International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930

DeDuplication Enabled USB drives

Afshan Butt – Bhilai Institute of Technology Raipur

Abstract

USB is being widely adopted as one the most used storage device. USB flash drives are small portable data storage devices that many users have been using for a long time now. The ability to store large amounts of data combined with the little space they occupy physically makes the device more desirable. Also, not to mention their fast data transfer rates adds even more to its popularity and usability among the other available storage devices. Often a user copies all the important data (or back-up) data to the USB drive consisting of various folders and it so happens that the same files (files with exact same data/duplicate data) are present in the different folders which user is unaware during the copy operation , which causes the wastage of storage space of the drive. According to a survey a fully filled USB drive consists of around 25~30% of duplicate data in across folders present in the drive , which a user is unaware and when drive space is fully occupied , the user either tries to delete some data to occupy more data , spends time in finding all the duplicate data on drive or buys the new drive. Data deduplication in USB drives can be a very effective solution to solve this issue. Data deduplication helps to overcome three issues on the flash-based USB drives; price per GB of storage space, write limit or disk endurance and the other is garbage collection overhead. In this paper we will propose a solution that will enable the data deduplication in the USB drives while the data is being copied to drive itself, so that the user doesn't have to spend time in cleansing the drive ones its out of space.

Introduction

USB drives are popularly being used into modern computer system storage device, getting spotlight as next generation storage media due to a high performance and small size. Data deduplication methodology is widely used in various archival storages and data centers due to its significant contribution to the storage industry by providing space utilization and IO performance by limiting the amount of write traffic. There are several kind of Storage Optimization/utilization methods that can be employed to store the data. The most common of them are Thin provisioning, Snapshots, Clones, Deduplication and Compression.

- Thin provisioning is the technology which employs on-demand allocation of blocks of data when applications run in out of storage space and thereby reducing the risk of application failures.
- Snapshot technology allows to store only the changes between each dataset
- Cloning is the real clone/copy of actual data.
- Deduplication is a technique which is used to track and eliminate the duplicate chunks in a storage unit.
- Data compression is a mechanism which saves the storage space by properly arranging the data in the required block size and adjusting the smaller blocks to fit the available free block space.

Data Deduplication

Data deduplication is a process that eliminates redundant copies of similar/already existing data and significantly optimizes storage capacity utilization. Deduplication can be an inline process as the data is being written into the storage system or as a background process to remove duplicate writes after the data is written to the device. Benefits of data duplication includes reduced data footprint, longer retention capability and garbage overhead.

The way that deduplication process works is that data is divided up into smaller chunks. A chunk is one or more contiguous blocks of data. The chunk sizes can vary from system to system, the design can also use fixed size chunks or variable size chunks. But for our proposal we will the using fixed size chunks of data (128Kb). The way the comparison works is that each chunk is passed through a hashing algorithm, such as SHA-1, SHA-2, or SHA-256, which creates a hash. If the hashes of 2 chunks matches, they are said to be identical. For our proposal we will be using SHA-1 hash is 160 bits long (20bytes). If we create a 160-bit hash for an 128kb chunk, you save almost 128kb every time same chunk is copied. That is why deduplication process is such a space saver.

Prior Solutions:

There are lots of third-party applications available that helps to locate the duplicate/redundant data on the USB drive, though most of the applications/tools are paid. Windows search operation also sometimes helps to find the find the duplicate files which a user can delete and free some space. All the tools available helps us to remove the data after the operation is done and is depending upon the Host computer to remove the duplicate data. Also, the available tools fail if the files having different names and the content of the file is duplicate. Hence a more fitting solution is needed which can help to detect the exactness of data even if files are having different names and copied in different location. There are no prior designs available that helps to detect the duplicate data and prevents it during copy operation itself. Hence we propose a solution which will help to solve this most occurring user issue where the duplicate data will be detected prior copying to drive resulting in saving a lot of space which can be made available to copy other files.

Description

Data deduplication in helps to overcome three issues on the flash-based USB drives

- price per GB of storage space

- write limit or disk endurance
- garbage collection overhead

By removing duplicate data, the deduplication system helps to improve storage efficiency and protects flash devices from unnecessary/duplicate writes. The aim here is mainly increasing the storage capacity of the USB device without adding any extra cells to the storage. The idea is to remove the duplicate data which takes the unnecessary memory to store the same data which was already copied by the user earlier and referencing on existing data signature. The solution utilizes chunk level deduplication to optimize memory utilization, decreases the latency and ensure effective deduplication. The deduplication process comprises of following stages: splitting the data into chunks, calculation of data hash/signature, searching the signature if existing and finally referencing if matching signature is found.

