ISSN: 0975-5665, Nov, 2013 www.ijrct.org

A Survey on Data De-Duplication

#D.Sujatha [1], *M.Tech, Computer Science Engineering*, E mail:diddisujath@gmail.com
#C V Guru Rao [2] *'Associate Professor, Department of CSE* Email:guru_cv_rao@hotmail.com
*#SR Engineering College, Warangal, A.P,**INDIA***

Abstract:

The earliest approach to data reduction was data compression, which searches for repetitive strings of information within a single file. The other approach was single-instance storage, which reduces the amount of storage by recognizing when files are repeated. Data de-duplication is the newest technique for reducing data. It recognizes differences at variable-length block basis within files and between files. Data de-duplication systems store data differently from how it was written. As a result, users are concerned with the integrity of their data. The various methods of de-duplicating data all employ slightly different techniques. However, the integrity of the data will ultimately depend upon the design of the de-duplicating system, and the quality used to implement the algorithms. Data de-duplication process involves removing copies of files and replacing those duplicates with pointers back to the original copy. Removing multiple copies frees up valuable storage space, makes backups smaller and faster, and reduces network traffic for over-the-network backups.

1. Introduction:

In computing, data de-duplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat synonymous terms are intelligent (data) compression and single-instance (data) storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the de-duplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced.[1]

This type of de-duplication is different from that performed by standard file-compression tools, such as LZ77 and LZ78. Whereas these tools identify short repeated substrings inside individual files, the intent of storage-based data de-duplication is to inspect large volumes of data and identify large sections – such as entire files or large sections of files – that are identical, in order to store only one copy of it. This copy may be additionally compressed by single-file compression techniques. For example a typical email system might contain 100 instances of the same one megabyte (MB) file attachment.

Each time the email platform is backed up, all 100 instances of the attachment are saved, requiring 100 MB storage space. With data de-duplication, only one instance of the attachment is actually stored; the subsequent instances are referenced back to the saved copy for de-duplication ratio of roughly 100 to 1.

2. Overview of De-Duplication

De-duplication may occur "in-line", as data is flowing, or "post-process" after it has been written.

2.1Post-process de-duplication

With post-process de-duplication, new data is first stored on the storage device and then a process at a later time will analyze the data looking for duplication. The benefit is that there is no need to wait for the hash calculations and lookup to be completed before storing the data thereby ensuring that store performance is not degraded. Implementations offering policy-based operation can give users the ability to defer optimization on "active" files, or to process files based on type and location. One potential drawback is that you may unnecessarily store duplicate data for a short time which is an issue if the storage system is near full capacity.

2.2In-line de-duplication

This is the process where the de-duplication hash calculations are created on the target device as the data enters the device in real time. If the device spots a block that it already stored on the system it does not store the new block, just references to the existing block. The benefit of in-line de-duplication over post-process de-duplication is that it requires less storage as data is not duplicated. On the negative side, it is frequently argued that because hash calculations and lookups takes so long, it can mean that the data ingestion can be slower thereby reducing the backup throughput of the device. However, certain vendors with in-line de-duplication have demonstrated equipment with similar performance to their post-process de-duplication counterparts.

2.3Source versus target de-duplication

Another way to think about data de-duplication is by where it occurs. When the de-duplication occurs close to where data is created, it is often referred to as "source de-duplication," whereas when it occurs near where the data is stored; it is commonly called "target de-duplication."

Source de-duplication ensures that data on the data source is de-duplicated. This generally takes place directly within a file system.[4][5] The file system will periodically scan new files creating hashes and compare them to hashes of existing files. When files with same hashes are found then the file copy is removed and the new file points to the old file. Unlike hard links however, duplicated files are considered to be separate entities and if one of the duplicated files is later modified, then using a system called Copy-on-write a copy of that file or changed block is created. The de-duplication process is transparent to the users and backup applications. Backing up a de-duplicated file system will often cause

duplication to occur resulting in the backups being bigger than the source data.

Target de-duplication is the process of removing duplicates of data in the secondary store. Generally this will be a backup store such as a data repository or a virtual tape library.

3. De-duplication methods

One of the most common forms of data de-duplication implementations works by comparing chunks of data to detect duplicates. For that to happen, each chunk of data is assigned an identification, calculated by the software, typically using cryptographic hash functions. In many implementations, the assumption is made that if the identification is identical, the data is identical, even though this cannot be true in all cases due to the pigeonhole principle; other implementations do not assume that two blocks of data with the same identifier are identical, but actually verify that data with the same identification is identical.[6] If the software either assumes that a given identification already exists in the de-duplication namespace or actually verifies the identity of the two blocks of data, depending on the implementation, then it will replace that duplicate chunk with a link.

Once the data has been de-duplicated, upon read back of the file, wherever a link is found, the system simply replaces that link with the referenced data chunk. The de-duplication process is intended to be transparent to end users and applications.

3.1 Chunking: Between commercial de-duplication implementations, technology varies primarily in chunking method and in architecture. In some systems, chunks are defined by physical layer constraints (e.g. 4KB block size in WAFL). In some systems only complete files are compared, which is called Single Instance Storage or SIS. The most intelligent (but CPU intensive) method to chunking is generally considered to be sliding-block. In sliding block, a window is passed along the file stream to seek out more naturally occurring internal file boundaries.

