Disk Scrubbing in Large Archival Storage Systems

Thomas J. E. Schwarz, S. J.[1,2 †] Qin Xin[2,3 ‡] Ethan L. Miller[2 ‡] Darrell D. E. Long[2 ‡]
Andy Hospodor[1] Spencer Ng[3]

[1]*Computer Engineering Department, Santa Clara University, Santa Clara, CA 95053*
[2]*Storage Systems Research Center, University of California, Santa Cruz, CA 95064*
[3]*Hitachi Global Storage Technologies, San Jose Research Center, San Jose, CA 95120*

Abstract

Large archival storage systems experience long periods of idleness broken up by rare data accesses. In such systems, disks may remain powered off for long periods of time. These systems can lose data for a variety of reasons, including failures at both the device level and the block level. To deal with these failures, we must detect them early enough to be able to use the redundancy built into the storage system. We propose a process called "disk scrubbing" in a system in which drives are periodically accessed to detect drive failure. By scrubbing all of the data stored on all of the disks, we can detect block failures and compensate for them by rebuilding the affected blocks. Our research shows how the scheduling of disk scrubbing affects overall system reliability, and that "opportunistic" scrubbing, in which the system scrubs disks only when they are powered on for other reasons, performs very well without the need to power on disks solely to check them.

1. Introduction

As disks outpace tapes in capacity growth, large scale storage systems based on disks are becoming increasingly attractive for archival storage systems. Such systems will encompass a very large number of disks and store petabytes of data (10^{15} bytes). Because the disks store archival data, they will remain powered off between accesses, conserving power and extending disk lifetime. Colarelli and Grunwald [2] called such a system a Massive Array of mainly Idle Disks (MAID).

In a large system encompassing thousands or even tens of thousands of active disks, disk failure will become frequent, necessitating redundant data storage. For instance, we can mirror disks or mirror blocks of data, or even use higher degrees of replication. For better storage efficiency, we collect data into large *reliability blocks*, and group m of these large reliability blocks together in a *redundancy group* to which we add k parity blocks. We store all reliability blocks in a redundancy group on different disks. We calculate the parity blocks with an erasure correcting code. The data blocks in a redundancy group can be recovered if we can access m out of the $n = m + k$ blocks making up the redundancy group, making our system k-available.

Our redundancy scheme implies that the availability of data depends on the availability of data in another block. The extreme scenario is one where we try to access data, discover that the disk containing the block has failed or that a sector in the block can no longer be read, access the other blocks in the redundancy group, and then find that enough of them have failed so that we are no longer able to reconstruct the data, which is now lost. In this scenario, the reconstruction unmasks failures elsewhere in the system.

To maintain high reliability, we must detect these masked failures early. Not only do we have to access disks periodically to test whether the device has failed, but we also have to periodically verify all of the data on it [6, 7], an operation called "scrubbing." By merely reading the data, we verify that we can access them. Since we are already reading the data, we should also verify its accuracy using signatures—small bit strings calculated from a block of data. We can use signatures to verify the coherence between mirrored copies of blocks or between client data and generalized parity data blocks. Signatures can also detect corrupted data blocks with a tiny probability of error. Our work investigates the impact of disk scrubbing through analysis and simulation.

† Supported in part by an SCU-internal IBM research grant.
‡ Supported in part by Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratory under contract B520714.

2. Disk Failures

Storage systems must deal with defects and failures in rotating magnetic media. We can distinguish three principal failure modes that affect disks: block failures, also known as media failures; device failures; and data corruption or unnoticed read error.

On a modern disk drive, each block consisting of one or more 512 B sectors contains error correcting code (ECC) information that is used to correct errors during a read. If there are multiple errors within a block, the ECC can either successfully correct them, flag the read as unsuccessful, or in the worst case, mis-correct them. The latter is an instance of data corruption, which fortunately is extremely rare. If the ECC is unable to correct the error, a modern disk retries the read several times. If a retry is successful, the error is considered a soft error; otherwise, the error is hard. Failures to read data may reflect physical damage to the disk or result from faulty disk operations. For example if a disk suffers a significant physical shock during a write operation, the head might swing off the track and thus destroy data in a number of sectors adjacent to the one being written.

