Designing an Efficient Deduplication Algorithm for Audio Files in Cloud Storage / Original article

Prof. Dr. Eng. Ammar Zakzouk
Al-Esraa University – Baghdad – Iraq: ammar.zakzouk@esraa.edu.iq
Homs University – Homs - Syria: azakzouk@homs-univ.edu.sy

Assoc. Prof. Dr. Eng. Alaa Al Sebae
Warwick University – Coventry – United Kingdom: a.al-sebae.1@warwick.ac.uk

Ph.D. Student Eng. Hasan Hasan
Homs University – Homs - Syria: h_hasan@homs-univ.edu.sy

Abstract

Data duplication is a significant challenge in large-scale data storage systems, as it consumes storage space and impacts data organization, management, and processing. An optimal storage system effectively utilizes available storage space. To solve this problem, hash algorithms are employed to generate hash keys for files. Matching files have the same hash key. However, the hash key for two different files in the data may match, and this is what we refer to as a collision. The collision issue is related to the length of the hash key. As the length of the hash key increases, the probability of a collision occurring decreases. When a file is uploaded to the cloud storage system, its hash key is compared with the existing keys stored in the system. However, as the amount of data stored in the cloud increases, the time required for searching and matching also increases. In this paper, we will introduce a File-Level Deduplication technique to deduplicate audio data in the cloud storage system. The proposed technique aims to reduce the search time for hash values by creating a table with multiple indexes. These indexes are categorized based on the format of the audio file, such as uncompressed formats, formats with lossy compression, and formats with lossless compression. Each table contains multiple indexes in the hash table, specifically designed for a particular audio file format. To reduce the probability of data collision, Message Digest-6 (MD6) algorithm will be used, which generates a 512-bit hash key.

Keywords: Deduplication - Hash Table - MD6 - Audio Files - Cloud Storage.

1. Introduction

The cloud storage system is responsible for managing and storing vast amounts of data generated from diverse sources, encompassing various formats such as audio, text, video, and images. Due to the nature of data collection and sharing, there is often a significant presence of duplicate data within the system, which leads to unnecessary consumption of storage resources and can hinder overall system performance [1]. To address these challenges, techniques aimed at eliminating duplicate data—commonly known as data deduplication—have been implemented to optimize storage utilization and enhance processing efficiency. There are two types of duplicate data detection methods: source-based detection (client-side) and target-based detection (server-side) [2]. The core process of deduplication typically involves generating a unique hash key for each file or data block, which serves as a digital fingerprint representing the content. This hash key is then compared against existing hash keys stored in a dedicated data structure, such as a hash table, to determine whether the data already exists within the system. If a matching hash is found, the system recognizes the data as a duplicate and avoids storing a redundant copy, thereby conserving storage space [3]. As the volume of stored data grows, the number of hash keys stored in the hash table also increases, which can lead to longer search times during comparison operations. This escalation in search duration can negatively impact system performance, especially in large-scale cloud environments. To mitigate such issues, one effective strategy involves creating multiple hash tables or indexes. These multiple structures can partition the stored data, enabling faster lookup operations by narrowing down search scopes and reducing the number of comparisons needed.

Additionally, employing multiple indexes helps address collision problems—situations where different data produces identical hash values—by distributing hash keys across various tables, thus lowering the probability of collisions and improving the accuracy of duplicate detection [4] [18]. The selection of an appropriate hash algorithm is another critical factor influencing deduplication efficiency. The hash function's length and complexity directly affect collision rates; shorter or less robust algorithms tend to produce more collisions, increasing the risk of false positives or negatives. Therefore, choosing a cryptographic hash function with a suitable balance of speed and collision resistance is essential for effective deduplication processes [5]. Based on these considerations, we propose an algorithm that involves creating a hash table specifically for audio files on the server. This table consists of multiple indexes, each containing hash keys generated using the MD6 algorithm. MD6 is a cryptographic hash function that employs a Merkle tree-like structure, enabling efficient parallel computation of hashes for extremely long inputs [6].

2. Audio File Formats

There are three main types of audio files formats: Uncompressed Audio Formats, Audio Formats with Lossy Compression, Audio Formats with Lossless Compression [7].

