International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 06 | June 2022 www.irjet.net p-ISSN: 2395-0072

Data De-Duplication Engine for Efficient Storage Management

Kajol Pawar[1], Vrushali Phatale[2], Ranu Kumari[3], Ashwini Kanade[4], Swapnil Ujgare[5]

[123]Student, Department of Information Technology, Bharati Vidyapeeth's College of Engineering for Women, Pune, Maharashtra, INDIA
[4] Assistant Professor, Dept. of Information Technology Engineering, Bharati Vidyapeeth's College of Engineering for Women, Pune, Maharashtra, INDIA
[5]Software Engineer, Veritas Technologies LLC, Pune, Maharashtra, INDIA

Abstract: *In today's world where we depend on internet for every small piece of information or for sharing the data, we generate new data every second. This data can include texts, videos, audios, images, etc. and storing or maintaining this big data is of concern. Businesses invest heavily in data storage and this is why there is a need to manage big data. De-duplication is emerging technology for removing redundant or duplicate data. De-duplication can be applied to any type of storage data like primary storage, secondary storage and also for cloud storage. De-duplication reduces the data to be stored hence the cost is reduced, also the bandwidth required for data sharing is reduced considerably. These are some of the advantages of using de-duplication. Data de-duplication is efficient for large scale storage systems rather than traditional approaches. This paper focuses on data de-duplication and its various types and the main stream is the implementation of de-dupe engine.*

Key Words: Big data, de-duplication, client-server, chunking, fingerprints, inter-process communication

1. INTRODUCTION:

In the era where data, specifically digital data has self-evident importance in the age of globalization, it has become essential to eliminate any circumstances or drawbacks in the field of data processing and management. One such challenge is data redundancy or duplication, where multiple copies of same digital data take up storage space on our system, and numerous other issues like bandwidth inefficiency, increased hardware and backup cost and network inefficiency. Reasons of redundancy:

An enormous portion of internet data is redundant for 2 reasons. Firstly, because of the significant decline in the storage cost per GB, people tend to store multiple copies of same file for data safety or user convenience. Secondly, while incremental (differential) data backups or disk image files for virtual desktops tend not to have duplicated whole file copies, there is still large fraction of duplicated data portion from the modification and revision of the files.

Chunks are the small divided portion of the complete data that may be fixed sized or variable sized. The techniques that track and eliminate any duplicate chunks of data in the specific storage unit are implemented, such techniques are called data de-duplication techniques. Data de-duplication is more important at the shared storage level, however implemented in software as well as database. Basically, de-duplication takes place at file level and block level. In file level de-duplication, it eliminates duplicate or redundant copies of same file. This type of de-duplication is called Single Instance Storage (SIS). In block level de-duplication, it eliminates redundant or duplicate blocks of data which is present in unique files. Block level de-duplication reduces more space than SIS, this type of de-duplication is known as variable block or variable length de-duplication. De-duplication on more finely grained chunks at block level shows better performance in removing duplicate data and thus creates more opportunities for space savings, but it may reduce the sequential layout of some files. Alternatively, whole file de-duplication is simpler and eliminate file-fragmentation concerns.

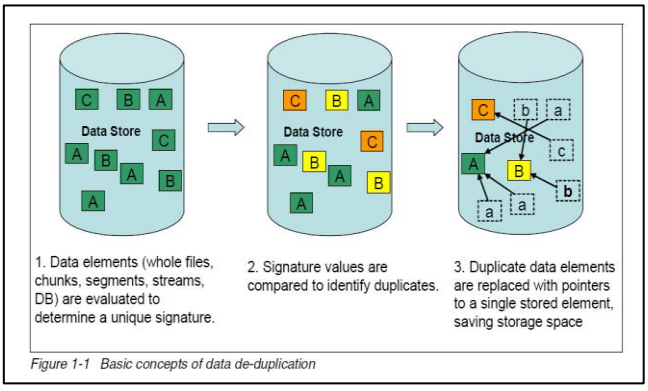

Fig.1. General De-duplication

With the help of data de-duplication on our storage environment, only one and unique copy of data is retained on storage media, and redundant or duplicate data is placed with pointer to unique data copy. That is, it looks at the data on a sub file or block level, and attempts to determine if it is seen the data before. If it has not, it stores it. If it has seen it before, it ensures that it is stored only once, and all other references to that duplicate data are mere pointers. De-duplication provides the benefits such as increased network efficiency, bandwidth efficiency and storage efficiency, improved speed replication, reducing backup window and thereby cost.

