International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 11 • Issue 1 pp. 839-852 Jan 2019-June 2019 www.csjournals.com

Performance Analysis of Deduplication Techniques for Data Storage Systems using Destor Tool

Naresh Kumar
Department of Computer Sc. and Engg, UIET, Kurukshetra University, Kurukshetra, INDIA

Abstract: The growing quantity of data requires some techniques that manage huge amount of data in a suitable way. Deduplication is one ofthe techniques that maximize the storage space to store plenty amount of data. In data deduplication numerous algorithms are feasible that basically detect and eliminate the redundant data and store unique copy of data. Chunking is a technique that divide the data stream in fix size or variable size chunks and pull out the redundancy. In this paper, results have been compared as deduplication ratio, total time, deduplication throughput of various deduplication techniques. The experimental results on DESTOR tool provides some guidelines to adopt the best chunking algorithm to clear away from clone data.

Keywords: Data Deduplication; Content Defined Chunking; Fixed Size Chunking; Basic Sliding Window; Two Divisors Two Thresholds.

I. INTRODUCTION

Big data is a term that describes all about large amount of data that may be structured, semi-structure and unstructured [1]. Structured data are relational data that stored in table with rows and columns, semi-structured data lies between structured and unstructured data mainly includes the XML and JSON.Unstructured data includes multimedia data like audio, video, photos, presentation and many more types of document. In order to define large amount of data, big data introduced to computing world by Roger Magoulas from O'Reilly media in 2005.It was introduced because traditional data management techniques can not manage and process due to the complexity and size of this big data. Big data requires different tools and architecture to solve problem in a better way but key challenge of big data storage is that how to handle large amount of data and provide the minimum input/output operation per seconds (IOPS) necessary to deliver the data.

In today's world, Big Data has emerged as a key buzzword in IT over past few years. Day by day the data is increasing, so some techniques are required that optimize and manage the storage space. According to the report of IDC (International Data Company), about 75% of data is duplicated in digital world and especially the duplicity in backup repository and archival system is more than 90% [2-3]. As seen the situation, deduplication is one of techniques that solve our purpose. It efficiently handles various types of data and also prevents redundant data from being stored in storage device and transmits over the network.

The rest of the paper organized as: section II explains the deduplication process and its various types, sections III describe chunking methods of deduplication. After that section IV contains experimental result. Paper is concluded with discussion given in section V.

II. DEDUPLICATION

In this section, we provide detail about the deduplication system and the various chunking algorithms. Deduplication [4] is the process of eliminating duplicate copies of data through a deduplication scanning process so that unique single copy is stored and will then serve all of the authorized users. This process is applied on single user where redundancy within data is identified and removed, but this is not practically possible. Another way is cross client de-duplication, where the client and server match the duplicity and then removes the duplicate data if it repeated; otherwise saves a single copy of unique data in storage device. These techniques divide a file into chunks and compute a unique hash value for each chunk by using MD5 or SHA1.

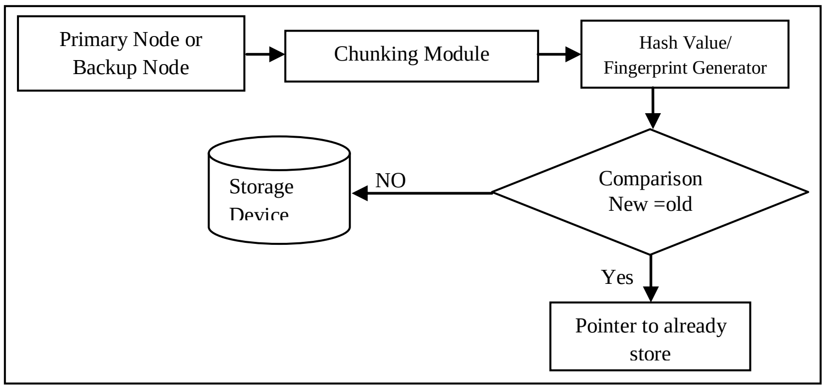
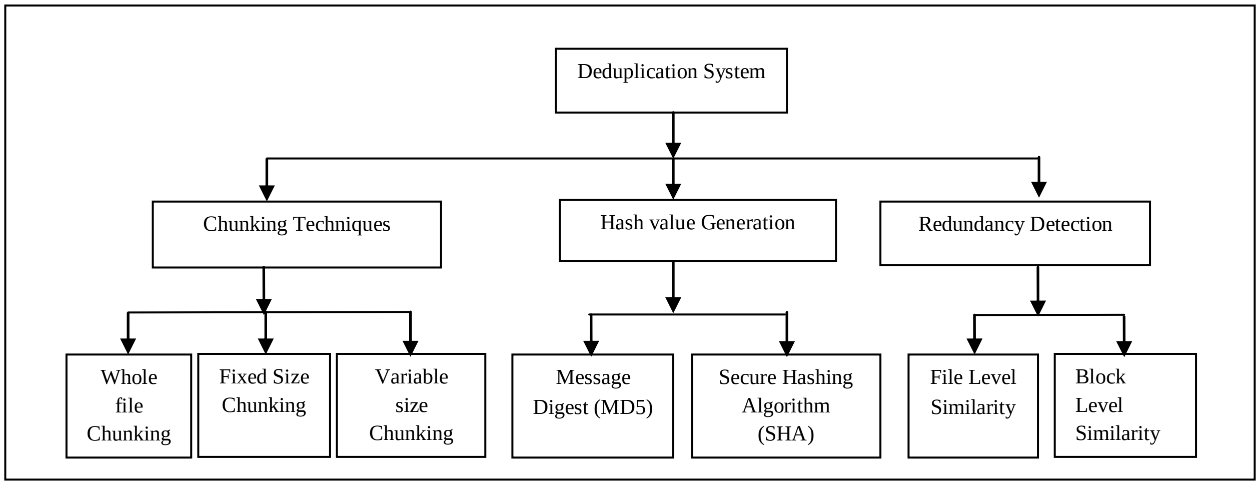