2.1. Block Failures

Many block defects are the result of imperfections in the machining of the disk substrate, non-uniformity in the magnetic coating and contaminants within the head disk assembly. These manufacturing defects cause repeatable hard errors detected during read operations. Disk drive manufacturers attempt to detect and compensate for these defects during a manufacturing process called self-scan. Worst-case data patterns are written and read across all surfaces of the disk drive to form a map of known defects called the P-list. After the defect map is created (during self-scan), the drive is then formatted in a way that eliminates each defect from the set of logical blocks presented to the user. These manufacturing defects are said to be "mapped-out" and should not affect the function of the drive.

Over the lifetime of a disk, additional defects can be encountered. Additional spare sectors are reserved at the end of each track or cylinder for these grown defects. A special *grown defect list*, or G-List, maps the grown defects and complements the P-List. In normal operation, the user should rarely encounter repeatable errors. An increase in repeatable errors, and subsequent grown defects in the G-List, can indicate a systemic problem in the heads, media, servo or recording channel. Self Monitoring Analysis and Reporting Technology (SMART) [13] allows users to monitor these error rates.

Note that a drive only detects errors after reading the affected blocks; a latent error might never be detected on an unread block. Therefore, scrubbing and subsequent reconstruction of failed blocks is necessary to ensure long-term data viability.

2.2. Disk Failure Rates

Some block failures are correlated. For example, if a particle left over from the manufacturing process remains within the disk drive, then this particle will be periodically caught between a head and a surface, scratching the surface and damaging the head [3]. The user would observe a burst of hard read failures, followed eventually by the failure of the drive. However, most block failures are not related, so we can model block failures with a constant failure rate λ_{bf}.

Device failure rates are specified by disk drive manufacturers as MTBF values. The actual observed values depend in practice heavily on operating conditions that are frequently worse than the manufacturers' implicit assumptions. Block failure rates are published as failures per number of bits read. A typical value of 1 in 10^{14} bits for a commodity ATA drive means that if we access data *under normal circumstances* at a rate of 10 TB per year, we should expect one uncorrectable error per year.

Block errors may occur even if we do not access the disk. Lack of measurements, lack of openness, and most importantly, lack of a tested failure model make conjectures on how to translate the standard errors per bits rate into a failure rate per year hazardous. Based on interviews with data storage professionals, we propose the following model:

We assume that for server drives, about 1/3 of all field returns are due to hard errors. RAID system users, who buy 90% of the server disks, do not return drives with hard errors. Thus, 10% of the disks sold account for one third of hard errors reported. Therefore, the mean-time-to-block-failure is 3/10 times the MTBF of all disk failures and the mean-time-to-disk-failure is 3/2 of the MTBF. If a drive is rated at 1 million hours, then the mean-time-to-block-failure is 3×10^5 hours and the mean-time-to-disk-failure is 1.5×10^6 hours. Thus, we propose that block failure rates are about five times higher than disk failure rates.

3. System Overview

We investigate the reliability of a very large archival storage system that uses disks. We further assume that disks are powered down when they are not serving requests, as in a MAID [2]. Disk drives are not built for high reliability [1] and any storage system containing many disks needs to detect failures actively. In a MAID, this may involve powering on disks that have been idle for a long time.

3.1. Redundant Data Storage

In order to protect against data loss, we incorporate two different defensive strategies. First, as in any large storage system, we store data redundantly. The simplest method is to store several copies of data, but this also multiplies the physical storage needs. An archival storage system would most likely employ a scheme based on error correcting codes (ECCs), in which we group m data items with an additional k parity items in a *redundancy group*. Using the ECC, we can access all data stored in a redundancy group when only m out of the $n = m + k$ items are available. The items can be complete disks or they can be subregions of disks. We call the latter *redundancy blocks* or *r-blocks*. The r-blocks in a redundancy group are placed on different disks so that a single device failure only affects one r-block per redundancy group. An r-block usually consists of many individual disk blocks. Previous research [15] discusses the advantages and disadvantages of using complete disks or smaller r-blocks for reliability and storage system operations.

