International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

Cross-Layer Fragment Indexing based File Deduplication using Hyper Spectral Hash Duplicate Filter (HSHDF) for Optimized Cloud Storage

K. Geetha[1*], Dr. A. Vijaya[2]
[1]Research Scholar (PT), PG and Research Department of Computer Science
Periyar University, Salem-636011
Mail-id: geethagacslm7@gmail.com
[2]Asst. Professor and Head, Department of Computer Applications
Sri Meenakshi Government Arts College For Women
Madurai-625002
Mail-id: vijayakathiravan@gmail.com

Abstract: Cloud computing and storage processing is a big service for maintaining a large number of data in a centralized server to store and retrieve data depending on the use to pay as a service model. Due to increasing storage depending on duplicate copy presence during different sceneries, the increased size leads to increased cost. To resolve this problem, we propose a Cross-Layer Fragment Indexing (CLFI) based file deduplication using Hyper Spectral Hash Duplicate Filter (HSHDF) for optimized cloud storage. Initially, the file storage indexing easy carried out with Lexical Syntactic Parser (LSP) to split the files into blocks. Then comparativesector was created based on Chunk staking. Based on the file frequency weight, the relative Indexing was verified through Cross-Layer Fragment Indexing (CLFI). Then the fragmented index gets grouped by maximum relative threshold margin usingIntra Subset Near-Duplicate Clusters (ISNDC). The hashing is applied to get comparative index points based on hyper correlation comparer using Hyper Spectral Hash Duplicate Filter (HSHDF). This filter the near duplicate contentdepending on file content difference to identify the duplicates. This proposed system produces high performance compared to the other system. This optimizes cloudstorage and has a higher precision rate than other methods.

Keywords: Cloudstorage; Deduplication;Cross-Layer Fragment Indexing; Syntactic Parser;Near-Duplicate Clusters.

I. Introduction

A storage system contains redundant copies of data within the same File or subfile region. With deductive technology, you can use this redundancy to reduce the space required to store files on your file system [1]. A scalable and reliable distributed system that supports data reduction has recently become popular for backup and archival data storage. This technology can be used in primary storage. Our research aims to develop a file type-aware reduction method to improve storage system capacity [2].

To understand the relationship between duplicate content and file types, we began our research by focusing on the relationship between the amount of duplicate content that can be extracted between different files [3]. Data deduplication is a new technology that introduces ways to reduce storage usage and efficiently handle data duplication in backup environments. In cloud data storage [4], deductive technology plays an important role in virtual machine architectures, sharing networks, social media processing of structured and unstructured data, and disaster recovery. Truncation algorithms are important in deductive scenarios as they are the first step in obtaining efficient data reduction rates and throughputs [5].

Deduplication is a technique that depends on the amount of duplicate data. Store duplicate data only once instead of repeatedly. An immediate practical question is how much duplicate data is created in the data center [6]. These issues can lead to poor server and application performance and quality, increasing operating costs [7]. So, to solve this problem and process the data properly, the data center has a dedicated deductive concept.

The deduction process is the basis of the rate and efficiency of the deduction system. Reads the entire existing metadata and checks for new chunks. The fragment index

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

table captures all available fragment information [8]. The index table is first searched to identify whether the new fragment's information is new or old. This process requires a lot of computer usage and disk access.

Deduplication technology recovers files for processes at the File or block level. At the file level, the entire File is considered a block, but at the block level, the data is divided into fixed or variable lengths. Each of these chunks derives a unique identifier from the hash method.

The main challenges are identifying maximum duplicate segments and selecting the storage nodes for distributing fragments of files. 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, and objective data backup methods to improve cloud storage performance [9]. The hash-based Indexing semantic relational approach is used to improve the deduplication accuracy.

File-level defragmentation is also called whole-file defragmentation. This way, individual files are treated as chunks, and files are not split into smaller chunks [10]. Only one index is generated for each File and compared to the stored metadata. Because only one index is created for each File, this copy system reduces the amount of fragment metadata stored in memory.

Compared to other fragment methods, this method requires less space and significantly reduces the total number of fragments required for comparison. The entire list can be stored in the main memory to reduce system resource usage. However, this method treats the File as a new part. If only a small part of the file changes, it will affect the detection algorithm. Calculates the hash value of the entire File, not just the changed part.

The inline process first removes duplicate data before storing it in storage. Otherwise, the deductive process occurs when the received data comes to storage. The post-processing method stores the received data on the staging storage disk and performs the derivation process.

II. Related work

Various works have addressed this deduplication methodology to optimize the storage problem. This is explained briefly in the literature review section, which various methodologies implement.

Managing cloud storage services is an important aspect of information management that stores / retrieve data based on distributed data centers [11]. Duplications add storage space because data and file structures stored in data centers containing identical information are multiple copies. Similarity data analysis does not detect duplicate content, so systems with potential compression do not perform efficient data reduction. This complex nature increases memory consumption in terms of cost.

Most storage services reduce the storage by finding duplicate data is essential to managing storage access. [12] described Data Deduplication as a more popular technique to attain space-efficient finding duplicates [13]. Deduplication remains the storage that only identifies those using hash values for comparison of a block of data and creates a logical pointer to another copy of the redundant data; actual data leads be stored as a copy to excess storage data [14].

The wrapping storage be optimized with conceptual-based similar data deduplication technology is recently got more attention for finding duplicate content; this is the most popular and effective method for the space backup storage system. The main challenge of centralized Deduplication is an extension of the fingerprint index search [15]. A scalable data deduplication system produces higher effectiveness for finding non-redundancy data because similar data produce very low overhead, lower throughput and balanced load, and stream based on storage access.

