Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.030259
Article

Tech Science Press

Hash-Indexing Block-Based Deduplication Algorithm for Reducing Storage in the Cloud

D. Viji* and S. Revathy

Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India
*Corresponding Author: D. Viji. Email: vijiresearchcse01@hotmail.com
Received: 22 March 2022; Accepted: 27 August 2022

Abstract: Cloud storage is essential for managing user data to store and retrieve from the distributed data centre. The storage service is distributed as pay a service for accessing the size to collect the data. Due to the massive amount of data stored in the data centre containing similar information and file structures remaining in multi-copy, duplication leads to increase storage space. The potential deduplication system doesn't make efficient data reduction because of inaccuracy in finding similar data analysis. It creates a complex nature to increase the storage consumption under cost. To resolve this problem, this paper proposes an efficient storage reduction called Hash-Indexing Block-based Deduplication (HIBD) based on Segmented Bind Linkage (SBL) Methods for reducing storage in a cloud environment. Initially, preprocessing is done using the sparse augmentation technique. Further, the preprocessed files are segmented into blocks to make Hash-Index. The block of the contents is compared with other files through Semantic Content Source Deduplication (SCSD), which identifies the similar content presence between the file. Based on the content presence count, the Distance Vector Weightage Correlation (DVWC) estimates the document similarity weight, and related files are grouped into a cluster. Finally, the segmented bind linkage compares the document to find duplicate content in the cluster using similarity weight based on the coefficient match case. This implementation helps identify the data redundancy efficiently and reduces the service cost in distributed cloud storage.

Keywords: Cloud computing; deduplication; hash indexing; relational content analysis document clustering; cloud storage; record linkage

1 Introduction

The cloud provides a massive storage service based on the "pay as usage" mode. The cloud offers a huge storage service based on the "pay as usage" mode. Among them is Amazon Simple Storage Service (Amazon S3), a product storage service that provides industry-leading scaling, data availability, and performance. Due to the increasing multi-copy of files, the storage is increased as per the high usage cost. To reduce the storage by finding duplicate data based on deduplication methodologies. Data Deduplication is a more popular technique to attain storage redundancy to find duplicates. To improve the deduplication performance based on similarity and locality-based indexing, content-based inline deduplication, semantic

deduplication, hash function, and record linkage, Deduplication is used to minimize the storage to find duplicate data.

Similarity and locality approaches work on the hashing policy to find similar files by accelerating contagious contents in the files. The hash policy divides the files into blocks, eliminates the content-wise presence in similar documents, and is indexed to group the relevant documents from similarity clustering. The file blocks are verified by cross-reference evaluation whether the content is present in similar documents. Indexing helps to find the frequency of contents present in similar documents.

The locality-based deduplication approach depends on the hash values for comparing the block of data and creating a logical pointer to another copy that is a copy of the redundant data instead of storing other actual copies of the excess data. Deduplication makes an index-based search model to find the duplicate content exploits stream of file content to make blocks. But the indexing doesn't create a locality index to filter duplicate files. The data filtering criteria need to perform the similarity content based on clustering to find the inconsistencies in data chunks depending on metadata forums. To eliminate the duplicate copies by applying hashing techniques using the flag asynchronous consistency model. The consistency service supports relational data duplication to find similar records. The similarity content analysis depends on finding the presence of data copies under different entity resolutions. The features have represented the size of data content count, file format, etc. The features are estimated using a semantic-based deduplication model to increase the duplicate data findings under the similarity measurement between the data to reduce the storage.

Dedication comes under cloud service management to manage the storage space through a data pooling server. Fig. 1 shows the general block-based deduplication in cloud storage. All the data are stored in a centralized cloud medium, which a cloud service provider provides. The user stores the data or document in cloud storage via network storage. All the documents are stored in the form of blocks and chunks processed along with a data pooling server.

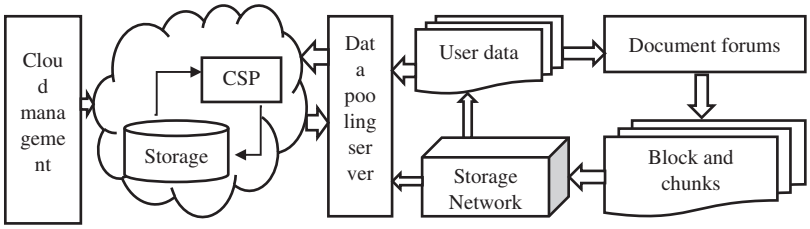

Figure 1: Process of block-based deduplication in cloud storage

The hash-based indexing scheme presents the match-case data deduplication model to find similar data. These models support the segment block-based deduplication methods to improve the accuracy of finding the duplicates. The files are segmented into groups to make indexing for creating hashing values to address the similarity of block content. These hash-based techniques support large-scale data storage management to achieve deduplication to improve storage performance.

The segmentation concepts divide the files between making block attains with hash values. It splits the file into the number of blocks to make the variable index for reference pointer to group the similar files into cluster format. The clustered index contains the similarity link between related data to make the semantic relation for comparing the other segments to find the duplicates. Similarly, the record linkage methods are used to find the relations among the duplicate contents, which make a related forum post of content match case to address the relational content.