2.1 Uncompressed Audio Formats

Uncompressed audio formats (UAF) consist of real sound waves that are captured and converted directly into a digital format without any additional compression or processing. Although these uncompressed audio

files tend to be the most accurate and true to the original sound, providing high-quality audio reproduction, they also typically require a significant amount of disk space to store due to their large file sizes. The Most Common Uncompressed Audio Formats are:

1- Pulse-Code Modulation (PCM).
2- Waveform Audio File Format (WAV).
3- Audio Interchange File Format (AIFF).

2.2 Audio Formats with Lossy Compression

Lossy compression is characterized by some data loss that occurs during the compression process, meaning that not all original information is preserved. Despite this, compression is essential because uncompressed audio files tend to occupy a significant amount of disk space, making storage and transfer more challenging. Therefore, lossy compression helps reduce file sizes, making it more practical to store and share audio files efficiently. The Most Common Audio Formats with Lossy Compression are:

1- MPEG-1 Audio Layer 3 (MP3).
2- Advanced Audio Coding (AAC).
3- Windows Media Audio (WMA).

2.3 Audio Formats with Lossless Compression

Opposite to lossy compression, there is lossless compression, which is a technique that reduces the size of an audio file without any data loss occurring during the process. This means that the compressed audio file remains an exact replica of the original source, with all the original data preserved perfectly, ensuring no quality is compromised even after

compression. The Most Common Audio Formats with Lossless Compression are:

1- Free Lossless Audio Codec (FLAC).

2- Apple Lossless Audio Codec (ALAC).

The figure (1) shows the most common audio files formats.

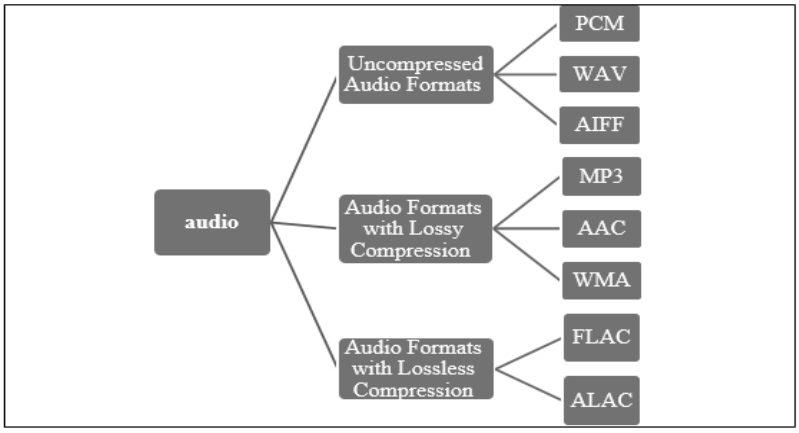


Figure (1): Most common audio files formats.

3. Related Work

In the cloud, File-Level Deduplication generally depends on generating a hash value for the incoming file using a hash algorithm and comparing it with the hash values already stored in the cloud. If this value doesn't exist in the hash table, the file is stored in the cloud and its hash value in the hash table.

We'll show some of the cloud deduplication techniques:

1- In 2015, Naveen A N and V Ravi, proposed a technique to detect duplicate user's files, and then the unique data is stored in the server. This technique is characterized by a low time-complexity, since the process works

with small amount of data. However, this technique is considered the least effective because the final user's data may match files on the server [8].

2- In 2016, V. Radia and D. Dingh, made a study of data deduplication techniques: file level, block level, inline post process, source based and target based. The study concluded that source-based deduplication technique is the best as it optimizes the uploading bandwidth and storage space over cloud. Distributed deduplication provides security. Both approaches together provide reliability [9].

3- In 2016, Parth Shah *et al*, proposed a technique to detect duplication between files of users. The technique involves detecting duplication not only within a user's files but also across files from different users. Once the unique files have been identified. This technique is considered more effective in saving storage space and has a medium time complexity since the process takes place at the level of users' data and not at the server or client. However, user files coming into the storage system may match files that already exist [10].

4- In 2017, Ishita Vaidya and Prof. Rajender Nath, proposed a technique to generate a hash key for the file using MD5 algorithm. Then, this key is compared with the stored keys in Hadoop [11].

