Optimizing RAID for Long Term Data Archives

Henning Klein
Fujitsu Technology Solutions GmbH
Bürgermeister-Ulrich-Strasse 100
86199 Augsburg, Germany
Henning.Klein@ts.fujitsu.com

Jörg Keller
Fernuniversität in Hagen
Dept. Of Mathematics and Computer Science
58084 Hagen, Germany
Joerg.Keller@FernUni-Hagen.de

***Abstract*—We present new methods to extend data reliability of disks in RAID systems for applications like long term data archival. The proposed solutions extend existing algorithms to detect and correct errors in RAID systems by preventing accumulation of undetected errors in rarely accessed disk segments. Furthermore we show how to change the parity layout of a RAID system in order to improve the performance and reliability in case of partially defect disks. All methods benefit of a hierarchical monitoring scheme that stores reliability related information. Our proposal focuses on methods that do not need significant hardware changes.**

Fault-Tolerant Software Design; Data Archival

I. INTRODUCTION

Storage media have always been a compromise between factors like volume and physical size, speed, reliability and price. Although the price/performance ratio and the speed are getting better, the reliability of storage media isn't increasing significantly. Moreover the failure types are changing with newer models and technologies which make error predictions almost impossible. The only way to get estimations on long term behavior is backwards analysis like in [13], [14], [6] and [11]. All these papers are based on observations of large disk drive populations. They analyze correlations between parameters like disk type, error type, built in health monitoring mechanisms, age, environment, etc. The studies give rough estimations on how future errors or disk lifetimes could be predicted. However, not all of the papers came to the same result and the observed disks are usually replaced by successors before the studies are even published. Especially when we have a look at a longer range of time the technologies are changing completely, i.e. current rotating hard disk drives are likely to be replaced by solid state disks with a complete different set of failure types and probabilities [8] which is again subject to continuous change [5].

The most common approach to overcome data loss is adding redundancy by organizing disks in arrays called RAID [1] which sacrifices the space of one or more media for redundancy. The original version has been enhanced several times for specific cases, but is primarily used to overcome a single error type: total disk loss. Tolerating disk failures however isn't enough, especially in applications like long term data archival. Moreover newer disk technologies are less likely to cause disk failures than undetected bit errors. We have proposed a technique to enhance the detection and correction capabilities of RAID, showed the advantages of integrating encryption and reported on the performance of parallel execution in modern multicore processors [4]. If the proposed scheme is implemented in Software, like the file system ZFS [2], the protection is "end-to-end" and covers errors occurring in data paths, controllers or cache memories as well.

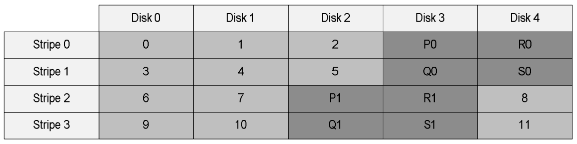
Long term data archival is a matter of scope. While there are existing approaches like Oceanstore [16] or LOCKSS [12] that spread data to different locations to overcome data loss as a result of environmental disasters, i.e. fire or theft, in this paper we concentrate on data loss caused by media errors. Despite whole media errors there are single faults which can be directly detected (visible) as the medium reports it during access or undetected (latent). To avoid data loss due to accumulation of latent errors in a redundant system a well-known technique called scrubbing has to be implemented, which frequently reads, checks and if necessary corrects data. Probabilities of combinations of visible and latent errors have been investigated in [17] and a PhD thesis [7] for disk arrays. Especially long term data archival is vulnerable to latent errors as data might not be accessed for a long period of time. To reduce cost and increase the lifetime of a medium, it is usually powered down when being idle. To check data, the medium has to be powered on. The impact of scrubbing on energy consumption has been investigated by [10]. The times when disks are running have been taken into account for one of the scrubbing algorithms proposed by [15]. A more sophisticated approach is being shown in [9] based on results of observations in [11]. Again, these publications are focused on observations of current disk technologies and don't consider the underlying disk array capabilities. Our approach monitors the disks behavior closely. We propose several mechanisms for different disk storage technologies to predict and prevent data loss. We figure that a solution that is robust to technology changes can't rely on studies from the past and that built in mechanisms to overcome wear-out of rarely accessed data over time might not work, if the device is powered down to conserve energy. The proposed schemes also rely on reading data as a mechanism to prevent data loss due to latent errors. That might be sufficient for HDDs but future technologies like SSDs need rewrite-cycles to avoid "forgetting" data, as stated in a SSD white paper [8].

