Optimizing Storage Efficiency in Hadoop Distribution File System Through Data Deduplication

Amrita Raj[a] and Dr. G. Uma Devi[b]
University of Engineering and Management, Jaipur, India

Keywords: Data Deduplication, Hadoop Cluster, Hadoop Ecosystem.

Abstract: The exponential growth of data in modern computing environments poses significant challenges in managing storage resources efficiently. This study addresses the optimization of storage efficiency in the HDFS through strategic utilization of data deduplication process. The primary objective is to improve the overall performance and cost- effectiveness of Hadoop clusters by minimizing storage requirements and maximizing data accessibility. The proposed approach involves the implementation of data deduplication strategies to eliminate redundant copies of information, thereby reducing the overall storage footprint. This research addresses the challenges of managing large datasets in distributed environments, aiming to streamline data storage, minimize redundancy, and mitigate network traffic. Through data deduplication the study seeks to provide a comprehensive approach to optimizing storage resources within the Hadoop ecosystem.

1 INTRODUCTION

The era of big data, where large volumes of data are flooded into organizations from various sources, efficient storage management is paramount. The Hadoop Distribution File System has emerged as a cornerstone for storing and processing massive datasets across distributed clusters of commodity hardware. However, as data volumes continue to escalate, optimizing storage efficiency becomes imperative to mitigate costs and enhance performance. One promising approach to address the challenge of storage optimization within Through data deduplication, HDFS operates. The goal of data deduplication is to remove unnecessary duplicates of data, thereby reducing storage requirements, and improving overall efficiency. Eighty percent of respondents to a recent IDC study predicted that the amount of data that must be evaluated will significantly expand over the next 24 months in the form of GIFs, films, sensors, and more media. In fact, by 2025, IDC projects that global data volumes would have increased by 175 zeta bytes. Since this space is quite expensive for both home users and software companies, it presents a significant challenge for efficiently storing such massive volumes of data. Additionally, deduplication tests carried out by Google, IBM, Microsoft, and other corporations revealed that nearly 75% of digital data is redundant. Since all users maintain several replications of their data at different locations because they wish to keep their data safe and secure. Since the data must be extremely secure and consistent, incremental backups of the data may also be the cause of data redundancy. Consequently, to overcome the problem of replication which means duplicate data, a very efficient and successful technique for addressing the difficulties of freeing up storage space is data deduplication. In many of area, techniques of data deduplication play a crucial role. In latest times, these techniques have been working on the storage. Niteesha Sharma, A. V. Krishna Prasad, V. Kakulapati [2] have highlighted the challenges posed by the exponential expansion of unstructured electronic data, emphasizing the impact on data analysis and storage systems. It underscores the financial importance of deduplication as a capacity optimization technique, addressing redundancy in backup systems. The review covers types and techniques of data deduplication, with a focus on approaches proposed by researchers in the context of Big Data storage. The study concludes by highlighting the ongoing challenges and outlining

[a] https://orcid.org/0009-0007-1584-6999
[b] https://orcid.org/0000-0001-8297-8392

Raj, A. and Devi, G. U.
Optimizing Storage Efficiency in Hadoop Distribution File System Through Data Deduplication.
DOI: 10.5220/0013398600004646

In *Proceedings of the 1st International Conference on Cognitive & Cloud Computing (IC3Com 2024)*, pages 13-19
ISBN: 978-989-758-739-9

future, including the application of k-means clustering and nature-inspired algorithms for more efficient deduplication in HDFS.[2]

Naresh Kumar, Preeti Malik, Sonam Bhardwaj, S.C. Jain [3] have proposed a novel data deduplication approach integrating HDFS with MD5 hash value generation. The technique identifies identical data chunks, eliminating redundancy by generating 128-bit hash codes stored in buckets for rapid retrieval. Mapper and Reducer elements in the Hadoop framework enable parallel data processing. Traditional deduplication methods struggle with extensive data, affecting storage efficiency and time management. The proposed MD5-based approach demonstrates reduced time consumption, increased storage space savings, higher deduplication ratios, and efficient detection of duplicate files in real datasets within the Hadoop framework. The method enhances storage efficiency by eliminating redundant files and utilizing a bucket approach for indexing unique hash values.

