Using Virtual Disks and Hamming Code for Data Preservation

Todor Cholakov, Dimitar Birov

Faculty of Mathematics and Informatics, Sofia University
todortk@abv.bg, birov@fmi.uni-sofia.bg

Abstract. Digitalizing archives and data happens more and more often. There are many obvious advantages of digital data- it won't change over time, it is much more compact compared to any analogue representation and it lasts over time. However there are risks concerning digital archives, too. Each media eventually wears out, which may cause errors when accessing the data. In this paper we propose an approach relying on a combination of virtual hard disks and checksums for data preservation.

Keywords: digital libraries, data preservation, vhd, crc, hamming code.

1 Introduction

Converting data into digital form usually requires a lot of time and efforts. In many cases the originals are destroyed after being digitalized. Even when they are left intact, it is important that the digital copies are preserved and accessible over time.

However most kinds of digital media have their issues:

- Hard disks and tapes rely on magnetic particles to store data. Even if the tape or disk is used as an archive it may develop bad sectors, causing that some of the stored data cannot be accessed [1,2].
- Optical discs like CDs or DVDs can get scratched. Sometimes they are destroyed by exposing to sunlight [3,4].
- Data on flash drives also wears out during time.

In order to guarantee data integrity all manufacturers implement data check and recovery mechanisms in their drives. Usually a 512 byte sector is stored in more than 512 bytes, some of them containing an error correcting code for the user data. This kind of sector structure allows recovery of single bits or bytes within the sector. However as the medium ages, the number of wrong bytes in a sector may increase over the limit that the error correcting code is able to correct. When this happens the whole sector is marked as unreadable and the data is lost. As we explained above, the process of wear of the media may happen at any time – both while the media is active (online), or while stored on the shelf (offline). The most common case is when the destroyed parts of the data are near each other – a scratch on a CD or DVD or a power

outage for a HDD causes errors in several adjacent sectors or tracks, leaving the rest of the media intact. Manufacturer defects are also commonly grouped together.

There are several common approaches for securing data on live systems – regular backups, RAID [5] arrays and so on.

RAID arrays rely on two or more disks, to provide data consistency. For a live system this is a good approach, but it's not so good for archives where keeping two or more archives of the same data would cost too much.

Backups rely on external storage to keep the data. The most common kind of storage for backup is magnetic tapes, but they are too expensive for large amounts of data. CD or DVD discs are much cheaper option but they are very unreliable. Hard disks are much more reliable when used for archiving, but even they can develop bed sectors over time.

Our approach is designed to work for backups and cheaper live systems where the organizations can't afford to pay much for high class disks and raid controllers or have too much data. It can transparently be combined with RAID arrays for online usage.

In the next section we w will give some basic terms. In section 3 we will present our approach. In section 4 we will consider some experimental data.

2 Terminology

2.1 Hamming Code

Hamming codes [6] are a class of linear binary codes. For each integer $r \geq 2$ there is a code with block length of $2^r - 1$ and message length $2^r - r - 1$ having a distance of 3. A Hamming code is able to detect up to two errors and to correct one error. The higher the value of r, the better is the ratio of the message length to the block length. In our approach we use a hamming code for block size of 31 which has 26 bits of message data and 5 bits of overhead.

Using hamming code with larger block size would improve the ratio of the message length to the block length. However this would increase the risk for the data. The problem is not only that we will be able to correct 1 out of 57 bits of data, but that the probability for having two errors increases. On the left picture bellow we have a disk that uses block size of 31. It is divided into 26 parts. When a problem appears (one or more bad sectors), it falls on one or two of the pieces of the disc, so it is correctable. On the right picture we have used block size of 63. The error area stays the same, but in this case it may cover 2 or three pieces of the disk and in some cases may be uncorrectable. It is possible to increase the block size when managing tape drives, because the errors there appear along the track instead of having width of several tracks.