The remainder of this paper is organized as follows. Section 2 gives an overview of the reliability of RAID and our previous work. Section 3 describes a hierarchical scheme to store data that is relevant for reliability and we show ways to increase the reliability of RAID systems. In the last section 4 we give a conclusion and an outlook on our future work.

II. DATA RELIABILITY OF RAID

In terms of reliability RAID is primarily designed to tolerate the loss of disks (total disk failure) rather than detecting and correcting single errors. The architectures of RAID systems are quite similar in most cases where data redundancy is pursued. First all disks are dissected into equally sized blocks. Blocks with the same offset across all disks are combined to a stripe. Within one stripe a certain amount of blocks is sacrificed to store redundancy information. I.e. the well-known RAID systems 5 and 6 use one and two blocks per stripe to store redundancy information which reduces the overall size of the RAID and enables the recovery of one or two disks.

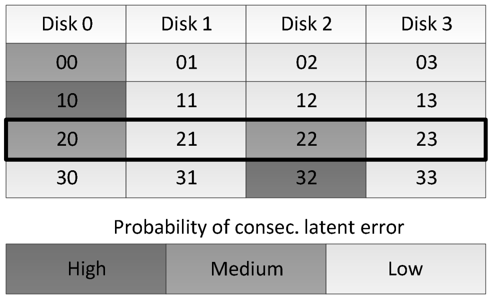
Redundant data can as well be used to detect and correct errors. In general at least one faulty block, whose data has been altered without further notice, can be detected and corrected for every two redundancy blocks that are stored in the stripe. In case of RAID 6 a single block, that has silently been altered, can therefore be recovered per stripe if no disk failed. In [3] we have shown how the capabilities of error detection and correction of RAID 6 systems can be improved by connecting two consecutive stripes and computing four redundancy blocks. We have shown that even after one disk failed a single block can still be recovered in every two stripes of the remaining disks. By adding block-encryption the capabilities can be further improved as bit errors are easier to detect as they spread across the whole block. We have shown how the scheme can be implemented efficiently in a software RAID [4]. Figure 1 depicts the RAID layout of the previously proposed scheme.



	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
Stripe 0	0	1	2	P0	R0
Stripe 1	3	4	5	Q0	S0
Stripe 2	6	7	P1	R1	8
Stripe 3	9	10	Q1	S1	11

Figure 1. Enhanced RAID Layout

When considering this scheme for long term data storage, more complex fault scenarios have to be considered. Rarely accessed data can develop undetected (silent) errors over time. Moreover recent investigations have shown that consecutive silent errors tend to be spatial and temporal correlated [11]. Combinations of visible and accumulated errors as investigated in [17] can lead to unrecoverable errors. Figure 2 depicts this constellation and shows how two errors with similar offsets affect the same stripe (black frame). If such errors arise in regions with similar offsets, nearby stripes are more likely to be impacted by multiple errors of different disks. When considering new storage technologies like Solid State Drives (SSD) for long term data archives another factor has to be taken into account. Cells of these disks are losing their state over time, so periodical write cycles are necessary to avoid data loss. Additionally physical defective areas of rotating hard disks can lead to decreased performance rates as disks try to recover damaged data in multiple cycles. If the disk is kept in the array, frequent accesses to defective areas would result in an overhead that prolongs the time of the initialization of a newly added replacement disk. On the other hand the disk could still store valuable information to support error detection and correction capabilities that would be lost if the disk is removed immediately.



Disk 0	Disk 1	Disk 2	Disk 3
00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

Figure 2. Multiple Errors affecting one stripe

We have analyzed the issues mentioned above and provide techniques for RAID systems to decrease the threats of data loss. We consider the proposed techniques to be implemented within a software RAID which has benefits of end to end data protection as described in [2] and is an inexpensive and flexible solution. However, the described methods could as well be implemented in a hardware RAID controller. The scope of the proposed solution can only be seen as part of a reliable long term archival as scenarios like natural disaster are not taken into account.