Boafft is a distributed deduplication cloud storage technology that solves issues with data center data transportation and storage capacity. By using numerous data servers for parallel deduplication with little loss of deduplication ratio, Boafft reaches the capacity and scalable throughput. Using a hot fingerprint cache based on the frequency, inside the memory similar indexing in each data server , and data routing algorithm based on same data, performance is optimized. In a prototype implementation of HDFS, a comparative examination against EMC's state the algorithm of routing which discloses the Boafft's greater deduplication ratio, lower network bandwidth overhead, increased storage space use, and better load balance.[10] the challenges of storing small files in HDFS, which leads to excessive metadata and name node memory bottlenecks. The proposed MBDC algorithm optimizes HDFS storage efficiency for small files by considering file distribution and correlation. MBDC reduces the number of HDFS blocks, making related files closer.[14] The challenges of handling small files in HDFS within cloud storage, where the performance degrades due to many small files. It introduces optimized strategies, including replication algorithms (HAR and sequence File), merging algorithms, replica placement algorithms, and SSF techniques such as FMP and SSF-FMP with three-level prefetching-catching technology. These strategies effectively increase access and storage efficiency for small files, reducing the time spent on reading and writing when requested by clients in HDFS.[15]

2 PROPOSED METHODOLOGY

Our proposed methods aim to enhance storage efficiency by implementing a robust data deduplication mechanism within the HDFS

2.1 Data Deduplication Process

By storing a single instance of the data, this strategy eliminates duplicate data. The findings and removing duplicate data from cloud storage is the primary goal of deduplication.
Some advantages of data deduplication are:

- The space of the storage is reduced
- The network traffic is prevented

The following techniques are covered here, which are necessary for the data deduplication process:

- Data Chunking: It is a technique often used in data deduplication pro- cesses to enhance efficiency and reduce storage requirements. It involves breaking up large amounts of data into smaller, more digestible pieces, or chunks. This is an explanation of data chunking's function in data deduplication.
- MD5 Algorithm: The MD5 is a algorithm of message-digest that was created by R. Rivest in 1991 the MD5 algorithm creates a unique 32-character hexadecimal hash for a 512-bit input block, making it a reliable method for identifying duplicates. This architecture finds and eliminates duplicate data blocks across a distributed file system, saving storage space and enhancing system performance MD5 hashing.

MD5 algorithm is a 5 steps process:[3]
Step 1 Appending Padding Bits: The initial message is padded or stretched so the length which is in bits equals 448, modulo 512. As per the requirements for padding, document margins must be the following:

- One bit ("1") is always used for the initial original message.
- After that, the message is padded with zeros or extra "0" bits to make it 64 bits long but not longer than a several of 512.

Step 2 Appending Length: The padded message has 64 bits tacked till the end to show the initial message length in bytes. The following are the appended length rules: document margins must be the following:

- Converting the message's original length (measured in bytes) into its 64-bit binary

representation. If an overflow occurs, only the last or- der of 64 bits is utilized.
- Dividing the length of 64-bit into two words, each comprising 32 bits.
- Last order words are appended first, followed by first order words.

Step 3 Initializing MD Buffer: The technique requires a 128-bit buffer with a predefined beginning value. The initializing buffer rules are made up of the four words A, B, C, and D in the buffer, each of which has 32 bits.

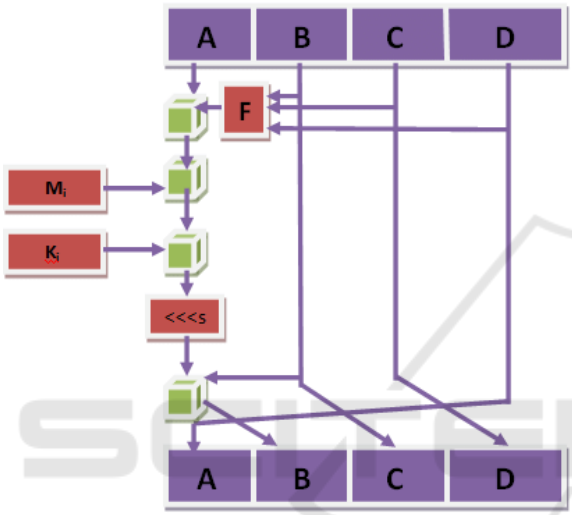

Figure 1: Single Step of MD5.