As an example, we can imagine a MAID with a steady write load. As data arrives for storage, we break it into r-blocks of fixed size, perhaps a few megabytes, and add parity r-blocks to form redundancy groups. In such a system, only a few of the thousands of disks in the storage system are powered on at any time because of read operations. We store the r-blocks on these active disks. In order to distribute load, we want to spread newly formed redundancy groups over all the disks in the system, but new read requests will presumably change the set of active disks.

3.2. Disk Scrubbing

We propose the use of *disk scrubbing* to detect failed disk sectors. The most basic version of disk scrubbing just reads all the data in a certain region, a *scrubbing block* or *s-block*. As with r-blocks, s-blocks contain many individual disk blocks, and could even encompass an entire disk.

If a disk sector suffers a failure, the internal ECC on the disk sector flags the sector as unreadable, but only when the sector is read. Whether we detect a sector failure during a normal read operation or during scrubbing, we use the redundancy in the storage system to recover the data and either rewrite the recovered contents in place or elsewhere if the disk sector suffered physical damage. If the failure is found while we are trying to repair another drive or sector failure, we might already have lost data.

3.3. Content Verification through Signatures

Disk scrubbing verifies that all the sectors in an s-block are accessible by simply reading them. We can also use a *signature scheme*, explained in more detail below, to verify the contents, protecting against rare instances of data corruption on the storage device as well as software failures. The most important instance of such failures are client data / parity data incoherence. In our system, we might on rare occasions update a block. The signature scheme detects whether the update changed all the parity blocks or mirrored blocks in the system. If not, then the mechanism to restore data will no longer work. If s-blocks encompass one or more r-blocks, then we can use our signature scheme to check whether parity blocks accurately reflect the contents of s-blocks.

3.4. Signatures for Disk Scrubbing

In our more elaborate scheme, we maintain an f-bit signature of the contents of an s-block in a data base. The signature scheme has the following characteristics:

1. The probability that two random s-blocks have the same signature is (or close to) the optimal value 2^{-f}.
2. If an s-block changes slightly then the signature detects this change, *i. e.*, the signature changes.
3. If we change a small portion of an s-block, then we can calculate the new signature from the old one and the change.
4. We can calculate the signature of a block composed of sub-blocks from the signature of the sub-block.
5. We can calculate the signature of an r-block containing parity data from the signatures of the data r-blocks in the same redundancy group.

Many signature schemes fulfill properties 1 and 2. Property 3 is related to the notion of *composable hash functions* [14]. We use the "algebraic signature" proposed by Litwin and Schwarz [8, 9] that has all the properties. However, according to Schwarz [11], the "algebraic signature" only has property 5 for Reed-Solomon codes [10], a convolutional array code, or a code using XOR to calculate parity as in Hellerstein, *et al.* [4].

3.5. Disk Scrubbing Operations

We periodically scrub an s-block by reading it into the drive buffer. A modified device driver allows each disk block of an s-block to be read directly into the buffer of the disk and validated based on the internal ECC. If the drive cannot validate a disk block, then we use redundant storage to recover the data and store it again on the device. Optionally, during some scrubs, we read all the data in an s-block, recalculate its signature, and compare the result with the signature maintained as metadata. This step validates the contents of the s-block, whereas disk scrubbing only insures that we can read the individual disk blocks that comprise an s-block.

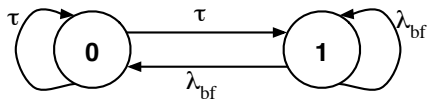


Figure 1. Markov Model for random scrubbing, constant rate block failures.

4. Modeling the Effect of Disk Scrubbing

We now investigate the impact of disk scrubbing on the probability that a disk contains one or more failed blocks. We model block failure as a Poisson process, where a disk block goes bad at a rate of λ_{bf}, $0 < \lambda_{bf} < 1$. We distinguish between three scrubbing disciplines. In *random scrubbing*, we scrub an s-block at random times, though with a fixed mean time between scrubs. *Deterministic scrubbing* of an s-block happens at fixed time intervals. Finally, *opportunistic scrubbing* piggy-backs as much as possible on other disk operations to avoid additional power on cycles.