5- In 2018, Manjunath R. Hudagi and Sachin A. Urabinahatti, proposed a technique to deduplicate data on file-level. This technique is based on building a hash table in the Hbase that contains the hash keys for the files stored within the system. To process each incoming file, a hash value is generated using a specific algorithm. This hash value is then compared with the values stored in the hash table. If there is no match between the generated hash and any of the stored values, the file is considered unique and subsequently stored in the system [12].

6- In 2020, Weiqi ZHANG *et al*. proposed a technique for deduplication in Hadoop. Hash table was designed in namenode. For every block of files, a hash key using SHA-512 was used. To determine the uniqueness of each block, the generated hash key is compared to the existing keys in the system. If there is no match, it indicates a unique block, which gets stored in the Hadoop Distributed File System (HDFS). SHA-512 algorithm is used to generate a hash value with 512 bits, which results in a lower collision rate compared to previous algorithms. This ensures a higher level of confidence in identifying and storing unique blocks of data [13].

7- In 2021, Niteesha Sharma and Dr. A. V Krishna Prasad, proposed a technique to solve the storage issues and deduplication in Hadoop. A table with hash keys is built in Hbase and SHA-256 is used. The process of reading from HBase is faster than Hadoop Distributed File System (HDFS) [14].

8- In 2021, G. Sujatha and Dr. Jeberson Retna Raj. Proposed an approach to improve the searching time of duplicated data. Dedicated hash tables were designed, each of which is used for each digital data type. When a file is received, its hash key is compared with the hash table corresponding to its type. Thus, the time required for the matching process is reduced compared to previous techniques. However, this technique did not give importance to the type of hash algorithm [15].

9- In 2024, N. A. Jaafar, this study examines the proposed model that includes computation of similarity using cosine coefficients, Euclidean similarity, and Jaccard similarity between training and test texts, providing a variety of metrics for comparison and analysis. These sequential steps combine automated analysis with human interpretation, enhancing the electiveness and accuracy of the plagiarism checker and making it easier to

use in many deferent fields and applications. The results showed that it is possible to accurately determine the similarity between texts [16].

10- In 2024, S. A. Talib, The study begins with the historical background of cryptography and then moves on to analyze trends in information technology before discussing on the challenges to security in cloud computing. The goal of its publication is to highlight the importance of using cryptographic technologies for the protection of information con_dentiality, its integrity and compliance. The combination of literature review and the case study method was applied, pulling knowledge from examples of good cryptographic use and the lessons learned from breaches in security. They reveal how cryptology is crucial in the furthering of cloud security and in maintaining customers' faith. Key information technology practitioners are encouraged not to relax their guard and adopt an integrated security strategy to ward o future attacks. Conclusion of the study is in synthesis of the subsequent analysis, highlighting the continued applicability of cryptography for Cloud Information System security [17].

Thus, the previous studies and their solutions as presented above are not sufficient because one of the parameters in the process of eliminating duplicate data is the time required to implement the technique, which is mainly related to the number of comparison operations for the hash key of the incoming file with the stored keys. The fewer the number of operations, the less the execution time. For any incoming audio file, it will be compared with all stored files (text, images, videos, audio) if a single indexing table is used. However, it will be compared with all audio files if multiple indexing tables are used. This requires a long execution time, which increases with the increase in the stored files.

4. Proposed Algorithm

We developed an algorithm to deduplicate audio files. The algorithm consists mainly of two phases:

1- building the indexing system.

2- deduplication.

4.1 Index System

Index System consists of a table with multi-indexes. Nine indexes for the most common audio files formats and one index for other formats. The first index is for Pulse-Code Modulation index. The second index is for Advanced Audio Coding. The third index is for Audio Interchange File Format. The fourth index is for MPEG-1 Audio Layer3. The fifth index is for Advanced Audio Coding. The sixth index is for Windows Media Audio and. The seventh index is for Free Lossless Audio Codec. The eighth index is for Apple Lossless Audio Codec. The last index is for other formats. The figure (2) shows the index system and it contains examples for hash values:

