تصميم خوارزمية إلغاء بيانات مكررة فعالة للملفات الصوتية في التخزين السحابي

أ. د. عمار زقزوق[1]

م. حسن حسن[2]

الملخص:

تشكل البيانات المكررة تحدياً مهماً في أنظمة تخزين البيانات الضخمة، فهي تستهلك مساحة تخزينية، ما يؤثر على تنظيم البيانات وإدارتها ومعالجتها. فأنظمة التخزين المثالية تستخدم مساحة التخزين بشكل فعال. لحل هذه المشكلة، تستخدم خوارزميات الاختزال لتوليد مفاتيح اختزال للملفات. الملفات المتطابقة لها المفتاح ذاته. لكن قد يتطابق مفتاح الهاش لملفين مختلفين، وهذا ما يسمى بالتصادم. ترتبط مشكلة التصادم بطول المفتاح المختزل. فكلما زاد طول المفتاح المختزل قلت احتمالية حدوث تصادم. عندما يُحمَّل ملف إلى نظام التخزين السحابي، تتم مقارنة مفتاح الاختزال الخاص به مع المفاتيح المخزنة في النظام. لكن مع زيادة البيانات المخزنة في السحابة، يزداد زمن عملية البحث والمطابقة.

في هذه الورقة، نقدم تقنية إلغاء بيانات مكررة على مستوى الملف لإلغاء تخزين البيانات الصوتية المكررة في نظام التخزين السحابي. تهدف التقنية المقترحة إلى تقليل زمن البحث عن قيم الاختزال عن طريق إنشاء جدول اختزال بفهارس متعددة. تُصمم هذه الفهارس بالاعتماد على صيغة ملف الصوت (ملفات غير مضغوطة، ملفات مضغوطة مع فقدان في البيانات، ملفات مضغوطة دون فقدان في البيانات). وبالتالي، يتضمن جدول الاختزال فهارساً متعددة، وكل فهرس من أجل صيغة محددة. لتقليل احتمالية حدوث تصادم، يتم استخدام خوارزميةMD6 ، والتي تنتج مفتاحاً بطول 512 بتاً.

الكلمات المفتاحية: إلغاء البيانات المكررة – جدول التجزئة – MD6 – الملفات الصوتية – التخزين السحابي.

1 أستاذ، قسم هندسة الحاسب، كلية الهندسة، الجامعة الوطنية الخاصة، حماه، سوريا

2 طالب دكتوراه، قسم هندسة التحكم الآلي والحواسيب، كلية الهندسة الميكانيكية والكهربائية، حمص، سوريا

Designing an Efficient Deduplication Algorithm for Audio Files in Cloud Storage

Prof. Dr. Ammar Zakzouk[1]

Eng. Hasan Hasan[2]

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 at the 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 Professor, Faculty of Engineering, Department of Computer Engineering, Al-Wataniya Private University, Hama, Syria

2 Ph.D Student, Faculty of Mechanical & Electrical Engineering, Automatic Control and Computer Engineering, Al-Baath University, Homs, Syria

1. Introduction

The cloud storage system handles a large volume of data in various formats (audio, text, video, images) from different sources. Some of data are duplicated, which are considered unnecessary [1]. To optimize storage space and improve the performance of the cloud system in terms of storage and processing, techniques for removing duplicate data have been employed. There are two types of duplicate data detection methods: source-based detection (client-side) and target-based detection (server-side) [2]. In general, the deduplication process involves generating a hash key for each file and comparing it with the hash keys of other files to determine if it should be stored. A hash table is used to store all the hash keys of the stored files [3]. However, as the size of the data stored in the cloud storage system increases, the number of hash values stored in the hash table also increases. This results in longer search times for hash values. To mitigate this issue, multiple tables or indexes can be created, reducing the time required to compare the hash key of an incoming file with the existing data in the cloud. This approach also helps address collision problems. By utilizing multiple indexes or tables, the number of comparison operations is reduced, thereby decreasing the probability of collisions [4]. The choice of hash algorithm also plays a crucial role in the deduplication process, as the length of the key directly impacts collision rates [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 Format

There are three main types of audio files format: 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 captured and converted to digital format without further processing. As a result, uncompressed audio files tend to be the most accurate but take up a lot of disk space. 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 when some data is lost during the compression process and compression is important because uncompressed audio takes up lots of disk space. 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 lossy compression is lossless compression, a method that reduces an audio file's size without ANY data loss between the source and the compressed audio files. The Most Common Audio Formats with Lossless Compression are:

1- Free Lossless Audio Codec (FLAC).

