasks are left to the cloud service providers. While cloud computing offers ease, it also presents a number of security risks. This paper first examines the design of cloud computing uses within the organization. It then discusses and studies virtualization, cloud computing, large-scale, dynamic, and extensible cloud computing, based on the security of cloud computing challenges, summarizes the benefits, and proposes a dynamic hash authorization scheme. The empowerment management system is shown with the user aspect as the letter and management technologies as the side. Pay attention to data encryption technology. Cloud storage security solutions come from data, design, and devaluation. Furthermore, we propose a combination of the cloud security standard and the evaluation system, improving the multi-dimensional cloud protection system and its implementation in light of the issues with the information security standard.

**Keywords:** Cloud Computing, Data Center, Extensible Authorization, Data encryption

### **INTRODUCTION**

Sharing digital files over a network is a common application of the networking technology. Files can be shared between: a) Users and machines (eg: when one downloads a file from a server, or uploads a file to a server), b) Machines (eg: automated backups), c) Different users (through machines: uploading the file to a server, from which the other party can download it; directly: using P2P file sharing services). It is common nowadays for users to share files across their own devices connected over a network using synchronization services such as Dropbox [1] or Google Drive [2]. They usually do so by allowing a user to upload their files from one device to central servers, and allowing other devices owned by the same user to download them from those servers. Note that the users also have the option to share their files with others, or make them public. Peer-to-peer (P2P) based synchronization systems split the files into chunks (or pieces), which are then replicated on a subset of peers. A cloud-based synchronization system (also a cloud-based storage service) is used to store users' files in a central server, owned and governed by a certain entity (eg: an enterprise, or a small company). Users upload their files to this server from one device, and download them on another (or on the same device, in

case the user loses the original file). Users can also share their files with others, and depending on the service provided, a cloud-based synchronization service can be extended to provide a collaboration platform to the users. These services are provided across many different platforms, using web as well as native application development technologies as their front-end. Some of them provide desktop applications that act as drives connected to the PC, to provide a seamless interaction with the actual cloud drive. These services usually employ a freemium model: a fixed amount of initial storage is given for free, with limited feature set, while allowing users to upgrade to a higher plan with more storage and additional features. A good comparison of some of the most popular cloud storage and synchronization services can be seen in [8]. Such a model makes cloud services much more accessible and convenient to the users.

#### **A. Objectives**

- To ensure that changes made to files are synchronized across all devices in real-time or near real-time, minimizing delays in accessing the latest version of a file.
- To implement mechanisms to detect that arise when the same file is edited on multiple devices simultaneously, ensuring that data integrity is maintained.
- To optimize the synchronization process, minimizing the amount of data transferred and reducing synchronization times.
- To access previous versions of files and open them if needed.
- To scale efficiently as the number of users and the time of conflict.

#### **B. Challenges of file synchronization**

File syncing often presents security concerns to enterprises whose employees use consumer-grade applications to access business files. In addition to creating "shadow IT," these applications may also contain vulnerabilities that threat actors can exploit to compromise the file and the organization. Another challenge with file syncing is loss of centralized control. Remote and hybrid teams often sync business files to their personal devices, especially if the firm has a "bring your own device" policy in place. If members of these teams leave the organization, the files will still be available on their devices. The company might not be able to control the files on the ex-employee's device, much less wipe them to protect enterprise data. Accidental moves and deletes are another challenge with file sync. To mitigate these concerns, it's important to use file sync software that enables file recovery. The software should also provide capabilities such as granular permissions, notifications and audit logs so managers can better control files as well as prevent accidental moves and deletes.