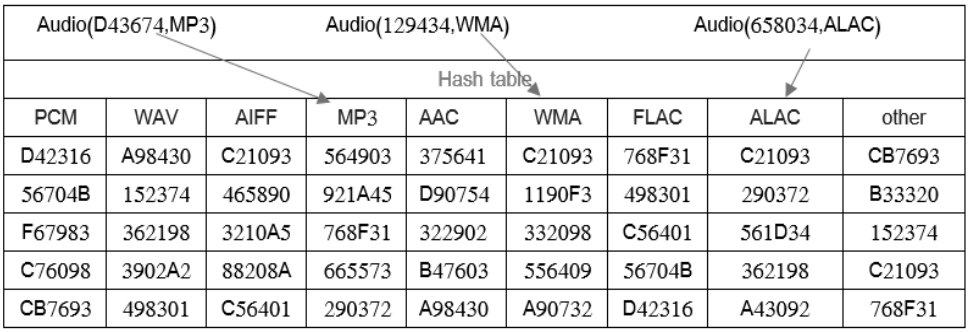
Hash table								
Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8	Index 9
PCM	WAV	AIFF	MP3	AAC	WMA	FLAC	ALAC	other
D42316	A98430	C21093	564903	375641	C21093	768F31	C21093	CB7693
56704B	152374	465890	921A45	D90754	1190F3	498301	290372	B33320
F67983	362198	3210A5	768F31	322902	332098	C56401	561D34	152374
C76098	3902A2	88208A	665573	B47603	556409	56704B	362198	C21093

Figure (2): Index system.

4.2 Deduplication

In this approach, when a file is uploaded to the cloud storage, the metadata of the audio file is read to determine its format. Then, a hash key

is calculated using the MD6 algorithm. Based on the format of the file, the file's hash value is compared with the hash values stored in the index specific to that format. If the hash value already exists in the corresponding index, it indicates that the file is a duplicate and therefore should not be stored again. However, if the hash value is not found in the index, it means the file is unique and can be stored in the cloud storage. Additionally, the hash value of the file is added to the corresponding index for future reference. The figure (3) shows the routing of hash values.



PCM	WAV	AIFF	MP3	AAC	WMA	FLAC	ALAC	other
D42316	A98430	C21093	564903	375641	C21093	768F31	C21093	CB7693
56704B	152374	465890	921A45	D90754	1190F3	498301	290372	B33320
F67983	362198	3210A5	768F31	322902	332098	C56401	561D34	152374
C76098	3902A2	88208A	665573	B47603	556409	56704B	362198	C21093
CB7693	498301	C56401	290372	A98430	A90732	D42316	A43092	768F31

Figure (3): Routing system.

4.3 Technique Scheme

Based on the provided information, the stages of the technique can be arranged as follows:

1. File Upload: The user uploads the file to the cloud storage system.
2. Read Metadata: The metadata of the uploaded audio file is read to determine its format.
3. Calculate Hash Key: MD6 algorithm is used to calculate the hash key of the file.
4. Index Comparison: Based on the file format, the calculated hash key is compared with the hash values stored in the corresponding index.

5. Determine Storage Decision: If the hash value is found in the index, it indicates that the file is a duplicate and should not be stored. If the hash value is not found, it means the file is unique and can be stored in the cloud storage.
6. File Storing: If the file is determined to be unique, it is stored in the cloud storage system.
7. Index Updating: The hash value of the stored file is added to the corresponding index for future reference.

By following these steps, the system can efficiently determine whether to store the uploaded file or not based on its uniqueness and thus, it avoids storing duplicated audio files. The figure (4) shows the general scheme of the proposed technique.