Mostly the clustering concepts are used in the deduplication file system to host the virtual machine file system [16]. This file system doesn't produce efficient storage and leads to wasted storage space, including a large number of duplicate blocks, increasing the cache footprint of the storage array [17]. These problems with deduplication address storing a single instance of each unique data block are achieved by sharing data among all sources [18].

The duplication detection techniques behind the forum posts become a problem associated with the increasing storage space. In contrast, the conventional method doesn't compare the contents of the binding posts to the content of the relevant document [19].

The content-based duplicate measures the similarity by checking the Content present in the File to make the subsidiary Intention-Based Segmentation method (CS-IBS) implemented by [20]. The content-based segmentation increases the probability of finding features with more significant terms. This increase the relatedness of non-relational feature to reduce the deduplication accuracy.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

III. Proposed System

Towards the development of efficient Deduplication techniques are implemented to reduce the storage. For repetitive data, reference pointers are used, and unique data is stored in the storage node. This increases the detection rate of duplicate data.We propose a Cross-Layer Fragment Indexing (CLFI) based file deduplication using Hyper Spectral Hash Duplicate Filter (HSHDF) for optimized cloud storage. Initially, the file storage indexing easy carried out with Lexical Syntactic Parser (LSP) to split the files into blocks. Then comparative sector was created based on Chunk staking. Based on the file frequency weight, the relative Indexing was verified through Cross-Layer Fragment Indexing (CLFI). Fragment placement algorithms are used to place fragments on different storage nodes. Use de-coloring to select the node. They are largely eliminating duplication and providing high protection for data fragments. It selects non-adjacent nodes preventing unauthorized access to data by other users.

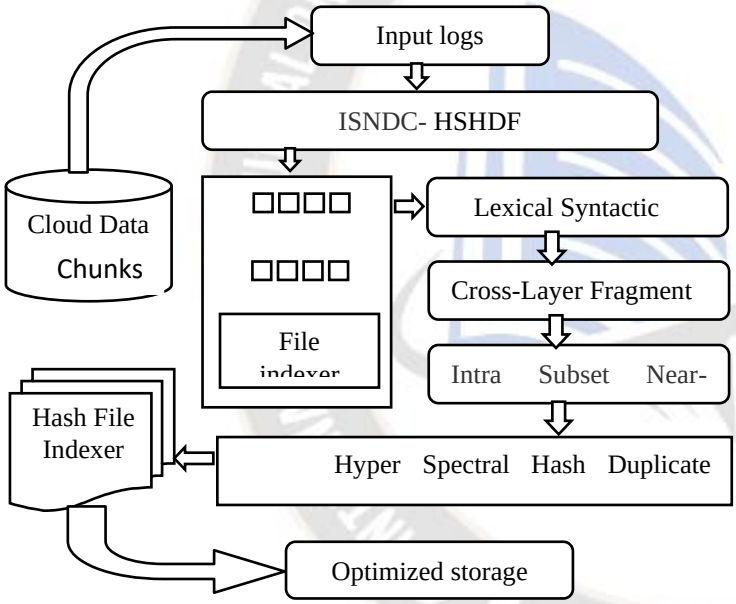

Figure 1 proposed architecture diagram ISNDC- HSHDF

Then the fragmented index gets grouped by maximum relative threshold margin using Intra Subset Near-Duplicate Clusters (ISNDC). Figure 1shows the proposed architecture diagram ISNDC- HSHDF. The hashing is applied to get comparative index points based on hyper correlation comparer using Hyper Spectral Hash Duplicate Filter (HSHDF). This filter the near duplicate content depending on file content difference to identify the duplicates.

3.1 Lexical Syntactic Parser (LSP)

In this stage, the preliminaries of the file index are verified based on the File's properties and its contents with format representation. This checks the intent properties of files indexed with staking queries.

$$\text{State matrix parser} = \begin{bmatrix} Fi_1 \\ Fi_2 \\ \cdots \\ Fi_n \end{bmatrix} \begin{bmatrix} Ci_1 & Ci_2 & \ldots & Ci_n \\ Cj_1 & Cj_2 & \ldots & Cn_1 \\ \ldots & \ldots & \ldots & \ldots \\ Cn_1 & Cn_1 & \ldots & Cn_n \end{bmatrix},$$

$Sp_{n=0} = 1, 2, 3 \ldots \ldots n$

This creates a state matrix Index, whether the files are split into tokenized indexes with original files. The Set of lexical tokens is formed based on the file size divided into File comparer staking (FCS) and file indexer. They make a syntactic rule for the comparison matrix taking Tcs→set={T1, T2,……Tn} be compared with other chunks. Each chunk gets a Difference matrix to compare blocks to other files. These are getting parsed to get the similarity index (Si)

3.2 Cross Layer Fragment Indexing (CLFI)

In this stage, the chunks are hashed with an internal layer file comparison using the FCS staking. The file indexer gets the blocks with other files—a general time complexity analysis of the project. Let S be the Staking withTcs set of inputs D that have a deterministic S and a stop property for D. Let W be the set of input weights, and |d| is a function of the size of the input of D. Let t (d) be the time cost function of S. On the other hand, the following two conditions apply at each size of W. A probability function is defined on the Set of inputs of size W and is denoted by P_W

By definition,

$$Fcs = S(Tcs) \sum_{|d|=w} P_W = 1 \rightarrow \{Ws1, Ws2, \ldots . WDn\}$$

Let 'w' be the Size of the file Block presented in chunks at the maximum comparison to reading the data at the time T is t(d) to the input W.mean time of data comparison in match case is,

$The\ average\ matchcase\ content\ (w) = \sum_{|d|=w} P_w(d) \cdot t(d) \rightarrow (H - index).$

This takes the sequential comparison of blocks with respective Tw. The probability gets maximum match case count terms is related to input file blocks P_{wkv}which is equal to chunk to verify the index at K state.