## I. LITERATURE SURVEY

**Csirmaz et. al.** states that synchronizing diverged copies of some data stored on a variety of devices and/or at different locations is an ubiquitous task. The last two decades saw a proliferation of practical and theoretical works addressing this problem. File synchronization is a feature usually included with backup software in order to make it easier to manage and recover data as and when required. File synchronization usually delivered through cloud services. Dedicated file synchronizing solutions frequently come with additional tools not just for managing the saved data, but also to allow for file sharing and collaboration with stored files and documents. These cloud storage services are easily accessible for the end-user because the service front-ends are very well integrated into web clients as well as desktop and mobile environments. Simple user interfaces hide the complex and sophisticated service back-ends. Collaboration services are frequently integrated into the “cloud storage” environment. For example, Google Docs is an application layer integrated into Google Drive storage, Office 365 is integrated with One Drive storage and Dropbox Paper service is an extension of

[1]

**Dürsch et. al.** states that an application that enable digitally conducting QDA are grouped as Computer assisted qualitative data analysis software. One representative of Computer assisted qualitative data analysis software (CAQDAS) is called QDAcity1. QDAcity is a cloud-based web application, developed and operated by the Professorship for Open Source Software at the Friedrich-Alexander-Universität Erlangen- Nürnberg. QDAcity provides an environment for multiple analysts or researchers to collaboratively conduct QDA. Since QDA deals with big amounts of fuzzy and subjective data, enabling researchers to share and discuss different interpretations, ideas, and conclusions can be very beneficial for the process of QDA. The approach of enhancing a process by promoting close collaboration and "shrinking the feedback loop" can also be found in other fields. Agile approaches of software development like Extreme Programming (Beck, 2000) serve as examples of this. However, currently QDAcity only allows the simultaneous collaboration of multiple researchers in a shared project, but not on a more granular level in a shared document. Real-time collaborative editing of a shared document is a classic form of digitally enabled, close collaboration.

[2]

**Martins et. al. states** that in recent years the cloud has become ubiquitous. Many apps and services with users spread across the world resort to these solutions. Cloud applications with global scale user base like social media tend to resort to distributed databases that prioritize lower latency over strong consistency. Such solutions don't require coordination which would require reads and writes to contact a majority of replicas in a communication process that can cross continents, penalizing performance. This kind of applications along with the database replicas usually run in multiple datacenters. Instances are usually geo-replicated to accommodate users from different parts of the globe with fast response time. When the amount of replicas grows it also becomes important to partition data in a way

that doesn't break the fault tolerance guarantees of replication, since having every piece of data in every replica of the database might not be necessary and can definitely become very expensive. In such a setting it is not enough to have database replicas close to the users. The coordination between replicas performing reads and writes also needs to be minimized in order to achieve the desired low latency. Consider that you have a local server close to a client. In order for a client database operation to complete, it would need to contact a majority of database instances. This would completely break the desired low latency. For this reason weak consistency models have been rising in popularity recently. [3]

## II. PROPOSED SYSTEM

### *A. Proposed System*

- **File Synchronization Algorithm**
- File comparison algorithms are used to determine the differences between two files. There are several approaches to file comparison, each with its own advantages and disadvantages. Here are a few common algorithms:
- **Step 1: Byte-by-byte comparison:** This is the simplest form of comparison where each byte of the files is compared. It's fast but doesn't provide detailed information about the differences.
- **Step 2: Line-by-line comparison:** This approach compares files line by line, which is useful for text files. It can highlight added or removed lines but may not detect changes within lines.
- **Step 3: Token-based comparison:** This method breaks the files into tokens (words, phrases, etc.) and compares them. It's more flexible than line-by-line comparison but requires tokenization logic.
- **Step 4: Tree-based comparison:** For structured data like XML or JSON, a tree-based approach can be used. It compares the structure and content of the trees, highlighting differences at different levels of granularity.
- **Step 5: Hybrid approaches:** Some tools use a combination of these algorithms to balance speed and accuracy based on the type of files being compared.