In this data deduplication, the hash indexing is generated to find the related document analysis based on semantic source content data analysis. This deduplication depends on hash-based indexing to form similar clusters and file groups. The semantic-based file duplication identification is carried through a data partition to find the relevant modules to implement content redundancy, which is larger than the file size. The record linkage concept makes a similarity block pointer link to find similar content between the files. Based on the study of semantic relationships, the increase in the processing files of the storage system reduces the loss of the original file over time. As a result, critical fragmentation must resonate with less essential data fragmentation to improve the Deduplication process.

This proposed system intends a new deduplication technique to reduce the storage space using hash indexing forums. It reduces the content search capability between the documents to make an ordered list of hash value indexes for finding the file content. By reducing the file comparison to find the similarity, content analyses based on the segmentation concept are used to split the files into a block that makes the block-based comparison. The compared files contain similarity count terms based on the content presented in the document, which is estimated by distance vector weightage correlation. Then the files are grouped into clusters to make similarity indexing values in the cluster. Depending on the cluster weight, the record linkage concepts are intended in the segmented bind linkage to compare the cluster weight to find the duplicates.

The rest of Section 2 contains the related work that defines the various author implementation and its challenges. Section 3 includes the implementation of the proposed Deduplication method. Section 4 contains results and a discussion on performance accuracy, and Section 5 concludes with the improvement of the proposed system.

2 Related Work

Data redundancy is essential for reducing duplicates in distributed cloud storage [1]. The deduplication techniques are the most crucial factor in considering the data redundancy [2,3]. The cross-user resource-based inline deduplication method is used to improve the redundant cloud storage using a file pointer indexing scheme to store the data. But the challenging issue is more complex to progress the data block to find similar content [4]. The content-based duplication measures the similarity of the file content through the Content Similarity over the Intention-Based Segmentation method (CS-IBS). The content files are segmented into the data block to find the related similarity documents and make comparisons based on the relational weight clusters [5]. The content-based segmentation increases the probability of finding features more significant to finding related segment leads [6]. The relative weights are not closest to important terms matching the file content comparisons [7,8]. This increases the non-relational point of content feature to reduce the deduplication accuracy, time-consuming frustration, etc.

The De-duplicated file is progressed into fragmentation files, storing multiple files into a single copy [9]. From the standpoint of forensics, it is not easy to recover a deduplication content file system because the semantic data requires specific knowledge of whether this creates a relational point to compare files [10]. The Correlated concept-based dynamic document clustering algorithms for deduplication to increase the number of documents in the corpus, such as relational contents, is non-regular in finding similarity [11]. It brings greater complexity to obtaining the clustering and documents by combining semantic features to make a large corpus. The quality of the cluster, even if it is improved based on semantic clustering, scalability of the system is still complex.

Data deduplication techniques are reviewed to progress duplicate findings in large storage systems. Deduplication allows redundant data to improve storage utilization and reduce storage costs not well to produce accuracy [12]. This describes many prior literature review systems deduplication systems in that point of hashing techniques, with repeating data cloud data storage deletion various primary art classification.

The Deduplication detection is based on Content-Defined Chunking (CDC) and secure-hash-based fingerprinting to remove redundant data at the chunk level. This eliminates the data redundancy at the block level, and file data reduction techniques do not well as to reduce the storage efficiently [13]. Most data deduplication is based on deduplication MD5, SHA-1, and SHA-2 Hybridization methods and creates a hash policy to find duplicates. This system doesn't make it efficient for redundant Cloud storage, which is a separate storage service that allows users to upload and download the data [14]. However, it raises privacy and confidentiality issues because all data is stored in cloud storage. The Reliable File De-duplicated HDFS doesn't consider the file record concept to waste disk space by repeating words, divided into the store number files are uploaded image copy [15]. Therefore, space is occupied more to delete duplicate files; the utility need is more complex.

The deduplication provisions to Automatic Dependent Surveillance-Broadcast (ADS-B) data processing model with support of Hash-Based Text Deduplication for Locality Sensitive Hashing (HBTD-LSH) deduplication technique to reduce the storage [16]. These works on machine learning depend on Locality Sensitive Hashing (LSH). It finds the relative documents which are grouped into singularized clusters. The text deduplication doesn't consider the semantic similarity for document finding leads to non-related content grouping. So the substring-based hashing creates multiple indexes, leading to storage complexity. This creates a cluster ensemble approach but doesn't make deduplication to manage the data in the redundant form in cloud storage.

The deduplication remains the physical storage data chunk to check the file similarity; this verifies the files based on reference-related content presented in the other files. It computes the XOR-based parity group of data chunks stored in a single indexing format to demonstrate the unique data chunks. Similarly, the individual chunks are separately stored without similarity indexing, so the chunk failures occur due to errors that lead to duplicate data. The per-file parity has not been of great importance to the reliability of deduplication storage systems. Because the relational file content analysis was not performed, they only used Redundant Array of Inexpensive Disks RAID-based intensive methods to index the similarity file. The parity file checker misses the hash indexing value, and the data be stored separately for increased storage. The non-related parity data block for each file holds individual data block [17], repeated calculation blocks PFP (Per-File Parity) increase the non-relational blocks, timing consumption, and chunk errors fail to produce the accuracy.