2- Apple Lossless Audio Codec (ALAC).

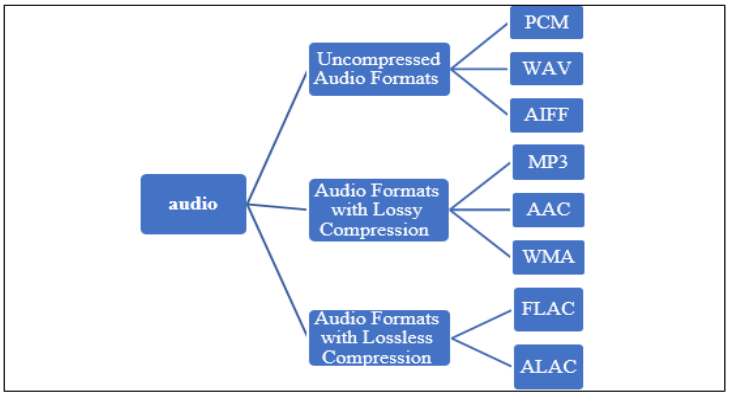


Figure (1): the most common audio files formats

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 stored in the cloud. If this value is not existed 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 is best to optimize upload 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, propose 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 to deduplication in Hadoop. Hash table was designed in namenode. For every block of file was used hash key using SHA-512. 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, indicating a unique block, it is 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. The SHA-256 was 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, hash table 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].

previous studies and their solutions 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 large execution time, which increases with the increase in 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 consist of a table with multi-indexes. nine indexes for the most common audio files formats and one index for other formats. First index for Pulse-Code Modulation index. Second index for Advanced Audio Coding. Third index for Audio Interchange File Format. Fourth index for MPEG-1 Audio Layer3. Fifth index for Advanced Audio Coding. Sixth index for Windows Media Audio and. Seventh index for Free Lossless Audio Codec. Eighth index for Apple Lossless Audio Codec. Last index for other formats. Figure (2) show index system contains examples for hash values:

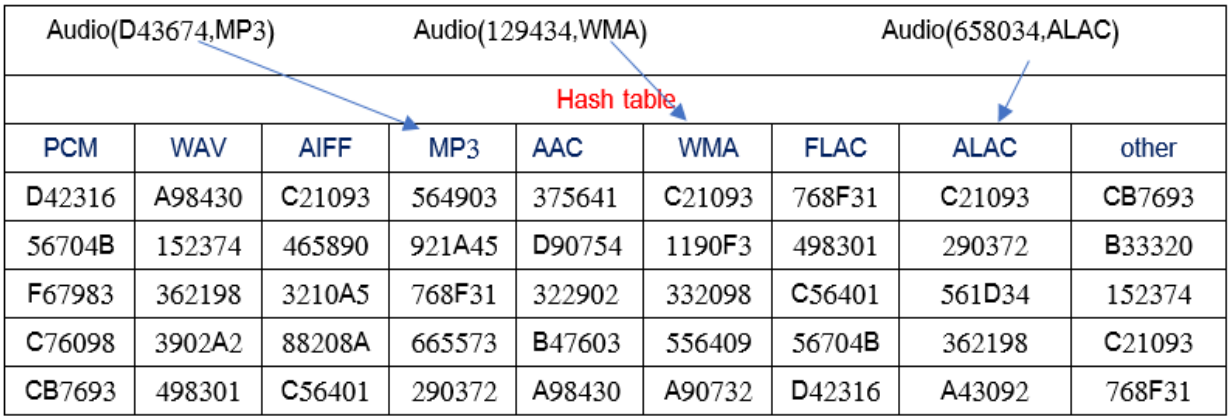
Hash table								
Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8	Index 9
PCM	WAV	AIFF	MP3	AAC	WMA	FLAC	ALAC	other
D42316	A98430	C21093	564903	375641	C21093	768F31	C21093	CB7693
56704B	152374	465890	921A45	D90754	1190F3	498301	290372	B33320
F67983	362198	3210A5	768F31	322902	332098	C56401	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. Figure (3) show 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. Store File: If the file is determined to be unique, it is stored in the cloud storage system.
7. Update Index: 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 avoid storing duplicated audio files. Figure (4) show the general scheme of the proposed technique.