### 3.3.2 Working of Checksums

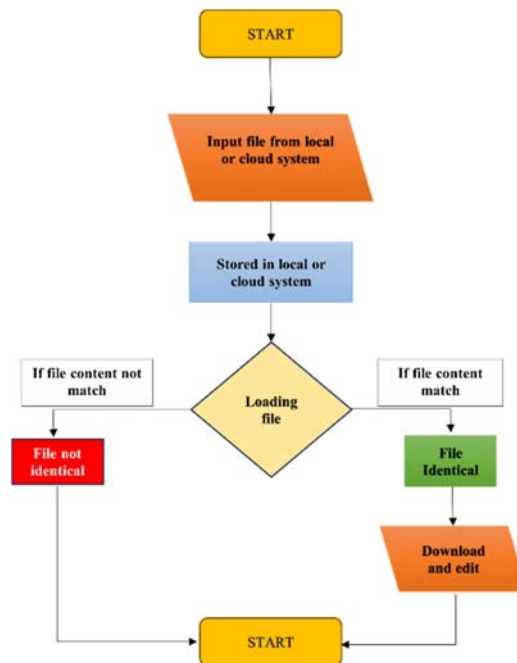
Checksums are widely used in file synchronization and conflict resolution to ensure the integrity of files and detect differences. Here's how checksums can be used:

- **Generating Checksums:** Each file is assigned a checksum, which is a unique identifier calculated based on the file's content. Common checksum algorithms include MD5, SHA-1, and SHA-256.

- **Comparing Checksums:** When synchronizing files, checksums are compared to determine if the files are identical. If the checksums match, the files are considered identical. If the checksums differ, it indicates that the files have different contents.
- **Conflict Resolution:** In case of conflicting changes, checksums can help identify the source of the conflict. For example, if two users modify the same file, their changes can be compared using checksums to determine the extent of the differences.
- **Checksum Verification:** Checksums can also be used to verify the integrity of files during synchronization. By recalculating the checksum of a file after transfer, it can be compared to the original checksum to ensure that the file was not corrupted during transmission.
- **Efficiency:** Checksums are efficient for detecting changes in large files because they provide a fixed-size representation of the file's content. This makes it easier to compare checksums than to compare the entire contents of large files.

While checksums are useful for detecting differences in files, they do have limitations. For example, checksum collisions (different files having the same checksum) are possible, although rare with modern algorithms. Additionally, checksums do not provide information on the specific changes that occurred in a file, only that the file contents are different. Checksum is the error detection method used by upper layer protocols and is considered to be more reliable than LRC, VRC and CRC. This method makes the use of Checksum Generator on Sender side and Checksum Checker on Receiver side.

**B. Flowchart**



**Fig. 3.3 Flowchart of the key agreement protocol mechanism****C. Libraries used**

- **OS Module**

In our proposed system we are working with file on local system so this module is important for us to implement the directory functionality of the file system. The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The `*os*` and `*os.path*` modules include many functions to interact with the file system.

- **Boto3**

Boto3 is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to interact seamlessly with AWS services like Amazon S3 and Amazon EC2. Boto3 simplifies the way you interact with AWS, offering a Pythonic programming approach to managing cloud resources.

- **Time**

As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module, we need to import it. The time module comes with Python's standard utility module, so there is no need to install it externally. In our system we make the use of timestamp functionality to provide the time in which the uploaded file was alter or open.

- **json**

**Python JSON JavaScript Object Notation** is a format for structuring data. It is mainly used for storing and transferring data between the browser and the server. Python too supports JSON with a built-in package called JSON. This package provides all the necessary tools for working with JSON Objects including parsing, serializing, deserializing, and many more.

- **hashlib**

A Cryptographic hash function is a function that takes in input data and produces a statistically unique output, which is unique to that particular set of data. The hash is a fixed-length byte stream used to ensure the integrity of the data. In this article, you will learn to use the hashlib module to obtain the hash of a file in Python. The hashlib module is a built-in module that comes by default with Python's standard library so there is no need to install it manually. The hashlib module implements a common interface for many secure cryptographic hash and message digest algorithms. There is one constructor method named for each type of hash. All return a hash object with the same simple interface. Constructors for hash algorithms are always present in this module.

#### **D. MD5 Hash**

In real life scenario, hash functions are used heavily in cryptographic algorithms, in digital signatures, fingerprints, to store password and many more areas. As a python programmer, we need hash functions to check the duplicity of data or files, to check data integrity when you transmit data over a public network, storing the password in a database etc. In our project we make the use of hash function accepts sequence of bytes and returns 128-bit hash value, usually used to check data integrity but has security issues.