The Learning-based Indexing and Prefetching Approach for Data Deduplication presence of sparse index, a technique used to solve large-scale sampling, is backed up, and the flow using the inherent locality block-based deduplication solutions problems faced [18]. The complexity of content-based deduplication fails to reduce the leverage of data presented in the backup medium. Also, the Leveraging Content Similarity to Optimize Collective on-Demand Data Access functions to spread the contents of the shared data set on a large scale to manage the storage space [19]. It eliminates duplicate data and improves storage efficiency and optimization techniques capacity. The Bucket-based data deduplication technique for the big data storage systems. The semantic-based information to the cloud. In that, hash-based deduplication is used to detect the duplicate content, which is similar to a block as a bucket [20]. It reduces the detection accuracy, which doesn't consider the content source similarity measure.

The author explores a proposed D3M middleware integrating Redhat's Linux Device Layer Compression and VDO (Virtual Data Optimizer). This two-tier detection support provides both client-side and server-side detection [21]. The author focuses on designing a system that will be equally beneficial to cloud storage providers. For the user's benefit, the data is encrypted before uploading the file to the storage server, which enhances the security model of the data stored in the cloud [22]. On the other hand, the cloud storage capacity is used optimally by avoiding unnecessary storage of duplicate files on the server.

2.1 Challenges and Issues

With the increasing size of files and the number of data in cloud storage increases, the storage becomes high in multi-copy backups and files during upload in a centralized cloud. However, standard methods of cloud storage are currently used to detect quantitative identification content in traditional methods. Semantic-based file deduplication requirements were required for different sizes in the data environment to make relation identification. Due to this fact increasing cost, storage size, and access to on-demand services. The primary relational analysis methods only use text features, file names, and format size to cluster the documents but miss the semantic relations between the document texts. So storage increases in the cloud will make higher complexity.

Deduplication doesn't remain the Chunks and blocks miss the sequence of repeated Block id leads data dimensionality. The earlier approach cluster the documents under specific classes but misses the subclasses because of replication in the similar document and file sizes. The methods suffer from data duplication in documents and duplicate data conventions. Due to this fact, documents replicate to higher storage and memory consumption, increasing the cost to maintain and find similar files. The performance on storage resembles a lower precision-recall rate, which makes data fragmentation produce higher time complexity and storage space.

2.2 Contribution of the Work

The main contribution is to reduce the centralized cloud storage based on finding duplicate content and remove the similarity content to improve cloud storage management. The contribution of this research is based on redundant storage by accessing multiple file contents, hashing techniques to achieve a balance between cloud storage, file types, volume storage capacity, error tolerance requirements objective data backup methods, which are to improve cloud storage performance. The hash-based indexing semantic relational approach is used to enhance the deduplication accuracy.

3 Hash-Indexing Block-Based Deduplication Based on Segmented Bind Linkage Methods

Amazon S3 (AWS) is a large-scale data storage medium with duplicate content due to the large amount of data stored. The proposed deduplication in cloud storage management is based on the document content level to find the duplicate file and optimize the storage. The Hash-Indexing Block-based Deduplication method is implemented to reduce the dimension of file contents searching, which makes an order of the document list. The contents are compared to reduce the storage size, combining file-level and chunk-level deduplication schemes based on file content to reduce lookup overhead by reducing metadata. By preparing the files for comparison using the segmentation concept, the files are segmented into the number of blocks. The block-based document comparisons make redundancy levels to analyze the similar content in other documents. The indexing methods routinely order the list of blocks in the storage whether the files are processed multiple times. Based on the indexing list, the relational documents are analyzed using the semantic content source deduplication, which finds the number of count terms present in the documents. Since the documents have been calculated with the document, distance-vector weight measures between the Documents are compared with similarity methods for redundant storage. This storage space comes under the related file grouping based on the clustering depending on document weightage and making a similar document to manage the data overlay to reduce the duplicate files.

By estimating the vector, dependencies are determined to make the cluster with document weightage values. The files are gathered into bins dependent on their size under cluster indexing. Fig. 2 shows the deduplication process of the proposed system. Applying the content filter reduces the system's metadata overhead in target storage before performing the deduplication process. The filters get the exact equivalence in file properties to remove the data from the index list to reduce the comparison among the

original and duplicate files. Then the cluster group has targeted weights, making a comparison under record linkage concepts because weights are linked with another cluster group.