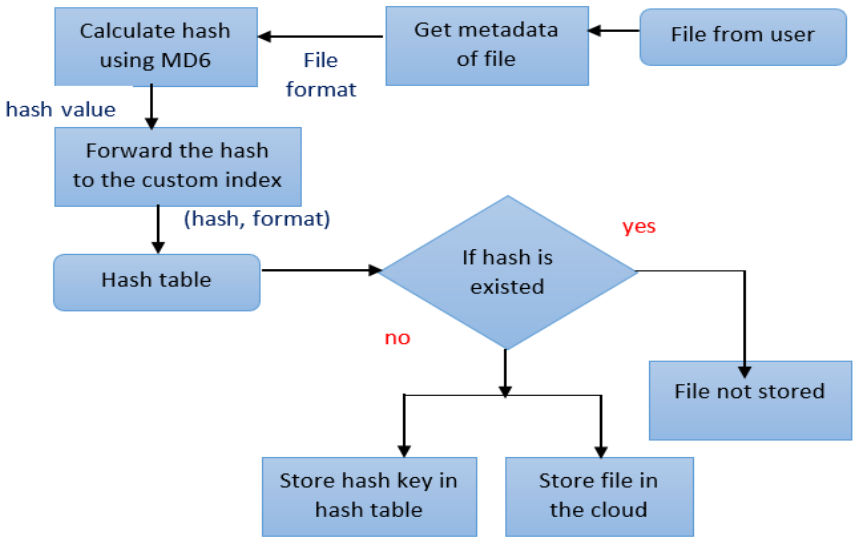


Figure (4): proposed algorithm

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, 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 any 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.

5.2 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.

5.2 Other Models

In the table (1) some 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): The effectiveness of the proposed algorithm

Experiment	Incoming files to the cloud	Number of audio files in the cloud								Number of Comparisons	
										Multi-tables	proposed
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 table (1) the following diagram illustrates the effectiveness of the proposed algorithm and compares the results obtained by this algorithm with the algorithm based on multi-table.

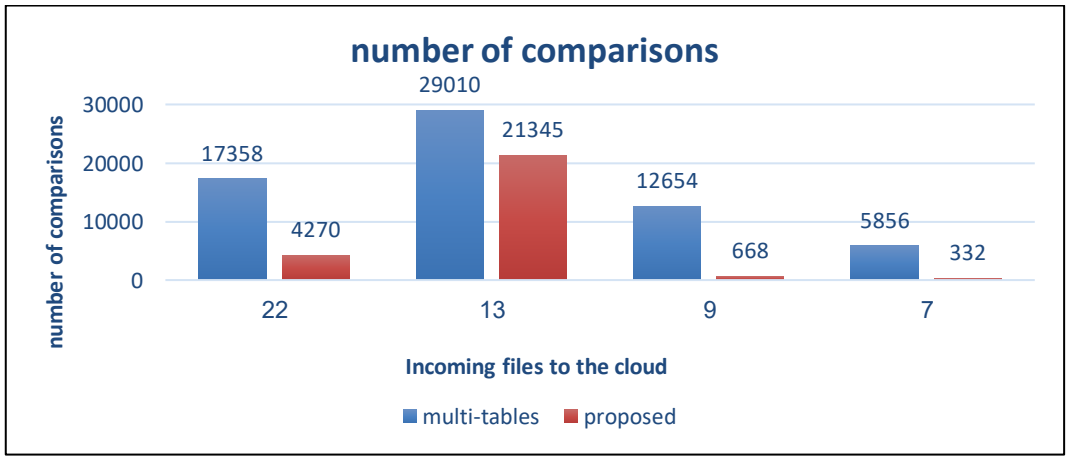


Figure (5): the effectiveness of the proposed algorithm

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 follow:

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.