III. Optimizing storage arrays

A. RAID health monitoring

Monitoring the health of data media is a common approach to estimate the reliability for future runtime. Current storage media implement records with a set of values representing the media health. The goal is to prevent data loss by indicating media faults in advance. However, latent faults are not detected by the storage media and data loss in redundant RAID systems is a combination of disk faults or errors in higher levels, i.e. the storage controller. In this section we present a hierarchical scheme that monitors faults for the entire RAID-system which enhances reliability estimation, fault localization and measurements to prevent data loss.

The proposed structure is used to assist methods like disk scrubbing or detecting the scope of defective areas. We consider this structure to be maintained by a Hard- or Software RAID controller which stores the content to the RAID disks either immediately for rare events like errors or periodically to store less critical information like disk usage statistics.

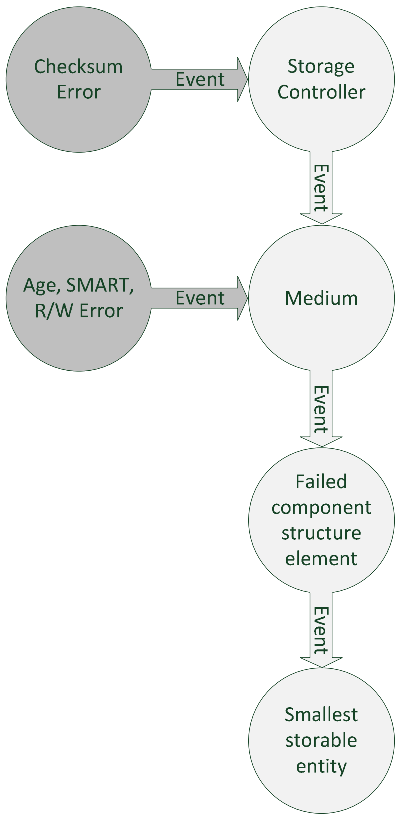


Figure 3. Hierarchical storage system

If the redundancy data in RAID systems is used to perform integrity checks, the minimum readable entity is a whole data stripe as all blocks within one stripe have to be read in order to calculate the parity values which are then compared to the values stored on disk. This simplifies keeping track of frequently accessed data areas as they are equal for all disks. Therefore the amount of data that needs to be stored depends only on the size of a single disk instead of the size of the RAID. We organize a RAID system in hierarchical components starting at the controller on the highest level, followed by the disks. The following layers represent data areas, starting from bigger sized regions down to a RAID block which is the smallest accessible entity per disk. Additionally errors that have been detected either by the disk (visible) or during redundancy checks of scrubbing passes (silent) have to be stored independently. Figure 3 depicts the hierarchical architecture. Further details are given in the sections below that are describing scrubbing and faulty area avoidance.

Symptoms like disk error rates, temperature or utilization are stored in the built in disk monitoring functions called SMART. These values could be combined with the observations stored in the proposed hierarchical scheme to predict future failures and adjust scrubbing rates to ensure data integrity before disks fail. Additionally the values can be used to make assumptions on the lifetime of other disks in the RAID with similar symptoms. As disks in new RAID systems are usually of the same brand and type and are often even of the same production batch, chances are higher that symptoms preceding disk failures are similar as the disks are equally accessed. Predicting disk failures has been analyzed in various papers like [13] or [18] and are not further analyzed in this paper.

B. Adjusting the RAID layout

1) Scheme

The ability to correct single errors depends on the amount of redundancy information per code word. By changing the RAID layout and doubling the stripe length the capabilities of classic RAIDs to correct errors can be increased as described in our previous work [3]. To avoid generating overhead due to increased transfer volume or seek times, two data stripes were positioned consecutive and created by splitting up blocks into two parts of equal size. Long term archives are usually quite rarely accessed and don't demand high performance. Keeping the data integrity however is the highest goal. Therefore we propose splitting up the RAID stripes, so that errors affecting a wider consecutive area affect only data of one of the double stripes. Depending on the access size and the storage technology used this would of course decrease the read and write performance.