### **III. CONCLUSION AND FUTURE SCOPE**

Across all the file synchronization methods, checksum method works better for files where maximum similarity is found. De-duplication works better for similar types of files, choosing a best chunk size may produce the maximum number of chunks. So, the probability of getting maximum number of similar chunks depends upon the chunk size. So, finding best chunk size is itself a problem and it varies according to the type of files used for synchronization. Proposed system resolves the issue of file synchronization at some extent by identifying the change in the content and warning the user before its opening. One more problem is to reconstruct the same file in the destination device using the same chunk. Chunking method works well over all file synchronization technique. Apart from other file synchronization methods,

### **REFERENCES**

- [1] Elod P. Csirmaz and Laszlo Csirmaz, "Synchronizing Many Filesystems in Near Linear Time", arXiv:2302.09666v2 [cs.IT] 17 May 2023
- [2] Martin Dürsch, "Scaling Real-time Collaborative Editing in a Cloud-based Web App", Erlangen, 19 April 2023
- [3] João Gonçalves Martins, "Query Processing in Cloud Databases with Partial Replication" NOVA University Lisbon March, 2023
- [4] Masoumeh Hajvali, Sahar Adabi, Ali Rezaee and Mehdi Hosseinzadeh, "Decentralized and scalable hybrid scheduling-clustering method for real-time applications in volatile and dynamic Fog-Cloud Environments" (2023) 12:66, <https://doi.org/10.1186/s13677-023-00428-4>, Journal of Cloud Computing: Advances, Systems and Applications
- [5] Elod P. Csirmaz I, and Laszlo Csirmaz, "Data Synchronization: A Complete Theoretical Solution for Filesystems" Future Internet 2022, 14, 344. <https://doi.org/10.3390/fi14110344>
- [6] Novak Bořskov, Ari Trachtenberg, and David Starobinski. Enabling costbenefit analysis of data sync protocols, 2023.
- [7] Elod P. Csirmaz and Laszlo Csirmaz. Data synchronization: A complete theoretical solution for filesystems. *Future Internet*, 14(11), 2022.
- [8] Elod Pal Csirmaz. Algebraic file synchronization: Adequacy and completeness. *CoRR*, abs/1601.01736, 2016.
- [9] John Day-Richter. What's different about the new Google Docs: Making collaboration fast, 2010.
- [10] <https://drive.googleblog.com/2010/09/whats-different-about-new-google-docs.html>, Last accessed on 12 Jan, 2023.
- [11] Shimon Even. *Graph Algorithms*. Cambridge University Press, USA, 2nd edition, 2011.
- [12] JiuLing Feng, XiuQuan Qiao, and Yong Li. The research of synchronization and consistency of data in mobile environment. In *2012 IEEE 2<sup>nd</sup> International Conference on Cloud Computing and Intelligence Systems*, volume 02, pages 869–874, 2012.
- [13] Rusty Klophaus. Riak core: Building distributed applications without shared state. In *ACM SIGPLAN Commercial Users of Functional Programming*, CUFPP '10, New York, NY, USA, 2010. Association for Computing Machinery.