Fig. 1 Traditional Deduplication System



Hash function is a part of cryptography but is does not require any special key for coding.It is a function that maps any value into a fixed size hash value called a hash code. A single bit change in the value generates the different hash code [5]. Traditional architecture of deduplication system is shown in Fig. 1 and different deduplication categories are as shown in Fig. 2.

A. Single Instance Storage (SIS) or File level Deduplication

It is applied on entire file that does not depend on the file size.It stores only one copy of similar file and creates reference for duplicated files. It utilizes small storage space and there is low CPU usage.

B. Block Level Deduplication

The Source file is divided in blocks that may be of fixed or variable size. After creation of block it checks if the new arrived block is matched with the previously stored block. If it ispresents then succeeding copy is not stored on the disk but a pointer is created to point the original block. It saves more space on disk as compare to SIS.

C. Byte Level Deduplication

It does not require the additional processing because data is compared byte by byte. This checks each byte more accurately for redundant data in post processing stage, but also requires lot of time in processing.

III. CHUNKING METHODS

The first component of any deduplication process is to divide the file into smaller units. The process of dividing the file into these units is called chunking and the resulting units are called chunks. There are several chunking algorithms:

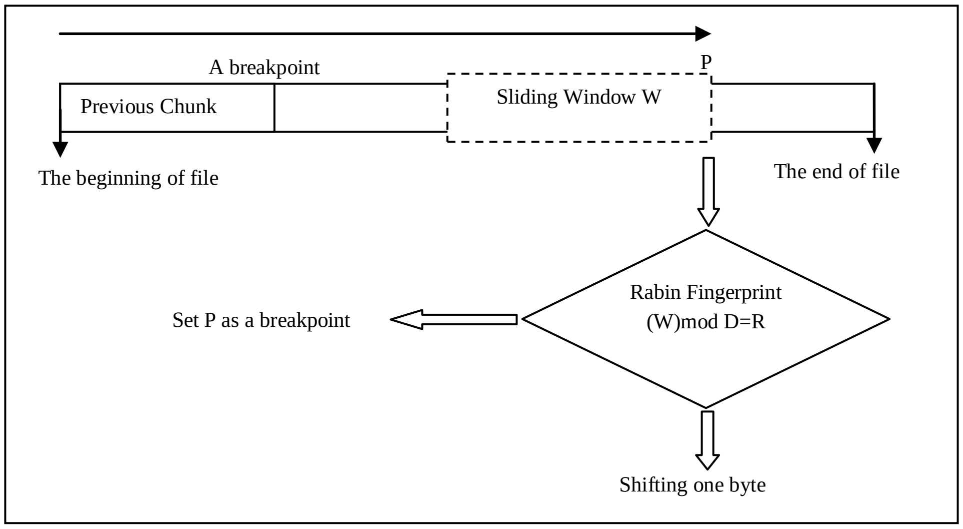
A. Fixed Size Chunking

A file is portioned into fixed size segment, e.g. 8KB segments. For backup application and large scale file systems, many of organizations use fixed size blocks. But there is a limitation in this method,for every insertion or deletion in the original file may generate a set of chunks that are entirely different from the original ones. The boundary of newly formed chunk is totally different from the previous chunks. It creates the lots of metadata with some minor changes that increase the storage data and also increase the CPU overhead.Frequency based chunking (FBC), byte index and multi-byte index are some of example of fixed size chunking.

Frequency based chunking (FBC) is a two stage algorithm [6] that identifies the high frequent chunk. At first stage, it identifies the fixed size high frequency chunk. Then at second stage, it is consist of coarse grained and fine grained based chunking. In coarse grained chunking, content defined chunking algorithm is used to partition the data stream into large size chunks and then fine grained chunking FBC scan each coarse grained chunk to find the frequent fix size chunk.

Byte index chunking [7] finds the duplicity at byte level by searching the high probability duplicate chunk byte by byte in the file. Index matrix of size 256x256 is used to find the high probable duplicate chunk in less time. After that hash function is applied to confirm that the chunk is definitely duplicate or not.After that multi-byte index chunking [8] is proposed that used two index matrix. First 32KB index matrix that used for files whose size is less than 5GB. If the file size is more than 5BG than second 4MB index matrix is to be used.

