



WAN Deduplication - Doing More With Less

The Benefits of Byte-Level WAN Deduplication

WAN deduplication (also known as disk-based data reduction) is a critical component of state-of-the-art WAN acceleration.

By eliminating the transfer of repetitive data across the WAN, deduplication saves bandwidth and increases application throughput (from 2x to 100x under the right circumstances.)

WAN acceleration appliances that utilize deduplication rely on storing commonly used patterns on the appliance's disk drives, which are later referenced instead of transmitting repetitive data across the WAN. In a deduplication scenario, the first time data is transferred across the WAN (called a "first pass") data patterns are stored in both appliances on either side of the WAN link. Subsequent transfers of the same or similar data (called "second passes") take advantage of the fact that both appliances have already "seen" and stored the data. During these passes, the originating appliance refers to the data in the remote appliance's data store, enabling the remote appliance to deliver the data to the end-station locally.

There are two ways of performing deduplication - "token-based" and "instruction-based". This paper discusses how the two methods work, the differences between the two, and why each performs differently in "dynamic data" scenarios, such as AutoCAD, Microsoft Excel, and video streaming.

Token-based Deduplication

Token-based schemes for deduplication rely on reference tokens (or "labels") to represent chunks (or "shingles") of original data. The token is typically a hash of the data, and can reference fixed or variable length byte patterns that begin and end on byte boundaries that are independent of higher level structures like file or block boundaries. Because of this, some people claim that token-based schemes have byte level granularity, and call these byte-level deduplication or byte-caching. However, this is inaccurate. As we will see shortly, there are big differences between what can be achieved with a token-based scheme and an instruction-based scheme with respect to byte-level deduplication.



TOKEN BASED SCHEMES

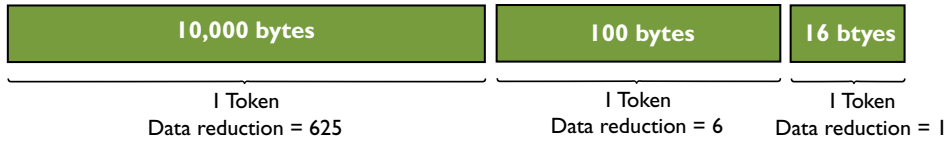


Figure 1: In a token based scheme, a token is used to represent a chunk of data.

To illustrate how token-based schemes are utilized for deduplication, let's assume that the tokens are 16-bytes and that the chunks of original data are variable in length.

(The token must be large enough to minimize the probability of hash-collisions, where different chunks of data result in the same hash. For this reason the token is usually 16 bytes (128 bits) or more.

When the chunks are large, token-based deduplication is very efficient. For example, representing 10,000 bytes of data with a 16 byte token yields data reduction factors up to 625 (10,000 / 16). If the chunks are smaller, data reduction factors will decrease. A 100 byte chunk would yield a 6.25x improvement and a 16 byte chunk would yield no improvement. Representing anything smaller than 16 bytes with a token would actually expand rather than reduce the amount of information transferred. Some token-based schemes use a hierarchical arrangement of tokens in order to get chunk-sizes that are as large as possible.

Token-based schemes work well when the data is relatively static, for example when the same file is transferred back-and-forth between the same two locations. These schemes also work well when changes to the

data are isolated to small contiguous regions of the file (for example, all changes are appended to the end of the file). In the first instance, the same tokens that represented the original data exactly represent the second-pass data. In the second instance, the same tokens that represented the original data represent the unchanged parts of the second-pass data and new tokens can be created for modified parts of the file.

Token-based schemes, however, face a much more difficult challenge when changes are dispersed throughout the file and the changes themselves are small. We'll refer to this class of data as "dynamic".

The Challenges of Deduping Dynamic Data

Dynamic data is increasingly prevalent in many Enterprise applications, including Microsoft Excel and AutoCAD 2007/2008. As described above, in these types of environments changes to the data set are intermixed throughout the data itself. In some instances, merely opening and saving a file can cause bytes within the file to change.

Figure 2 contains a screenshot of a binary-level diff executed on two Microsoft Excel spreadsheets. Test1.xls shows part of the original file content; test2.xls is the same section of file content after it was opened in Microsoft excel and then saved from Microsoft excel (with no changes made). The white font represents bytes that are the same from one file to the next. The red font represents bytes that changed.