Step 4 Processing Message in 512-bit Blocks: The primary and most crucial step of the MD5 algorithm. The appended and padded message is cycled through in units of 512 bits apiece. There are four rounds of operations for every input, incorporating every sixteen operations.

Step 5 Output: The materials of buffer words A, B, C, and D are back, with their last order byte coming first in the sequence.

2.2 Proposed Algorithm

Data Processing and Storage in HDFS

Input: A file provided by the user.
Output: No output or data saved in HDFS.

```
While data is being accepted
     Retrieve the data (any format)
          If the data size exceeds
     available storage space
          Notify the user that
the request to save the
data is denied
Else

     Divide the data into
multiple blocks.
     Optimize the data to
save space.

     Calculate the message
digest using MD5 in a
distributed MapReduce
framework.

     If any collisions
occur in the digests

          Print the name of
     the file.

     Else
          Save the file to
HDFS.

     Measure and record the
time taken, storage space saved,
deduplication ratio, and store the
generated hash values in their
respective buckets.
End
```

2.3 Work Flow

Hadoop platform is used to offer the distributed environment needed for the suggested technique.

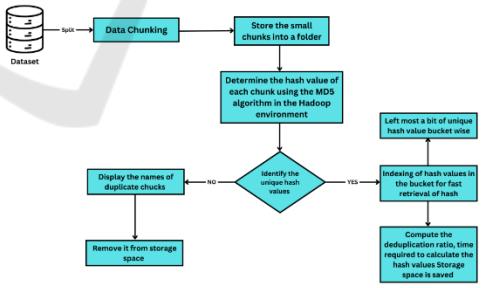

Figure 2: Flow Diagram of Data Deduplication Process.

Fig. 2 demonstrate the flow diagram of data deduplication process where the data is taken from the dataset then data chunking is the initial step in the dataset is the first step in the proposed system's architectural design. The dataset is divided into more manageable, smaller pieces throughout this procedure. After that, these tiny pieces are kept in a specific folder. The system then determines which of these pieces have unique hash values. The following

stage involves determining each chunk's hash value using the MD5 technique in a Hadoop environment, if a chunk's hash value is discovered to be unique or if it did not find the hash values to be unique then the that data is marked as a duplicate data and then it will show the list of duplicate chunks then after it will be removed from the storage. This Hadoop connection makes data reduction and processing more efficient. After the MD5 procedure, the resultant 128-bit hexadecimal hash codes are indexed for quicker retrieval. In particular, the leftmost bit of the hash code is used to determine which unique hash values are kept in which bucket. When needed, hash values may be retrieved quickly and effectively. The system makes sure that duplicate chunks are recognized and handled efficiently by adhering to this organized flow, which uses hash-based indexing to optimize storage and retrieval operations.

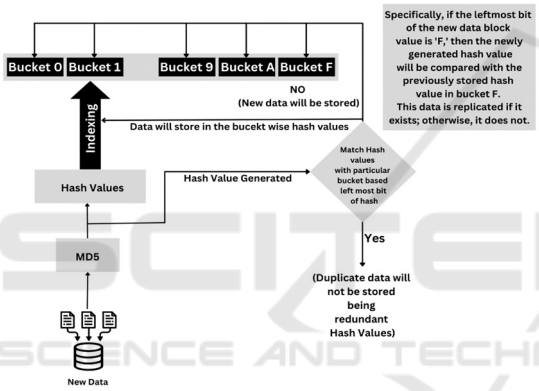

Figure 3: Bucket Approach's Work Flow from Bottom to Top.

Fig. 3 demonstrates according to the above-described the work flow of the bucket approach. By generating the unique hash values, the newly received input data stream is first sent via the MD5 algorithm technique. Whether any redundant hash values occur, the produced hash values match. The process of indexing distinct hash values involves grouping them into appropriate buckets according to the leftmost hash code bit. When necessary, this bucket technique facilitates a quicker hash code retrieval. When indexing hash values, the leftmost hash code bit is used as the bucket number i.e.

Bucket 0 stores this hash value if the leftmost bit is 0. Bucket 1 stores this hash value if the leftmost bit is 1. Bucket F stores this hash value if is the leftmost bit F. Bucket A stores this hash value if is the leftmost bit A.