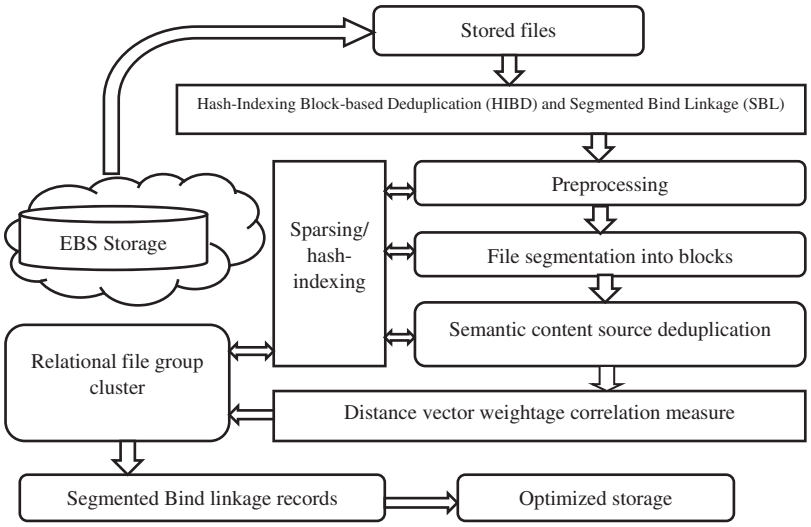

Figure 2: Workflow for the proposed system

3.1 Preprocessing

In this stage, the system loads several documents from storage directories that hold the start of data points. The point is to find a list of individual files and names of the terms that are effective and similar information to be obtained through the Sparse Indexing Augmentation (SPD) methodology. This finds the size forums about file content and name to remove exact duplicates. They perform cleaning, stemming, and Tokenization, which then deletes every data word that has been blocked and executed, and this method is tested on all dimensions. The sparse creates redundant data to increase the processing file easier to get similarities. If any data points are identified as incomplete, they will be removed from the data. At the preprocessing end, make indexing the order of the file to compare the files.

Algorithm: Preprocessing-Sparse Indexing Augmentation

Input: Collective text document D

Output: Redundant Index List

Step 1: initialize the doc file D→d1, d2, From Directory Index List (Idl)

Step 2: Select the Reader: The corpus reader Cr←D

Step3: For each D→Do property and read content

If check doc Text, size, file name Equalize content: The empty text

Remove the document from the corpus

Keep singular file index list SG←Idl

Singular Get (SG) list, SG List++;

End if

End for

(Continued)

Algorithm (continued)

Step 4 Get SG→ d1, d2,..Document = Reader.GetDocument

Create Temp Content holder Tch← SG(d1) do ++

For each D and compare files, D (n)

Remove stop words, stemming, cleaning

D++ Get document data content terms (Pts)

Pts→Check Text = doc(Text)

Step 5. Relist the doc File to make the index.

Step 6: return Pts←Index list Idl

The above preprocessing reduces the dimension among the most duplicate copy data, making similar files check with content verification. The document received the reduction of non-redundant terms of access to closeness relations before processing the document's non-relational terms.

3.2 File Segmentation into Blocks

In this stage, the file is split into several blocks, which makes for a better comparing another file to find the duplicates. By reducing the file size, segmentation makes the fixed block of size hold the preprocessed unique file content in the form block. The block size is 4 kb fixed because the contents are comparable to other indexing blocks. Further, the segmented block is hazed into an indexing block for comparing another file block.

Algorithm: Segmentation Block

Input: Preprocessed files→ Preprocess term set (Pts)

Output: Number of Blocks and index

Begin

Step 1 compute Each file from PtsIndexDh←(d1, d2…idn);

Step 2 For i = 1 to n Do

Read Hash id→Hid

Split File 4 Kb→[d1(1), d2(2),…]

Return Hid← Block countDbkc (i..n);

Step3 Check the Block content rate count term indexed

Step 4 End

Step 5 return Hid ←Dbkc(i)

End

The above algorithm returns the Data block counts (Dbkc), which are used to compare the documents to find the duplicates for further analyzing duplicate content.

3.3 Semantic Content Source Deduplication

Each file is compared in this stage with a related content count group. Consider the term set identified in the preprocessing stage where the data blocks are considered preprocessed term set (Pts) get blocked from Dbkc (i), which contains K number of terms. The number of occurrences of the terms in the Semantic class Sc in the ontology O has been measured as follows:

$$\text{Number of terms occurrence NT} = \sum_{i=1}^{size(Tts)} Pts(Dbkc(i) \in O(sc) \tag{1}$$

From comparisons, the semantic term equivalence to the new term (NT) is averaged into the size of the blocks. From the above Eq. (1), each block is compared with other blocs to count the occurrence. Now the value of Semantic Count Weight (SCW) towards the Semantic class (Sc) is measured as follows:

$$SCW = \frac{NT}{size(\sum Terms \in O(sc))} \tag{2}$$

The value of SCW towards different semantic classes can be measured based on the above Eq. (2). According to the value of SCW, the concept with maximum SCW can be selected as a result. Further, the relational documents estimate the distance vector weightages between the number of count terms present comparison file and the target file.

$$\sum = \begin{vmatrix} a_{ij11} & a_{ij21} & \ldots & a_{ij1m} \\ a_{ij12} & a_{ij22} & \ldots & a_{ij2m} \\ \ldots & \ldots & \ldots & \ldots \\ a_{ij1n} & a_{ij1n} & .. & a_{ij3m} \end{vmatrix} \tag{3}$$

Before that, the size of the file is varied by increasing the dimensionality, and the proposed make dimensionality by splitting the files into blocks by matrix representations from block-wise an (i, j) as row and column representation which is shown in Eq. (3) make cosine indexing similarity by comparing the blocks, after that the similarity be marginalized into distance vector.

$$\text{Similarity} = \cos(\theta) = \frac{A.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4}$$

Distance vector measures similarity relative cluster group based on the cosine representation create adjacent links between A and B between the similar documents. Eq. (4) shows the similarity measure from modulo representation. The cosine of 0° is 1, which is observed by max distance weight, and cluster weight values less than 1 depending on the relation of margins between two clusters is (0, π] radians. If the radii of a cluster depend on the cluster centroid between two clusters under the marginal coverage weight, two cluster vectors oriented at 90° relative to each other have a similarity of 0, and two vectors opposed have a similarity of −1, the independent value is based on the cluster value equivalence.