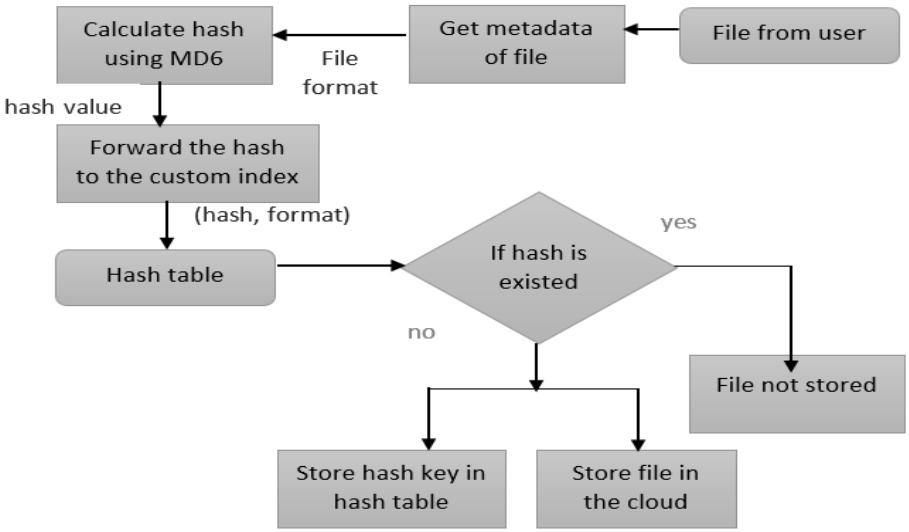


Figure (4): proposed algorithm

Figure (4): Proposed algorithm.

5. Experiments

In this section, we will present multiple models that showcase the outcomes of the proposed technique and we will compare the results of the proposed algorithm with the algorithm based on multiple hash table [15]. These models vary in the number of incoming audio files transmitted to the cloud and the file format. The number of audio files stored in the cloud is 1923, distributed as follows:

1- 800 audio files in MP3 format.
2- 450 audio files in WMA format.
3- 296 audio files in WAV format.
4- 200 audio files in FLAC format.
5- 177 audio files in AIFF format.

5.1 First Model

The number of incoming audio files transmitted to the cloud is 56 audio files, distributed as follows:

1- 22 audio files in MP3 format.
2- 17 audio files in WAV format.
3- 11 audio files in WMA format.
4- 6 audio files in AAC format.

Using the proposed algorithm: The number of comparisons for MP3 audio files is 800*22= 17600, for WMA audio files is 450*17= 7650, for WAV audio files is 296*11=3256, AAC files are unique data, and can be stored in the cloud without the need for comparison processes since the cloud system does not store any AAC audio files. Therefore, the total number of comparison operations: 17600+7650+3256= 28506 comparisons.

Using the proposed based on multi-tables [15]: The hash key of any incoming file to the cloud will be compared with all the hash keys stored in the audio files hash table, and therefore, the number of comparison operations is 1923*56= 107688.

5.2 Second Model

The number of incoming audio files transmitted to the cloud is 23 audio files, distributed as follows:

1- 9 audio files in MP3 format.
2- 6 audio files in PCM format.
3- 11 audio files in WMA format.
4- 6 audio files in AAC format.

Using the proposed algorithm: The number of comparisons for MP3 audio files is 800*9= 7200, for WMA audio files is 450*11=4950, and there isn't a comparison process for AAC and PCM format. Therefore, the total number of comparison operations: 7200+4950= 12150 comparisons.

Using the proposed based on multi-tables [15]: The hash key of any incoming file to the cloud will be compared with all the hash keys stored in the audio files hash table, and therefore, the number of comparison operations is 1923*23= 44229.

5.3 Third Model

The number of incoming audio files transmitted to the cloud is 11 audio files, distributed as follows:

1- 4 audio files in WMA format.
2- 4 audio files in WAV format.
3- 3 audio files in FLAC format.

Using the proposed algorithm: The number of comparisons for WMA audio files is 450*4= 1800, for WAV audio files is 296*4=1184, and for FLAC audio files is 200*3=600, the total number of comparison operations: 1800+1184+600=3584 comparisons.

Using the proposed based on multi-tables [15]: The hash key of any incoming file to the cloud will be compared with all the hash keys stored in the audio files hash table, and therefore, the number of comparison operations is 1923*11= 21153.

5.4 Other Models

The table (1) illustrates the experimental models that differ in the number of files stored in the cloud, the number of files incoming to the cloud, and the maximum number of required comparison operations. This is achieved using the proposed algorithm and the technique based on multiple tables.

Table (1): Effectiveness of the proposed algorithm.

