Mathematical Statistician and Engineering Applications
ISSN: 2326-9865
DOI: https://doi.org/10.17762/msea.v70i1.2299

Improving the Performance of Primary Storage Systems Using Leveraging Data Deduplication

Mahantesh K Pattanshetti

Professor, Department of Computer Science, Graphic Era Hill University, Dehradun, Uttarakhand India 248002

Article Info
Page Number: 190-197
Publication Issue:
Vol. 70 No. 1 (2021)

Article History
Article Received: 25 January 2021
Revised: 24 February 2021
Accepted: 15 March 2021

Abstract

Data deduplication is a specialised data compression technique used in computers to get rid of redundant copies of repeated data. Intelligent (data) compression and single-instance (data) storage are words that are similar and partly interchangeable. This method may be used to increase storage efficiency and the quantity of data that has to be sent should be reduced over networks. Unique data chunks, or byte patterns, are found and saved throughout the deduplication process through a process of analysis. Recent studies have demonstrated that main storage systems in the Cloud offer moderate to high levels of data redundancy. It demonstrates that as a result of the small I/O requests to redundant data's comparably high temporal access locality, data redundancy displays a substantially greater degree of intensity on the I/O path than that on discs. The system in this project suggests using a performance-oriented I/O deduplication technique known as POD. It is used to increase the I/O performance of main storage systems in the Cloud without compromising the latter's capacity savings.

Keywords: Data Deduplication, Primary Storage System, Data Compression, Single Instance Storage, Storage Efficiency.

1. Introduction

Deduplication can take place "post-process," after the data has been written, or "in-line," as the data is being processed. Data deduplication is an advanced data compression method used in computers to get rid of redundant copies of repeated data. Intelligent (data) compression along with single-instance (data) storage are words that are similar and partly interchangeable. This method may be used to increase storage efficiency and cutting down on the quantity of data that needs to be sent over networks. Unique data chunks, or byte patterns, are found and saved throughout the deduplication process through a process of analysis.

The superfluous chunk is substituted with a brief reference that leads to the saved chunk if a match is found as the analysis proceeds by comparing more chunks to the stored copy. The quantity of information that must be saved or communicated can be drastically decreased since Depending on the chunk size, the identical byte pattern may show up dozens, hundreds, or possibly thousands of occasions.

Compared to the deduplication carried out by common file-compression programmes like LZ77 and LZ78, this method of deduplication is distinct. In order to keep just one duplicate of a big volume of data, storage-based data deduplication inspects the volume and identifies significant chunks that are similar, such as complete files or substantial sections of files. These tools only recognise short repeated substrings within individual files. Single-file compression methods might be used to further compress this copy. A typical email system, for instance, may include 100 copies of a single 1-MB (megabyte) file attachment. All 100 copies of the attachment are preserved whenever the email platform is restored, consuming 100 MB of storage. Only one duplicate of the file is really stored when data deduplication is used; subsequent copies are linked to the preserved Approximately 100 copies are made for every one deduplicated copy.

Comparing data chunks to find duplicates is one of the most popular ways to conduct data deduplication. Each data chunk is given an identifier for that purpose, which is computed by the programme often using cryptographic hash methods. The pigeonhole principle prevents this from always being true, however, in many implementations, it is deemed sufficient if the identification is the same, the data must also be identical. In contrast, in other implementations the assumption the similarity between two chunks of data with the same identifier is tested to ensure that it is true.

Depending on the implementation, the programme will either presume that a certain identifier exists already in the deduplication namespace or actually verify the identity of the two blocks of data before replacing the duplicate chunk with a link. Shortening the backup window, improving storage space economy, and maximising network bandwidth consumption have all been demonstrated to be possible using data deduplication in cloud backup and archiving software. To enhance the I/O performance of main storage systems in the Cloud by taking into account the workload characteristics, we suggest a Performance-Oriented data deduplication technique, known as POD, as opposed to a capacity-oriented one (such as iDedup). This will deal with the serious performance problem with cloud main storage as well as the aforementioned problems brought on by deduplication. POD uses a request-based selective deduplication technique as part of a two-pronged strategy to boost main storage system performance while minimising deduplication's performance cost.