Cluster similarity documents are based on the vector weightage, and the average divisible value is fixed to group the similarity files into the number of clusters to be indexed.

3.4 Segmented Bind Linkage Redundancy

In this stage, cluster margins are combined to find the marginal relevance between the clusters. The data redundancy estimates the semantic correlation using the average mean and standard deviation values to represent the cluster closeness using the distance vector weightage correlation. Based on the Cluster group weightage mean average, the clusters are segmented into subgroups. Let us consider semantic clusters. It creates links between the clusters to compare the documents and return the distance matrices.

For each cluster, '$Cl = cl_1, cl_2, \ldots, cl_n$' and semantic counts terms grouped into clusters presented as semantic cluster weight (*ScwCl*) are denoted as Eq. (5) given below,

$$Cl = \sum_{i=1}^{n} ScwCl_i \tag{5}$$

The relative content words in the corresponding cluster hold similar contents in another cluster, with additional weight given below.

$$\text{The cluster weight is } Clw = \sum_{i,j=1}^{n} ScwCl_i w_{ij} \tag{6}$$

Corresponding clusters Cl_i be represented from W_{ij} of document content with the relative form of *Clw*. The variations from the equivalence link compare to the clusters '$cl_1, \ cl_2, \ cl_3$' to form a relative document cluster group. Once the comparison is based on the stream of correlation measures by relative content remains in the cluster singular file. Eq. (6) shows the estimation of cluster weight. The weightage initiates i and j remains the corresponding strength of size in Relative Set (Rs) forum similar content.

$$Rs = \sum_{i,j=1}^{n} ScwCl_i(Fw_{ij}) \tag{7}$$

The document has similar content with singular presentation of file content '*Rs*' is evaluated, which is shown in Eq. (7) by the frequent semantic match case by the term frequency '*F*' of words 'w_{ij}' in-group of paired document 'Cl_i'. The stand deviation ad means are estimated to correlate as follows,

$\mu\ (Mean) = ScwCl_i \sum_{i,j=1}^{n} \frac{w_{ij}}{n}$, resembles the document on similarity content available on relational documents. This points to the centroid coverage between the two clusters and creates margins to estimate the deviated values. Eq. (8) below shows the deviated cluster point's representation estimated from standard deviation.

$$\sigma\ (Standard\ deviation) = \sqrt{\frac{1}{n-1} ScwCl_i \sum_{i,j=1}^{n} (w_{ij} - \ \mu)^2} \tag{8}$$

The deduplication means the singular transformation with relatively compared with the similar cluster objects as given below.

$$\text{Distance flow } (DFS) Dist\ (ScwCl_{Wij}, \ \ w_{ik}, \ \ldots, w_{in}) = \frac{n * D(w_{ij}, \ \ w_{ik}, \ \ldots, w_{in})}{\sum_{i,j=1}^{n} D(w_{ij})} \tag{9}$$

The document contents are stored to check the cosine transformation with the stream of file content weightage ordered and grouped into a singular cluster. Eq. (9) shows the distance flow between the file variations based on the weights. The cosine transformation remains the cluster weights which they fix dimensionality checks in multi-point file progress to make marginal clusters. The marginal cluster ranks the order of the similarity measure in which the content is duplicated to similar content.

3.5 Hash-Indexing Block-Based Deduplication

It creates the Hash grouped index for the cluster based on similar files containing the content block. Each block is considered a target block and compares with another queuing block transform matrix content indexing block. The Deduplication process was read to determine if the similar cluster in stored data depended on the weightage values. Suppose the block generated in this method already has storage considered cluster form compared with different clusters. In that case, the non-related cluster will be discarded due to the low value, and only one index will be created of all the files from which the cluster weight margins are analyzed to find the duplicates.

Algorithm: Compute deduplication transformation

Input: Initialized the dataset D = DFS→ $\{d_1, d_2, d_3,..d_n\}$ with co-references resolved.

Output: Redundant cluster data:

Step 1: Compute dataset forums:

For each cluster document, D ←ScwCl do create a block stream from centroid values

For each document w in d do

Remains the considering cluster for the document content possible senses using the semantic case content weight comparison.

Step 2: Finding a cluster closeness from most similar clusters and select the margins from the centroid cluster

If cluster Count == 1 on comprehensive max weight (w)

Substitute the manuscript size by corresponding cluster id

End if

Step 3: Calculate cosine transformation on the discrete document by byte streams

If document index count > 1 document similar

Simulate the singular document content weight MAX form D

Remains Comprehend data (Cd)

End if

Step 4: If the terminal index is singular as the comprehensive cluster, else repeating

Select the compacted similarity value as a suitable Max weightage MAX→Cds

Substitute the document w ←Cd by matching cluster based on the maximum weight matching cluster.