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.

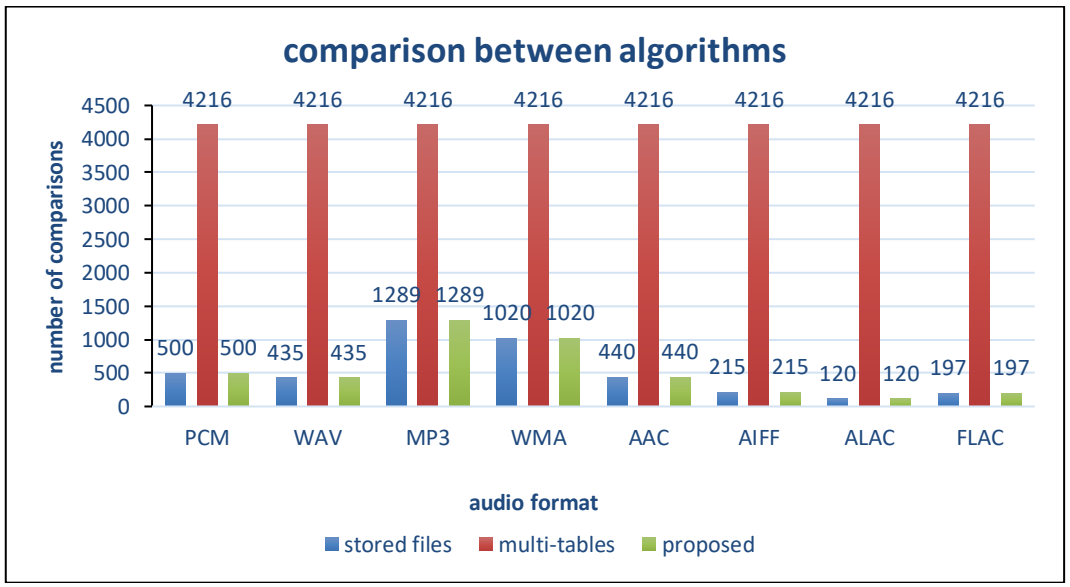


Figure (5): the role of proposed algorithm

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. Figure (6) illustrates the change in the number of comparisons using the proposed technique.

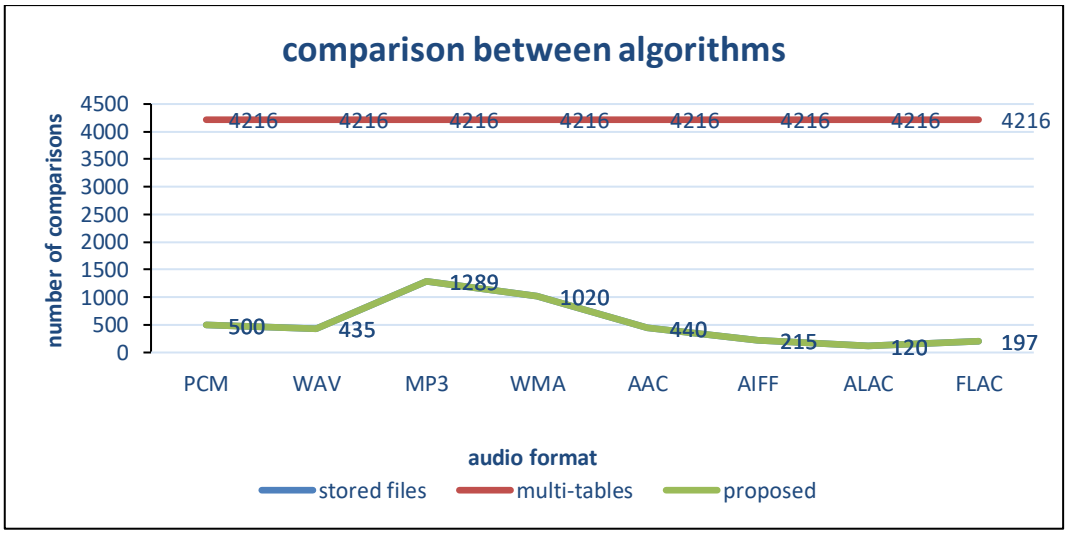


Figure (6): the change in the number of comparisons

6. Results and Discussion

Using the proposed technique based on multi-tables, all hash keys of audio files are stored together, and the key of each file needs to be compared with the keys of all other files in order to find a match. This results in a high time complexity, and the time complexity increases as the number of audio files increases.