B. Variable Size Chunking
Variable size chunking solves the problem of "boundary shift problem" that come in fixed size chunking. It partitions the file according to the content. Some of variable size chunking algorithm are leap based chunking, bimodal chunking, multi-modalchunking and basic sliding window.
Leap-based CDC algorithm [9] improves the deduplication performance by adding another judgment function. The pseudo-random transformation is used to define whether a window is qualified or not. This is the replacement of rolling hash function that is used in the sliding window CDC.Bimodal chunking [10] introduced as opposed to the (uni-modal) baseline CDC approach. This is the improved version of CDC that mixes chunks of different average size together. The algorithm first chunks the data stream into large chunks and then splits parts of them into small chunksA Multimodal Content Defined Chunking (MCDC) was proposed as a new enhancement in Bimodal Content Defined Chunking. MCDC [11] determines the optimal chunk size according to data size and compressibility.Basic Sliding Window (BSW) algorithm isused in variable size chunking. A signature is created for each chunk, if signature matches the predefined bit pattern;the algorithm set the chunk boundary atthe end of window. After each comparison, the window slides one byte position and compute hash function.



To find the duplicated data, the content defined chunking proposed a low bandwidth network file system (LBFS) that [12] determines the file similarity and saves bandwidth. Content defined chunking break the file into variable sized chunks according to content. Various algorithms that use CDC[13] are :

1) Basic Sliding Window(BSW) Algorithm
2) Two Thresholds (TD) Algorithm
3)Two Thresholds Two Divisors (TTTD) Algorithm
4)Asymmetric Extremum (AE) Algorithm

1) Basic Sliding Window (BSW) Algorithm
The BSW algorithm [14] establishes a window of byte stream starting from the first content to last content of a file. It performs file chunking, fingerprint generation and redundancy detection. It avoids the boundary shift problem by making chunk boundaries depend on the local content of the file.
There are three parameter need to be desired: a fixed size sliding window W, an integer divisor D and an integer remainder R, where R<D.The parameter R must lie between 0 and D-1, and usually taken as D-1. Algorithm for BSW is given below:

```
Fixed size sliding window W is moved across the file and fingerprint is generated
{
    If the (Fingerprint mod D = R), Then this point is breakpoint and compute the hash
     value. Check duplication;
    {
       If occur create a reference pointer for that chunk.
       Else, Store the chunk in storage device.
    }
     Else, Shift the window by one byte and same procedure are followed for rest of data.
}
```

2) Two Divisors (TD) Algorithm

There are two problems in BSW algorithm. Firstly, it may determine the breakpoint in each shift if the file contains lot of continuous repeating string like 'aaaaaaaa'. Due to this, the metadata size is equal to or larger than original file. Thesecond problem is that after scanning complete file there is no breakpoint detected, so whole file is taken as chunk. To resolve these problems Two Threshold algorithm [14] was introduced that takes second divisor D', where D'<D. It has greater chance of searching the chunk boundary. In this the entire stream is to be scanned and at every position the primary and secondary divisor both compute the chunk boundary. If chunk boundary found by primary divisor D before going up to the T_{max}(Maximum Threshold) then it declare a breakpoint; otherwise the breakpoint is determined by secondary divisor D'.

3) Two Thresholds, Two Divisors (TTTD) Algorithm

In BSW and TD algorithms, the maximum threshold value cause chunk to vary greatly in size. The small sized chunks increase the quantity of chunks that results to be memory overhead. Two divisor two threshold is combination of SCM (small chunkmerge) algorithm and TD algorithm.

The TTTD algorithm [15] uses four parameters D (Primary Divisor), D' (Second Divisor), T_{max}(Maximum Threshold), T_{min}(Minimum Threshold).To control the variance in chunk size, minimum and maximum threshold is to be set. The second divisor is half of the primary divisor.

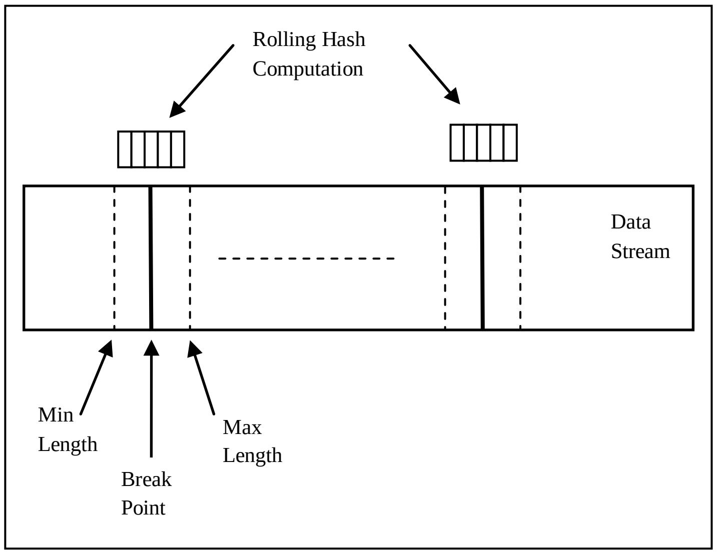


Fig 4. Rolling Hash Computation in TTTD

4) Asymmetric Extremum (AE) algorithm

A new content defined chunking algorithm was presented that mainly focus to improve the chunking throughput and the chunk size variance. The Rabin based and MAXP based CDC algorithm limitations are removed by AE algorithm [16]. It only requires maximum value of window size is searched by content of data stream. A variable size window is used that find maximum value without going in reverse direction as opposed to fixed size window that is used in Rabin based CDC. The AE algorithm requires one comparison and two conditional branch operations per byte.