Let S (w)=
$\sum_{k\geq 0} k\big(staking\ (s) \sum_{|d|=w,t(d=k)} P_w(d)\big) \rightarrow (Max) \sum_{k\geq 0} kp_{wk} \longrightarrow (H - index)$

For each block of file content size S (w) of W, the time complexity limit of the input quantity s is equal to the mean value E (w) of the random variable TW. A part of the function E (w) forms the statistical properties of the time of the function. S is characterized by the standard deviation D (w) of

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

the variance functions V (w) and t (•), where w is in the range w.

$$\mathrm{V(w)} = \mathrm{S}\sum_{k\geq 0}(K - T_{ave}(w))^2\, P_{wk}$$

A probability function is defined on the Set of inputs of size w and is denoted by

By definition,

Probability to get file match case at P(s→)$\sum_{|d|=w} p_{w=1}$

For each quantity w of W, the range t (•) of the input of quantity w to the time cost function is a random variable. Tw denotes it. Assume that the random variable Tw is a natural number. Pwk denotes the probability distribution of Tw. For an input of size d, TW is the probability equal to k. The average time complexity is as follows.

$$Average\ T(w) = \sum_{|d|=w} P_w(d)\cdot t(d)$$

$$= \sum_{k\geq 0} k\left(\sum_{|d|=w,t(d=k)} P_w(d)\right) = \sum_{k\geq 0} kp_{wk}$$

For each size $_w$ in W, that is, the average time complexity of S for input size $_w$ is equal to the mean E(w), the statistical properties of the running V(w) and standard deviation D(w) of $_{Tw}$ with $_w$ ranging over W, where

$$\mathrm{V(w)} = \mathrm{S}\sum_{k\geq 0}(K - T_{ave}(w))^2\, P_{wk}$$

This returns the match case vectors for predominate average match case content equal to the chunks defined responsibility. This returns the average match case blocks of files returned to the content

3.3 Intra Subset Near-Duplicate Clusters (ISNDC)

In this content analysis, the average vectors are scaled into File content comparative features, and relative features are grouped into clusters to make Indexing. Then comparing the cluster with one another, be scaled into Intra subset values. Based on the feature dependencies, the Blocks are input depending on the file content size. This estimates the probability function is pointed by the input size w and is denoted by $P_{CB.}$ By the definition,

$\sum_{|d|=w} P_{wB} = 1,2,\ldots . n$
(1)

For each size w in W, the restriction of the time cost function *t(·)* to inputs of size w is a random variable;$_{Tw}$.the random variable $_{Tw}$ denotes it assumes natural numbers are valid. The probability that for an input *d* of size $_{w}$, $_{Tw}$ is equal to k.

Notice that the average time complexity is

$MAx(w) = \sum_{|d|=w} P_w(d)\cdot t(d) =$
$\sum_{k\geq 0} k\left(\sum_{|d|=w,t(d=k)} P_w(d)\right) = \sum_{k\geq 0} kp_{wk}$ (2)

For each size w in W, that is the average time complexity of S for input size $_w$ is equal to the mean value E(*w*) of the random variance function V(*w*) and standard deviation D(*w*) of $_{Tw}$ with $_w$ ranging over W, where

$$\mathrm{V(w)} = \mathrm{S}\sum_{k\geq 0}(K - T_{ave}(w))^2\, P_{wk} \qquad (3)$$

$$D(w)\sqrt{V(w)} \qquad (4)$$

These quantities determine how much the random variables $_{Tw}$ are concentrated around their mean values. the smaller the standard deviation, the better concentration of $_{Tw}$ around its mean value is

To find the statistical quantities E(w), V(w), and D(w), the method of generating function is used. the generating function for random variables $_{Tw}$ is

$$P_w(z) = \sum_{k\geq 0} P_{wkz^k}$$

With arguments and values being real numbers.therefore,

$$P'_w(1) = \left(\sum_{k\geq 0} Kp_{wk}\, z^{k-1}\right)(1) = \sum_{k\geq 0} kp_{wk}$$

From the above equation, the content similarity at exponential levels, $E(w) = P'_w(1)$

Next

$$P''_w(1) = \sum_{k\geq 0} k(k-1)P_{wk}\, z^{k-2}(1) = \sum_{k\geq 0} k(k-1)P_{wk}$$

From the above equation, we get vectored equivalence of content match case, $V(w) = \sum_{k\geq 0}(k - p'_w(1))^2 P_{wk} = P''_w(1) + P'_w(1) + P'_w(1)^2$

Based on the estimated weight, the duplicate files are discarded, and only one index will be created of all the files from which the original file index is to be created.

Algorithm:

Input: The dataset D= {d_1, d_2, d_3,..d_n} with co-reference resolved.

Output: paired redundant reduction data:

Step 1: compute dataset forums:

For each cluster document d in D, do create byte stream

For each document w in d do

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

Check the semantic cluster terms compared with other cluster groups based on distance flow.