3 EXPERIMENTAL RESULT AND ANALYSIS

The rate at which unstructured data is growing has increased, posing several data storage issues. One of the newer technologies that saves storage space and lessens the quantity of redundant data preserved is data deduplication. It consists of various result that we have achieved in the implementation. The results of various datasets are tested. The Hadoop 3.3.6 version is employed in this study. The explanation of several parameters that are utilized in the experimental analysis and aid in decision making.

Here, Data deduplication has two techniques which is used in this paper:

Backup Storage Deduplication: It refers to the traditional method of deduplication employed in backup systems or primary storage systems. Usually with a centralized design, these systems remove redundant data at the primary storage level by using deduplication techniques. The re- search may assess the effective of current techniques regarding dedupli-cation ratio, time efficiency, and space consumption by utilizing backup storage deduplication as a benchmark.

Distributed Storage Deduplication: This study presents a new method of data deduplication using the Hadoop platform and distributed storage. By leveraging distributed computing, efficient storage management, and parallel processing, the method gets around the limitations of traditional deduplication methods. It tries to demonstrate potential improvements in processing speed, storage capacity, and deduplication efficiency.

Here, come the various parameters which are used in this experimental analysis and assisted in making decision are elaborated below:

- Deduplication Ratio (DR): It is the proportion of data that would be delivered or stored with deduplication as opposed to data that would be stored without it. It is expressed as follows:

$$DR = \frac{\text{Total input data size before deduplication}}{\text{Total input data size after deduplication}} \quad (1)$$

- Hash Time: The whole time is required to extract the hash code from an input string is its definition. for a particular set of data, it is said that computing the hash values takes a specific amount of time.

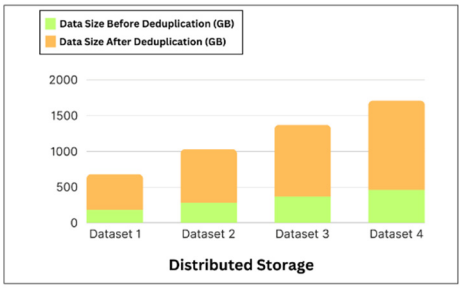

Figure 4: Graph shows the before and after data size of distributed storage deduplication.

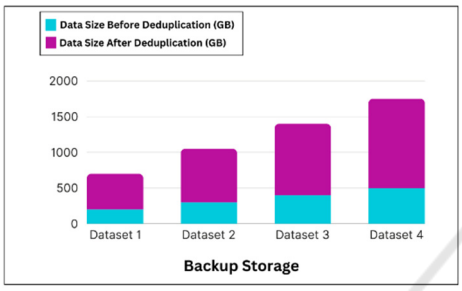

Figure 5: Graph shows before and after data size of backup storage deduplication.

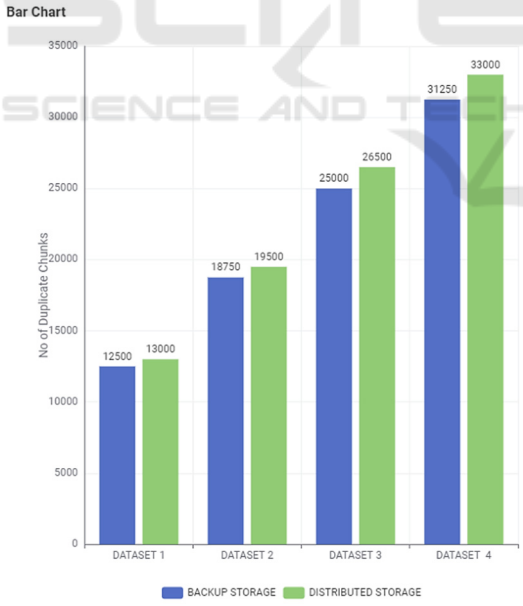

Figure 6: Graph shows the number of duplicate chunks.

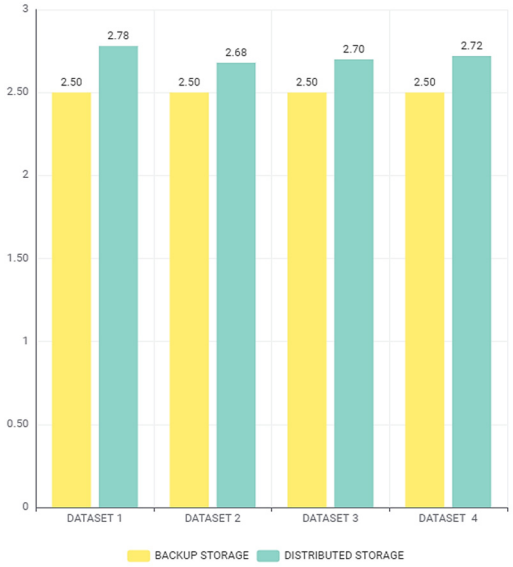

Figure 7: Graph shows the Deduplication Ratio.

