Portable Data Deduplication

Chris Keiser

Abstract. The desire of deduplication is like that of compression for reducing data down to its smallest lossless form. The efficiency of data compression has evolved through the decades bound by a theoretical maximum entropy rate. In this paper we will explore the effects of deduplication as part of a compression strategy for the distribution of large datasets using a Stack Overflow public extract in the form of a Microsoft SQL database as a working example.

1. Introduction

Data deduplication [1] allows for the storage of unique instances of segmented data within a dataset. Once the redundant copies of data are removed through deduplication you have the choice to further compress the result to save even more space. Starting with Windows Server 2012 R2, Microsoft has given us a new server role called Data Deduplication. There are other deduplication engines available but for the context of this paper this is the one we will be using.

Lossless compression [2] involves no loss of information. If data has been compressed using a lossless algorithm, the original data can be recovered exactly from the compressed data. In a 1948 paper called "A Mathematical Theory of Communication" by Claude E. Shannon [3] suggests that there is a fundamental limit to lossless data compression. This limit, called the entropy rate and the precision of its value depends on the statistical nature of the information source.

We will be using the Stack Overflow dataset which is publicly offered from the Stack Exchange network [4] through an anonymized dump of all user-contributed content on the network. The data is available to the public through a cc-by-sa 4.0 license.

The availability of the dataset was brought to our attention after taking interest in the content of a cool guy named Brent Ozar [5] whom is a SQL server guru. He has taken the time to import the Stack Overflow dataset into SQL server then offer the dataset as SQL databases packaged and delivered in a well-chosen compressed 7z archive for other researchers to easily consume. Although we do have an interest in the contents of the dataset, for our purpose here we are more interested in packaging and delivery.

2. Setup

The hardware used to perform the tasks was an Intel Xeon twelve core processor running VMWare ESXi. Storage was running over NFS (Network File System) to a network attached storage device.

The virtual machine used for processing was given six CPU (Central Processing Units) cores and sixty-four gigabytes of memory for processing and running Windows Server 2019. When running Windows Server as a virtual machine the Hardware Virtualization setting needs to be enabled in the virtual machine options to allow for the Hyper-V role to be installed. In addition to the Hyper-V role (for the tools), the Data Deduplication role also needs to be enabled.

To streamline the processing of building our deduped virtual disk containers we used a PowerShell script to provision and prepare the drive as well as ease the deduplication process [6].

Compression was done using a 7-Zip [7] archive with the LZMA2 compression method enabled.

3. Execution

All input payloads are represented as four separate one hundred gigabyte files totaling four hundred gigabytes for this database. The final pass does also include a small added amount of information relating to the SQL log and supporting archive info for the sake of completeness.

- First pass was a 100 GB input which reduced to about 32 GB
- Second pass was a [100 GB + 32 GB] 132 GB input which reduced to about 64 GB
- Third pass was a [100 GB + 64 GB] 164 GB input which reduced to about 96 GB
- Final pass was a [100 GB + 96 GB] 196 GB input which reduced to about 121 GB

Each pass took about twenty minutes to extract from its 7z archive into the mounted virtual disk container. An added thirty minutes per pass was needed to execute the deduplication process for compaction. All in it took about four hours to transform the four hundred gigabyte dataset from its 7z archive into a one hundred and twenty-one gigabyte deduplicated virtual disk container.

Four passes were needed because the virtual disk container was constrained to a max of two hundred gigabytes. A single large container with all archives extracted in a single pass then the deduplication process in another single pass should have similar overall results. The final deduplication savings are reported as two hundred and eighty-four gigabytes or about seventy percent.

Subsequently once the deduplication process was complete, we tested further compressing the final virtual disk container into a 7z archive. Regardless of the 7-zip settings used the compressed result comes out at about hundred and four gigabytes or a seventeen gigabyte drop.

4. Results

Deduplication of this dataset resulted in a seventy percent decrease of storage requirements over the uncompressed version. The usefulness of this result can be considered in two separate ways depending on if the data is in motion or if the data is at rest.

From a data in motion point of view, our best compressed output [104 GB] was twice as large as Brent's [54 GB] result by compressing the databases directly without deduplication.

From a data at rest point of view, mounting the extracted and deduped container [121 GB] to a SQL server consumes about one third less of the total SQL database footprint on disk [400 GB] where no deduplication is used.

Since the dedupe, engine keeps its metadata within the container, it is portable between different Windows instances that have support for it. Content consumers do need to have the deduplication engine enabled to use the data.

5. Conclusion

The benefit of using deduplication on this dataset is more in favor of disk space savings for data at rest. Quick mounting of the duplicated dataset could be worth the disk space savings for data that is infrequently accessed. We are currently unsure of its performance on a high-volume setup though this setup should be sufficient for low volume queries.

For data in motion, Brent's distribution method still makes more sense for the sake of bandwidth conservation. If the consumer wants added disk savings from deduplication, they can enable deduplication on the volume storing the databases to achieve the benefit after delivery.

Compressing the deduplicated container did yield some added savings, but it may not be worth the time investment. It could be worth it maybe if you are putting the archive in cold storage but even then, you lose the ability to quickly mount without first extracting the archive which requires even more free space.

References

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

[2] https://en.wikipedia.org/wiki/Lossless_compression

[3] https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

[4] https://archive.org/details/stackexchange

[5] https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/

[6] https://docs.microsoft.com/en-us/powershell/module/deduplication/?view=windowsserver2022-ps

[7] https://7-zip.org/