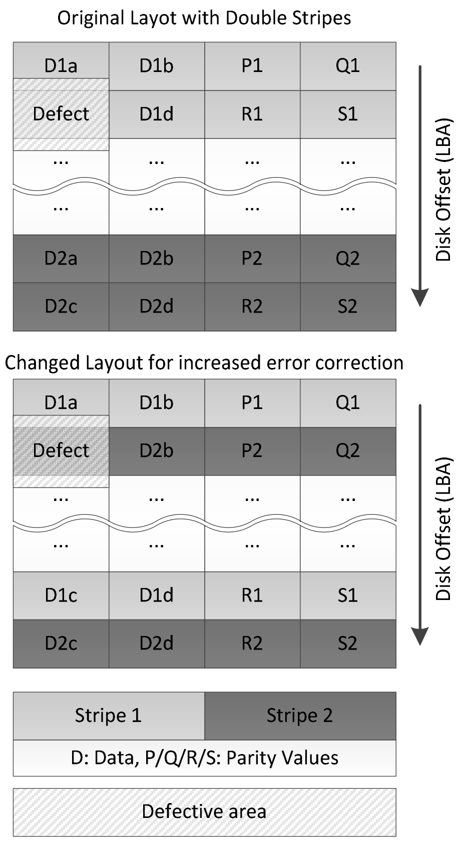


Figure 4. Splitting up double stripes

However, in case of a partially working disk this increases the capabilities to detect and correct silent and known errors until a replacement drive has been added and initialized. Figure 4 depicts how a defective area would affect blocks of different double-stripes. In the original layout data of two blocks in one stripe would fail (upper example). If the two halves of the double-stripes are distributed evenly across the disk only one block of each stripe would contain faulty data, which leads to increased failure correction abilities, as three instead of two blocks could still be recovered. The defective area could be as large as 50% of the storage size of the medium. The proposed hierarchical scheme is used to store areas that have been detected as defective and to estimate its scope. If faulty areas are skipped, performance issues can be overcome that are caused by storage media trying to recover data in repeated access cycles. The time intensive search for boundaries of a defective area can be cut short as there's still enough information to even detect and correct up to three visible and two latent errors.

2) Medium considerations

For HDDs the proposed scheme has a major drawback as splitting up the stripes results in increased head movements and therefore increases mechanical wear-out. At the same time, the performance will decrease. However, one of the latest studies [13] didn't see a significant impact of utilization on disk lifetimes except for very young and old drives. Therefore we figure, that this solution fits best for archives which demand maximum reliability. Large files would decrease the impact on performance as the number of head movements can be minimized if data is pre-fetched.

Little is known about failure rates and types of SSD drives. However, it can be assumed, that this technology is less vulnerable to large defective areas, as pages and blocks are independent of each other and automatically mix up during write cycles to enable an even wear-out of the medium. However in [8] the effects of "disturb faults" are shown, where program and read cycles could affect the content of adjacent cells. On the other hand direct cell access and no moving parts should make the overhead for the changed RAID layout small to irrelevant and be beneficial if the wear-out algorithm exchanges bigger blocks for performance reasons.

C. Scrubbing techniques

In this section we describe methods to decrease the probability of data loss in a running storage system. We'll show how to optimize RAID systems for two different storage technologies: Rotating hard disk drives (HDD) and chip based Solid State Disks (SSD).

1) HDD Arrays

A lot of strategies have been developed to implement efficient ways of scrubbing. A quite new algorithm based on the results of the study about latent errors of [11] is [9]. It dissects a storage medium into 128 MB regions that are further divided into 1 MB segments which fits perfectly in our proposed hierarchical scheme. We refer to their work for further details regarding HDD error detection capabilities. Basically the idea of "staggered scrubbing" in this publication is to go through the disk checking one segment per region in one pass. In the following cycle the next set of segments is checked until all regions have been checked completely. If errors are detected, the algorithm is being changed to perform a mix of sequential and staggered scrubbing. The reason why only one segment per region is checked is that latent errors often occur in multiple numbers spreading within the region. So there's a chance to find one of multiple faults by checking only a part of the region.

We adapt from this model and extend it first by adding immediate sequential scrubbing in case the age of the medium gets critical, based on saved history of other RAID members, SMART errors were detected or one medium failed. That way the content of the

RAID is being checked before media failures are expected. Secondly, we consider that some of the segments could have been accessed between two scrubbing cycles. Latent errors, especially if they tend to occur in multiples are more likely to remain unnoticed in rarely accessed areas. Therefore we propose to change the order in which segments are checked. First all segments are checked with higher priority that were not accessed between two scrubbing cycles. In order to get an evenly distributed coverage we choose the segment that has the maximum distance to the neighbor segment that has already been checked or accessed. For completion we continue with segments that have been partially accessed and skip only those which have been read out completely.