2. CLASSIFICATION:

There are various ways to classify, but generally 4 ways of classification are-

i. Point of application
ii. Time of application
iii. Granularity
iv. Algorithm

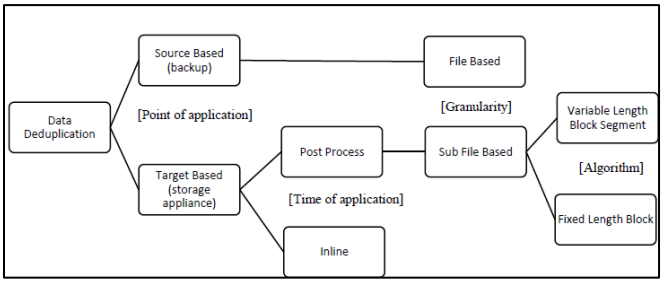

Fig.2. General Classification

2.1. Source De-duplication

Source de-duplication is in where the redundancies are removed before transmitting the data to backup target. This

offer a reduced bandwidth and network storage. For large amounts of data, source de-duplication may be slower than the target de-duplication but the rate of transfer is pretty good as the duplicate data has been removed before the backup transmission. Due to extensive workload on the servers overall time required for backup may increase. This type of de-duplication is suited for backing up small backup sets.

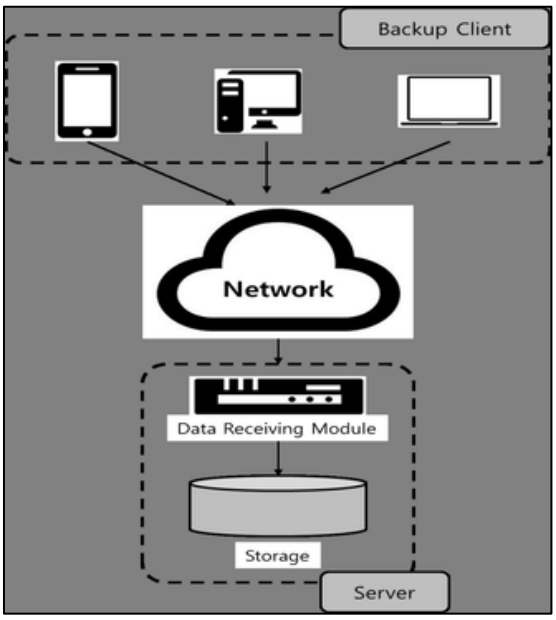

Fig.3. Source De-duplication

This de-duplication takes place at client-side or at server-side. Only the unique segments or block of data which has been changed since previous backup is only considered for the backup. The cost of traditional backup infrastructure like hardware, software, tapes, courier, drives, etc. gets eliminated. Advantages of source based de-dupe are-

1. Rapid backup window
2. Reduction in network traffic
3. Data transfers are efficient

2.2. Target De-duplication

The redundancies are removed from the backup transmission as it passes through the appliance between source and the backup target. Intelligent Target Disks (ITD) and Virtual Tape Libraries (VTL) use target de-duplication. Same as source de-duplication, target de-duplication reduces the amount of storage required but it does not reduce the amount of data to be transferred

for backup. This type of de-duplication requires hardware but it is sometimes considered as a drawback.