- [14] Zhenhua Li, Christo Wilson, Zhefu Jiang, Yao Liu, Ben Y. Zhao, Cheng Jin, Zhi-Li Zhang, and Yafei Dai. Efficient batched synchronization in dropbox-like cloud storage services. In David Eysers and Karsten Schwan, editors, *Middleware 2013*, pages 307–327, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [15] Erik Liu. A CRDT-based file synchronization system. Master’s thesis, Department of Computer Science, 2021.
- [16] Jakub T. Mo’scicki and Luca Mascetti. Cloud storage services for file synchronization and sharing in science, education and research. *Future Generation Computer Systems*, 78:1052–1054, 2018.
- [17] Agustina Ng and Chengzheng Sun. Operational transformation for real-time synchronization of shared workspace in cloud storage. In *Proceedings of the 2016 ACM International Conference on Supporting*
- [18] *Group Work*, GROUP ’16, page 61–70, New York, NY, USA, 2016.
- [19] Andrea Petroni, Francesca Cuomo, Leonisio Schepis, Mauro Biagi, Marco Listanti, and Gaetano Scarano. Adaptive data synchronization algorithm for iot-oriented low-power wide-area networks. *Sensors*, 18(11):4053, Nov 2018.
- [20] Nuno Preguic, a, Joan Manuel Marques, Marc Shapiro, and Mihai Letia. A commutative replicated data type for cooperative editing. In *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, ICDCS ’09, page 395–403, USA, 2009. IEEE Computer Society.
- [21] Nuno M. Preguic, a. Conflict-free replicated data types: An overview. *CoRR*, abs/1806.10254, 2018.
- [22] Yuechen Qian. *Data synchronization and browsing for home environments*. PhD thesis, Mathematics and Computer Science, 2004.
- [23] Bin Shao, Du Li, Tun Lu, and Ning Gu. An operational transformation based synchronization protocol for web 2.0 applications. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW ’11, page 563–572, New York, NY, USA, 2011. Association for Computing Machinery.
- [24] Marc Shapiro, Nuno Preguic, a, Carlos Baquero, and Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Technical Report 7506, INRIA, Inria-Centre Paris-Rocquencourt, jan 2011.
- [25] Marc Shapiro, Nuno M. Preguic, a, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In Xavier D’efago, Franck Petit, and Vincent Villain, editors, *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings*, volume 6976 of *Lecture Notes in Computer Science*, pages 386–400. Springer, 2011.
- [26] Jian Shen, Member, IEEE, Tianqi Zhou, Debiao He, Yuexin Zhang, Xingming Sun, Senior Member, IEEE, and Yang Xiang, Senior Member, IEEE, “Block Design-based Key Agreement for Group Data Sharing in Cloud Computing”, 1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See [http://www.ieee.org/publications\\_standards/publications/rights/index.html](http://www.ieee.org/publications_standards/publications/rights/index.html) for more information
- [27] Mehdi Bahrami 1 and Mukesh Singhal 2, “A Dynamic Cloud Computing Platform for eHealth Systems”, 2015 IEEE 17th International Conference on e-Health Networking, Applications and Services (Healthcom)
- [28] Tessema Mengistu\*, Abdulrahman Alahmadi\*, Abdullah Albuai, Yousef Alsenani, and Dunren Che, “A "No Data Center" Solution to Cloud Computing”, 2017 IEEE 10th International Conference on Cloud Computing
- [29] Zhao Tianhai,” The Key of Application Software Service in Science Cloud Computing”, 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics
- [30] Wang Xiaoyu, Gao Zhengming , “Research and Development of Data Security Multidimensional Protection System in Cloud Computing Environment” , 2020 International Conference on Advance in Ambient Computing and Intelligence (ICAACI)
- [31] Yanhong Shang1 and Jing Zhang,” Computer Multimedia Security Protection System Based on the Network Security Active Defense Model”, Hindawi, *Advances in Multimedia*, Volume 2021, Article ID 8792105, 9 pages, <https://doi.org/10.1155/2021/8792105>
- [32] L. Zhou, V. Varadarajan, and M. Hitchens, “Cryptographic rolebased access control for secure cloud data storage systems,” *Information Forensics and Security IEEE Transactions on*, vol. 10, no. 11, pp. 2381–2395, 2015.
- [33] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, “Secure cloud storage meets with secure network coding,” in *IEEE INFOCOM*, 2014, pp. 673–681.
- [34] D. He, S. Zeadally, and L. Wu, “Certificateless public auditing scheme for cloud-assisted wireless body area networks,” *IEEE Systems Journal*, pp. 1–10, 2015.
- [35] J. Shen, H. Tan, S. Moh, I. Chung, and J. Wang, “An efficient rfid authentication protocol providing strong privacy and security,” *Journal of Internet Technology*, vol. 17, no. 3, p. 2, 2016.
- [36] Michał Antkiewicz and Krzysztof Czarnecki. *Design Space of Heterogeneous Synchronization*, pages 3–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.