If latent errors are detected, their position is being saved in the scheme to detect constellations that could lead to data loss. If errors are detected in two regions on different hard drives, a future failure is more likely to appear, if both regions have a common intersection with a RAID stripe. The best way would be to replace faulty sectors of at least one of the disks with space of a spare pool on the disk. SCSI disks implement a command that enables sectors to be marked as defect and replaced. However, consumer drives do not implement such methods and the overhead space could be exhausted soon if large regions are marked as defective. Therefore we propose adding a simple translation layer that remaps regions until further investigations prove that it is unlikely that additional latent errors will follow. The drawback of the overhead with additional seek times and administration structures should be marginal as the size of 128 MB for the regions is large enough so that the additional positioning time during consecutive accesses would be smaller than a few percent on modern disk drives. Figure 5 depicts this process.

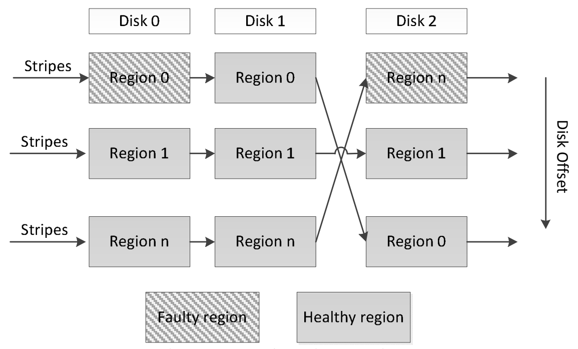


Figure 5. Exchanging Regions

We have implemented a simulator to show the advantages of the proposed scrubbing algorithm. We use access patterns to describe user accesses on the disks. We define three different patterns, which are simulating different data transfer volumes per day: low (5 GB), medium (250 GB) and high (2.5 TB). The accesses are split up in small (< 256 KB), medium (500 KB – 10 MB) and large (50 MB – 5 GB) consecutive chunks and are randomly distributed across the storage array. For simplification only the status value for each RAID block is stored, as this is the smallest accessible entity. Like in [9] we use 128 MB region and 1 MB segment sizes. Each scrubbing cycle is being done daily and applied in multiple partial passes throughout one day. Before a scrubbing cycle starts a random error is being injected. Disk areas, which have already been accessed, are skipped by our proposed scrubbing algorithm. Therefore frequently accessed areas are not additionally stressed by scrubbing passes. By exploiting that disks are accessed equally in a RAID system with integrated redundancy checking, storing accesses can be simplified. After each scrubbing cycle the access history structure is being reset. Our results show that although the amount of data that has to be checked is reduced as depicted in Figure 6, single errors can still be detected as good as in the original proposal, see Figure 7. We have measured the average time needed to detect single errors on a 2 TB RAID system. Additionally, our proposal detects latent errors earlier, if they spread in multiple consecutive segments, see Figure 8.