IV. EXPERIMENTAL RESULT

In this section, experimental results have been compared as a performance result analysis on various datasets. The analysis is performed at file based, fixed size chunking and variable sized chunking. When the duplicity is checked over a file, the entire file is treated as single one and then duplicate data is to be removed. In case of fixed size chunking, the whole file is partitioned into fixed size blocks and redundancy is removed. Finally, the variable size chunking is analyzed that partitions the file according to the contents. The results have been performed as analysis on a machine with configuration Intel i5 CPU with Install memory 4.00GB on 64bit OS in Ubuntu version 14.04. The analysis performed on different datasets [17] with the help DESTOR tool [18]. Destor is a platform for data deduplication that includes file level deduplication, fixed sized block level chunking and variable size content defined chunking. The testing results are shown from Fig. 5 to Fig 8 including file based, fixed size, Rabin based CDC and TTTD techniques respectively. Table 1 show results on various datasets with different deduplication algorithms.

A. Deduplication Elimination Ratio (DER) or Deduplication Ratio (DR)

The overall deduplication ratio is defined as non-redundant Stored Capacity divided by Total Capacity. It may be expressed as:

$$\mathrm{DER} = \frac{\mathrm{StoredCapacity}}{\mathrm{TotalCapacity}}$$

B. Throughput

Throughput is a measure of how many units of information a system can process in a given amount of time.

C. Hash Time

The amount of time required to complete the process of hashing during deduplication is the hash time.

Table 1. Experimental Result on Various Datasets with Different Deduplication Techniques

Datasets	Data size before Deduplication(GB)	Techniques	Data size after Deduplication(GB)	Deduplication Ratio	Throughput (MB/s)	Hash time (MB/s)
Dataset 1	2.51	File	0.039	0.9844	121.24	196.72
		Fixed	0.037	0.9871	102.22	172.72
		CDC	0.037	0.9651	95.65	180.31
		TTTD	0.035	0.9858	102.28	176.26
Dataset 2	2.89	File	0.83	0.9713	91.40	167.37
		Fixed	0.079	0.9725	92.13	173.73
		CDC	0.080	0.9722	136.19	169.22
		TTTD	0.078	0.9727	106.08	139.71
Dataset 3	2.67	File	0.10	0.9595	100.14	197.09
		Fixed	0.091	0.9657	105.71	180.64
		CDC	0.099	0.9629	158.53	161.72
		TTTD	0.091	0.9658	116.91	142.71