The general architecture of USB data flow is shown in fig1:

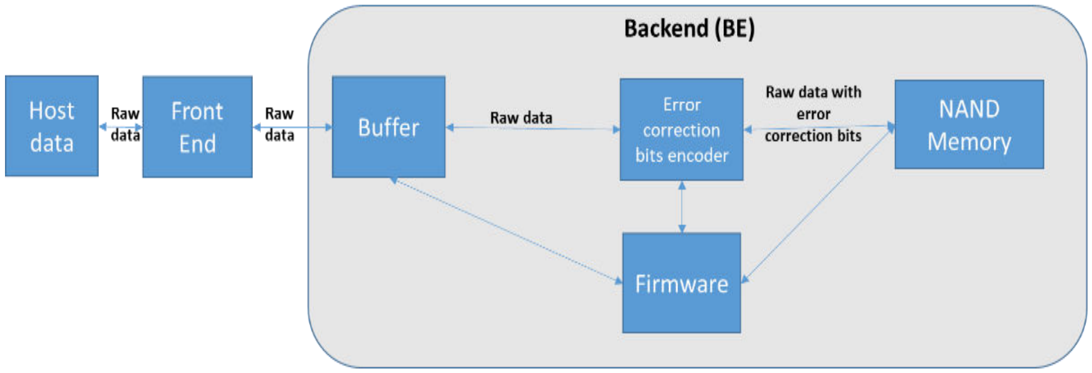

Fig1: USB dataflow architecture

Currently, the storage system receives the raw data coming from the host, append/pad the encoded bits for error correction and store them on the NAND device as shown in fig 1 below.

The components involved are as follows:

The data from the host is given to the front end (FE) as a write command. The front end transfers the data to the backend by writing the host data into a buffer. The data in the buffer maybe padded with error correction/parity bits. This is processed by firmware to determine the address in the memory where data needs to be written in the NAND flash.

The current process doesn't evaluate if the host data is the duplicate or not. It just writes everything requested by the host to the NAND.

Proposed Solution

The idea proposed here is to have a hardware (controller) to calculate the hash of incoming data between the front end. The hash of the incoming data from the host is calculated by the controller and if no matching hash is found in the buffer then the data is written into the buffer and processed by firmware and written to NAND. And if a matching hash of the incoming data is found in the buffer, then that write chunk can be ignored since the same (duplicate) data is already present in the memory .The hashing of the data thus helps in avoiding the duplicate writes to the memory and saves number of bytes to be re-written in flash and enabling the de-duplication in the device.

The deduplication of data ensures only unique hash write chunks are written to the flash and results in increase of storage capacity in case of multiple copies of same data is written.

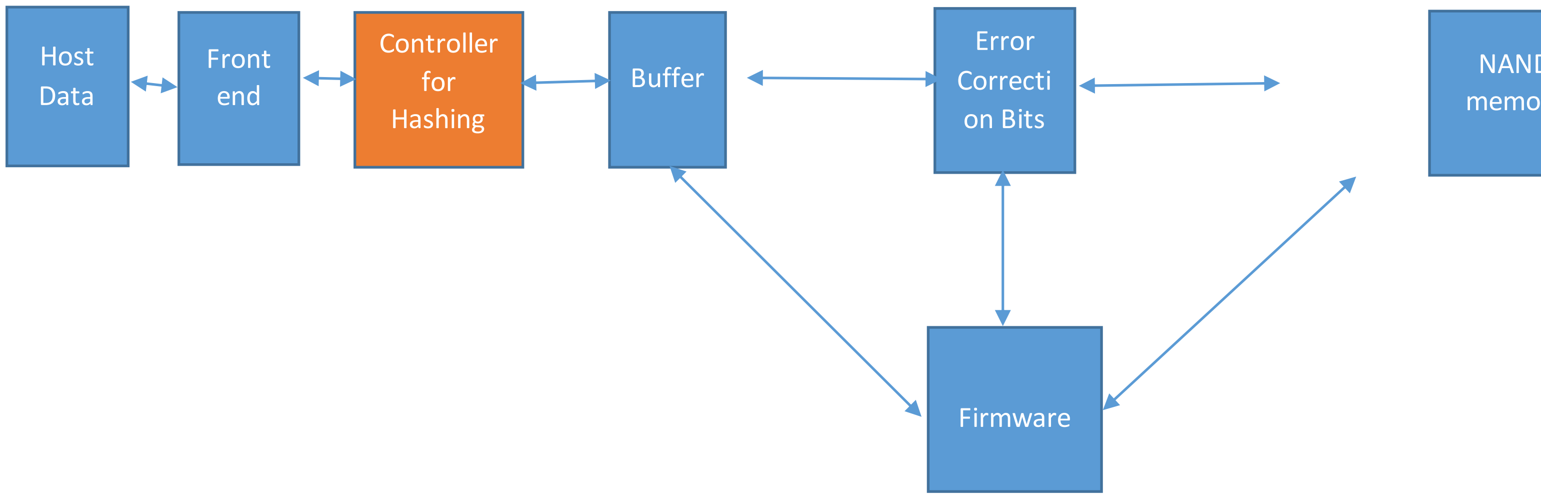

Fig2 : Hashing controller added in FE to calculate Hash before writing data to NAND

Working Methodology

For better understanding of the working process we will consider 2 scenarios , the first where we copy 2 unique files (files with different data) each of 4MB to the drive and in other case 2 files with same content (duplicate data) and see how our proposal adds value .