2. Literature Survey

This work provides range writes, a straightforward but effective modification to the disc interface that eliminates the need for block placement micromanagement by the file system. Range writes allow the disc to choose the request's final on-disk location by letting a file system define a selection of potential address destinations, which increases efficiency by a factor of three. This problem has been addressed in the past by shifting to a higher-level object-based interface or remapping blocks instantly. In order to solve the issue of micromanagement range writes, this work offers an evolutionary adjustment to the disc interface. The file system gives the disc a list of potential placement locations so that it may choose one determined by internal positioning data. Expand-and-cancel scheduling and hierarchical range scheduling, two unique methods for scheduling range writes, are developed. The simulation demonstrates that both of these schedulers operate superbly, drastically lowering write latency as the number of targets rises [1].

When publishing dirty pages to disc or network file systems, Operating system memory managers fail to consider the number of read vs write pages in the buffer pool or open I/O requests into account.Bursty I/O patterns result from this, which slow down data reading operations and lower storage efficiency. We overcome these constraints by opportunistically In addition to writing dirty pages to disc before the operating system submits them for write-back, you may also do this by adjusting how much RAM is allocated across write buffering as well as read caching. For mixed read/write workloads, the author shows performance increases of over 30%. To maintain a balance between read as well as write workloads while developing adaptive destaging techniques that respond to workload, read, write, and free page populations in memory, operating system memory managers must be improved. For the purpose to lessen the effects to write traffic on read performance and to enhance disc throughput while increasing disc throughput, the author defines an adaptive system for destaging dirty pages to disc. Performance is increased by modern operating systems delaying the writing of dirty memory buffers to storage. In contrast to read operations, writing actions are less crucial since As a consequence, a procedure is not disrupted of a write system call. However, there are two downsides to maintaining the page in volatile memory for an extended length of time [2].

The file system information from more than 60,000 Windows PC file systems in a major company is examined in this article with regard to temporal changes. It presents a generative model that describes namespace structure, directory size distribution, and major temporal patterns related to the frequency of different file kinds, the source of file content, and the way the namespace is operated, as well as the level of variation between file systems. The data set contains the most file-system metadata that has ever been gathered also it covers the most time of any significant metadata collection. The author created a programme that iterates over each local, fixed-disk file system mounted on a computer's directory tree and captures a snapshot of all the metadata related to each file or directory, such as name, size, timestamps, and characteristics. A significant portion of the Microsoft employees were sent the scanning programme by email, and they were entered into a lottery with each computer they scanned as an entry. The same file system is also described as having two snapshots that have the same volume ID, drive letter, user name, machine name, as well as amount of space. The author adds an abscissa for the zero value in addition to using a logarithmic scale for nonzero values in order to express these ranges concisely [3].

Inline chunk-based data deduplication systems struggle with chunk fragmentation, which slows down the pace at which the most recent backup may be restored. To solve this issue, three methods are investigated: enlarging the cache, container capping, and employing a forward assembly area. Although employing a forward assembly area lowers container capping lowers chunk fragmentation but sacrifices some deduplication in order to reduce the amount of RAM required for a given degree of caching at restoration time. Rearranging chunks can be costly and has the most impact on most recent backups. The two methods for enhancing restoration speed in deduplication systems are examined in this research. The first strategy is capping, which restricts the number of locations that may be accessed each backup MB. The second strategy is the forward assembly area technique, which develops a novel, more effective caching and prefetching mechanism for recovering deduplicated data using knowledge of accesses in advance. The backup programme creates a stream out of the source data and saves chunks in a number of files referred to as chunk containers. When a fresh piece is received, when an open container exceeds a certain size, it is closed and replaced with a new, empty one. It is appended to the end of the relevant open container [4].

In order to effectively read data for further data analysis, visualisation, checkpoint restart following a failure, as well as other read-intensive tasks, this study looks at several read-intensive read-out techniques. It establishes "read" benchmarks to identify the typical read patterns employed by analytical algorithms and compares experimentally the read performance seen with various data volumes, organisational structures, along with read process counts. The findings show that from high to low Data structures that allow for flexibility in data layout and placement on parallel storage targets are required for high-performance IO, including strategies that can balance the performance of data writes vs. reads. Experimental evaluations centred around these patterns measure how well data is presented in two distinct manners, and this work offers six frequent read patterns for petascale scientific programmes. The second assessment utilises a log-based data structure, which maximises the quantities of relevant data received by each read, whereas the first evaluation employs a logically contiguous architecture suggested by the NetCDF and HDF communities. The benefit of this method is that it maximises the quantity of usable data collected by each read while taking use of the numerous storage targets offered by parallel file systems [5].

