NETAPP WHITE PAPER

Looking Beyond the Hype: Evaluating Data Deduplication Solutions

Larry Freeman, Network Appliance, Inc.
September 2007 | WP-7028-0907

Table of Contents

The Deduplication Hype

Deduplication is without a doubt one of this year's hottest topics in data storage. The rationale behind deduplication is simple: Eliminate your duplicate data and reduce the capacity needed during backups and other data copy activities. Unfortunately, the many different deduplication approaches from various vendors, with much hype about their unique benefits, can leave users bewildered. As they consider the variety of deduplication offerings, they often fail to understand the basic design nuances that are important to them.

This paper looks beyond the hype and focuses on the important design aspects of deduplication, giving evaluators the information they need to make informed decisions when examining deduplication solutions.

What Is Deduplication?

Deduplication is the process of "unduplicating" data. The term *deduplication* was coined by database administrators many years ago as a way of describing the process of removing duplicate database records after two databases have been merged.

In the context of disk storage, deduplication refers to any algorithm that searches for duplicate data objects, such as blocks, chunks, or files, and discards these duplicates. When a duplicate object is detected, its reference pointers are modified so that the object can still be located and retrieved, but it "shares" its physical location with other identical objects. This data sharing is the foundation of all types of data deduplication.

How Does Deduplication Work?

Regardless of operating system, application, or file system type, *all* data objects are written to a storage system using a data reference pointer, without which the data could not be referenced or retrieved. In traditional (non-deduplicated) file systems, data objects are stored without regard to any similarity with other objects in the same file system. In Figure 1, five identical objects are stored in a file system, each with a separate data pointer. Although all five data objects are identical, each is stored as a separate instance and each consumes physical disk space.

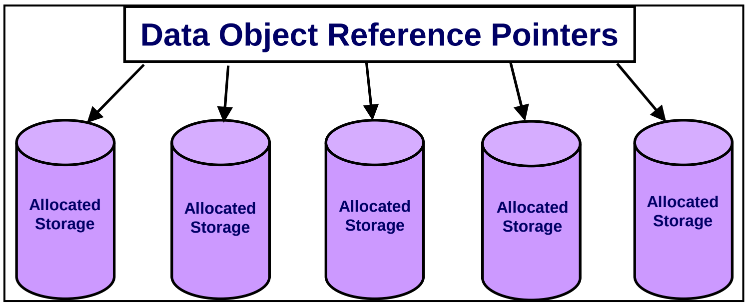

Figure 1) Non-deduplicated data

In a deduplicated file system, two new and important concepts are introduced:

- A catalog of all data objects is maintained. This catalog contains a record of all data objects using a "hash" that identifies the unique contents of each object. Hashing is discussed in detail in "Deduplication Design Considerations," later in this paper.

- The file system is capable of allowing many data pointers to reference the same physical data object.

Cataloging data objects, comparing the objects, and redirecting reference pointers forms the basis of the deduplication algorithm. As shown in Figure 2, referencing several identical objects with a single master object allows the space that is normally occupied by the duplicate objects to be given back to the storage system.

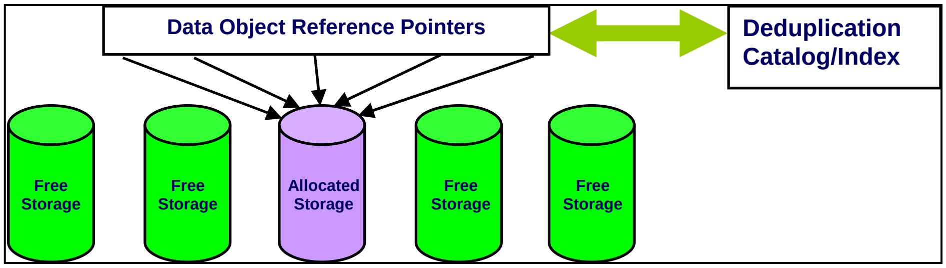

Figure 2) Deduplicating identical objects creates free space.

Deduplication Design Considerations

Given the fact that all deduplication vendors must maintain some form of catalog and must support some form of block referencing, there is a surprising variety of implementations (and they all have subtle differences that allow them all to be patented). The following sections explain the methods that vendors use when designing deduplication.