From the distance flow weight, each cluster is ordered into Indexing

Step 2: Finding a semantic group pair of the most similar clusters and merging index values

If cluster Count ==1 on compression

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 compressed from D

Other non-similar data to compressed data Cd

End if

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

Select the compacted similarity value as a suitable sense

Substitute the document w ←Cd by matching cluster based on the recognized intellect data singular compression.

Step 5 End if

End for

End for

The cosine transformation checks the index values by checking the stream of files in the cluster. This makes the redundant comparison to improve the Deduplication. Finally, only one index file-sized value is compared with all indexed clusters to find the duplicates.This returns the vector weight of each block represented to the comparative chunk with the content match case.

3.4 Hyper Spectral Hash Duplicate Filter (HSHDF)

In this stage, Hyper spectral Deduplication makes an index-based search model to find the duplicate content and exploits stream of file content to make blocks. This Hashed Indexing creates a locality index to filter duplicate files. Such that needs the data filtering criteria performing the similarity content based on clustering to find the inconsistencies in data chunks depending on metadata forums.Let us consider T (n).assume that h(x) =j when an unsuccessful search for x happens. Denoting by $P_{nk}$the probability that the list H[j] has length k, we have

$$P_{nk} = \binom{n}{k}\left(\frac{1}{M}\right)^{k}\left(1-\frac{1}{M}\right)^{n-k}$$

Since the value *j* appears *k* times in a sequence $h_1,\ldots,h_n$ with the probability defined by the Bernoulli schema, now we have

$$P_n^{-}(Z) = \sum_{k\geq 0} P_{nk} Z^{k+1}$$

Which can easily be transformed into a simpler form

$$P_n^{-}(Z) = \left(\frac{z}{M}+1-\frac{1}{M}\right)^{n} z$$

By different (12), we have

$$P_n^{-1}(Z) = \left(\frac{z}{M}+1-\frac{1}{M}\right)^{n-1}\left(\frac{nz}{M}+\frac{z}{M}+1-\frac{1}{M}\right)$$

$$P_n^{-"}(Z) = \left(\frac{z}{M}+1-\frac{1}{M}\right)^{n-2}\left(\frac{n-1}{M}\left(\frac{nz}{M}+\frac{z}{M}+1-\frac{1}{M}\right)+\left(\frac{z}{M}+1-\frac{1}{M}\right)\left(\frac{n}{M}+\frac{1}{M}\right)\right)$$

Based on the above equation, we get redundancy to check the file properties of equivalent content. We get, $P_n^{-1}(I) = \frac{n}{M}+1$

Similarly, the Inverse comparison of each chinks is referred as $P_n^{-"}(I) = \frac{n(n-1)}{M^2}+\frac{2n}{M}$ and the exponential blocks comparison rate is,

$$E^{-}(n) = P_n^{-1}(1) = \frac{n}{M}+1$$

$$V^{-}(n) = P_n^{-"}(1)+P_n^{-1}(1)-\left(P_n^{-1}(1)\right)^2$$

$$= \frac{n(n-1)}{M^2}+\frac{2n}{M}+\left(\frac{n}{M}+1\right)-\left(\frac{n}{M}+1\right)^2$$

$$= \frac{n(n-1)}{M^2}$$

And denoting by $\alpha={}^{n}/_{M}$, that is, the definition of table H, we obtain

$E^{-}(n) = \alpha+1, V^{-}(n) \cong \alpha, D^{-}(n) \cong \sqrt{\alpha},$

To estimate$E^{+}(n)$, that is, the average cost in a successful case, consider the function $nE^{+}(n)$. Its value equals the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

number of steps performed when all n elements of 'A' are searched for. But the list of length k contributes $\frac{1}{2}k(k+1)$

Steps to the total. Consequently, since there are M lists, we have

$$nE^{+}(n) = M\sum_{k\geq 0}\frac{k(k+1)}{2}P_{nk}$$

and according to the definition of $P_n^{-1}(Z)$, we obtain

$$E^{+}(n) = \frac{M}{2n}\sum_{k\geq 0}k(k+1)\,P_{nk} = \frac{M}{2n}P_n^{-"}(1) = \frac{n-1}{2M}+1$$
$$\cong \frac{1}{2}\alpha + I$$

By this evaluation, the block size results related to other similarity files are indexed as duplicated and match case similarity levels. I.e., $E^{+}(n)$ produce higher results in comparative definition rate for each chunk.

Performance evaluation and its outcomes

The outcome of the deduplication system efficiency is based on the duplication ratio. Figure 1.9 explains the key process outcome results. It is calculated based on the total number of sizes before and after the Deduplication, as shown in equation 1.1.

$$TotalDedupe = \frac{OriginalSize}{DedupeSize}$$

A good deduplication system provides a more effective ratio and detects more duplication elements. The throughput is calculated based on the number of Hits on the main memory for the index lookup, as shown in equation 1.2. Data Skew concepts are purely used in the data directing nodes. It is calculated on Max node utilization divided by the average node utilization as shown in equation 1.3.

$$DataSkew = \frac{MaxNodeUtilization}{AverageNodeUtilization}$$

Hit ratio is more, and I/O access less means the deduplication system's throughput is more. Data Skew concepts are purely used in the data directing nodes. It is calculated on Max node utilization divided by the average node utilization as shown in equation 1.3. Hit ratio is more, and I/O access less means the deduplication system's throughput is more.

$$EffectiveDedupe = \frac{TotalDedupe}{DataSkew}$$

In the distributed or cluster model system, the data traversal between the node is challengeable, and it can affect the data deduplication performance. So the Data Skew also needs to be handled properly. The efficiency of the system is based on the total Deduplication and Data Skew, as shown in the equation.