3. Proposed System

The suggested system includes an I/O deduplication that is focused on performance. Specifically, the POD (performance-oriented data deduplication) approach. It is suggested to deal with the significant performance issue of cloud-based main storage. Additionally to lessen issues brought on by deduplication. By taking into account the workload characteristics, it helps to optimise the I/O performance of main storage systems in the Cloud.

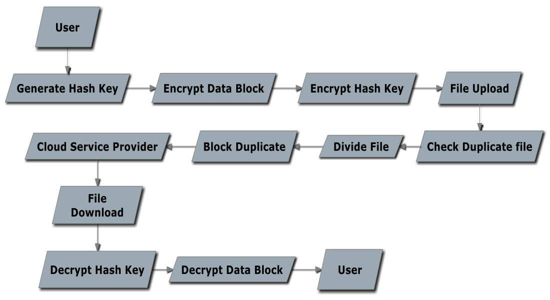


Fig 1: System Architecture

It incorporates small-I/O-request dominance workload factors into design decisions. If the write data is already sequentially recorded on the discs, it deduplicates every write request, including the short write requests that capacity-oriented deduplication techniques would ordinarily skip. Storage controllers can offer speedy write response times by obscuring disc delay using non-volatile write-back caches. Effective write cache management is essential for storage controller performance. The pace of destages must still be taken into account as it is a crucial component of write caching. Destage at a consistent rate while monitoring the write cache's usage to avoid under- or over-committing for the optimal performance.

Analysing data chunks to look for duplicates is how one of the most popular data deduplication solutions operates. Each data chunk is given an identifier for that purpose, which is computed by the programme often using cryptographic hash methods. The pigeonhole principle prevents this from always being true, however, it is often assumed in implementations that if the identification is the same, the data must be the same as well. Instead than assuming that two blocks of data with the same identity are similar, some systems actively verify that data with the same identification is identical. In accordance with the implementation, the software will either assume that a particular identifier is currently present in the deduplication namespace or will actually validate the identification of the two blocks of data before replacing the duplicate chunk with a link.