Data deduplication approach was applied some techniques for the removing the duplicate data from the dataset which we have used in this project.

Table 1: Experimental Results on Various Datasets with Some Deduplication Techniques.

Dataset	Data Size Before Deduplication (GB)	Chunk Size (MB)	Techniques	Number Of Duplicate Chunks	Hash Time (ms)	Saved Space (GB)	Data Size After Deduplication (GB)	Deduplication Ratio
1	500	4	BACKUP STORAGE	12500	50	300	200	2.5
			DISTRIBUTED STORAGE	13000	60	320	180	2.78
2	750	8	BACKUP STORAGE	18750	70	450	300	2.5
			DISTRIBUTED STORAGE	19500	80	470	280	2.68
3	1000	16	BACKUP STORAGE	25000	90	600	400	2.5
			DISTRIBUTED STORAGE	26500	100	630	370	2.7
4	1250	32	BACKUP STORAGE	31250	110	750	500	2.5
			DISTRIBUTED STORAGE	33000	120	460	460	2.72

The result of the deduplication is presented in the figures Fig. 4, Fig. 5, Fig. 6, and Fig. 7 which illustrates the performances of completion times were improved due to the decreased amount of data that needed to be processed. This led to increased throughput, allowing more data processing to be completed in the same amount of time contrasted to the pre-deduplication state. The ratio of deduplication which compares the volume of data before and after deduplication, highlighted the effectiveness of the process in identifying and removing redundant data. Overall, the implementation of data deduplication in the Hadoop environment enhanced storage utilization and system performance, demonstrating its value in managing large-scale data efficiently. Table 1

presents the results of an investigation comparing the proposed distributed storage deduplication with traditional backup storage deduplication on a range of datasets. Figures from Fig. 4, to Fig. 7 shows the tested results. The results of data deduplication in a Hadoop environment demonstrated above significant improvements in storage efficiency and overall system performance. By utilizing MD5 hashing to identify and eliminate redundant data chunks, the total storage space required for the dataset was markedly reduced. The storage decreases the expenses and made room for more data by reducing the need for additional storage infrastructure.

4 CONCLUSIONS

The utilization of data deduplication methods in HDFS has heavily relied on data deduplication techniques to optimize storage efficiency. The method such as MD5 hashing can be effectively employed to identify and eliminate redundant data, hence decreasing storage expenses and enhancing processing effectiveness. This paper provides a hash-based method for data deduplicate using MD5. It is utilized in a distributed setting made possible by the Hadoop architecture. By making use of the mapper and reducer functions, this method maximizes storage space by ensuring that only unique files are kept and redundant ones are destroyed. Each file is uniquely hashed by MD5, acting as a fingerprint that enables bucket-based indexing of those files. By accelerating hash value computation, this all- encompassing method not only improves storage efficiency but also dramatically increases computational performance. Additionally, by greatly raising the deduplication ratio, this method reduces the overall storage footprint. The method's re- markable ability to recognize and handle unnecessary components is among the factors that make it so successful for data management jobs. With Hadoop's dis- tributed processing capability and the benefits of MD5 hashing, this endeavour can provide a dependable solution for data redundancy issues in large-scale storage systems.

ACKNOWLEDGEMENTS

The endless thanks go to Lord Almighty for all the blessings he has showered onto me, which has enabled me to write this last note in my research work. During the period of my research, as in the rest of my life, I have been blessed by Almighty with some extraordinary people who have spun a web of support around me. Words can never be enough in expressing how grateful I am to those incredible people in my life who made this thesis possible. I would like an attempt to thank them for making my time during my research in the Institute a period I will treasure. I am deeply indebted to my research supervisor, Professor Dr G UMA DEVI such an interesting thesis topic. Each meeting with her added in valuable aspects to the implementation and broadened my perspective. She has guided me with her invaluable suggestions, lightened up the way in my darkest times and encouraged me a lot in the academic life.