4.1. Random Scrubbing

We model random scrubbing as a simple Markov model with two different states and the transitions depicted in Figure 1. State 0 models the situation without block failures, and State 1 depicts the state where the scrubbing block contains a failed block. p is the probability that the system is in State 0; and τ is the mean scrubbing rate. The system is in balance if the flow into State 0 equals the flow out of it, when $\lambda_{bf} \cdot p = \tau \cdot (1-p)$. The equilibrium condition is equivalent to

$$p = \frac{\tau/\lambda_{bf}}{1+\tau/\lambda_{bf}}.$$

The probability that the disk has a bad block when we access it at random t is simply $P_{failure} = 1/(1+\tau/\lambda_{bf})$. As expected, this probability depends only on the ratio of scrubbing rate to failure rate, not on their absolute values. If the size of the scrubbing blocks is doubled, the scrubbing rate τ should also be doubled to keep the ratio constant. In this way, the probability of a block failure does not change, making the effect of scrubbing independent of the size of the scrubbing area.

4.2. Deterministic Scrubbing

We assume that we scrub parts of the disk regularly. We call the time between scrubbing operations T. Assume that the scrubbing area consists of N blocks and that each block suffers bit-rot or is corrupted at a constant rate λ_{block}. Then the probability $F(t)$ that at least one block is corrupted before time t is $F(t) = 1 - \exp(-N \cdot \lambda_{block} \cdot t) = 1 - exp(-\lambda_{bf} \cdot t)$, where we set $\lambda_{bf} = N \cdot \lambda_{block}$ to be the scrubbing area block failure rate.

We can now compute the probability $P_{failure}$ that, at random time t, we find that the area contains at least one failed block, despite scrubbing with interval T. Since our random time t needs to fall into one of the scrubbing intervals, we can assume that it falls with uniform probability in the interval $[0,T]$. The probability density of t is then $T^{-1}dt$. The chance of suffering a failure at time t is given by $F(t)$. To compare with our previous result, we write $\tau = T^{-1}$. Thus

$$\begin{aligned} P_{failure} &= \int_0^T (1-\exp(-\lambda_{bf} \cdot t) T^{-1} dt \\ &= 1 - (\tau/\lambda_{bf})(1-\exp(-\lambda_{bf}/\tau)) \end{aligned}$$

Again, as has to be the case, the probability only depends on the ratio τ/λ_{bf}. As Figure 2 shows, deterministic scrubbing always outperforms random scrubbing. In Figure 2, we choose τ/λ_{bf} to be between 0.5 and 100 and give the probability of finding a block in error at a random time t. As Figure 2 shows, most of the benefits of scrubbing accrue at rather low ratios. Since block failure rates are measured at least in hundreds of thousands of hours, it appears that scrubbing, while important, need be done only rarely. However, as we will see, the system mean time between data loss can be very sensitive to small changes in the probability of reconstruction so that aggressive scrubbing is needed for optimal reliability.

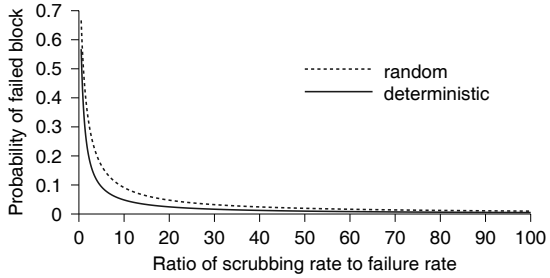


Figure 2. Failure probability for a single scrubbing region for fixed and random scrubbing intervals.

4.3. Opportunistic Scrubbing

Since disk power-ups negatively affect disk drives (unless of course the disk has not been powered up for a long time), we want to avoid the mandatory power-ups of a deterministic scrub. For this reason, we explored a third scrubbing strategy, *opportunistic scrubbing*. In this strategy, we set a scrub interval target, and maintain the time

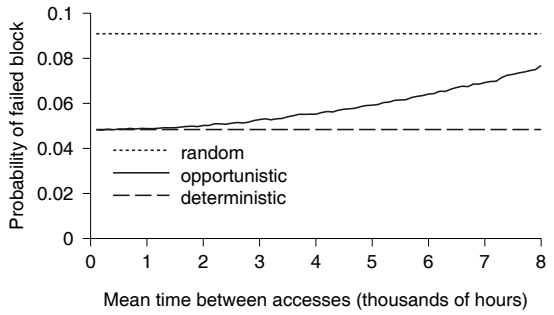


Figure 3. Failure probability for a single scrubbing region under opportunistic scrubbing. The average scrub interval is 10^4 hours and the block MTBF is 10^5 hours.

since the last scrub of the disk portion. When the disk is accessed, the portions of the disk for which the scrubbing interval was exceeded are scrubbed immediately. If we access disks rather frequently, then the actual scrubbing interval is very likely to be close to the scrubbing interval target with relatively low variation. If disks are accessed less frequently, then the variation of the actual scrub intervals is much larger and the probability of having a block failure is closer to that obtained with random scrubbing.