Experiment	Incoming files to the cloud Multi-tables proposed				Number of audio files in the cloud					Number of Comparisons	
Experiment1	22	PCM	MP3	WAV	PCM	MP3	WAV	FLAC	WMA	17358	4270
		12	6	4	123	415	76	92	83		
Experiment2	13	MP3	AIFF	AAC	MP3	WAV	WMA	AIFF	AAC	29010	21345
		5	4	4	625	120	870	1700	2855		
Experiment3	9	WMV	WMA	AAC	PCM	WMA	FLAC	AIFF	AAC	12654	668
		4	3	2	413	210	31	521	19		
Experiment4	7	MP3	FLAC	WMA	WAV	WMA	AAC	ALAC	AIFF	5856	332
		3	3	1	47	332	498	12	87		

Based on the table (1), the figure (5) illustrates the effectiveness of the proposed algorithm and compares the results obtained by this algorithm with the algorithm based on multi-table.

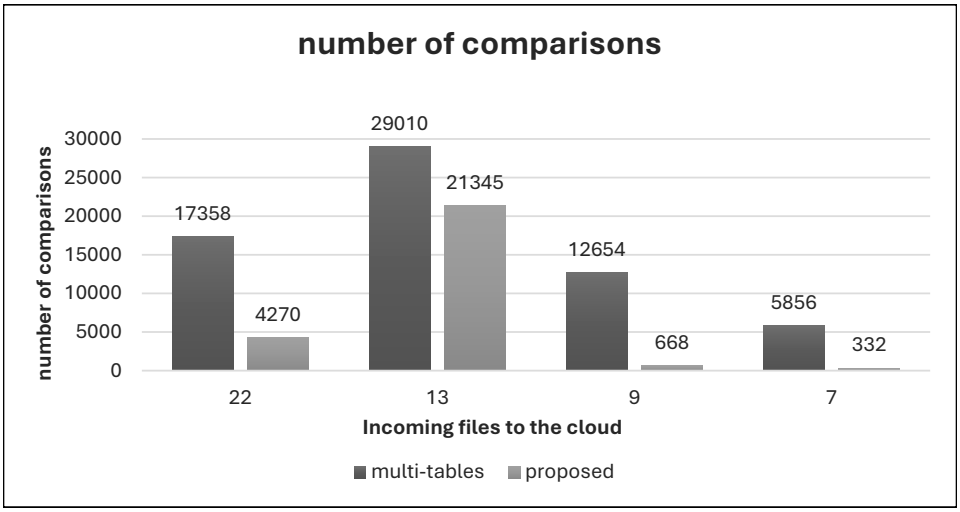


Figure (5): Effectiveness of the proposed algorithm.

From the figure above, it can be observed that the proposed algorithm is more efficient compared to the technique based on multiple tables. With the proposed algorithm, each audio file coming to the cloud will only be compared with other audio files of the same format.

For further clarification, let's consider that the total number of audio files stored in the cloud is 4216, distributed as follows:

1- 500 audio files in PCM.

2- 435 audio files in WAV.

3- 1289 audio files in MP3.

4- 1020 audio files in WMA.

5- 440 audio files in AAC.

6- 215 audio files in AIFF.

7- 120 audio files in ALAC.

8- 197 audio files in FLAC.

The figure (6) illustrates the difference in the number of comparison operations between the proposed algorithm and the algorithm based on multiple tables, in the case of an audio file of a specific format being received in the cloud.