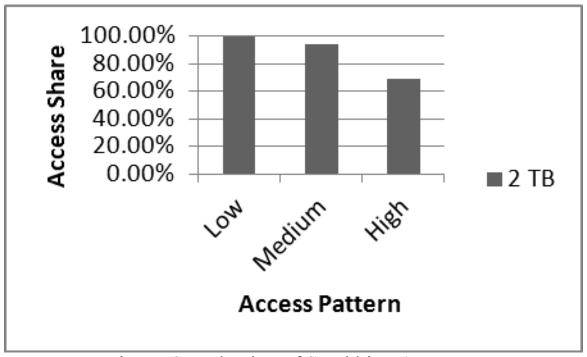


Figure 6. Reduction of Scrubbing Accesses

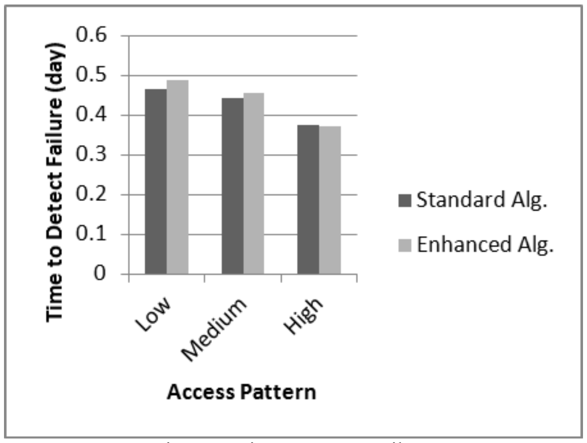


Figure 7. Time to Detect Failure

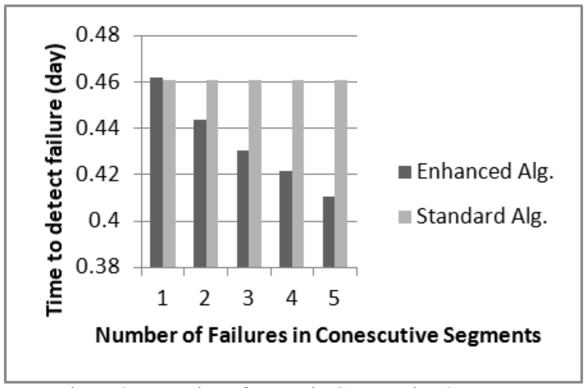


Figure 8. Detection of Errors in Consecutive Segments

2) SSD Arrays

SSDs store information in chips, organized in blocks which consist of pages that store the data bits in cells. Data cannot be overwritten before an entire page is cleared. Therefore, especially write accesses should be aligned to the page size and positions. The disks are less vulnerable to device failures, but due to the technology are not capable of storing data for years if written once only. Rewriting a cell reloads it with the correct capacity and therefore prolongs the time the cell will hold the correct state. However, the number of write cycles is the main factor limiting the lifetime of SSDs. Therefore an algorithm that checks and reloads data cells must carefully choose the reload-frequency to keep lifetime and data protection as high as possible. Due to a lack of published investigations on SSDs and fast changing technologies that try to close the gap in size between SSDs and HDDs we figure it's currently impossible to estimate how long data cells for a particular disk model will hold data, especially as the wear-out level is likely to reduce it. SSDs have several advantages over HDD drives, i.e. low power consumption, little impact by environmental factors like vibrations and higher transfer speeds. Therefore it is likely to completely replace HDDs in the future even for scenarios like long term data archival.

For this reason we propose an algorithm that avoids data loss while minimizing wear-out. This algorithm consists of two parts: one that monitors the disk detects and corrects latent errors and another part that refreshes the cell's contents. We'll keep the proposed HDD algorithm to detect latent errors with a few modifications. SSDs are less impacted by random read accesses. Therefore the size of a segment can be reduced and set to the page size of the SSD which enables a better distribution of the checking spots. As the disks implement an algorithm to enable an even wear-out of the cells, the method to avoid double latent failures on different disks can be simplified. Instead of remapping regions the data can simply be rewritten as the disk will automatically change the data position.

For the refreshing algorithm we propose using two calculated time limits which represent an upper and a lower value of the reliability of a cell in terms of losing its state. Above the upper value, the cell is considered to hold a correct value at a sufficient probability. The lower value is used as a line that shouldn't be crossed, as the cell's state is considered to become unstable and so an immediate scrubbing with rewriting should follow. If a cell reaches the age between these lines there's a chance that data might get corrupted. There should always be enough data blocks left above the line to reconstruct and check data. An Algorithm for fault avoidance would therefore work as follows. Let k be the minimum number of blocks necessary to check and reconstruct all data within one dataset. A medium will be rewritten (refreshed), if it passes the lower value or if less than k disks are beyond the upper value. If more than one combination is possible, the disks with a lower number of previous write-scrubs will get a higher priority. Figure 9 shows how the cell capacity in a SSD decreases over time and is being refreshed by rewrite cycles.