REFERENCES

Sais, N., Mahdaoui, J., 2023 *Distributed storage optimization using multi-agent systems in Hadoop*, E3S Web of Conferences 412, 01091 ICIES'11

Sharma, A., Kakulapati., 2019 *Data Deduplication Techniques for Big Data Storage Systems,* International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), vol.-8 Issue-10

Kumar, S., Bhardwaj, P., 2017 *Enhancing Storage Efficiency Using Distributed Deduplication On Big Data Storage System*. vol. 9, pp, Number-1

Balamurugan, K., 2021 *A Survey On Deduplication Techniques Handling Bigdata In HDFS*. vol 4- Issue 1, Paper 14

Phyu, T., 2001 *Capacity Optimized Deduplication for Big Unstructured Data in Scale-out Distributed Storage System.* Banff, Canada, pp. 174-187

R., S., 2017 *Image Storage Optimization using Deduplication,* International Journal of Scientific Engineering and Research (IJSER) vol 5 Issue 5

Fu, N., Jiang, F, Hu, W., 2017 *Application-Aware Big Data Deduplication in Cloud Environment,* IEEE

Kumar, S., 2017 *Secure Data Deduplication in Hadoop Distributed File Storage System,* Journal of Network Communications and Emerging Technologies (JNCET) vol 7, Issue 9, ISSN: 2395-5317

Powar, B., 2018 *Massive Volume of Unstructured Data and Storage Space Optimization*, International Journal of Engineering & Technology, 252-257

Luo, G., Li, S., Wu., 2015 *Boafft: Distributed Deduplication for Big Data Storage in the Cloud*, IEEE Transactions On Cloud Computing, vol. 61, No

Alange, A., 2022 *Optimization of Small Sized File Access Efficiency in HDFS by Integrating Virtual File System Layer,* International Journal of Advance Computer Science and Applications (IJACSA), Vol. 13, No. 6

M.S.Ali, B., 2020 *Big Data Optimization Techniques: An Empirical Study,* International Journal Of Scientific &

Technology Research, Volume 9, Issue 03, Issn 2277-8616

Chakravarthy, N.S., Reddy., 2023 *An Intelligent Storage Optimization Technique for Heterogeneous Hadoop Clusters,* IEEE, pp.195-199

Cail, C., Liang, 2018 *An optimization strategy of massive small files storage based on HDFS,* Joint International Advanced Engineering and Technology Research Conference (JIAET)

Deepika., 2015 *An Optimized Approach for Processing Small Files in HDFS,* International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value :78.96

Jiang, W., Xia, H., Tian, L., Feng, D., 2012 *Accelerating Data Deduplication by Exploiting Pipelining and Parallelism with Multicore or Manycore Processors.* In Proceedings of the 10th USENIX Conference on File and Storage Technologies

Jiang, H., Xia, W., Tian, L., Feng, D., F U, M., and Wang, Z., 2012 *P-Dedupe: Exploiting Parallelism in Data Deduplication System,* IEEE

Sengupta, S., Debnath, B., L.I., J., 2010 *Chunk Stash: speeding up inline storage deduplication using flash memory,* In Proceedings of the USENIX Conference on USENIX Annual Technical Conference

Xia, M., Zhou , Y., Jiang, H., Tian , D., Feng, L., F U , W.,.: D-delta, 2014 *A Deduplication- Inspired Fast Delta Compression Approach.,* Performance Evaluation 79, pp. 258–272.

Eshghi, K., Lillibridge, M., Bhagwat, D, 2013 *Improving Restore Speed for Backup Systems That Use Inline Chunk-Based Deduplication,* In Proceedings of the 11th USENIX Conference on File and Storage Technologies, pp. 183–197.

Bhagwat, D., Long, K., Pollack , J.-F, Schwarz , T., Pâris, E. L., Miller, 2006 *Providing High Reliability in a Minimum Redundancy Archival Storage System*, IEEE, pp. 413–421.

F Eng, D., Zhou, Y., Xia, F., F U, M., Huang, W., Z Hang, Y, L I, C., 2015 *SecDep A User- Aware Efficient Fine Grained Secure Deduplication Scheme with Multi-Level Key Management*, IEEE

https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

https://www.kaggle.com/code/rtatman/data-cleaning-challenge-deduplication

http://www.freedb.org/en/download da tabase.10.html