Case-1: Unique files are copied

- Take 2 different files each of 4MB size
- Start copying the files to the Drive simultaneously. When the first file is received from the Host to FE of drive, the hashing controller will divide the 4MB files to 128KB chunks and calculate the Hash for each chunk and store them to the Buffer. (4MB/128KB= 32 chunks each of 20bytes as per SHA-1 algorithm)
- Since it is the 1st file being copied the chunks will be matched to chunks present in buffer, and since no matching chunks will be found in buffer these chunks will be processed by FE to buffer and from buffer to NAND.
- Similar process will happen when the 2nd 4MB file will be copied. The hashing controller will divide the file in 128KB chunk size and see the buffer, if the matching chunks are not present, they will be processed by the Front end to buffer and later to NAND and both the files will be stored in the drive.

Case-1: Duplicate files are copied

In this case 2 files with same content/data will be copied. Here both the cases are possible , files could be the same name or different name , but the content in file will be exact same.

- Take 2 files with exact same data each of 4MB size
- Start copying the files to the Drive simultaneously. When the first file is received from the Host to FE of drive, the hashing controller will divide the 4MB files to 128KB chunks and calculate the Hash for each chunk and store them to the Buffer. (4MB/128KB= 32 chunks each of 20bytes as per SHA-1 algorithm)
- Since it is the 1st file being copied the chunks will be matched to chunks present in buffer, and since no matching chunks will be found in buffer these chunks will be processed by FE to buffer and from buffer to NAND.
- Similar process will happen when the 2nd 4MB file will be copied. The hashing controller will divide the file in 128KB chunk size and see the buffer, in this case matching chunk will be found and this chunk will be ignored and next chunk will be matched. In this case all the chunks will be matched with already existing chunks, hence all the chunks will be ignored, hence the 4MB complete file will not be copied and that space is saved on the USB drive.
- Though to the user it will appear that the file is being copied but that copy time will be utilized to check if all the chunks are already present in buffer. This second file will only have references to already existing file, so that when user wants to read the 2nd file which has duplicate data, it will be available through the reference made to 1st file. Hence enabling to achieve the 4MB space.

Hence by this scenario we can see how addition of hashing controller helps the drive to save the space and provide as a feature add to the drive. Also helping user to save more data in space of redundant data being copied. The copy of buffer data which has all the signatures is also preserved by the firmware (in form of snapshot) so that when power failure happens all the data is preserved , and ones power is up the data signature record is copied back to buffer and process continues swiftly.

Advantages of the Solution

- Inline Deduplication is achieved, data is filtered out during copy operation itself. No background work to be done by the user to cleanse the device.
- Irrespective of the data type, one can consistently achieve the increased capacity
- The reduction of number of writes yield in longer device endurance
- SHA-1 provides the fastest calculation of hashes as compared to (SHA-2, SHA-3)

Conclusion

The proposed solution to remove duplicate data from the USB drive from the device side (inline deduplication) comes with an added cost of a controller. Also, the hash calculation adds a little delay in write performance (can be neglected) since each chunk has to go through the hashing controller while read performance does not have any impact. Hence this DeDuplication enabled USB device offers an added advantage to the users to save the space which utilized by the redundant/duplicate data and also improves the efficiency of the device. The idea can be utilized by any other flash product following the similar a architecture.

References

1. E. Manogar, S. Abirami, A Study on Data Deduplication Techniques for Optimized Storage, 2014 Sixth International Conference on Advanced Computing(lCoAC)

2. Survey on Deduplication Techniques in Flash-Based Storage, PROCEEDING OF THE 22ND CONFERENCE OF FRUCT ASSOCIATION

3. Dutch T. Meyer and William J. Bolosky. A study of practical deduplication. In Proceedings of the 9th USENIX Conference on File and Stroage Technologies, FAST'11, pages 1–1, Berkeley, CA, USA, 2011. USENIX Association.

4. Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, and Yukun Zhou. A comprehensive study of the past, present, and future of data deduplication. Proceedings of the IEEE, 104(9):1681–1710, September 2016.

5. SHA-1 wikipedia https://en.wikipedia.org/wiki/SHA-1, https://en.wikipedia.org/wiki/Secure_Hash_Algorithms