Figure 3 confirms that opportunistic scrubbing can perform nearly as well as deterministic scrubbing. We used an average scrub time of 10^4 hours (about once per year) and a block MTBF of 10^5 hours. We varied the access rate between once per 10^2 hours (about 4 days) to once per 8×10^3 hours. In our model, the system's goal was to scrub an s-block every 10^4 hours, but it is unlikely that one of the random accesses to the disk would fall conveniently at the correct moment for scrubbing. Instead, the scrub is done at the first access that falls after 10^4 hours have elapsed since the last scrub. Actually, in order to maintain the exact mean time between scrubs, we scrub at the first access after 10^4 hours minus the average disk access time have elapsed. The difference in practice is minute, since we assume that disk accesses are much more frequent than scrubbing operations in order to save power-on hours. We see that the probability of having a failed disk stays for small MTBA close to the deterministic probability,but that at a value of 20% to 30%, the probability moves up towards the value for random scrubbing. We recall our earlier observation that the absolute values of the three numbers (MTBA, MTBS, MTBF) do not matter, but only their relation. As we move towards more realistic scrubbing intervals, the difference of the failure probability between random and deterministic scrubbing becomes much smaller.

Our model shows that opportunistic scrubbing gains most of the benefits of deterministic scrubbing with no additional power up / down cycles. In addition, it is not the size of the scrubbing area but but the rate at which each disk area is scrubbed that affects reliability. Of course, in many practical systems, cooling can be a problem and running a disk for the several hours to completely scrub it could lower the disk drive's reliability.

5. Power Cycling and Reliability

Turning a disk on and off has a significant impact on the reliability of the disk. This is especially true for commodity disks that lack techniques used by more expensive laptop disks to keep the read/write heads from touching the surface during power-down. Before a commodity disk powers down, it parks its heads over a specially textured area, typically near the spindle to enable the heads to break free from the surface during power-on. Even for laptop disks that ramp load their heads, the spin-up period is more likely to cause a head crash than normal disk operations.

Disk manufacturers are reluctant to publish actual failure rates because they depend so much on how disks are operated; large consumers of disk drives share this reluctance. We base our modeling on a report published by Seagate [1, 12]. The Mean Time Between Failure (MTBF) of disks is specified as a rate per operating hours. The MTBF value needs to be multiplied by a factor dependent on the expected Power On Hours (POH) of the disk. For example, if the POH is 492 hours per year (80 minutes per day), the MTBF increases by a factor of 2.05. For 2400 POH, the multiplier is one, and for 8760 h per year (the disk runs continuously), the multiplier is about 0.6. Because of the shape of this curve, we have to assume that these data reflect the same number of power on / off cycles per year.

Based on this assumption, we can now estimate the impact of power cycling on reliability. Since the disk does not remain powered off indefinitely, we can disregard any impact of the power-off time on the failure rate of the disk. Therefore, we are left with two failure causes: operation of the disk and power on/off. We capture the former with a constant failure rate λ_t and the latter with a constant failure rate per power up λ_p. Thus, the failures per year of a set of these disks are given by $\lambda_p \cdot N_p + \lambda_t \cdot t$, where N_p is the number of power-ups and t is the POH value. The multiplier is then given by $\lambda_p \cdot N_p/t + \lambda_t$. By using two of our Seagate values to solve for both λ_t and $\lambda_p \cdot N_p$, we obtain $\lambda_t \approx 0.53$, which means that the impact of power on / off is about the same order of magnitude than running the disks. Unfortunately, without knowing N_p it is impossible to estimate λ_p. While not stated in the Seagate report, we can assume that these numbers involve a single power on / off per day. If this is the case, our calculations suggest that a power on / off cycle is approximately equivalent to running the drive for eight hours in terms of drive reliability, though

this equivalence may vary with different drives. In a drive designed for power cycling, such as a laptop drive, the reliability effect of a power cycle would likely correspond to a much shorter operating time.