Check similar clusters to form deletion. And make a similar comprehensive file reorder.

Step 5: End if

End for

End for

The cosine transformation remains compared to the comprehensive file to find the related cluster. The algorithm finalizes the duplicate indexing by comparing the cluster's maximum weights to make a singular comprehensive cluster. It supports cluster level weight comparison based on centroid value and makes its redundant spatial storage compared to these other reprint approaches. This weightage highlights the marginal dependencies of duplicate weight to be considered as duplicate files to make deletion to remove from storage.

4 Results and Discussion

The proposed storage management was implemented in AWS cloud management as a homogeneous storage reduction method, and its performance evaluation was used to reduce cloud storage. This method implements a real-time cloud environment with cloud services in circles using the tool depending on a visual configuration of Internet data such as document forums. The proposed method measures the

consequences and compares the file balancer (Per File Parity) with other methods. Table 1 below shows the parameters and values processed for deduplication in the cloud.

Table 1: Parameters and values processed

Parameters	Values processed
Cloud environment	AWS, S3 cloud storage
Data used and size	Content file type,=<= 5 Gb
Simulation framework	Visual studio/c#.net, VS 4.5
Output type definition	Redundant storage space

Table 1 shows the presentation of the technique has been restrained in gathering, correctness, time complexity, and recall to produce the best performance under different levels of testing.

Fig. 3 shows the precision performances from testing the various data sizes using different methods. Estimating the precision rate by actual relevant data is intersected with retrieved relevant data by average total data in the file. The results tested 5 GB of data, the comparison was carried out with differential methods, and the performance of the proposed HIBD method produced 89.7% best performance compared to other systems like CS-IBS make 76.3%, HBTD-LSH has 84.7%, and PFP produces 86.9%.

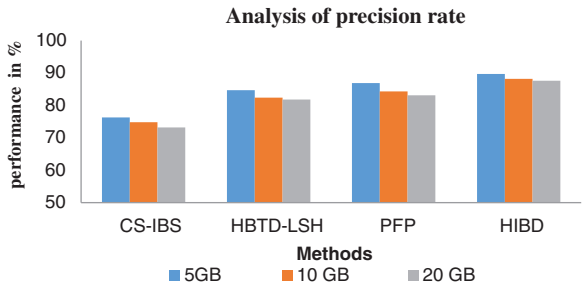

Figure 3: Analysis of precision rate

Table 2 shows the performances result in comparing the accuracy of produce in various ways. The proposed HIBD performs the best precision rate accuracy compared to the other methods.

Table 2: Analysis of precision rate

Analysis of precision rate in %				
Storage/methods	CS-IBS	HBTD-LSH	PFP	HIBD
5 GB	76.3	84.7	86.9	89.7
10 GB	74.8	82.4	84.3	88.2
20 GB	73.2	81.8	83.1	87.6

Fig. 4 shows that Recall performance has been measured in various ways. The proposed HIBD produces the best performance up to 87.6% compared to the methods by testing 5 GB of data, the CS-IBS have 75.3%,

HBTD-LSH makes 78.8%, and PFP produces 84.3%. And the proposed HIBD algorithm attains improved recall performance is higher than other methods.

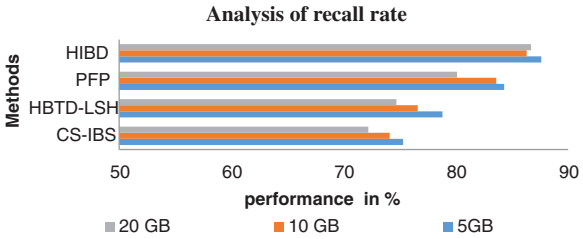

Figure 4: Analysis of recall rate

Table 3 recounts a variety of state comparison techniques. The proposed HIBD system creates a high return state with a maximum rating of 87.6% compared to another system.

Table 3: Comparison of recall

Impact of recall in %				
Storage/methods	CS-IBS	HBTD-LSH	PFP	HIBD
5 GB	75.3	78.8	84.3	87.6
10 GB	74.1	76.6	83.6	86.3
20 GB	72.2	74.7	80.1	86.7

Fig. 5 shows the measure of the rate of incorrect redundancy rate production by various methods, and it is presented. The proposed HIBD produces the best performance up to 4.1% compared to the methods by testing 5 GB of data, the CS-IBS have 6.6%, HBTD-LSH has 5.3%, and PFP has 5.2%. The results of the proposed HIBD algorithm show that it has a false classification rate less than other methods.

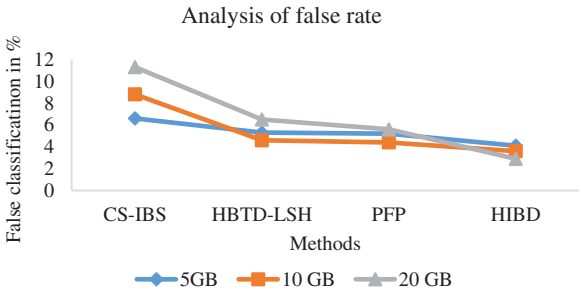

Figure 5: Analysis of false classification rate

To measure the rate of incorrect rate production by various methods is presented in Table 4. The results of the proposed HIBD algorithm show that it produces a false classification rate less than other methods.