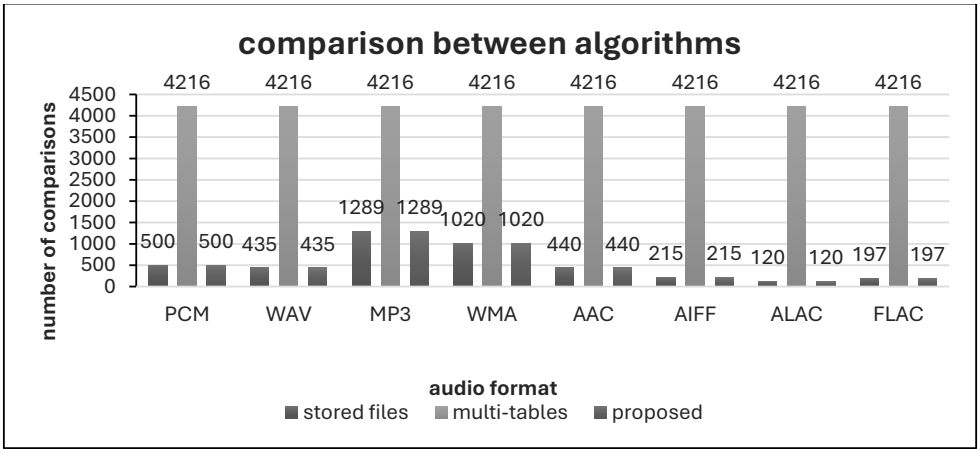


Figure (6): Role of proposed algorithm.

Using the proposed algorithm, the number of comparison operations does not change based on the total number of audio files stored in the cloud. Instead, it depends on the number of files stored for each individual format. For example, let's consider a scenario where there are 500 MP3 files stored in the cloud and no WAV files. If a WAV file is received, it will be automatically stored in the cloud without any comparison operations being performed. This is because there are no other WAV files to compare it with.

The figure (7) illustrates the change in the number of comparisons using the proposed technique.

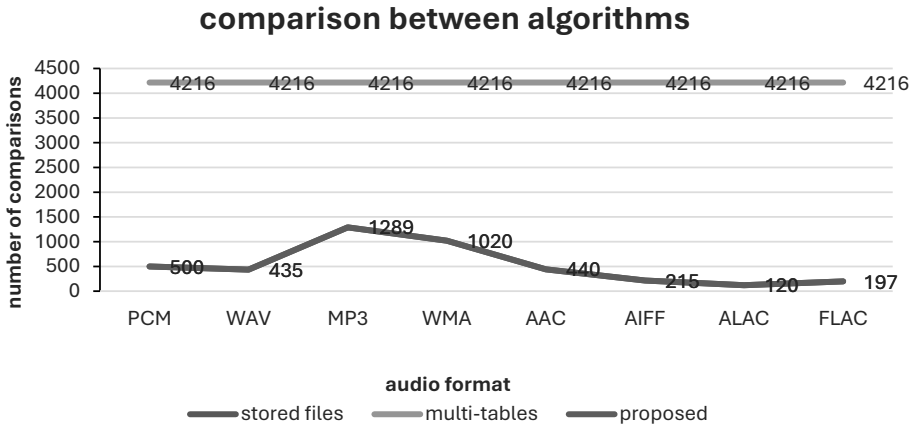


Figure (7): Change in the number of comparisons.

6. Results and Discussion

The multi-table-based technique stores all hash keys of audio files in a centralized manner. To find a match for a specific file, its key must be compared against the keys of all other files in the system. This approach results in a high overall time complexity, as the number of comparisons increases significantly with the total number of audio files. Consequently, the process becomes less efficient, particularly when handling large datasets.

To address this issue, the proposed algorithm intelligently divides the main table of audio files into nine separate indexes, each corresponding to a specific file format. Each index contains only the keys of files that share that particular format. When a new audio file is uploaded, it is assigned a unique key and stored in the index that matches its format. Consequently, when searching for a specific file, the comparison is limited to the relevant format index, drastically reducing the number of comparisons required. This proposed approach eliminates unnecessary comparisons across different

formats, leading to improved efficiency and quicker processing times when managing diverse audio formats in a cloud environment.

The proposed technique enhances the search process by reducing the number of comparison operations, leading to faster performance and a lower likelihood of data collisions. Unlike methods that rely on multi-table comparisons—where all files are compared with each other, increasing the chances of similar keys and subsequent collisions—our approach confines comparisons to files within the same format index. This targeted comparison strategy effectively minimizes collision probability while maintaining efficient search speeds.

7. Conclusion

With the increasing volume of data received by the cloud system, which includes duplicate data, it is not enough to design techniques that only focus on efficient duplicate detection. In the face of this massive amount of data, it is necessary to minimize the time required to process duplicate data as much as possible. In this research, we have built a multi-indexing system based on metadata (file format) to reduce the number of comparisons and thus accelerate the search process for the partition key. Reducing the number of comparisons also reduces the probability of collisions between partition keys. In this research, we also used the MD6 algorithm to calculate the hash value for the files, which produces a 512-bit key. The longer the key length, the lower the collision rate. In the future, it is possible to build a duplicate data elimination system that relies on multiple tables and multiple indexes to minimize the time complexity as much as possible.