6. Finding the Optimal Scrubbing Interval

We now assemble our various models to determine optimal scrub time intervals. Since systems vary widely, we cannot derive generic prescriptions; rather, we present a methodology that can provide guidance in choosing scrubbing parameters. Our approach only uses system reliability as a metric. For example, we do not include the cost of replacement drives necessary due to aggressive scrubbing that increases the power-on hours of individual drives which then fail more frequently. Heat generation is also an issue. For example, scrubbing an archive consisting of 250 Seagate ST3200822A disks of 200 GB each generates 13,658 BTU per single scrub operation. If the system scrubs disk drives aggressively, it needs millions of BTU in cooling, or failing this, reduce the lifetime of disk drives considerably. We also neglect the consequences of the switches of power on and off for both random and deterministic scrubbing in our modeling, although we take the total power-on hours of disk drives into account. In our simulation part, we will show the impact of the power-on/off switches on system reliability and will present the advantage of opportunistic scrubbing.

Scrubbing increases system reliability because it lowers the possibility of encountering a block failure during a reconstruction. However, too frequent scrubbing lowers system reliability because it increases the operating hours of devices. As we will see, small changes in the probability of unmasking a block failure can have a great impact on system reliability. In general, we assume that block failures are aggressively repaired by powering up disks that contain data in the same redundancy group. In a very large system with a high degree of failure tolerance, we might instead adopt a lazy strategy where repair operations are piggy-backed on disk accesses, or even start data reconstruction only when a certain threshold of r-blocks are not available.

We construct standard Markov models of the systems, neglecting the failure mode that consists *only* of block failures. This is because the chances that an individual disk sector has failed are so minute that we need not worry about multiple block failures in the parallel sectors within the reliability group. Under this assumption, a block failure only comes into play when this block is needed in order to deal with a device failure. A spare drive is assumed to be available for reconstructing data of the failed driver. Under this assumption, we model the situation after a device failure by two possible transitions. The first transition is a second device failure that brings the system closer to data loss or even induces data loss. The second transition is the repair transition which depends on the availability of all needed blocks. If these are not available, then we have a data loss.

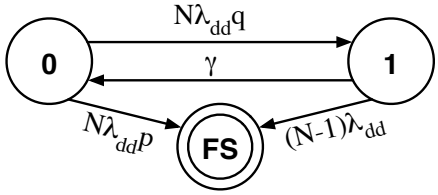


Figure 4. Markov model for a two-way mirroring system with N disk drives.

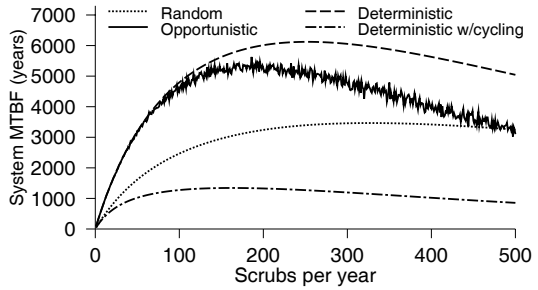
In our first example, we study a system that uses declustered mirroring. The system contains $N = 250$ disk drives. We assume that two simultaneous drive failures lead to data loss. We can model this system with three states, as shown in Figure 4. State 0 models a system without device failure, and State 1 models a system with a device failure, but one from which we can recover, and State FS is the failure state. One of the N disks in the system fails with rate $N \cdot \lambda_{dd}$. With probability p, the mirrored copy—which we can treat as a virtual disk of the same size as a physical disk, distributed throughout the system—has a block failure on it, preventing recovery. This is a transition into the absorbing failure state. However, with probability $q = 1 - p$, the mirrored copy—or rather the blocks that make up a mirror of the failed device—is free from block errors and we can proceed with recovery. From State 1, we have a repair transition to State 0 taken with rate γ, which is set by the length of time needed to do distributed recovery. Another failure in State 1 leads to data loss; thus, there is a transition from State 1 to the failure state taken at rate $(N-1) \cdot \lambda_{dd}$.