Fig. 6 presents the results of various methods' storage performance rates in memory of performance analysis. The proposed HIBD produces the best performance up to 72.6% compared to the processes by testing 5 GB of data, the CS-IBS have 56.1%, HBTD-LSH produces 59.2%, and PFP produces 63.2%. The proposed method by minimizing data replication and reducing memory consumption.

Table 4: Analysis of false rate

Analysis of false rate %				
Storage/methods	CS-IBS	HBTD-LSH	PFP	HIBD
5 GB	6.6	5.3	5.2	4.1
10 GB	8.8	4.6	4.4	3.6
20 GB	11.3	6.5	5.6	2.9

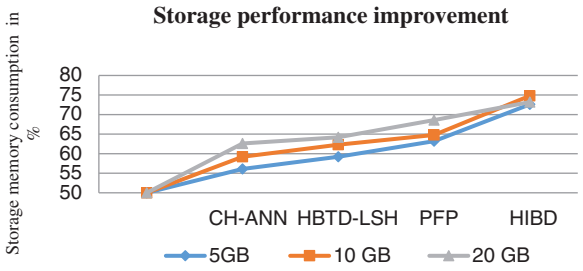

Figure 6: Storage performance rate

Above Table 5 shows the changes in the data set of measures that were compared, analyzing the performance of the amount of storage performance rate. As a result of the comparison, the proposed system reduces storage consumption more than other methods.

Table 5: Storage performance rate

Storage performance rate in %				
Storage/methods	CS-IBS	HBTD-LSH	PFP	HIBD
5 GB	56.1	59.2	63.2	72.6
10 GB	59.2	62.3	64,8	74.8
20 GB	62.6	64.2	68.6	73.2

Fig. 7 shows the Measuring the time process preparation by different methods. By testing 5 GB of data, the proposed system produces 3.8 seconds (s) execution time compared to the CS-IBS method has 6.3 (s), HBTD-LSH makes 5.3 (s), and PFP has 4.7 (s). The HIBD attained a high-performance rate in redundant execution time shown in Fig. 7 presents the less time over time gave more than the other methods reduced.

$$t(i) = i - 1 \tag{10}$$

The above equation analyses the time complexity process to find the duplication files in the cloud. Let's assume t refers to the overall time process of i^{th} files processing iteration. Then 1 refers to the least end time to detect deduplication files.

Above Table 6 shows the time complexity for identifying the duplicate files in different storage spaces. The proposed and existing methods are compared with varying storage sizes shown in Fig. 7. The process of comparison takes the execution time at 'n'-number be in O (n) meantime comparison by identifying the duplicates with least time execution in the proposed system.

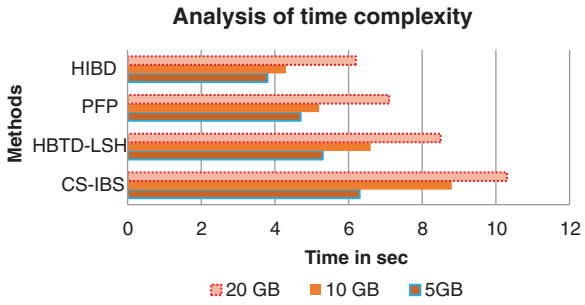

Figure 7: Analysis of the time complexity process

Table 6: Analysis of the time complexity for identifying the duplication files

Analysis of time complexity process (s)				
Storage/methods	CS-IBS	HBTD-LSH	PFP	HIBD
5 GB	6.3	5.3	4.7	3.8
10 GB	8.8	6.6	5.2	4.3
20 GB	10.3	8.5	7.1	6.2

Fig. 8 defines the Analysis of cloud storage optimization performance with various cloud storage platforms like AWS (S3), Some and pcloud. The proposed method quickly detects duplicate documents and consumes less cloud storage space.

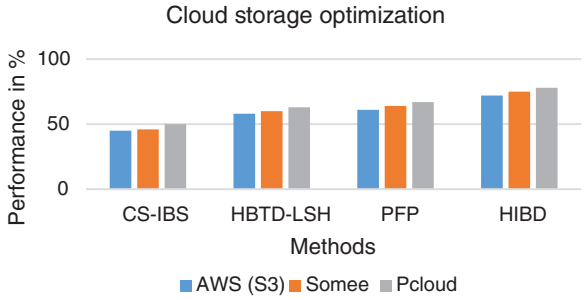

Figure 8: Analysis of cloud storage optimization performance

5 Conclusion

To conclude, this storage management improves the redundancy in data storage based on Hash-Indexing Block-based Deduplication to strengthen the cloud distributed storage size with semantic bind linkage technique in a single stream of a duplicate detection system. Hash-Indexing Block-based Deduplication (HIBD) based on Segmented Bind Linkage Methods performs as well reducing storage in the cloud environment. As well relational terms are proceeds of the high performance compared to other existing systems, they reduced duplication based on relation identification. The proposed solution is optimal for all cloud storage platforms like AWS, Azure, somee, and p cloud. The proposed system produces high performance in precision rate up to 89.7% by testing 5 GB of data and recall rate have 87.6% well than other methods. The proposed algorithm efficiently reduces the cloud storage compared with previous algorithms.