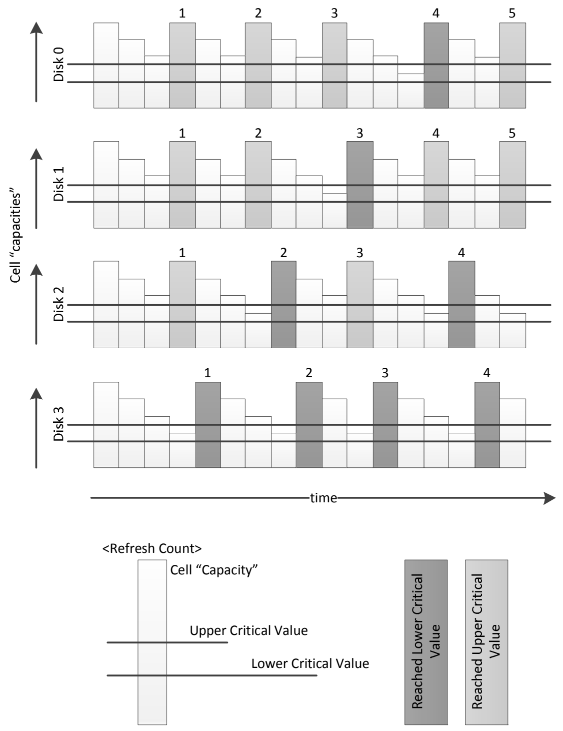


Figure 9. Unbalanced write scrubbing on a SSD

Three out of four disks or data blocks are being refreshed before the upper critical time value is reached. The upper and lower values have to be adjusted by the number of errors found during the read-scrubbing and the number of writes performed on the medium. In [8] the time an SSD holds data is considered to be "several years". If the array is written regularly and data pages are exchanged throughout the medium there's no need for extra rewrites. However, in a scenario like long term data archival, the disks might be written once and rarely accessed or even shut down for some time. Even if the cells are accessed regularly, they'll get worn out and the ability to keep data over time might decrease.

We propose to set the initial upper value to 6 months and keep the lower value twice as much as the upper value. If no error was found during the first scan we add 10% to both values. If an error was found during a scan we'll decrease both intervals to 50%. We figure that all cells might have been rewritten during two cycles and that no refresh might be necessary. Therefore we propose to skip the refresh cycle if 100% of the volume of the medium was written between two cycles. As two pages are exchanged in one write cycle that would mean, that all pages of the medium have been written twice in an optimal wear-out distribution, excluding some percentage additional space reserved by the disk to replace defective pages. In case the archive is actively been written, our algorithm won't increase the wear-out of the disk. In between and towards the other extreme end, where little or no data is written we'll slowly increase the confidence in the medium and double the refresh rate as soon as latent errors arise. The unbalanced refreshing will help minimizing the overall wear-out and give an ahead-warning if one medium develops massive data loss.

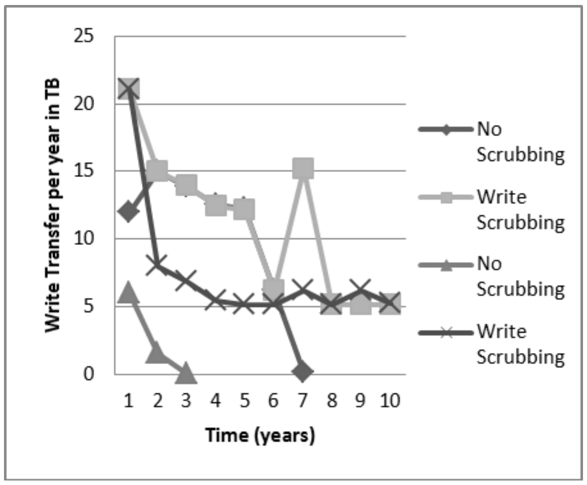


Figure 10. Unbalanced write scrubbing simulation

We have simulated a scenario in which the data volume written to the disk decreases below a critical limit so cells have to be refreshed in order to keep their state. For simulation purposes we assumed that each data cell can reliably keep its state for two years without being rewritten. After that we linearly increase the chance of data loss to its maximum after one year, so we assume that the cell loses its content after not being accessed for three years. Additionally we consider that the content of any cell has the chance to be swapped with a different cell that is written. As cells are swapped out randomly, all cells can be refreshed, if at least half of the disk's size has been written. In practice this takes more write accesses as SSDs also swap out cells with hidden additional cells that are used to replace defective cells and it would require an optimal distributing swapping algorithm. The simulated RAID consists of six disks, holding a total volume of six terabytes and a usable storage space of four terabyte. To ensure that data cells are refreshed, at least two TB of data have to be written in two years. We reduce the size of written data over time significantly below that value to show how our algorithm adapts to occurring errors and compensates low access rates by additional write cycles, see Figure 10. After one year the write scrubbing algorithm adapts to the initially sufficient write transfer rates by skipping unnecessary rewrite cycles. If the write transfer volume drops, the algorithm automatically increases write cycles. Without write scrubbing unrecoverable errors were detected quite soon after the transfer rate dropped.