Hashing

Data deduplication begins with a comparison of two data objects. It would be impractical (and very arduous) to scan an entire data volume for duplicate objects each time a new object is written to that volume. For that reason, deduplication vendors create small *hash values* for each new object, and store these values in a catalog.

A hash value, also called a *digital fingerprint* or *digital signature*, is a small number that is generated from a longer string of data. A hash value is substantially smaller than the data object itself, and is generated by a mathematical formula in such a way that it is unlikely (although not impossible) for two nonidentical data objects to produce the same hash value.

A hash value can be as simple as a parity calculation or as elaborate as a SHA-1 or MD-5 encryption hash. In any case, once the hash values have been created, they can be easily compared and deduplication candidates can be identified.

If matching hash values are discovered, there are two approaches to handling these "candidates." First, you can assume that identical hash values always indicate that the data objects are identical, and move straight to the deduplication phase. Or, as an alternative, you can add a secondary operation to scan each data object to validate that the data objects are indeed identical before performing the deduplication.

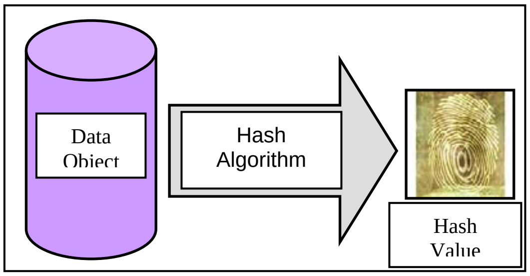

Figure 3) A hash value is a digital fingerprint that represents a much larger object.

Beyond the Hype: What You Should Know about Hashes

Understanding a vendor's hashing algorithm is a major criterion when evaluating deduplication. If the vendor depends solely on the hash match to decide whether two objects are to be deduplicated, then you are accepting the possibility that false positive matches could occur, with data corruption as the likely result.

If any possibility of data corruption, however small, is not acceptable, you should make sure that the vendor has taken steps to perform secondary data object validation after the hash compare is complete. Alternatively, if you are willing to accept that false positive hash matches may occur, and speed of deduplication is a greater priority, then a "trusted hash" design may be adequate for your needs.

Indexing

When the duplicate objects have been identified (and optionally validated), it's time to deduplicate. As you might expect, different vendors employ varying methods when modifying the data pointer structure. However, all forms of data pointer indexing fall into two broad categories:

- **Hash catalog:** A catalog of hash values is used to identify candidates for deduplication. A system process identifies duplicates, and data pointers are modified accordingly. The advantage of catalog deduplication is that the catalog is used only to identify duplicate objects; it is not accessed during the actual reading or writing of the data objects. That task is still handled by the normal file system data structure.

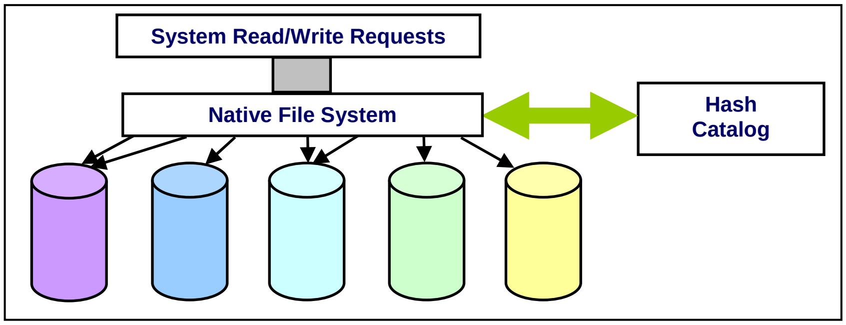

Figure 4) Catalog indexing: The file system controls block sharing of duplicate blocks.

- **Lookup table:** A lookup table extends the functionality of the hash catalog to also contain a hash lookup table in order to index the deduplicated object's parent data pointer. The advantage of a lookup table is that it can be used on file systems that do not support multiple block referencing; a single data object can be stored and referenced many times by using the lookup table.