IV. Results and Discussion

The results are tested with an Amazon web service (AWS) cloud environment, making it an EC2 server instance with EBS storage. The collected content files are grouped into the duplicate dataset to make fixed storage for deduplicationredundancy. The proposed HSHDFimplementation applies the block-based comparison with indexed hash table files called lookup indexing tables. This result test with a confusion matrix to test the efficiency as precision, Recall, false rate, and storage optimization accuracy compared with other methodsSiLo, CFBC, andEHFDD. 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, EBS cloud storage
Data used and size	Content file type,=<= 5Gb
Simulation framework	Visual Studio/ c#.net , VS 4.5
Output type definition	Redundant storage space

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.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

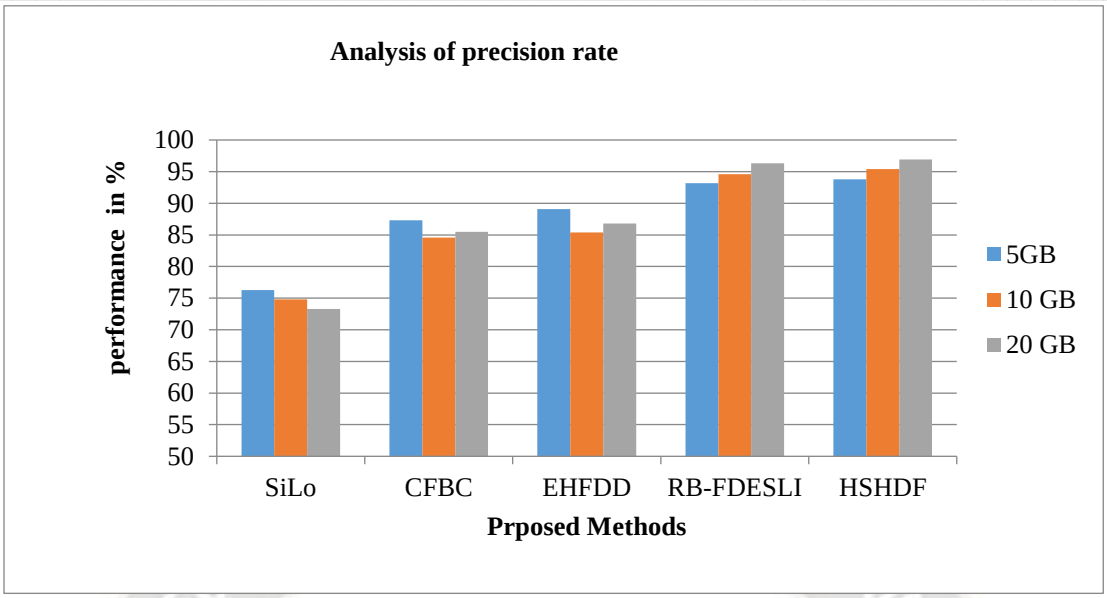

Figure 3: Analysis of precision rate

Figure 3 shows the precision performances from testing the various data sizes using different methods. The proposed SLCSD produces the best performance up to 93.8 % compared to the methods by testing 5 GB of data, RB-FDESLI produces 93.6 %, SiLoproduces 76.3 %, CFBCproduces 87.3 %, and EHFDD produces 89.1 %.

Table 2: Analysis of precision rate

Analysis of precision rate in %					
Storage /methods	**SiLo**	**CFBC**	**EHF**	**RB-FDESLI**	**HDF**
5GB	76.3	87.3	89.1	93.2	93.8
10 GB	74.8	84.6	85.4	94.6	95.4
20 GB	73.2	85.5	86.8	96.3	96.9

Table 2 shows the performances result in a comparison of the accuracy of produce in various ways. The proposed HSHDFperforms the best accuracy in precision rate compared to the other methods.