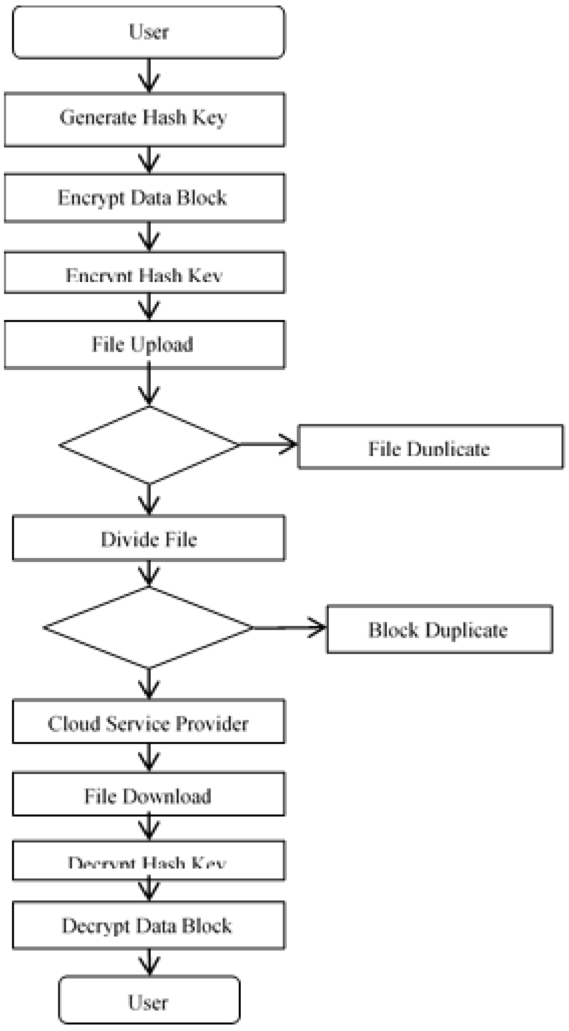


Fig 2: Flow Diagram

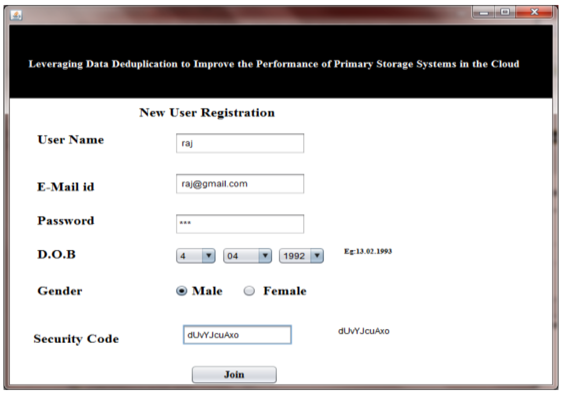
Data deduplication has mostly been utilized to secondary storage systems up to this point. There are two explanations for this. Data deduplication first needs extra work to find and get rid of duplicate data. This overhead may have an effect on performance in systems with main storage. Deduplication is used on secondary data for a second reason: secondary data typically contains more duplicate data. Particularly backup applications can produce substantial amounts of duplicate data over time. In some situations when the system architecture doesn't necessitate a lot of overhead or have an influence on speed, data deduplication has been implemented successfully using main storage. The following benefits of the suggested strategy are listed:

- By lowering the user I/O intensity during recovery, it is possible to considerably increase the performance of online RAID reconstruction.
- It is intended to successfully handle the deduplication-related issues while maintaining the desirable benefits of data deduplication's capacity to reduce write traffic.
- It can successfully avoid the read amplification issue brought on by deduplication and reduce write traffic.
- The performance in both reading and writing is greatly improved.

4. Results

By using data deduplication on the I/O channel to eliminate duplicate write requests while also reducing storage space, this research offers a Performance-Oriented Data Deduplication scheme (POD) to enhance the performance of main storage systems in the Cloud. With the help of data deduplication on the I/O channel, duplicate write requests will be eliminated, enhancing the performance of main storage systems in the Cloud while also conserving storage space.

It works best in situations where several copies of data that is extremely close or even identical are kept on a single disc. The outcome demonstrated the high efficiency of this POD that can be seen from following screenshots.

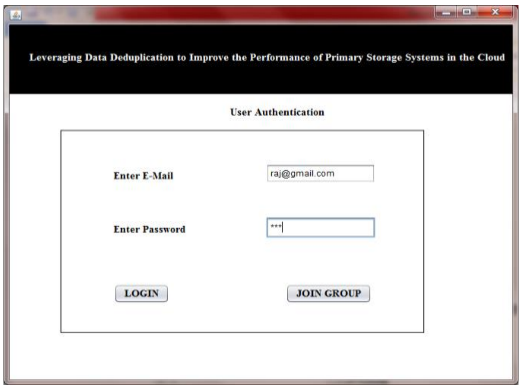


Leveraging Data Deduplication to Improve the Performance of Primary Storage Systems in the Cloud

New User Registration

User Name	raj	
E-Mail id	raj@gmail.com	
Password	***	
D.O.B	4 ▾ 04 ▾ 1992 ▾	Eg:13.02.1993
Gender	[● Male] ○ Female	
Security Code	dUvYJcuAxo	dUvYJcuAxo

Fig 3: User Registration

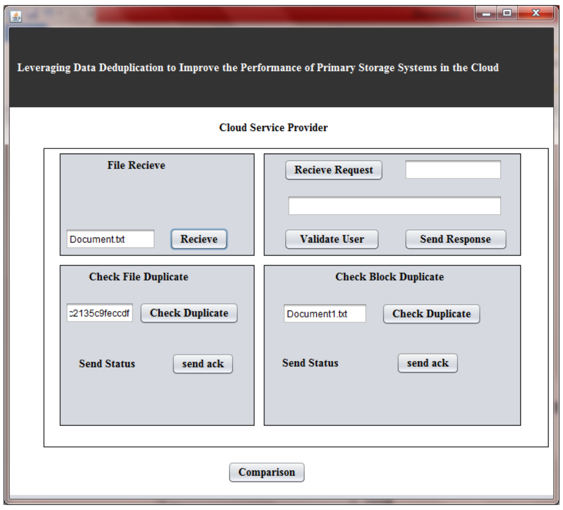


Leveraging Data Deduplication to Improve the Performance of Primary Storage Systems in the Cloud