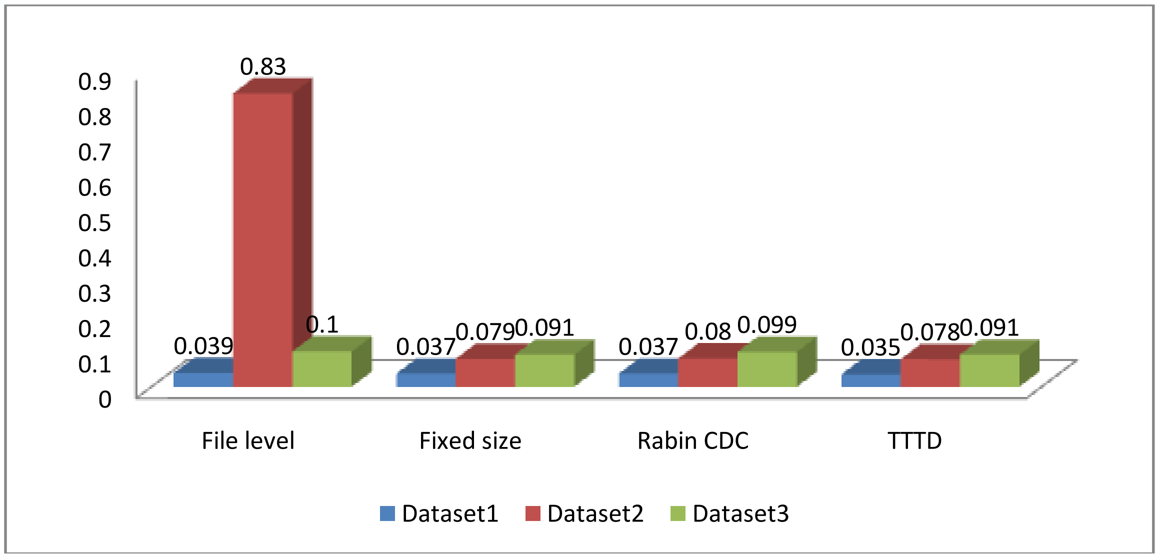


Fig 5. Datasize (GB) after Deduplication

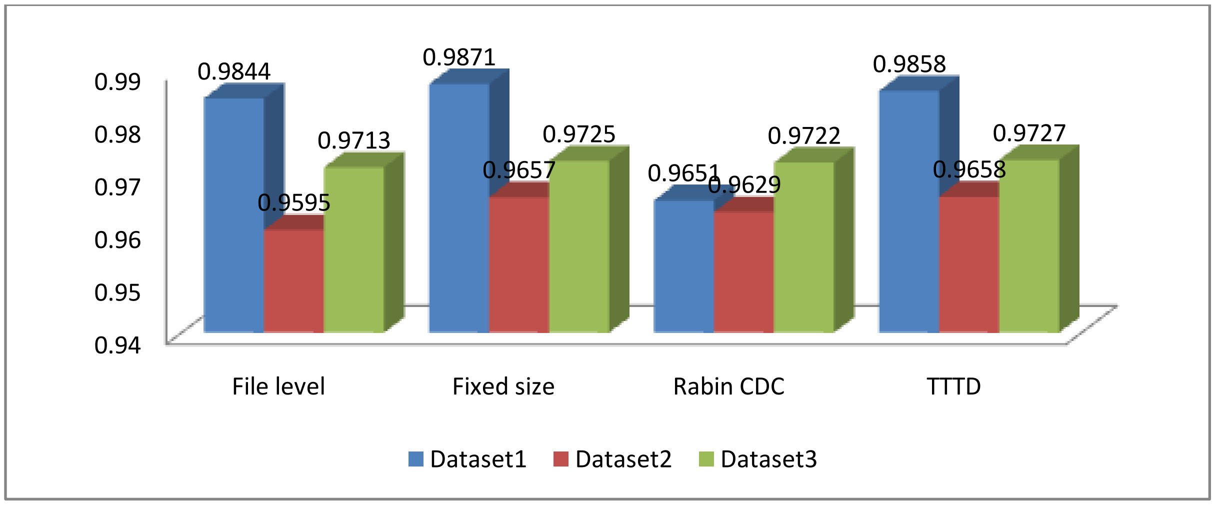


Fig 6. Deduplication Ratio for Different Deduplication Techniques

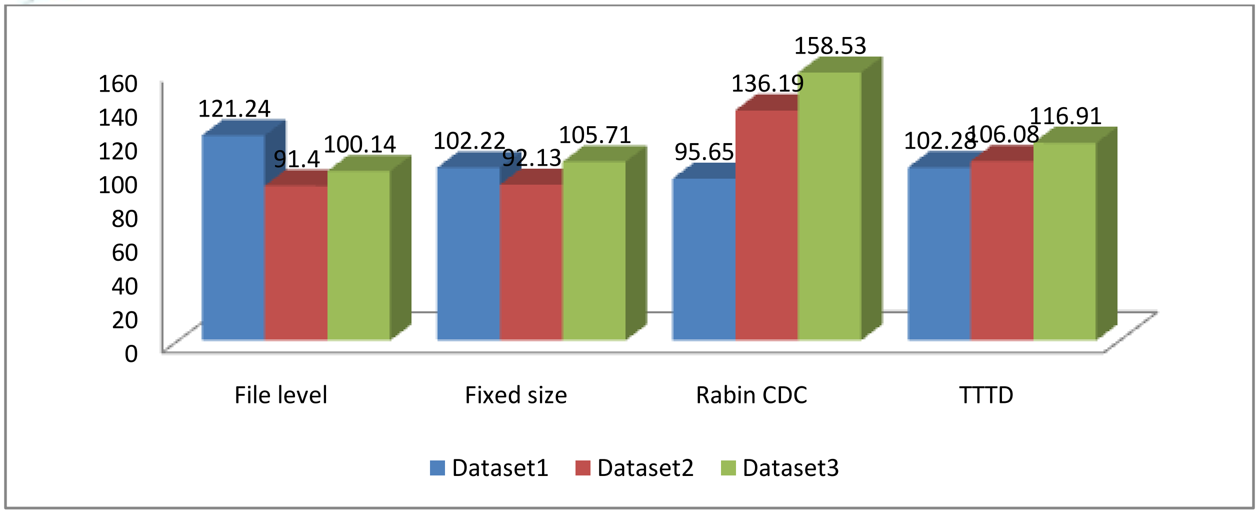


Fig 7. Throuhput (MB/s) for Different Deduplication Techniques

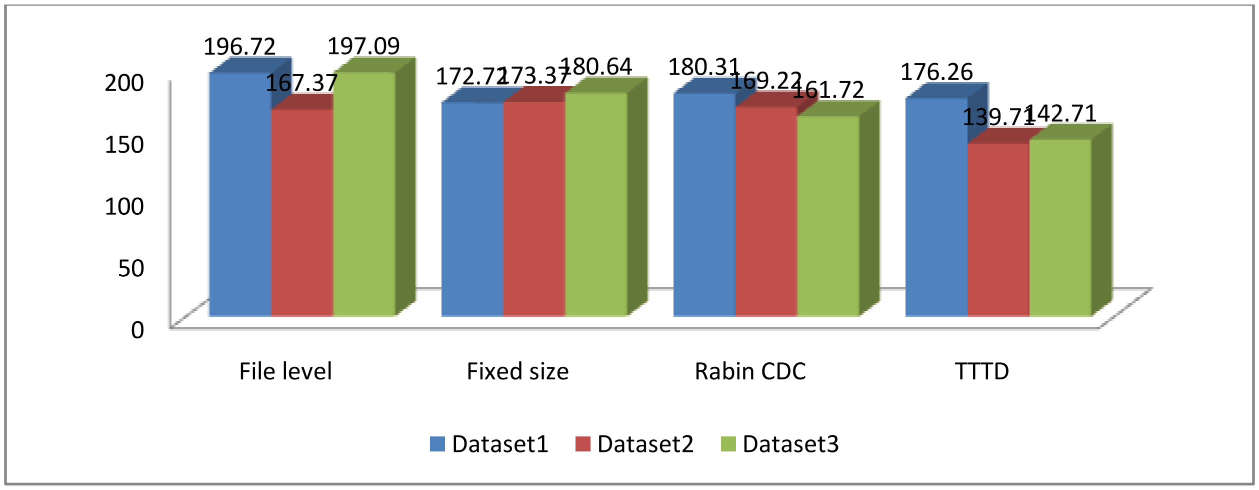


Fig 8. Hash Time (MB/s) for Different Deduplication Techniques

V. DISCUSSION

In this paper, experimental results have been evaluated for different chunking algorithms of data deduplication. Mainly it has been focused for file level deduplication, fixed size deduplication, Rabin based content defined chunking and Two thresholds and Two Divisor (TTTD). For performance analysis the structured, semi-structured and unstructured data is taken. The experimental results on different datasets show that the duplicated data is detected with less computation and the outcomes are only the unique data that saved in the storage device. The Rabin based CDC and AE algorithms computes the duplicacy more efficiently by maximizing storage space in less time.

VI. REFERENCES

[1] K.V.Kanimozhi , Dr.M.Venkatesan,” Unstructured Data Analysis-A Survey”, International Journal of Advanced Research in Computer and Communication Engineering, IJARCCE, Vol. 4, Issue 3, pp. 223-225, 2015.

[2] Yu-xuan Xing, Nong Xiao, and Fang Liu, Zhen Sun, Wan-hue He ”AR-Dedupe: An Efficient De-duplication Approach for Cluster De-duplication System” , Journal of ShandhaiJiaotong University, vol. 15, no. 1, pp. 76-81, 2015.

[3] ShengmeiLuo, Guangyan Zhang, Chengwen Wu, “Boafft: Distributed De-duplication for Big Data Storage in the Cloud”, Transactions on Cloud Computing, IEEE, vol. 4, no. X, pp. 1-13, 2016.

[4] Qinlu He, Zhanhuai Li, Xiao Zhang,”Data De-duplication Techniques”, International Conference on Future Information Technology and Management Engineering, IEEE, pp. 430-433, 2010.