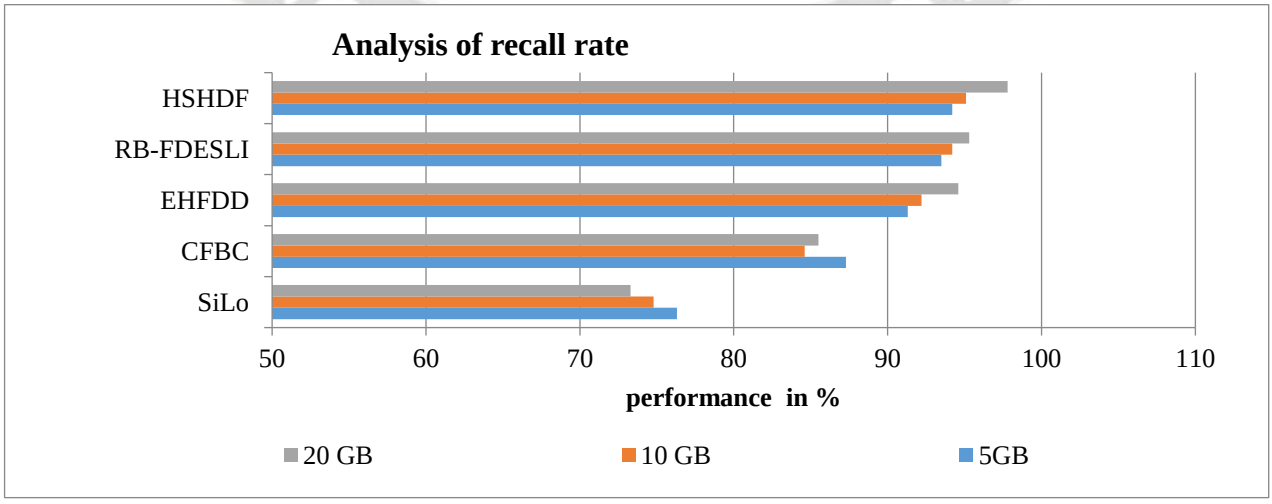

Figure 4: Analysis of recall rate

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

Figure 4 shows that Recall performance has been measured in various ways. The proposed RB-FDESLI produces the best performance up to 93.5% compared to the methods by testing 5 GB of data, the SiLoproduces 75.3 %, CFBC produces 85.3 %, and EHFDD produces 91.3 %. And the the proposed HSHDFalgorithm attain improved Recall performance is higher than other methods.

Table 3: comparison of Recall

Impact of Recall in %					
Storage /methods	**SiLo**	**CFBC**	**EHF**	**RB-FDESLI**	**HDF**
5GB	75.3	85.3	91.3	93.5	94.2
10 GB	74.1	84.6	92.2	94.2	95.1
20 GB	72.2	83.5	94.6	97.3	97.8

Table 3 recounts a variety of state comparison techniques. The proposed HSHDFsystem creates a high return state with a maximum rating of 97.8 % compared to another system.

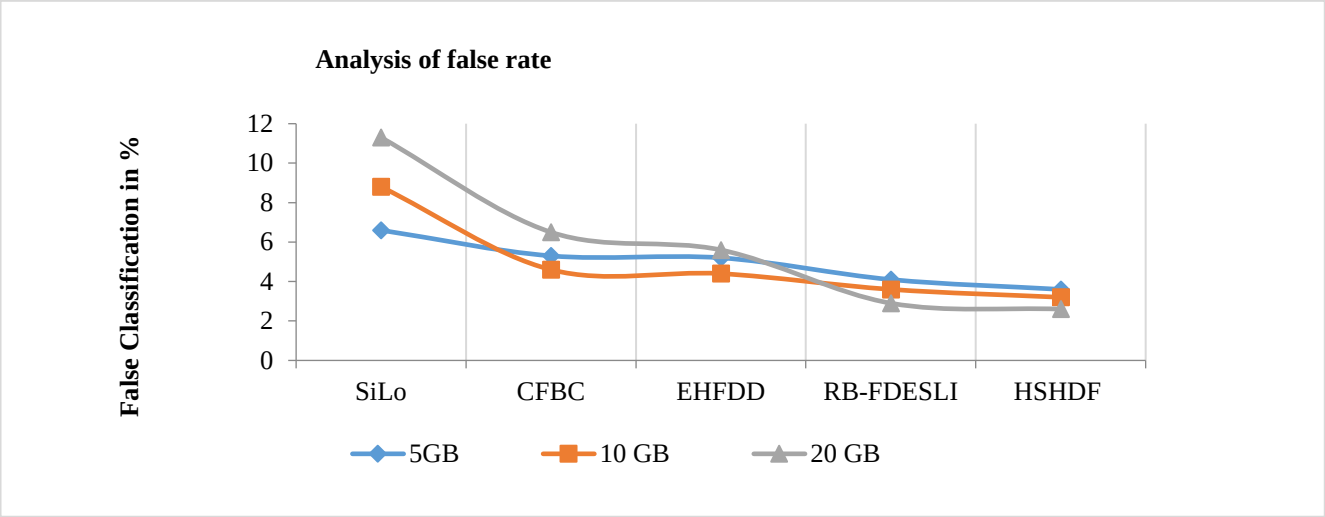

Figure: 5 Analysis of false classification rate

Figure: 5 shows the measure of the rate of incorrect redundancy rate production by various methods, it is presented. The proposedHSHDF and RB-FDESLI produce the best performance up to 4.1 % compared to the methods by testing 5 GB of data, the SiLoproduces 6.6 %, CFBC produces 5.3 %, and EHFDD produces 5.2 %. The results of the proposed HSHDFalgorithm show that it produces a false classification rate less than other methods.

Table 4: Analysis of false rate

Analysis of false rate %					
Storage /methods	SiLo	CFBC	EHF	RB-FDESLI	HDF
5GB	6.6	5.3	5.2	4.1	3.6
10 GB	8.8	4.6	4.4	3.6	3.2
20 GB	11.3	6.5	5.6	2.9	2.6