8. References

[1]. Y. Himeur, H. Elouadrhiri, L. Khoukhi, and N. Beni-Hssane, (2023),"AI-big data analytics for building automation and management systems: a survey, actual challenges and future perspectives", Artificial Intelligence Review, Vol. 56, No. 6, pp. 4929-5021.

[2]. M. Akbar, *et al.*, (2024),"Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach", Cluster Computing, Vol. 27, No. 3, pp. 3683-3702.

[3]. M. Muniswamaiah, T. Agerwala, and C. Tappert, (2019),"Big data in cloud computing review and opportunities", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 11, No. 4, pp. 43-57.

[4]. V. Schmitt and J. Jordaan, (2013),"Establishing the validity of MD5 and SHA-1 hashing in digital forensic practice in light of recent research demonstrating cryptographic weaknesses in these algorithms", International Journal of Computer Applications, Vol. 68, No. 23, pp. 40-43.

[5]. M. Eichlseder, F. Mendel, and M. Schläffer, (2014),"Branching heuristics in differential collision search with applications to SHA-512", in Proceedings of the Fast Software Encryption - FSE 2014, p. 16.

[6]. M. Kathiravan, *et al.*, (2023),"A cloud based improved file handling and duplicate removal using MD5", in 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), IEEE.

[7]. Wikipedia, (2023),"Audio file format", available at: https://en.wikipedia.org/wiki/Audio_file_format

[8]. N. A. Naveen and V. Ravi, (2015),"Client-side deduplication scheme for secured data storage in cloud environments", International Journal of Engineering Research & Technology (IJERT), Vol. 4, No. 5, pp. 1465-1467.

[9]. V. S. R. and D. K. Singh, (2016),"Secure deduplication techniques: a study", International Journal of Computer Applications, Vol. 137, No. 8, pp. 41-43, doi: 10.5120/ijca2016908874.

[10]. S. Parth, G. Amit, P. Sandipkumar, and P. Priteshkumar, (2016),"Efficient cross-user client-side data deduplication in Hadoop", Journal of Computers, DOI: 10.17706/jcp.12.4, pp. 362-370.

[11]. I. Vaidya and R. Nath, (2017),"An improved de-duplication technique for small files in Hadoop", International Research Journal of Engineering and Technology (IRJET), Vol. 4, No. 7, pp. 2040-2045.

[12]. R. Hudagi and A. Urabinahatti, (2018),"Efficient deduplication using Hadoop", International Journal of Latest Trends in Engineering and Technology, Vol. 10, No. 3, pp. 236-238.

[13]. W. Zhang, B. Shao, G. Bian, and Q. He, (2020),"Research on multifeature data routing strategy in deduplication", Scientific Programming, Vol. 2020, Article ID 8869237, pp. 11.

[14]. N. Sharma and Dr. A. V Krishna Prasad, (2021),"File-level deduplication by using text files – Hive integration", in Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), p. 6.

[15]. G. Sujatha and Dr. Jeberson Retna Raj, (2021),"Improving the efficiency of deduplication process by dedicated hash table for each digital data type in cloud storage system", Webology, Vol. 18, Special Issue on Artificial Intelligence in Cloud Computing, pp. 288-301.

[16]. N. A. Jaafar, (2024),"A study on improving the accuracy and effectiveness of similarity detection processes in text files using NLP techniques", Volume 6, Issue 9, Al-Esraa University College Journal for Engineering Sciences.

[17]. S. A. Talib, (2024),"The importance of cryptography in cloud computing", Volume 6, Issue 10, Al-Esraa University College Journal for Engineering Sciences.

[18]. Ammar Zakzouk, Bassim Oumran, and Hasan Hasan, (2024),"ALLI: a high-performance approach to data deduplication in Hadoop using enhanced hashing and two-level indexing techniques", International Journal of Performability Engineering, Vol. 20, No. 12.