The device failure rate λ_{dd} and probability p that the distributed mirror has a block failure are determined by the intensity of scrubbing operations. Following [12], we assume that $\lambda_{dd} = \mu(t) \cdot \lambda_0 \cdot t$, where t is the power-on hours per year, $\mu(t)$ is the scaling function in MTTF of disk drives, and λ_0 is the specified rate of device failures per year, with the result measured in years. If we use *opportunistic scrubbing*, then we do *not* pay a penalty for powering up the disk, but instead run the disk longer. If we scrub disks at a rate of τ (measured in the number of scrubs per year), and if it takes S hours to completely read a disk, then we have $S \cdot \tau$ additional power-on hours per year for disk scrubbing. We simplify by assuming that μ remains constant, resulting in a failure rate with scrubbing of $\lambda_{dd} = \mu \cdot \lambda_0 \cdot (t + S \cdot \tau)$.

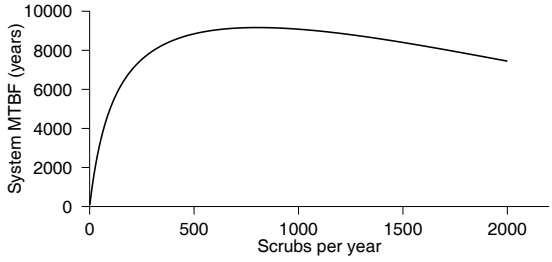
For our experiment, we chose a device failure rate of $\lambda_{dd} = 5 \times 10^{-5}$ hours and a block failure rate of 10^{-5} hours, and set $S = 4$ hours. The results of this model are shown in Figure 5(a); the middle line is the result of opportunis-

tic scrubbing. Since we did not have a functional expression of the probability of a block failure at a random moment, we used simulation to estimate the number, resulting in a non-smooth curve. We obtain lower and upper limits by calculating system MTBF for random and deterministic scrubbing, *excluding the costs of power-ons into account*. Figure 5(a) shows that, if few scrubs are needed each year, opportunistic scrubbing functions just as deterministic scrubbing, since there are many accesses on which to piggy-back a scrub operation. The graph also shows the large impact of block failure probability, and suggests almost daily disk scrubbing for highest reliability.

When the cost of power-cycling is included, however, deterministic scrubbing has much lower reliability. Assuming that each power on operation has the same impact on device reliability as running the disk for eight hours dramatically reduces system reliability with deterministic scrubbing, as the "deterministic w/cycling" curve in Figure 5(a) shows. Opportunistic scrubbing does not require additional disk power-on events, so system MTBF is unaffected even if power-on cycles are considered.



(a) Mirrored reliability blocks.



(b) Mirrored disks using opportunistic scrubbing.

Figure 5. System MTBF for different storage system configurations.

In an archival system, we might forego distributed mirroring for the simplicity of dedicated mirrors. Our second example models a system of the same size, in which entire disks rather than reliability blocks are paired; the resulting model can be solved explicitly. If a disk failure is discovered, then we can recover from it under two conditions: the mirror disk does not fail until we have reconstructed all the data on a spare disk, or the mirror disk does not have a block failure. The probability p_{mf} that the mirror disk fails during the reconstruction time T_{rec} is given by $p_{mf} = \lambda_{dd}(t) \cdot T_{rec}$. The device failure rate λ_{dd} itself is given by the number t of power-on-hours, the time it takes to scrub (read) a disk S and the rate τ of scrubbing, resulting in $\lambda_{dd} = \mu \cdot \lambda_0 \cdot (t + S \cdot \tau)$. We already calculated the probability p_{bf} that a disk has one or more block failures, depending on the scrubbing discipline. As long as we use opportunistic scrubbing and piggyback scrubbing operations onto accesses to disks, and as long as we scrub significantly less than we access the disk, so that we can assume opportunistic scrubbing to be like deterministic scrubbing, we have

$$p_{bf} = 1 - (\tau/\lambda_{bf})(1 - \exp(-\lambda_{bf}/\tau)$$

. The probability P that we cannot recover the disk is then $P = p_{mf} + p_{bf} - p_{mf} \cdot p_{bf}$, and system MTBF is $MTBF_{system} = 1/N \cdot \lambda_{df} \cdot P$. For a system with parameters similar to those used in Figure 5(a) but using disk mirroring, the system MTBF is nearly twice as large and the optimal scrubbing rate is close to three times per day, as Figure 5(b) shows. This rate is clearly too high, since, at this rate, opportunistic scrubbing is no longer performing as well as deterministic scrubbing.