3.2Client backup de-duplication: This is the process where the de-duplication hash calculations are initially created on the source (client) machines. Files that have identical hashes to files already in the target device are not sent, the target device just creates appropriate internal links to reference the duplicated data. The benefit of this is that it avoids data being unnecessarily sent across the network thereby reducing traffic load.

Primary storage and secondary storage: By definition, primary storage systems are designed for optimal performance, rather than lowest possible cost. The design criteria for these systems is to increase performance, at the expense of other considerations. Moreover, primary storage systems are much less tolerant of any operation that can negatively impact performance. Also by definition, secondary storage systems contain primarily duplicate, or secondary copies of data. These copies of data are typically not used for actual production operations and as a result are more tolerant of some performance degradation, in exchange for increased efficiency.

Presently, data de-duplication has predominantly been used with secondary

storage systems. The reasons for this are two-fold. First, data de-duplication requires overhead to discover and remove the duplicate data. In primary storage systems, this overhead may impact performance. The second reason why de-duplication is applied to secondary data, is that secondary data tends to have more duplicate data. Backup application in particular commonly generate significant portions of duplicate data over time.

3.3Advantages

Storage-based data de-duplication reduces the amount of storage needed for a given set of files. It is most effective in applications where many copies of very similar or even identical data are stored on a single disk—a surprisingly common scenario. In the case of data backups, which routinely are performed to protect against data loss, most data in a given backup isn't changed from the previous backup. Common backup systems try to exploit this by omitting (or hard linking) files that haven't changed or storing differences between files. Neither approach captures all redundancies, however. Hard linking does not help with large files that have only changed in small ways, such as an email database; differences only find redundancies in adjacent versions of a single file (consider a section that was deleted and later added in again, or a logo image included in many documents).

Network data de-duplication is used to reduce the number of bytes that must be transferred between endpoints, which can reduce the amount of bandwidth required.

Virtual servers benefit from de-duplication because it allows nominally separate system files for each virtual server to be coalesced into a single storage space. At the same time, if a given server customizes a file, de-duplication will not change the files on the other servers—something that alternatives like hard links or shared disks do not offer. Backing up or making duplicate copies of virtual environments is similarly improved.

4.Conclusion

Data de-duplication will recognizes the differences at variable-length block basis within files and between files. Data de-duplication systems store data differently from how it was written. As a result, users are concerned with the integrity of their data. However, the integrity of the data will ultimately depend upon the design of the de-duplicating system, and the quality used to implement the algorithms. Data de-duplication process which involves removing copies of files and replacing those duplicates with pointers back to the original copy. In this removing multiple copies frees up valuable storage space, makes backups smaller and faster, and reduces network traffic for over-the-network backups.

5.References:

1. Bitton, Dina. 1983. “Duplicate record elimination in large data files.” *ACM Transactions on database systems (* 8:255-265. http://portal.acm.org/citation.cfm?id=319987&dl= (Accessed November 2, 2010).

2. Chatterjee, A, and Arie Segev. 1991. "Data manipulation in heterogeneous databases." *ACM SIGMOD Record* 20:64-68. http://portal.acm.org/citation.cfm?id=141356.141385 (Accessed September 8, 2010).

3. Chou, C, T Du, and V Lai. 2007. "Continuous auditing with a multi-agent system." *Decision Support Systems* 42:2274-2292. http://linkinghub.elsevier.com/retrieve/pii/S0167923606001175 (Accessed August 19, 2010).

4. Elmagarmid, Ahmed, Panagiotis Ipeirotis, and Vassilios Verykios. 2007. "Duplicate Record Detection: A Survey." *IEEE Transactions on Knowledge and Data Engineering* 19:1-16. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4016511.

5. Fellegi, Ip. 1969. "A theory for record linkage." *Journal of the American Statistical Association* 64:1183-1210. http://www.jstor.org/stable/2286061 (Accessed November 2, 2010).

6. H. 2008. "Industry-scale duplicate detection." *Proceedings of the* 1:1253-1264. http://portal.acm.org/citation.cfm?id=1454159.1454165 (Accessed August 21, 2010).

7. Hernandez, Ma, and Sj Stolfo. 1995. "The merge/purge problem for large databases." *Proceedings of the 1995 ACM SIGMOD*. http://portal.acm.org/citation.cfm?id=223807&dl=GUIDE, (Accessed September 10, 2010).

8. Inspector General. 1997. "Audit report duplicate payments." http://www.va.gov/oig/52/reports/1997/7AF-G01-035--duppay.pdf.

9. Kimball, R. 2009. "The data warehouse toolkit." http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:the+data+warehouse+etl+toolkit#0 (Accessed September 8, 2010).

10. Levenshtein, Vi. 1966. "Binary codes capable of correcting deletions, insertions, and reversals." *Soviet Physics Doklady*. http://www.mendeley.com/research/binary-codes-capable-of-correcting-insertions-and-reversals/ (Accessed November 2, 2010).

11. McMullan, Michael. 2001. "Duplicate Medicare Payments by Individual Carriers." *Public Law*.

12. Newcombe, Hb. 1988. "Handbook of record linkage: methods for health and statistical studies, administration, and business." http://portal.acm.org/citation.cfm?id=63465 (Accessed September 8, 2010).