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

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

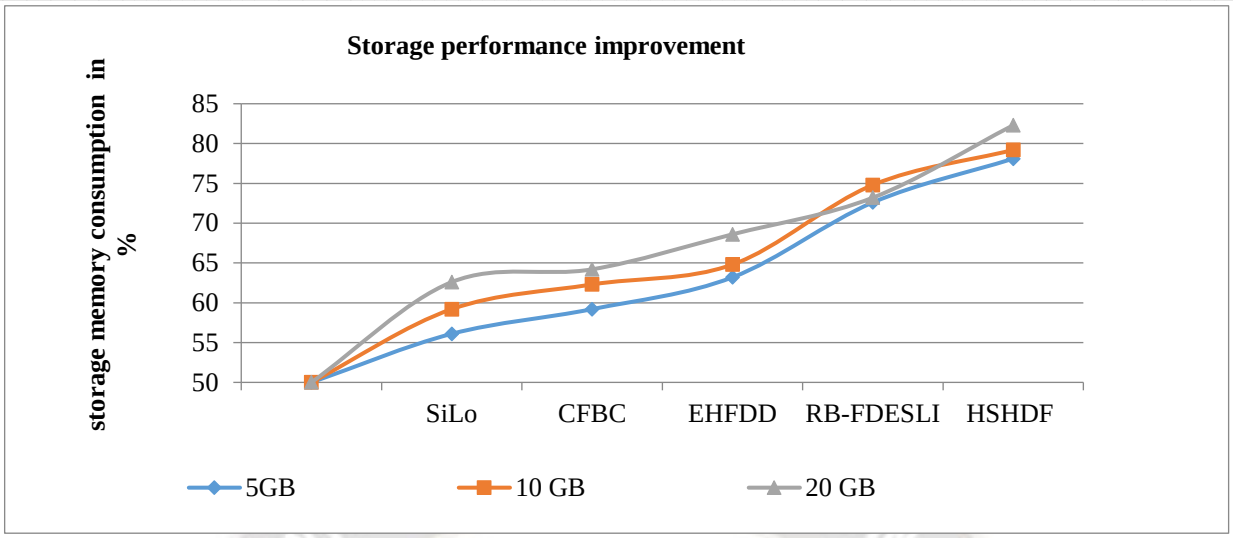

Figure 6: storage performance rate

Figure 6 presents the results of various methods' storage performance rates in memory of performance analysis. The proposed HSHDFproduces the best performance up to 72.6 % compared to the methods by testing 5 GB of data, the SiLoproduces 56.1 %, CFBC produces 59.2 %, and EHFDD produces 63.2 %. The proposed HSHDFmethod minimizes data replication and reduces memory consumption.

Table 5: storage performance rate

storage Redundant performance rate **in %**					
Storage /methods	SiLo	CFBC	EHF	RB-FDESLI	HDF
5GB	56.1	59.2	63.2	72.6	68.1
10 GB	59.2	62.3	64,8	74.8	71.2
20 GB	62.6	64.2	68.6	73.2	72.3

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

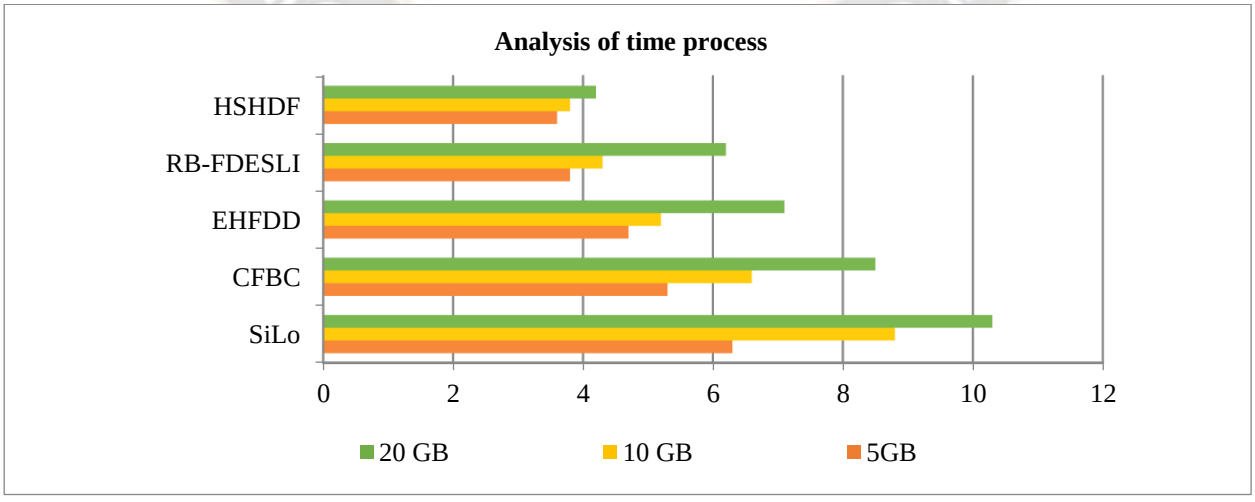

Figure 7: Analysis of the time process

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

The above figure 7 shows the Measuring the time process preparation by different methods. The proposedHSHDF, RB-FDESLI produces the best performance up to 8.3 (s) compared to the methods by testing 5 GB of data, the SiLoproduces 6.3 (s), CFBC produces 5.3 (s), and EHFDD produces 4.7 (s). The proposed HSHDFalgorithm presents less time compared to the other methods reduced.

Table 6: Analysis of the time process

Analysis of time process (s)					
Storage /methods	SiLo	CFBC	EHF	RB-FDESLI	HDF
5GB	6.3	5.3	4.7	3.8	3.6
10 GB	8.8	6.6	5.2	4.3	3.8
20 GB	10.3	8.5	7.1	6.2	4.2

Table 6 shows the time process in different storage spaces, and the variety of proposed method options are compared to prove the performance. Applying the projected process HSHDFalso creates less time to process files and higher performance of up to 5.8 ms.