User Authentication

Enter E-Mail	raj@gmail.com
Enter Password	***

Fig 4: User Authentication

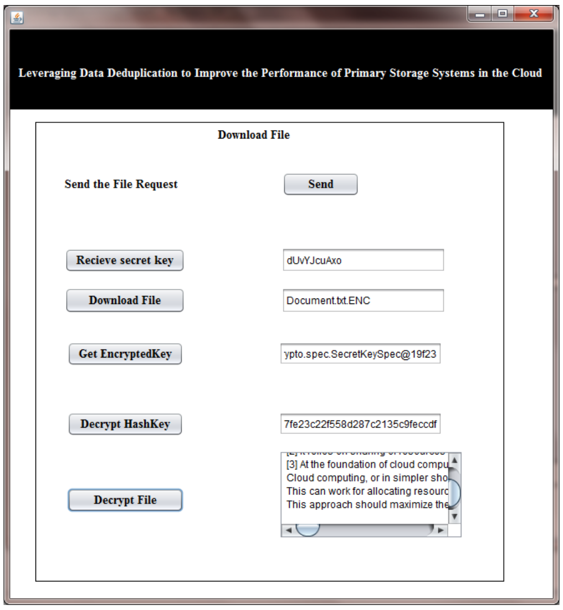


Document.txt

:2135c9feccdf

Document1.txt

Fig 5: Cloud Service Provider



dUvYJcuAxo

Document.txt.ENC

ypto.spec.SecretKeySpec@19f23

7fe23c22f558d287c2135c9feccdf

[3] At the foundation of cloud compu
Cloud computing, or in simpler sho
This can work for allocating resourc
This approach should maximize the

Fig 6: File Download

5. Conclusion

The system's suggestion for this project is a performance-focused I/O deduplication. Performance-Oriented data deduplication strategy (POD) exactly what it sounds like. With the help of data deduplication on the I/O channel, duplicate write requests will be eliminated, enhancing the performance of main storage systems in the Cloud while also conserving storage space. The interplay between read and write requests in deduplication-based main storage systems requires more thorough modelling and analysis. If there is no index-lookup disc barrier, deduplication decreases write traffic and directly enhances write performance. The quantity of storage required for a certain collection of files is decreased using storage-based data deduplication.

It works best in applications where several copies of data that are very close or even identical are kept on a single disc. When it comes to data backups, which are frequently carried out to safeguard against data loss, the majority of the data are consistent from backup to backup. Common backup systems attempt to take advantage of this by recording differences between files or ignoring files that haven't changed. However, neither method completely eliminates redundancies. Large files that have only minor changes, like an email database, are not helped by hard-linking since differences only reveal redundancy in subsequent iterations of a single file (think of a part that was removed and subsequently brought back in or a logo picture used in several publications).The outcome demonstrated how effective this POD is.

Reference

1. “Avoiding File System Micromanagement with Range Writes”, Ashok Anand, Sayandeep Sen, Andrew Krioukov, Florentina Popovici, 2008.
2. “AWOL: An Adaptive Write Optimizations Layer”, Alexandros Batsakis, Randal Burns, 2008.
3. “A Five-Year Study of File-System Metadata”, Nitin Agrawal, William J. Bolosky, John R. Douceur, Jacob R. Lorch., 2007.
4. “Improving Restore Speed for Backup Systems that Use Inline Chunk-Based Deduplication”, Mark Lillibridge,Kave Eshghi ,and Deepavali Bhagwat, 2013.
5. “Six Degrees of Scientific Data: Reading Patterns for Extreme Scale Science IO”, Jay Lofstead, Milo Polte , Garth Gibson , Scott A. Klasky, 2011.
6. “I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance”, Ricardo Koller, Raju Rangaswami., 2010.
7. “The Effectiveness of Deduplication on Virtual Machine Disk Images.”, Keren Jin, Ethan L. Miller, 2009.
8. “Understanding and Improving Computational Science Storage Access through Continuous Characterization”, Philip Carns, Kevin Harms, William Allcock, Charles Bacon, 2011.
9. “stdchk: A Checkpoint Storage System for Desktop Grid Computing.”, Samer Al Kiswany , Matei Ripeanu , Sudharshan S. Vazhkudai , Abdullah Gharaibeh, 2008.
10. “Nitro: A Capacity-Optimized SSD Cache for Primary Storage”, Cheng Li , Philip Shilane, Fred Douglis, Hyong Shim, Stephen Smaldone, and Grant Wallace, 2011.