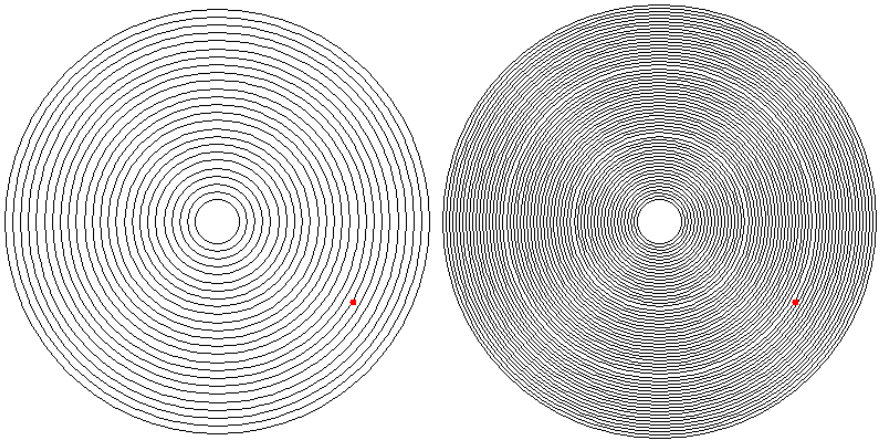

Fig. 1. Disks using 31 and 63 block size

2.2 Virtual Hard Disks

A virtual hard disk is a file containing the information usually contained in a whole hard drive. A virtual hard disk may be partitioned and accessed by a virtual machine or by special software. There are several existing virtual hard disk formats represented by the different software vendors:

- Vmdk discs are the proprietary format for VMWare virtual machines[7].
- VDI discs are specific to Oracle VirtualBox.
- VHD discs are used by Microsoft VirualPC [8].

Most of the current virtual machine software implementations are able to read all of the above listed formats. However Microsoft has integrated support for the VHD format allowing virtual disks to be mounted and used as regular disks.

Being located in files, virtual hard disks have some features, not available for regular hard drives:

- Variable size – a virtual disk may be created with a minimum size containing only the disk header. The file containing the disk is extended whenever a new sector of the disk is to be written. Overwriting already al-located sectors doesn't change the size of the containing file. There is no guarantee that consecutive pieces of the file containing the virtual disk contain consecutive sectors.
- Differencing disks – this feature allows a virtual disk to inherit the data from a parent disk image and store only the sectors that are different from its parent.
- Easy movement – virtual hard disks may be moved between partitions or physical hard drives and still remain accessible.

Being regular files virtual hard disks may be easily accessed by high level programs using standard file operations. This gives us the ability to write custom software for processing virtual drives both as files or as disk images.

3 Data Preservation Using Virtual Disks

Our approach relies on the following assumptions:

1. Whenever there are problems in storage media, they are located in consecutive sectors or tracks rather than being spread all over the media. This assumption is valid in most cases for both hard disks, CD/DVD disks and tape devices:

- Hard disks develop bad sectors either because their surface wears out, which causes correctable errors on sector level or because of mechanical issue. The mechanical issues happen when the head or a small particle touches the surface of the disk. Because of the high speed of the rotation of disk, the scratch left is alongside a track or several tracks.
- CD/DVD disks also develop bad sectors mostly due to scratches. When the scratch is not parallel to the tack, the error correcting mechanisms of the CD/DVD device are able to correct the data. Again the problem is when the scratch is alongside the track. This is the case where a bad sector is developed and the case that is the target of our approach.
- Tapes may develop bad sectors if there is a problem in the mechanics or part of the tape itself. As the tape is read and written successively, the damage will happen on small part of it, which corresponds to error alongside the track on hard disks.

2. The most current data is easily recoverable. This assumption is true in many cases but depends on the definition of "current". The data from the last few days may be recovered easily. The data from the last six months may not be recovered so easy. When there is a high risk for the current data the approach may be modified to use a RAID array for it. It will still be cheaper to have two 40GB disks in RAID 1 array and a 3TB one for the rest of the data, than using a RAID array of 3TB disks.

The approach is as follows:

We create a fixed size virtual hard disk in VHD format. Let Ld be the size of the created disk in MB (1MB=1048576bytes). The size of the disk must be chosen in such a way that there is at least $5(\frac{Lb}{26}+1)$MB free space left on the containing partition.

A checksum file is created for the virtual disk. Its length is $5(\frac{Lb}{26}+)$MB depending on Ld. The checksum creation process goes as follows:

Let $Ls = (\frac{Ld}{26} + 1$*1048576 and Ci is the current MB index.

For each Ci from 0 to $\frac{Ld}{26}$ we read 26MB of data from the virtual disk file. The data read is 1MB at positions Ci*1048576,Ci*1048576+Ls, Ci*1048576+2*Ls,…. Ci*1048576+25*Ls. If the MB at Ci*1048576+25*Ls would pass the end of the file, it is considered containing only zeroes.