The proposed algorithm divides the table of audio files into 9 separate indexes based on file format. Each index contains only the keys of the files with that specific format. When a new audio file is uploaded, it is assigned a key and stored in the corresponding index based on its format. This means that when searching for a specific file, the comparison is only made within the index that contains files of the same format, significantly reducing the number of comparisons needed. that means that the proposed algorithm eliminates the need for unnecessary comparisons, resulting in improved efficiency and faster processing times when handling different audio formats in the cloud.

By reducing the number of comparison operations, the proposed technique not only speeds up the search process but also minimizes the probability of data collisions. Using the proposed technique based on multi-tables, all files are compared with each other, there is a higher probability of two files having similar keys, resulting in collisions. However, with the proposed technique, the chances of collisions are reduced since the comparison is limited to files within the same format index.

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] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issue", Information Systems 47, pp. 98-115, 2015.

[2] N. Sharma, A. V. Krishna Prasad, V. Kakulapati, "Data Deduplication Techniques for Big Data Storage Systems", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8 Issue-10, pp. 1145-1150, 2019.

[3] M. Muniswamaiah, T. Agerwala, C. Tappert, "BIG DATA IN CLOUD COMPUTING REVIEW AND OPPORTUNITIES", International Journal of Computer Science & Information Technology (IJCSIT), Vol. 11, No. 4, pp. 43-57, 2019.

[4] V. Schmitt, J. Jordaan, "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, Volume 68– No.23, pp. 40-43, 2013.

[5] M. Eichlseder, F. Mendel, M. Schl¨affer, "Branching Heuristics in Differential Collision Search with Applications to SHA-512", Conference: Fast Software Encryption - FSE 2014, P. 16.

[6] Ronald L. Rivest, "The MD6 hash function A proposal to NIST for SHA-3", Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, P. 237, 2008.

[7] Audio file format, Audio file format - Wikipedia, 2023.

[8] Naveen A N and V Ravi, "Client Side Deduplication Scheme for Secured Data Storage in Cloud Environments", International Journal of Engineering Research & Technology (IJERT), Vol. 4 Issue 05, pp. 1465-1467, 2015.

[9] V. S. R. and D. K. Singh, "Secure Deduplication Techniques: A Study," Int. J. Comput. Appl., vol. 137, no. 8, pp. 41–43, 2016, doi: 10.5120/ijca2016908874.

[10]] Parth. S, Amit. G, Sandipkumar. P, Priteshkumar. P," Efficient Cross User Client Side Data Deduplication in Hadoop", Journal of Computers, DOI: 10.17706/jcp.12.4, pp. 362-370, 2016.

[11] i. Vaidya, Prof. R. Nath, "An Improved De-Duplication Technique for Small Files in Hadoop", International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 07, pp. 2040-2045, 2017.

[12] Manjunath R. Hudagi and Sachin A. Urabinahatti, "EFFICIENT DEDUPLICATION USING HADOOP", International Journal of Latest Trends in Engineering and Technology, Vol. (10) Issue(3), pp.236-238, 2018.

[13] w. Zhang, B. Shao, G. Bian, Q. He, "Research on Multifeature Data Routing Strategy in Deduplication", Scientific Programming, Volume 2020, Article ID 8869237, P. 11, 2-020.

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

[15] G. Sujatha, Dr. Jeberson Retna Raj, "Improving the Efficiency of Deduplication Process by Dedicated Hash Table for each Digital Data Type in Cloud Storage System", Webology, Volume 18, Special Issue on Artificial Intelligence in Cloud Computing, pp. 288-301, 2021.