[5] AnakAgungPutriRatna, Ahmad Shaugi, Prima DewiPurnamasari, Muhammad Salman,” Analysis and Comparison of MD5 and SHA-1 Algorithm Implementation in Simple-O Authentication based Security System”, International Conference on QiR (Quality in Research), IEEE , page. 99 – 104, 2013.

[6] Guanlin Lu, Yu Jin, and David H.C. Du,”Frequency Based Chunking for Data De-Duplication” 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, pp. 287-296, 2010.

[7] IderLkhagvasuren, Jung Min So, Jeong Gun Lee, Jim Kim and Young WoongKo, ”Design and Implementation of Storage System using Byte-index Chunking Schemes”, International Journal of Security and Its Applications, vol. 8, no. 1, pp. 33-42, 2014.

[8] IderLkhagvasuren, Jung Min So, Jeong Gun Lee, Jin Kim and Young WoongKo,”Multi-level Byte Index Chunking Mechanism for File Synchronization”, International Journal of Software Engineering and Its Applications, SERSC, vol.8, no.3 (2014), pp.339-350, 2014.

[9] Chuanshuai Yu, Chengwei Zhang, Yiping Mao, Fulu Li, “Leap Based Content Defined Chunking- Theory and Implementation”, 31st Symposium on Mass Storage Systems and Technologies (MSST), IEEE, pp. 1-12, 2015.

[10] Erik Kruus, Christian Ungureanu, CezaryDubnicki,”Bimodal Content Defined Chunking for Backup Streams”,Fast 10 Preceding of the 8th USENIX Conference on file and Storage Technologies ,USENIX ,pp. 239-252, 2010.

[11] Jiansheng Wei, Junhua Zhu, Yong Li,” Multimodal Content Defined Chunking for Data Deduplication”, https://www.researchgate.net/publication/261286019,Research gate,2014.

[12] AthichaMuthitacharoen, Benjie Chen, And David Mazieres,” A Low-bandwidth Network File System”, proceeding of the 18th ACM Symposium on operating System principle (Sosp’01), Chateau lake louise,Banff, cnada, pp 174-187, 2001.

[13] Cai Bo, Zhang Feng Li, and Wang can,” Research on Chunking Algorithms of Data De-duplication,proceeding of the ICCEAE, Springer, AISC 181, pp. 1019-1025, 2013.