Funding Statement: The authors received no specific funding for this study

Conflicts of Interest: The authors declare they have no conflicts of interest to report regarding the present study.

References

[1] W. Xia, H. Jiang, D. Feng and Y. Hua, "Similarity and locality based indexing for high-performance data deduplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 1162–1176, 2015.

[2] A. Khan, P. Hamandawana and Y. Kim, "A content fingerprint-based cluster-wide inline deduplication for shared-nothing storage systems," *IEEE Access*, vol. 8, no. 3, pp. 209163–209180, 2020.

[3] M. Padmanaban and T. Bhuvaneswari, "An approach based on artificial neural network for data deduplication," *International Journal of Computer Science and Information Technologies*, vol. 3, no. 4, pp. 4637–4644, 2012.

[4] Y. Tan, H. Jiang, D. Feng, L. Tian, Z. Yan *et al.,* "SAM: A semantic-aware multi-tiered source deduplication framework for cloud backup," in *Int. Conf. on Parallel Processing*, San Diego, CA, USA, pp. 614–623, 2010.

[5] A. T. Clements, I. Ahmad, M. Vilayannur and J. Li, "Decentralized deduplication in SAN cluster file systems," in *Proc. of the Conf. on USENIX Annual Technical Conf.*, San Diego, California, pp. 1–8, 2009.

[6] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 9, pp. 1537–1555, 2012.

[7] M. Mishra and S. S. Sengar, "E-David: An efficient distributed architecture for inline data deduplication," in *Conf. on Communication Systems and Network Technologies*, Bengaluru, India, pp. 438–442, 2012.

[8] T. Botsis, J. Scott, E. J. Woo and R. Ball, "Identifying similar cases in document networks using cross-reference structures," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 6, pp. 1906–1917, 2015.

[9] D. Papadimitriou, G. Koutrika, Y. Velegrakis and J. Mylopoulos, "Finding related forum posts through content similarity over intention-based segmentation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1860–1873, 2017.

[10] D. Lanterna and A. Barili, "Forensic analysis of de-duplicated file systems," *Digital Investigation*, vol. 20, no. 4, pp. 99–106, 2017.

[11] J. Jayabharathy and S. Kanmani, "Correlated concept based dynamic document clustering algorithms for newsgroups and scientific literature," *Decision Analysis, Springer*, vol. 1, no. 3, pp. 1–21, 2019.

[12] R. Kaur, I. Chana and J. Bhattacharya, "Data deduplication techniques for efficient cloud storage management: A systematic review," *The Journal of Super Computing*, vol. 74, no. 5, pp. 2035–2085, 2018.

[13] W. Xia, D. Feng, H. Jiang, Y. Zhang, V. Chang *et al.,* "Accelerating content-defined-chunking based data deduplication by exploiting parallelism," *Future Generation Computer Systems Syst.*, vol. 98, no. 10, pp. 406–418, 2019.

[14] M. Kaur and J. Singh, "Data deduplication approach based on hashing techniques for reducing time consumption over a cloud network," *International Journal of Computer Applications*, vol. 142, no. 5, pp. 4–10, 2016.

[15] C. I. Ku, G. H. Luo, C. P. Chang and S. M. Yuan, "File deduplication with cloud storage file system," in *IEEE 16th Int. Conf. on Computational Science and Engineering*, Sydney, NSW, Australia, pp. 280–287, 2013.

[16] T. Tran, D. Pham, Q. Duong and A. Mai, "An adaptive hash-based text deduplication for ADS-B data-dependent trajectory clustering problem," in *IEEE-RIVF Int. Conf. on Computing and Communication Technologies (RIVF)*, Danang, Vietnam, pp. 1–6, 2019.

[17] S. Wu, B. Mao, H. Jiang, H. Luan and J. Zhou, "PFP: Improving the reliability of deduplication-based storage systems with per-file parity," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 2117–2129, 2019.

[18] G. Xu, B. Tang, H. Lu, Q. Yu and C. W. Sung, "LIPA: A learning-based indexing and prefetching approach for data deduplication," in *Symp. on Mass Storage Systems and Technologies (MSST)*, Santa Clara, CA, USA, vol. 23, no. 3, pp. 299–310, 2019.

[19] B. Nicolae, A. Kochut and A. Karve, "Discovering and leveraging content similarity to optimize collective on-demand data access to IaaS cloud storage," in *IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, China, vol. 23, no. 4, pp. 211–220, 2015.

[20] N. Kumar, R. Rawat and S. C. Jain, "Bucket based data deduplication technique for big data storage system," in *2016 5th Int. Conf. on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, pp. 267–271, 2016.

[21] H. B. Chen, S. Tang and S. Fu, "A middleware approach to leverage the distributed data deduplication capability on HPC and cloud storage systems," in *IEEE Int. Conf. on Big Data (Big Data)*, Atlanta, GA, USA, pp. 2649–2653, 2020.

[22] S. Muthurajkumar, M. Vijayalakshmi and A. Kannan, "An effective data storage model for cloud databases using temporal data deduplication approach," in *Eighth Int. Conf. on Advanced Computing (ICoAC)*, Chennai, India, pp. 42–45, 2017.