For each such set of 1MB pieces we create a 5MB hamming code and write it in the checksum file.

Because of the distance between the different chunks it is more likely that at most one of them would fail if the hard disk starts to wear out. At the same time the checksum file does not take too much space.

We create a child virtual disk of the currently created virtual disk and mount it. All written data will go to the child while any reads which are not contained in the child will be read from the original disk. If the host drive develops bad sectors on the master drive, they will be corrected by using the checksum file. The data in the child is still vulnerable to damage, but we assume that it may be easily recovered from other sources. We may use a RAID volume in order to ensure its consistency if available.

On a regular basis or when the child volume reaches a prespecified size, we transfer the contained data to the parent virtual drive. We created a custom tool that transfers the data and at the same time updates the checksum. The algorithm which is used ensures that even if a power outage happens, there will be no data loss and it might be performed again to finish its job. After the checksum and the parent volume are updated, the child volume is deleted and then created again.

When the data needs to be archived we must perform the parent virtual disk update and then copy the parent virtual disk and the checksum to the external storage. Even if the external storage develops bad sectors the data and the checksum file will be enough to recover the original state.

The proposed approach is suitable in cases where there is enough time available to perform the maintenance tasks. It is not suitable for 24/7 systems, because the maintenance tasks will stop the system for a while.

4 Experiments and Tests

The proposed approach is currently being used for a 1,5TB hard disk. The size of the virtual drive is about 1TB and the checksum is about 200GB.

The data is moved from the child to the parent whenever the child reaches size of about 20GB. The process takes about 2 hours using a E4500 Core 2 Duo processor and a WD Caviar Green host hard disk.

The initial checksum creation takes about 16 hours for this setup.

When the host disk started to develop bad sectors we were able to restore the data without any loss to another hard drive. A disk check utility showed that the bad sectors were concentrated on a single place of the drive.

For testing purposes I created an 8GB virtual disk and copied it. Then I changed parts of the copy using a hex editor. After restoring using the generated checksum, the resulting file was the same as the original one.

5 Improvements and Future Work

Using a RAID array to store the child virtual disk would eliminate the risk of losing some of its data.

An additional checksum file might be created on the generated checksum file because its data is more important.

A multi-threaded approach to the checksum generation and update process allowed a double increase in speed.

An extension to this approach would be to intercept writes to the child virtual image and create a checksum for them on the fly.

6 Conclusion

Digital data is often destroyed by corrupt media. All the effort made to transform it in digital form may be lost if the containing media starts to deteriorate. The proposed approach tries to eliminate some of the most common risks and to prevent data loss especially for archived data. Additionally it is compatible and may be used together with other data loss prevention methods such as RAID arrays reducing their cost (we need to host only the child virtual disk in a raid array, which may be much smaller than the original drive).

7 Acknowledgements

The authors gratefully acknowledges financial support by the Bulgarian National Science Fund within project DO 02-102/23.04.2009.

References

1. A. Sebstian, How long do hard drives actually live for?, http://www.extremetech.com/computing/170748-how-long-do-hard-drives-actually-live-for
2. Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz André Barroso (February 2007). "Failure Trends in a Large Disk Drive Population". USENIX Conference on File and Storage Tech-nologies. 5th USENIX Conference on File and Storage Technologies (FAST 2007)
3. Marken. A., CD and DVD Longevity: How Long Will They Last?, http://www.audioholics.com/audio-technologies/cd-and-dvd-longevity-how-long-will-they-last
4. Bradley, Kevin. (2006) Memory of the World Programme: Sub-Committee on Technology: Risks Associated with the Use of Recordable CDs and DVDs as Reliable Storage Media in Archival Collections–Strategies and Alternatives. p. 11. Paris: UNESCO. Accessed on Oc-tober 8, 2007
5. Patterson, David; Gibson, Garth A.; Katz, Randy (1988). "A Case for Redundant Arrays of Inexpensive Disks (RAID)". SIGMOD Conferences. pp. 109–116.
6. Moon, Todd K. (2005). Error Correction Coding. New Jersey: John Wiley & Sons. ISBN 978-0-471-64800-0.
7. VMWare, Virtual Disk Format 1.1, http://www.vmware.com/app/vmdk/?src=vmdk
8. "Virtual Hard Disk Image Format Specification". Microsoft TechNet. Microsoft Corporation. 27 February 2009.