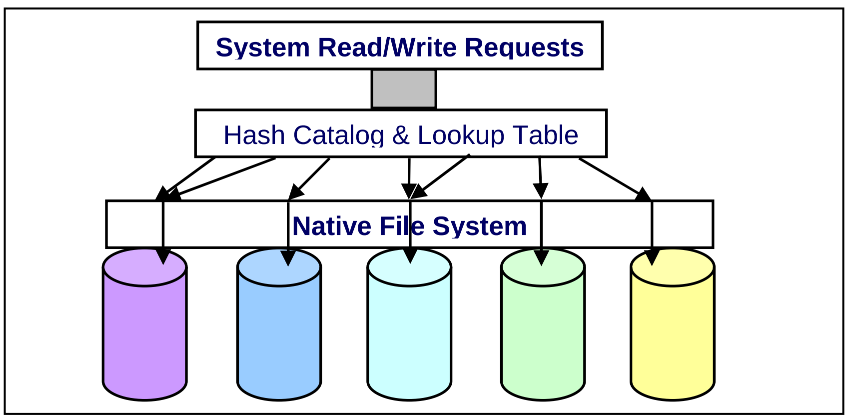

Figure 5) Lookup table indexing: Block sharing is managed and controlled by the lookup table.

Beyond the Hype: What You Should Know about Indexing

A seemingly subtle design difference, deduplication indexing is actually an important consideration in vendor evaluation, particularly when it comes to resiliency. When you are using lookup tables to index data objects, the table itself becomes a single point of failure. Any corruption of the table is likely to render the entire file system unusable. Catalog-based deduplication, on the other hand, is used only for discovery and has no dependency for the actual reading and writing of objects. Catalog deduplication, however, requires that the native file system support multiple block referencing.

If resiliency is paramount, catalog-based deduplication should be your first choice. If breadth of product support, including remote systems, is more important, then lookup deduplication may be your only alternative.

Inline or Postprocessing

Another vendor design distinction is *when* to perform deduplication. Again, there are two options:

- **Inline deduplication:** Deduplication is performed *as the data is written to the storage system.* With inline deduplication, the entire hash catalog is usually placed into system memory to facilitate fast object comparisons. The advantage of inline deduplication is that it does not require the duplicate data to actually be written to disk. The duplicate object is hashed, compared, and re-referenced on the fly. The disadvantage of this method is that significant system resources are required to handle the entire deduplication operation in milliseconds. Duplicate object validation beyond a quick hash compare is generally not feasible when done inline; therefore these systems usually rely on "trusted" hash compares without validating that the objects are indeed identical.

- **Postprocessing deduplication:** Deduplication is performed *after the data is written to the storage system.* With postprocessing, deduplication can be performed at a more leisurely pace, and it typically does not require heavy utilization of system resources. The disadvantage of postprocessing is that all duplicate data must first be written to the storage system, requiring additional, although temporary, physical space on the system.

Beyond the Hype: What You Should Know about Inline and Postprocessing

The decision regarding inline versus postprocessing deduplication has more to do with your application than with any technical advantages or disadvantages.

When performing data backups, the user's objective is completion of backups within an allowed time window. When adding deduplication to the backups, the user's objective is to free up the redundant storage space required for these backups.

These two objectives should not compete—the additional time required for deduplication should not drive backups beyond their allotted time window. An assessment should be made to determine if the time penalty of deduplication is offset by the space savings realized after deduplication, regardless of whether the deduplication is performed inline or postprocessing.

Other applications, such as primary storage and data archiving, do not lend themselves well to inline deduplication. In the case of primary storage, systems rarely have the performance headroom to facilitate inline deduplication. In the case of archival data, users may simply want to "sweep" their file systems of duplicate data during periods of low activity, similar to other occasional storage housekeeping chores. In these environments, postprocessing deduplication is the preferred method.

The bottom line is that you should evaluate your application environment and the "cost" of deduplication. If your priority is high-speed data backups with optimal space conservation, choose inline deduplication. If you are interested in deduplicating primary storage or archival data, postprocessing deduplication is probably your best bet.

Note: Some vendors have recently designed their products with the ability to perform either inline *or* postprocessing deduplication. If you are evaluating a product that makes this claim, make sure that the deduplication mode is user selectable, so that you can set the mode based on your application's characteristics and the storage system workload.

Source or Destination Deduplication

The final design criterion in choosing a deduplication vendor is *where* the deduplication occurs. Once again you have two choices: at the *source* or at the *destination*.

- Source deduplication refers to the comparison of data objects at the source, *before they are sent to a destination* (usually a data backup destination). The advantage of source deduplication is that less data is required to be transmitted and stored at the destination point. The disadvantage is that the deduplication catalog and indexing components are dispersed over the network so that deduplication potentially becomes more difficult to administer.