[14] KaveEshghi, HsiuKhuern Tang, ,”A Framework for Analyzing and Improving Content Based Chunking Algorithms” Technical Report TR 2005-30, Hewlett-Packard Development Company, http://www.hpl.hp.com/techreports/2005/HPL-2005-30 R1.html, 2005.

[15] A. Venish and K. Siva Sankar,” Study of Chunking Algorithm in Data Deduplication” ,Proceedings of the International Conference on Soft Computing Systems, Springer,pp. 13-20, 2016.

[16] Yucheng Zhang, Hong Jiang, Dan Feng, Wen Xia, Min Fu, Fangting Huang, YukunZhou,”AE: An Asymmetric Extremum Content Defined Chunking Algorithm for Fast and Bandwidth-Efficient Data De-duplication”, 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, pp. 1337-1345, 2015.

[17] www.freedb.org

[18] https://github.com/fomy/destor.

Experimental Results

Datasets	Data size before Deduplication (GB)	Techniques	Data size after Deduplication(GB)	Deduplication Ratio	Throughput (MB/s)	Hash time (MB/s)
Dataset 1	2.51	File	0.039	0.9844	121.24	196.72
		Fixed	0.037	0.9871	102.22	172.72
		CDC	0.037	0.9651	95.65	180.31
		TTTD	0.035	0.9858	102.28	176.26
Dataset 2	2.89	File	0.83	0.9713	91.40	167.37
		Fixed	0.079	0.9725	92.13	173.73
		CDC	0.080	0.9722	136.19	169.22
		TTTD	0.078	0.9727	106.08	139.71
Dataset 3	2.67	File	0.10	0.9595	100.14	197.09
		Fixed	0.091	0.9657	105.71	180.64
		CDC	0.099	0.9629	158.53	161.72
		TTTD	0.091	0.9658	116.91	142.71

1. **Deduplication on Dataset1 using different deduplication techniques**

1.1 File level deduplication:

```
number of chunks: 208963 (9132 bytes on average)
number of unique chunks: 4837
total size(B): 2705124862
stored data size(B): 42168114
deduplication ratio: 0.9844, 330.2293
total time(s): 15.011
throughput(MB/s): 121.24
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 2.327s, 782.11MB/s
chunk_time : 11.933s, 172.52MB/s
hash_time : 11.539s, 196.72MB/s
dedup_time : 0.324s, 5621.54MB/s
rewrite_time : 0.017s, 109271.19MB/s
filter_time : 0.377s, 4824.41MB/s
write_time : 0.015s, 121741.37MB/s
```

1.2 Fixed Size based deduplication:

```
number of chunks: 234702 (9146 bytes on average)
number of unique chunks: 4062
total size(B):  2705124862
stored data size(B): 40189046
deduplication ratio: 0.9871, 180643.0786
total time(s): 20.028
throughput(MB/s): 102.22
number of zero chunks: 0
```

```
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 8.436s, 242.67MB/s
chunk_time : 12.745s, 160.63MB/s
hash_time : 11.853s, 172.72MB/s
dedup_time : 0.423s, 4844.47MB/s
rewrite_time : 0.017s, 120813.89MB/s
filter_time : 0.374s, 5481.41MB/s
write_time : 0.013s, 157412.90MB/s
```

1.3 Rabin CDC based Deduplication:

```
number of chunks: 219473 (9145 bytes on average)
number of unique chunks: 4080
total size(B):  2705124862
stored data size(B): 40265426
deduplication ratio: 0.9651, 101853.0263
total time(s): 20.013
throughput(MB/s): 95.65
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 7.754s, 246.86MB/s
chunk_time : 11.545s, 165.81MB/s
hash_time : 10.616s, 180.31MB/s
dedup_time : 0.189s, 10123.77MB/s
rewrite_time : 0.016s, 117459.16MB/s
filter_time : 0.349s, 5491.61MB/s
write_time : 0.015s, 126235.29MB/s
```

1.4 TTTD based deduplication:

```
number of chunks: 234702 (9146 bytes on average)
number of unique chunks: 3523
total size(B):  2705124862
stored data size(B): 38284168
deduplication ratio: 0.9858, 206456.6060
total time(s): 20.016
throughput(MB/s): 102.28
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 10.787s, 189.79MB/s
chunk_time : 12.408s, 165.00MB/s
hash_time : 11.615s, 176.26MB/s
dedup_time : 0.425s, 4821.37MB/s
rewrite_time : 0.018s, 115177.88MB/s
filter_time : 0.426s, 4801.53MB/s
write_time : 0.007s, 274067.84MB/s
```

2. Deduplication on Dataset2 using different deduplication techniques

2.1 File level deduplication:

```
number of chunks: 157656 (9142 bytes on average)
number of unique chunks: 7906
total size(B): 3109540864
stored data size(B): 89272961
deduplication ratio: 0.9713, 24.2513
total time(s): 15.040
throughput(MB/s): 91.40
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 8.394s, 163.77MB/s
chunk_time : 8.943s, 153.71MB/s
hash_time : 8.213s, 167.37MB/s
dedup_time : 0.151s, 9088.17MB/s
rewrite_time : 0.012s, 117379.46MB/s
filter_time : 0.378s, 3637.08MB/s
write_time : 0.101s, 13605.90MB/s
```