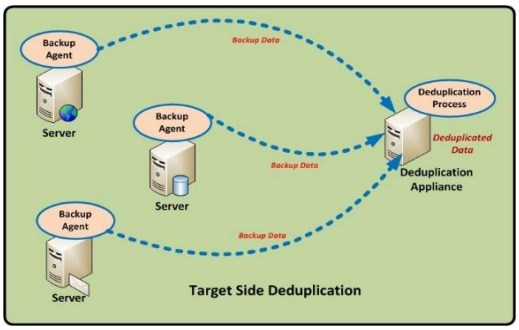

Fig.4. Target De-duplication

Target de-duplication provides faster performance for big datasets. This de-duplication can be used for protecting large SQL or Oracle databases. Target de-duplication can integrate with the existing infrastructure of the backup system. This de-duplication reduces the consumption of primary storage resources. Advantages of target de-duplication-

1. Reduction in backup windows
2. Make minor changes in system to improve efficiency

2.3. Post Process De-duplication

Post process de-duplication is also called as asynchronous or offline de-duplication. In this, the removal of redundant data is done after backup is complete and data has been written on storage. As compared to inline de-duplication post process de-duplication is faster. In this technique, duplicate data is removed and replaced with a pointer. This requires a lot of storage space as duplicates are removed after storing data and this is a drawback. Advantage of this de-duplication is that it is straight-forward and takes less time. Post process products are ExaGrid EX.

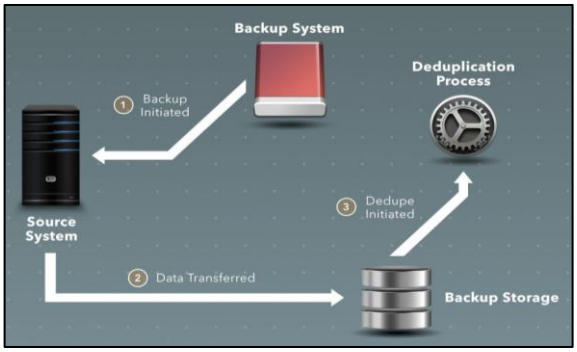

Fig.5. Post Process De-duplication

2.4 Inline De-duplication

Inline de-duplication is also called as synchronous or online de-duplication. Redundancies and duplicate data is removed while writing data on backup device. Inline de-duplication can make system work slower because the devices are in the data path between the servers and backup disk systems. This system can work as source or target-based.

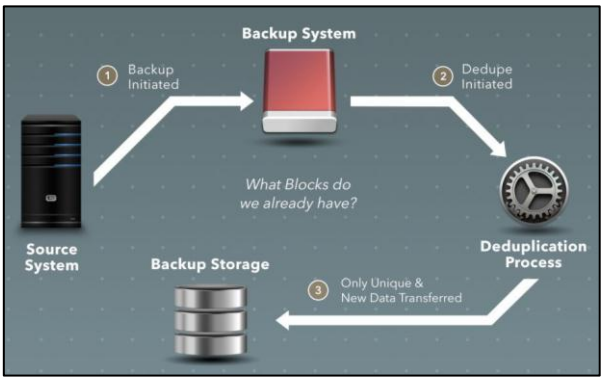

Fig.6. Inline De-duplication

Inline de-duplication requires less space than post process as the redundancies are removed before storing and do not require temporary space. Inline de-duplication can cause bottleneck slowing down the overall process. NetApp's SolidFire division offer products with inline de-duplication. Some famous inline de-duplication products are IBM Spectrum Virtualize, Veritas NetBackup.

2.5. File-based De-duplication

File-based de-duplication is commonly known as Single Instance Storage (SIS). In this de-duplication, whole file is compared with the files already stored by checking its attributes. If the new file is unique then it is stored and if the file is repeated only a pointer to existing file is stored. Duplicates are not stored. This technique requires less processing power due to smaller indexes and less number of comparisons.