- Destination deduplication refers to the comparison of data objects *after they arrive at the destination point.* The advantage of destination deduplication is that all the deduplication management components are centralized. The disadvantage is that the entire data object must be transmitted over the network before deduplicating.

Beyond the Hype: What You Should Know about Source and Destination

The decision to select source or destination deduplication is determined by your objectives. If your main objective is to reduce the amount of network traffic when copying files, source deduplication is the only option. If your goal is to simplify the management of deduplication and the amount of storage required at the destination, destination deduplication is the preferred choice.

Deduplication Space Savings

Now that you have carefully evaluated all aspects of deduplication design and made your selection—what should you expect for space savings?

Deduplication vendors often claim that their products offer 20:1, 50:1, or even greater data reduction ratios. These claims actually refer to the “time-based” space savings effect of deduplication on repetitive data backups. Figure 6 shows one vendor’s theoretical space savings calculation over time. The space savings shown in this figure depend on repeating backups of highly redundant data. Because the backups contain mostly unchanged data, once the first full backup has been stored, all subsequent full backups see a very high occurrence of deduplication.

But what if you don’t retain 64 backup copies? What if your backups have a higher change rate than the example in Figure 6? Realizing that space savings numbers from a vendor’s marketing department often don’t represent a real-life environment, what should you expect for space savings on your backup data sets?

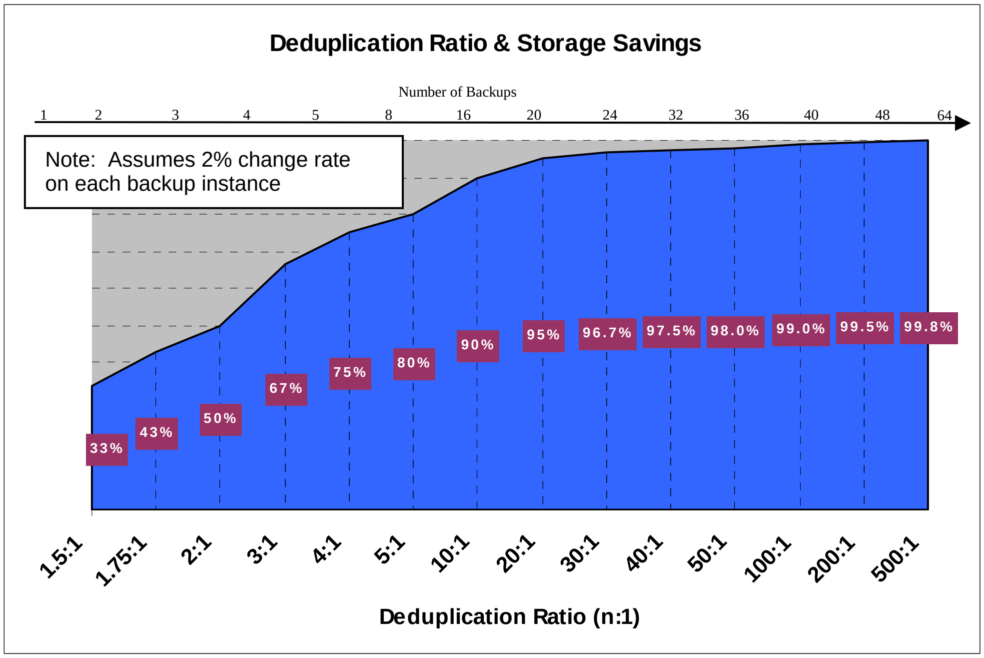

Figure 6) Deduplication ratios increase as backups accumulate over time.

Another area to consider is your nonbackup data volumes, such as primary storage and archival data, where the rules of time-based data reduction ratios do not apply. In those environments, volumes do not receive a steady supply of redundant data backups, but they may still contain a large number of duplicate data objects. The ability to reduce space in these volumes through deduplication is measured in spatial terms. In other words, if a 500GB data archival volume can be reduced to 400GB through deduplication, the spatial (volume) reduction is 100GB, or 20%.