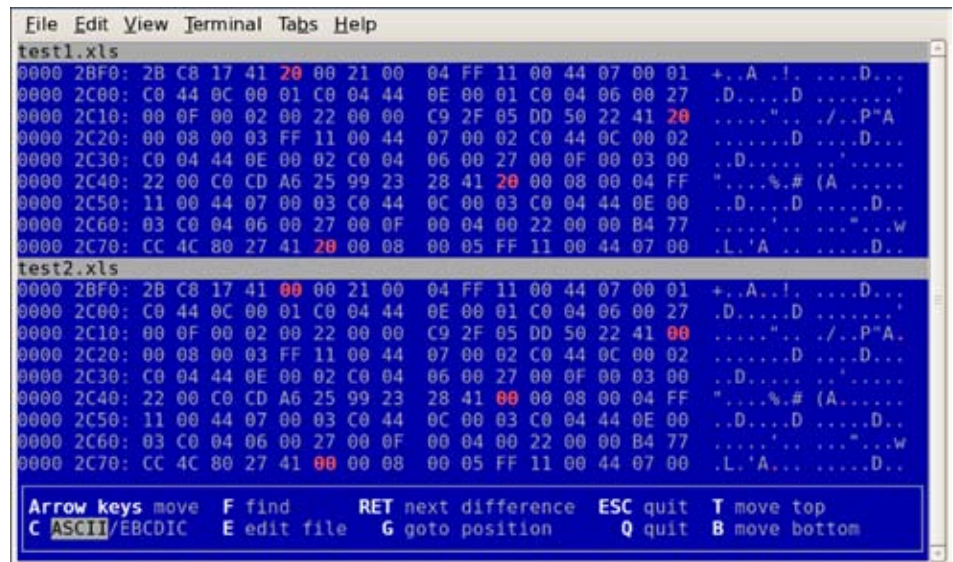


Figure 2: Opening and saving an Excel file causes byte level changes within the file shown (in red)



The best level of granularity is achieved when every byte of data is examined in real-time.

REAL PERFORMANCE IMPROVEMENTS

| File Format | | Save (seconds) | Improvement (%) |
|-----------------|-------------------------------|----------------|-----------------|
| 2007 DWG format | Baseline | 186 | |
| | With byte-level deduplication | 29.8 | 84% |
| 2004 DWG format | Baseline | 117 | |
| | With byte-level deduplication | 40.2 | 66% |

Figure 5 shows how these differences translate to real performance improvements when saving a 14 MB CAD image across the WAN:

In the Excel example provided in Figure 4, Silver Peak's deduplication algorithm would reference the second-pass of the Excel file using the following instructions: "retrieve 20 bytes from location a to location b with 1 delta "XX". Network Memory would match the 19 out of 20 unchanged bytes and only send the single changed byte "XX" as first-pass data. More importantly, Silver Peak will do this repeatedly throughout the entire file, ensuring that all repetitive data is found. This is in stark contrast to the token based scheme, which will find little or no matches in both the Excel and AutoCAD environments.

Conclusion

While all deduplication technologies examine bytes of data, token-based schemes lack true byte level granularity when doing pattern matches. The best level of granularity is only achieved when every byte of data is examined in real-time and referenced using specific start/stop

instructions with the ability to detect and encode small changes to previous data, as is the case with Silver Peak's Network Memory technology.

The difference in the Silver Peak approach becomes quite apparent when attempting to dedupe applications with dynamic data formats, such as AutoCAD 2007 or later, Microsoft Excel, video streaming, and dynamic HTML. In these environments, Silver Peak delivers consistent and reliable performance, even when noticeable changes occur at the binary level. Similarly, Silver peak's differentiated architecture is apparent when supporting latency sensitive applications, such Citrix, RDP, SRDF/A, and VoIP.

Silver Peak offers the only WAN acceleration architecture designed to work on all enterprise applications, regardless of transport protocol, latency requirements, or data format. By delivering significant performance gains across all traffic types, Silver Peak provides the best return on investment for WAN acceleration. ■