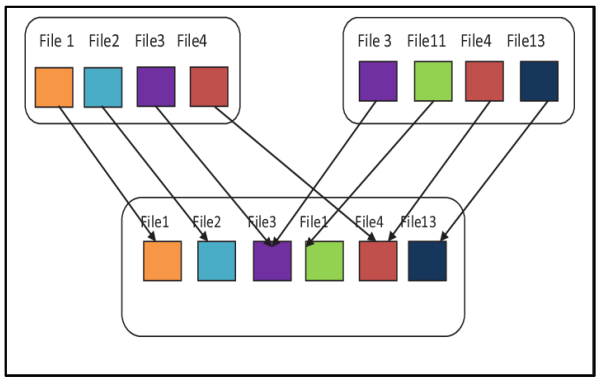

Fig.7. File level De-duplication

The time required for computation is less. The main big advantage of file-level de-duplication is that it requires less resources but it also has a major drawback that it cannot eliminate smaller redundant chunks of data. Hashing algorithms like MD5 or SHA-1 are used for calculating the hash of file and then comparing hash values to find the duplicates.

2.6. Sub-file based De-duplication

Block-level de-duplication works on sub-file level. In this, a file is broken into chunks or blocks that are then checked for redundancy with previously stored information. For this hashing algorithm is the best. Using the hashing algorithm, it creates a unique id or fingerprint for each block. If the id exists previously then the block is already processed.

The size of block depends on different peoples. Some may have fixed size block and some have variable size block. For fixed block the usual size is 8 KB or 64 KB. Smaller block size has more probability of finding the redundancies. This means very less storage space is required as there is almost no duplicates available after processing the data. There is an issue with this fixed size blocks that if a file is updated and the system uses the same old blocks of data then it might not detect any redundancy.

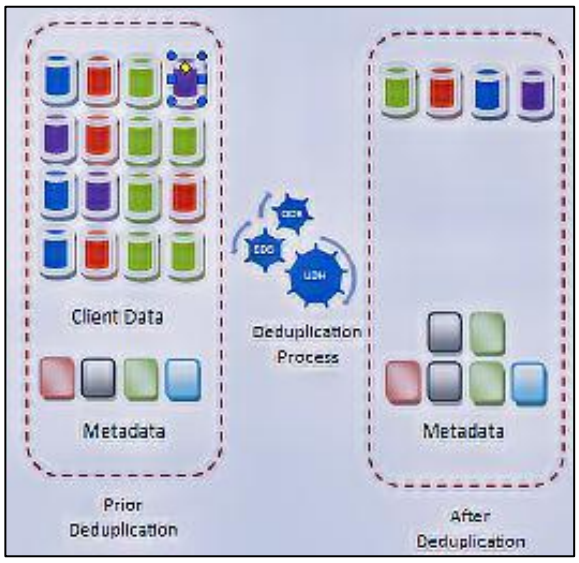

Fig.8. Block level De-duplication

The variable sizes of blocks used in variable-sized block de-duplication creates confusion. This approach can find redundant data even if the file is been shifted to another place. There is high de-duplication ratio as compared to file-base de-duplication.

3. METHODOLOGY

For implementing de-dupe engine for files with de-duplicate data, we established client server communication using sockets for inter process communications. We first created server that will be a daemon process. When client requests server to fulfill its requirements, server receives those messages through sockets. Thus, server could bind and listen to client before accepting the client requests. The messages between client and server are simultaneously stored at log file created. The client utility is basically a process that could parse user entered arguments and perform actions accordingly. Our implementation is able to parse arguments that could either be name of input file or name of output file.

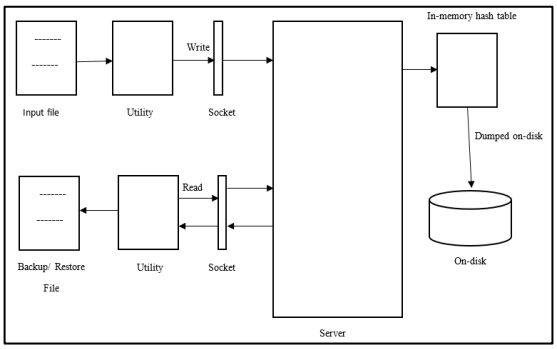

Fig.9. Architecture of De-dupe Engine

The input file is used by client for asking the server to fulfill its different kind of requirements for the given file. The requirements (tasks) may include:

- identifying unique chunks of data,
- writing the unique chunks and its related information to metadata file,
- incrementing the reference count of any duplicate chunks in the metadata file at the
- corresponding metadata position,
- creating backup file for original file using only the metadata file created above.