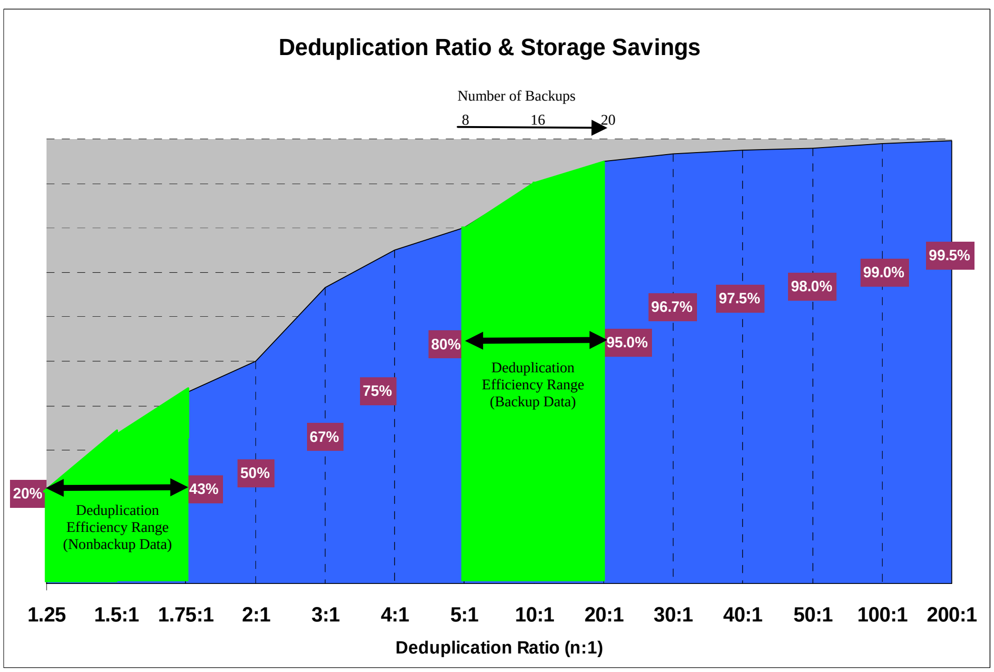

Figure 7) Users should look for opportunities to reduce both backup and nonbackup data.

Beyond the Hype: What You Should Know about Space Savings

When evaluating deduplication space savings, you should have two goals.

First, examine your backup data. As shown in Figure 7, it is reasonable to expect 5:1 to 20:1 space savings (over time) with a backup change rate of 2%. If you store more than 20 backup copies on disk, or if your change rate is less than 2%, your deduplication ratio will increase. Conversely, if you retain a lower number of backup copies, or if your data change rate is higher than 2%, your deduplication space savings ratio will be reduced.

Next, examine your nonbackup data. Are there any opportunities to eliminate duplicate data on those volumes? Generally speaking, if you can reduce this data by 1.25 to 1.75:1, deduplication would be economically feasible. Think of it as receiving a “storage rebate” by reducing the storage capacity of these volumes by 20% to 40%. Enterprise storage systems today can easily exceed $10,000 per TB. Saving just a few TBs across your enterprise could justify the implementation of volume-based deduplication.

Summary

Data deduplication is an important new technology that is quickly being embraced by users as they struggle to control data proliferation. By eliminating redundant data objects, an immediate benefit is obtained through space efficiencies. When choosing a deduplication product, however, it is important to consider all aspects of design, including hashing, indexing, inline or postprocessing, source or destination, and of course space savings efficiency.

Many vendors currently offer deduplication, with more sure to follow, all with various approaches and techniques. It is clear that data deduplication will someday be a requirement for every storage vendor, much as snapshots became a requirement years ago.

Well-designed deduplication must perform without compromising data integrity and reliability. Deduplication magnifies the effect of data corruption. If a deduplicated data object becomes corrupt, it has far-reaching implications, because it is referenced by many other files and applications. Vendors will be required to provide 100% assurance that their design will prevent any such data inconsistencies

Deduplication must operate seamlessly in existing user environments. Users will not build a storage infrastructure around deduplication; rather, deduplication must fit into their existing environment with minimal disruption. Ultimately, deduplication must be a transparent background process.

Finally, deduplication will be required to have minimal impact on system performance. Users will not implement deduplication if it has a negative impact on their system workloads. This is particularly true as deduplication makes its way from backup applications to more performance-sensitive primary storage environments.

NetApp, a leader in data storage efficiency since 1992, has established A-SIS deduplication as the first deduplication product to be used broadly across many applications, including data backup, data archival and primary data. A-SIS deduplication combines the benefits of granularity, performance, and resiliency to provide users with significant data deduplication benefits.