V. Conclusion

To conclude, the proposed resultant performance and deduplication improvement produce the best performance to optimize the storage space. The new Cross Layer Fragment Indexing based file deduplication using Hyper Spectral Hash Duplicate Filter (HSHDF)attains high performanceto reduce the cloud storage.The clusters are further prepared for Deduplication remain the cosine transformation for redundancy to make singular byte stream compression. This implementation proves the data redundancy to reduce the storage space and improved results to improve the quality of memory management and service cost reduction in distributed cloud storage.This potential deduplication system makes efficient data reduction to find similar data analysis using the parsing methods.This resultant of proposed HSHDF proves the best evaluation in recall rate up to 97.8%, precision rate up to 96.9 %, and storage optimization 82.3 % improved as well as other methods.

Reference

[1] Y. Tan et al., "Improving the Performance of Deduplication-Based Storage Cache via Content-Driven Cache Management Methods," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 214-228, 1 Jan. 2021, DOI: 10.1109/TPDS.2020.3012704.

[2] X. Yang, R. Lu, J. Shao, X. Tang, and A. A. Ghorbani, "Achieving Efficient Secure Deduplication With User-Defined Access Control in Cloud," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 1, pp. 591-606, 1 Jan.-Feb. 2022, DOI: 10.1109/TDSC.2020.2987793.

[3] B. Wang et al., "A Data Structure for Efficient File Deduplication in Cloud Storage," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0071-0077, DOI: 10.1109/UEMCON51285.2020.9298159.

[4] B. Wang et al., "A Data Structure for Efficient File Deduplication in Cloud Storage," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0071-0077, DOI: 10.1109/UEMCON51285.2020.9298159.

[5] F. Rashid, A. Miri, and I. Wolfgang, "Proof of Storage for Video Deduplication in the Cloud," 2015 IEEE International Congress on Big Data, 2015, pp. 499-505, DOI: 10.1109/BigDataCongress.2015.79.

[6] J. Ren, Z. Yao, J. Xiong, Y. Zhang, and A. Ye, "A Secure Data Deduplication Scheme Based on Differential Privacy," 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), 2016, pp. 1241-1246, DOI: 10.1109/ICPADS.2016.0169.

[7] M. Aman, P. Verma, and D. Rajeswari, "Secure Cloud Data Deduplication with Efficient Re-Encryption," 2021 International Conference on Intelligent Technologies (CONIT), 2021, pp. 1-4, DOI: 10.1109/CONIT51480.2021.9498487.

[8] K. Vijayalakshmi and V. Jayalakshmi, "Analysis on data deduplication techniques of storage of big data in cloud," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 976-983, DOI: 10.1109/ICCMC51019.2021.9418445.

[9] N. Chhabra and M. Bala, "A Comparative Study of Data Deduplication Strategies," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 68-72, DOI: 10.1109/ICSCCC.2018.8703363.

[10] Yong-Ting Wu, Min-Chieh Yu, Jenq-ShiouLeu, Eau-Chung Lee and Tian Song, "Design and implementation of various file deduplication schemes on storage devices," 2015 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015, pp. 80-84.

[11] Kulkarni, L. . (2022). High Resolution Palmprint Recognition System Using Multiple Features. Research Journal of Computer Systems and Engineering, 3(1), 07–13. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/35

[12] P. Bartus and E. Arzuaga, "Using file-aware deduplication to improve capacity in storage systems," 2017 IEEE Colombian

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7239
***Article Received**: 25 April 2023 **Revised:** 18 June 2023 **Accepted:** 01 July 2023*

Conference on Communications and Computing (VOLCOM), 2017, pp. 1-6, DOI: 10.1109/ColComCon.2017.8088193.

[13] L. Conde-Canencia and B. Hamoum, "Deduplication algorithms and models for efficient data storage," 2020 24th International Conference on Circuits, Systems, Communications and Computers (CSCC), 2020, pp. 23-28, DOI: 10.1109/CSCC49995.2020.00013.

[14] Y. Tan et al., "Improving the Performance of Deduplication-Based Storage Cache via Content-Driven Cache Management Methods," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 214-228, 1 Jan. 2021, DOI: 10.1109/TPDS.2020.3012704.

[15] W. Xia et al., "The Design of Fast Content-Defined Chunking for Data Deduplication Based Storage Systems," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 9, pp. 2017-2031, 1 Sept. 2020, DOI: 10.1109/TPDS.2020.2984632.

[16] Y. Won, K. Lim and J. Min, "MUCH: Multithreaded Content-Based File Chunking," in IEEE Transactions on Computers, vol. 64, no. 5, pp. 1375-1388, 1 May 2015, DOI: 10.1109/TC.2014.2322600.

[17] B. Wang et al., "A Data Structure for Efficient File Deduplication in Cloud Storage," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0071-0077, DOI: 10.1109/UEMCON51285.2020.9298159.

[18] N. Chhabra and M. Bala, "A Comparative Study of Data Deduplication Strategies," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), 2018, pp. 68-72, DOI: 10.1109/ICSCCC.2018.8703363.

[19] K. Vijayalakshmi and V. Jayalakshmi, "Analysis on data deduplication techniques of storage of big data in cloud," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 976-983, DOI: 10.1109/ICCMC51019.2021.9418445.

[20] Y. Tan et al., "Improving the Performance of Deduplication-Based Storage Cache via Content-Driven Cache Management Methods," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 214-228, 1 Jan. 2021, DOI: 10.1109/TPDS.2020.3012704.

[21] N. Sharma, A. V. Krishna Prasad and V. Kakulapati, "File-level Deduplication by using text files – Hive integration," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-6, DOI: 10.1109/ICCCI50826.2021.9402465.