The output file name is given so that it is used to read or restore from metadata created. The in-memory fingerprint is dumped on disk, which is referred to as metadata file. Dumping is done to have the in-memory fingerprints available for reference all time. The metadata is essentially an array of user defined structure namely de-dupe footprint. The elements of this structure are:

- data buffer
- data length
- data checksum
- reference count
- dedupe offset
- submeta (sub structure which includes the elements filename and offset)
- next pointer referencing this structure type itself (for purpose of adding structures dynamically)

Suppose client utility wants to write a file, server process read that input file in chunk of 128 KB. While reading the file from beginning, the first 128 KB data is read and stored into data buffer, which forms our chunk that we are stating throughout project. The checksum of this data chunk gets calculated. Checksum is calculated using md5sum checksum algorithm. Since each unique chunk has unique checksum, and duplicate chunks have same checksum being calculated, we will be using this fact of matter during our project implementation. We search that checksum in in-memory fingerprint.

At the start the first data chunk will be unique with its respective checksum. This checksum is also used to determine the hash index position in the in-memory fingerprint and metadata file. The remainder obtained on dividing the checksum value by the hash table size, is used as our hash index position where the data and the metadata of the chunk gets stored directly.

As we move further to read next 128 KB of data, we may face following situations: -

1. The checksum of this data chunk is found in the in-memory fingerprint –

   In this case, data and metadata of this duplicate chunk is not stored, just the reference count of respective metadata will be incremented, its de-dupe offset will be stored and submeta for additional information of the duplicate chunk will be updated in the in-memory fingerprint as well as metadata file for backup or restore purpose. This optimizes the storage space and cost.

2. The checksum is not found in the in-memory fingerprint but hash index calculated will be one which is occupied by metadata of earlier chunks –

   In this case, the chunk is unique, but since hash index position is already occupied, it will append this node of metadata to the previous node at that particular hash index using its next pointer, in the in-memory fingerprint as well as in the metadata file. This allows dynamic insertion of data structures and optimize space by eradicating the use of static structures with large hash index values.

3. The checksum is not found in the in-memory fingerprint as well as hash index will be one which is unoccupied—

   In this case, the chunk is unique as well as its hash index position is unoccupied, so the data of this chunk and its metadata can be directly stored at this hash index position of the in-memory fingerprint as well as of the metadata file.

Whatever the data remaining at end, which may be less than 128 KB are stored in metadata file along with its respective metadata, similarly.

Thus, the complete data file is read and simultaneously its chunks are searched and identified if duplicate or not, and instead of storing each chunk in system and occupying more storage space, we use it to create metadata that optimizes storage space as well as cost. This could also be used for backup purpose, when original file contents may be lost. This is done by creating a restore file that completely replicates the original file, without any data loss, by only using this metadata file created.

The restore file is created as described below: -

The particular filename in in-memory fingerprint is searched for and matched with filename of original file of whose restore file is to be created. If found, we look for the corresponding offset and accordingly write the corresponding complete data from its data buffer at the offset position in restore file. The filename and offset of next node in in-memory fingerprint is searched and written to its restore file at particular offset in similar way. This is done in iteration until complete restore file is written of original input file is written.

Thus, we used de-duplication technique with fingerprinting phenomenon is used to write de-duped metadata for backup and storage efficiency purposes, and log files for referring the communication between client and server processes.