2.2 Fixed size based deduplication:

```
number of chunks: 157372 (9270 bytes on average)
number of unique chunks: 7924
total size(B): 3109540864
stored data size(B): 85358412
deduplication ratio: 0.9725, 320.6170
total time(s): 15.103
throughput(MB/s): 92.13
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 3.917s, 355.24MB/s
chunk_time : 8.728s, 159.42MB/s
hash_time : 8.009s, 173.73MB/s
dedup_time : 0.224s, 6214.23MB/s
rewrite_time : 0.012s, 115872.56MB/s
filter_time : 0.297s, 4687.22MB/s
write_time : 0.077s, 18062.83MB/s
```

2.3 Rabin CDC based deduplication:

```
number of chunks: 156367 (9134 bytes on average)
number of unique chunks: 7426
total size(B):  3109540864
stored data size(B): 86163214
deduplication ratio: 0.9722, 48134.8397
total time(s): 10.002
```

```
throughput(MB/s): 136.19
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 0.540s, 2521.34MB/s
chunk_time : 8.645s, 157.56MB/s
hash_time : 8.049s, 169.22MB/s
dedup_time : 0.157s, 8682.66MB/s
rewrite_time : 0.011s, 125584.73MB/s
filter_time : 0.272s, 5004.31MB/s
write_time : 0.012s, 113290.53MB/s
```

2.4 TTTD based deduplication:

```
number of chunks: 309456 (9275 bytes on average)
number of unique chunks: 7654
total size(B):  3109540864
stored data size(B): 84682546
deduplication ratio: 0.9727, 1120.6486
total time(s): 450.158
throughput(MB/s): 106.08
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 442.045s, 6.19MB/s
chunk_time : 21.535s, 127.12MB/s
hash_time : 19.594s, 139.71MB/s
dedup_time : 1.397s, 1959.87MB/s
rewrite_time : 0.029s, 95989.27MB/s
filter_time : 0.580s, 4720.72MB/s
write_time : 0.143s, 19096.21MB/s
```

3. <u>Deduplication on Dataset3 using different deduplication techniques</u>

3.1 File level deduplication:

```
number of chunks: 114730 (9187 bytes on average)
number of unique chunks: 9861
total size(B): 2876482560
stored data size(B): 116405672
deduplication ratio: 0.9595, 143075.4179
total time(s): 10.038
throughput(MB/s): 100.14
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 0.521s, 1928.83MB/s
```

```
chunk_time : 5.633s, 178.44MB/s
hash_time : 5.100s, 197.09MB/s
dedup_time : 0.296s, 3395.74MB/s
rewrite_time : 0.008s, 118875.08MB/s
filter_time : 0.200s, 5030.14MB/s
write_time : 0.039s, 25697.46MB/s
```

3.2 Fixed size based deduplication:

```
number of chunks: 539491 (7718 bytes on average)
number of unique chunks: 9334
total size(B): 2876482560
stored data size(B): 98626547
deduplication ratio: 0.9657, 19.0088
total time(s): 695.095
throughput(MB/s): 105.71
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 684.015s, 5.81MB/s
chunk_time : 23.668s, 167.79MB/s
hash_time : 21.984s, 180.64MB/s
dedup_time : 1.321s, 3005.72MB/s
rewrite_time : 0.039s, 102018.08MB/s
filter_time : 3.196s, 1242.57MB/s
write_time : 0.362s, 10966.65MB/s
```

3.3 Rabin CDC based deduplication:

```
number of chunks: 311762 (8886 bytes on average)
number of unique chunks: 9621
total size(B): 2876482560
stored data size(B): 106582642
deduplication ratio: 0.9629, 754.4806
total time(s): 45.146
throughput(MB/s): 158.53
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 24.657s, 107.16MB/s
chunk_time : 17.593s, 150.19MB/s
hash_time : 16.338s, 161.72MB/s
dedup_time : 0.474s, 5575.57MB/s
rewrite_time : 0.024s, 109394.37MB/s
filter_time : 1.670s, 1582.28MB/s
write_time : 0.051s, 51828.21MB/s
```

3.4 TTTD based deduplication:

```
number of chunks: 343152 (9051 bytes on average)
number of unique chunks: 9154
total size(B): 2876482560
stored data size(B): 98266484
deduplication ratio: 0.9658, 443.2899
total time(s): 175.182
throughput(MB/s): 116.91
number of zero chunks: 0
size of zero chunks: 0
number of rewritten chunks: 0
size of rewritten chunks: 0
rewritten rate in size: 0.000
read_time : 159.778s, 18.54MB/s
chunk_time : 22.414s, 132.16MB/s
hash_time : 20.757s, 142.71MB/s
dedup_time : 0.632s, 4690.47MB/s
rewrite_time : 0.031s, 94639.99MB/s
filter_time : 0.921s, 3217.98MB/s
write_time : 0.054s, 55311.96MB/s
```