IV. CONCLUSIONS

We have presented a new hierarchical architecture to store relevant data to predict and optimize the reliability of storage arrays (RAID). Furthermore we have shown how to extend the reliability, specifically for an application like long term archiving. We have shown that new storage technologies like SSDs have great advantages over HDDs but carry some threats that have to be addressed. Therefore we have introduced a new scrubbing algorithm that refreshes data and adjusts to the changing reliability. In our future work we will further improve the parameters used for the scrubbing algorithms by more detailed simulations.

REFERENCES

[1] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in SIGMOD '88: Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, 1988, pp. 109–116.

[2] Sun Microsystems, "Sun on-disk specification", http://opensolaris.org/os/community/zfs/ docs/ondiskformat-0822.pdf

[3] H.Klein and J. Keller, "Storage Architecture with Integrity, Redundancy and Encryption", 14th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems (DPDNS'09), Rome, May 2009

[4] H. Klein and J. Keller, "Optimizing a Highly Fault Tolerant Software RAID for Many Core Systems", International Conference on High Performance Computing & Simulation (HPCS 2009), Leipzig, June 2009

[5] Silicon Systems, Inc., "NAND Evolution and its effects on Solid-State Drive (SSD) usable life",

http://www.siliconsystems.com/technology/pdfs/WP-001-00R.pdf

[6] J. Gray and C. van Ingen, "Empirical Measurements of Disk Failure Rates and Error Rates", Microsoft Research Technical Report MSR-TR-2005-166, 2005

[7] H. Kari, "Latent Sector Faults and Reliability of Disk Arrays", PhD Thesis, University of Helsinki, 1997

[8] A. Olson and D. J. Langlois, "Solid State Drives (SSD) Data Reliability and Lifetime", http://www.imation.com/PageFiles/83/SSD-Reliability-Lifetime-White-Paper.pdf, 2008

[9] A. Oprea and A. Juels, "A clean slate look at disk scrubbing", http://www.rsa.com/rsalabs/staff/bios/aoprea/publications/scrubbing.pdf

[10] G. Wang, A. Raza Butt and C. Gniady, "On the Impact of Disk Scrubbing on Energy Savings", HotPower, 2008

[11] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy and J. Schindler, "An analysis of latent sector errors in disk drives", In Proceedings of the 2007 SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 2007

[12] M. Baker, M. Roussopoulos, M. Shah, P. Maniatis, P Bungale, T. Giuli and D. Rosenthal, "LOCKSS: A Peer-to-Peer Digital Preservation System", ACM Transactions on Computer Systems (TOCS), 2005, Vol. 23, Issue 1, pp. 2-50

[13] E. Pinheiro, W. Weber and L. Barroso, "Failure trends in a large disk drive population", FAST'07: Proceedings of the 5th conference on USENIX Conference on File and Storage Technologies, 2007

[14] B. Schroeder and G. Gibson, "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?", 5th USENIX Conference on File and Storage Technologies, 2007

[15] T. Schwarz, Q. Xin, A. Hospodor and S. Ng, "Disk scrubbing in large archival storage systems", In Proceedings of the 12th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '04), 2004

[16] J. Kubiatowicz, D. Bindel, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage", http://oceanstore.cs.berkeley.edu

[17] M. Baker, M. Roussopoulos, M. Shah and P. Maniatis, P. Bungale and T. Giuli, D. Rosenthal, "A Fresh Look at the Reliability of Long-term Digital Storage", Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems, 2006

[18] J.F. Murray, G.F. Hughes and K. Kreutz-Delgado, "Hard drive failure prediction using non-parametric statistical methods", Proceedings of ICANN/ICONIP, 2003