4. IMPLEMENTATION

For initial testing of code, a large text file of size 50 MB was taken onto the system which is having LINUX environment (Ubuntu). For ease, we took small portion of this file of around 1 MB which also includes duplicate data portions inside it. Our task was to perform block level de-duplication over this file. We divided this file into fixed size chunks or blocks of data of size 128 KB, performed hashing over these chunks and stored the resultant hash value into the memory.

We used MD5 hashing algorithm for generating the checksum. This checksum will be unique for every unique chunk. Small changes to any particular block of data or chunk produces different checksum that is uncorrelated with its previous checksum. Thus, we verified that adding or deleting some portion of data into the file at some places, produces different checksum and assured that the algorithm performed well cryptographically. Thus, the chunks are produced in the cryptographic algorithm to yield a corresponding checksum or digital print for it.

Because of the hash properties each and every block maintain its unique identifier if those blocks are unique, that is, there is no any duplication. If the block is repeated, we can identify it easily with the help of its repeated checksum. Using this principle, only the unique blocks are saved into the hard disk or secondary memory, and its metadata is created into the hash table with its reference count of occurrence. This hash table is implemented using arrays of linked list and the nodes of these linked list being structures that represents the metadata.

Even though, same hash index is generated from any hashing formula for two or more chunks, its checksum will be different for different chunks. Thus, metadata of these chunks will be stored at same index in the hash table array with the help of linked list at the particular index.

If duplicate chunks are already present in the memory, according to user requirement we have implemented function to search and delete those duplicate chunks, and corresponding changes are made to the hash table. Such kind of hash table is dumped into the hard disk or secondary memory, which is then used by the client-server architecture model.

In the client-server model, the server accepts and processes the requests from clients, with the help of sockets. Sockets act as carrier of utility requests and required inputs between clients and server, which are requested by clients. Server accepts and performs the utilities with the help of the hash table in hard disks, and successfully returns the result to clients. Server thus acts as daemon process in this architecture which continuously feed utilities to clients when requested. Thus, we aim to perform block level de-duplication with the help of client-server architecture.

5. RESULTS AND EVALUATIONS

In the results, we found that after de-duplication process the read/write time required for a particular time is decreased. For this, we took text file of few kilo bytes (KB). We chunked it and created a metadata in hash table and dumped it into memory. The in-memory fingerprint of hash table is used for maintaining the records even after the program termination. The server can read data from this file and create a restore file if there is any data corruption or data loss.

As the chunks are of small size, duplicate data identification is done on large scale. It is very difficult to identify duplicate data in a larger chunk size. Also, for future use if the client

requires on a specific data chunk then only that can be send. This also requires less bandwidth and transmission time.

The client-server communication allows many clients/users to connect at a time with the single server. Simultaneous read/write processes can be done. This saves time of users and there is user satisfaction. The checksum algorithm helps to detect if the data is corrupted or not. Fingerprints are very useful to identify the duplicate data chunks. They work as unique identifiers for each unique chunk.

Hence, we can measure de-duplication engine performance based on time and space. It can be used for backup and restore of data. In big organizations, the performance of de-dupe engine will help gain more profit.

6. CONCLUSION AND FUTURE WORK

Data storage is important issue. Many individuals as well as organizations face this issue. There are many ways for storage which aren't cost effective. Also the methods should be reliable. The data must remain safe. Due to big data there is problem of network traffic. Due to multiple copies of same data there is a lot of confusion between employees and end consumers.

Data de-duplication is efficient way for storing backups and big data. De-duplication reduces the space required for the storage. As the space is reduced the cost also reduces. The problem of network traffic reduces and bandwidth wastage for transferring the data is also reduced. As de-duplication cleans the data and stores only the unique copies of data, productivity increases.

Also the brand reputation is intact as there is no confusion for end consumers. Time is saved to greater extent and can be utilized for more creativity and productivity rather than cleaning data manually.

In the future, we will add an important module of deleting the duplicate chunks from the input file and receive a file without duplicate data. This will reduce the storage space required and reduce the read/write time required.

ACKNOWLEDGEMENT

To complete any type of project work is team work. It involves all the technical/non-technical expertise from various sources. The contribution from the experts in the form of knows-how and other technical support is of its vital importance. We are indebted to our inspiring guide **Prof. A.V. Kanade** and our project coordinator **Prof**. **Dr. K.A. Malgi.** We are thankful to **Mr. Swapnil Ujgare** for guiding us in technical coding part of the project and for lending support throughout the project.

We are also very thankful to **Prof. Dr. D. A. Godse** for her valuable guidance. We have great pleasure in offering big thanks to our honorable principal **Prof. Dr. S. R. Patil**. Last but not least, we would like to thank all the direct and indirect help provided by all the staff and our entire class for successful completion of this project.

REFERENCES

1. E. Manogar, S. Abirami, International Conference on Advanced Computing (ICoAC), "A Study on Deduplication Techniques for Optimized Storage", 2014.

2. Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, Yukun Zhou, "A Comprehensive Study of the Past, Present and Future of Data Deduplication", IEEE 2016, Vol. 104.

3. Zhou Lei, Zhao Xin Li, Yu Lei, YanLing Bi, Luokai Hu, Wenfeng Shen, "An Improved Image File Storage Method Using Data Deduplication", IEEE 2014.

4. Chao Tan, Luyu Li, Chentao Wu, Jie Li, "DASM: A Dynamic Adaptive Forward Assembly Area Method to Accelerate Restore Speed for Deduplication based Backup Systems", published by Springer International Publishing AAG 2016.

5. Panfeng Zhang, Ping Huang, Xubin He, Hua Wang, Ke Zhou, "Resemblance and Mergence based Indexing for High Performance Data Deduplication", Journal of Systems and Software, Science Direct, 2017.

6. Meyer D. T., & Bolosky W. J., "A Study of Practical Deduplication", ACM Transactions on Storage, 7(4), 1–20. doi:10.1145/2078861.2078864

7. Ahmed Sardar M. Saeed, Loay E. George, "Fingerprint-based Data Deduplication Using a Mathematical Bounded Linear Hash Function", Symmetry 2021, 13.
8. Dutch T. Meyer, William J. Bolosky, "A Study of Practical Deduplication", ACM Transactions on Storage, Vol. 7, No. 4, Article 14, January 2012

9. https://linux-kernel-labs.github.io/refs/heads/master/lectures/intro.html

10. https://www.tutorialspoint.com/unix/unix-file-system.htm

11. https://github.com/libfuse/libfuse

12. https://www.kernel.org/doc/html/latest/filesystems/fuse.html

13. https://docs.microsoft.com/en-us/windows-server/storage/data-deduplication/overview

14. https://opensource.com/article/19/4/interprocess-communication-linux-networking

15. https://beta.computer-networking.info/syllabus/default/exercises/sockets.html

16. https://vdc-repo.vmware.com/vmwb-repository/dcr-public/c509579b-fc98-4ec2-bf0c-cadaebc51017/f572d815-0e80-4448-a354-dff39a1d545e/doc/vsockAppendix.8.3.html

17. https://en.wikipedia.org/wiki/Cyclic_redundancy_check#:~:text=This%20is%20a%20practical%20algorithm,%C2%A7%20Multi%2Dbit%20computation).

18. https://www.thegeekstuff.com/2012/02/c-daemon-process/

BIOGRAPHIES

Ms. Kajol Pawar is perusing B.E. (IT) from Bharati Vidyapeeth College of Engineering for Women, Pune and presently is in Fourth Year.

Ms. Vrushali Phatale is perusing B.E. (IT) from Bharati Vidyapeeth College of Engineering for Women, Pune and presently is in Fourth Year.

Ms. Ranu Kumari is perusing B.E. (IT) from Bharati Vidyapeeth College of Engineering for Women, Pune and presently is in Fourth Year.

Mrs. A.V. Kanade is working as a assistant professor in department of IT, Bharati Vidyapeeth College of Engineering for Women.

Mr. Swapnil Ujgare is currently working as Software Engineer in Veritas Technologies LLC.