7. Simulation Results

In this section we present our results from an event-driven simulation. We compared the data loss occurrence under various disk scrubbing schemes in an archival storage system and showed how scrubbing rates impact system reliability.

7.1. System Parameters

We simulated a disk-based storage system that contains one petabyte ($1\,\mathrm{PB} = 10^{15}B$) of archival data. We assume that disk drives have an MTTF of 10^5 hours and are replaced after reaching their retirement age. We distinguish block error and device erasure failure on hard disk drives, as described in Section 2. We consider the effect of power-on hours on disk reliability using the model described by Anderson, *et al.* [1]. We assume that data loss is only noticed when that piece of data is accessed.

Data is distributed across 10,000 disk drives evenly using 10 GB reliability blocks; the placement algorithm [5] guarantees that the original data and its replicas or parity blocks will never be put on the same drive. We assume user data traffic is about 1 TB/day; data access comes not

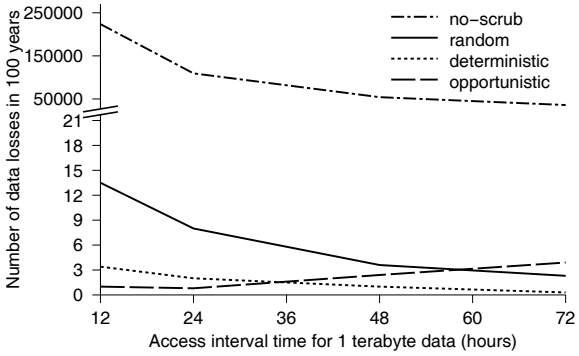
only from the user activity such as reads/writes, but also from internal maintenance. We considered two different redundancy schemes: two-way mirroring and RAID 5. For a large-scale data archival system, mirroring is expensive in terms of storage cost, although it is simple in implementation and requires fewer disks to be powered on during a data access. We assume that data on a failed disk will be reconstructed to a spare disk. Since the main concern in our system is the power-on hours of disk drives, not bandwidth, we do not use distributed recovery schemes.

7.2. Comparison of Data Scrubbing Schemes

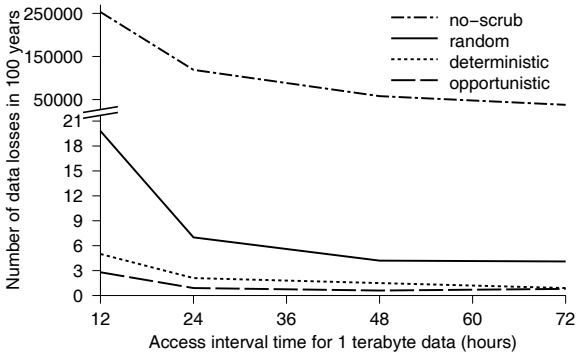
We first explored the three data scrubbing schemes—random, deterministic, and opportunistic. Figure 6 shows the number of data losses for the one petabyte archival system over 100 years under various scrubbing schemes, and compares this to the case when no scrubbing is done. Please note that the designed lifetime of a disk drive is about 5–7 years, so it requires disk replacement for many times during the simulated time. Two data redundancy schemes are configured: two-way mirroring, shown in Figure 6(a) and RAID 5 across four disk drives, shown in Figure 6(b). We vary the data access frequency to emulate the data "temperature" in such a system. The simulations graphed in Figure 6 assume that one terabyte of data is accessed in 12, 24, 48 and 72 hours. Note that only 1%–3% of the disk drives need be powered on to access one terabyte data. We set up the Mean-Time-To-Scrub of a single disk drive as about three times per year for random and deterministic schemes. For the opportunistic scheme, we set the ratio between data access rate and disk scrub rate so that each disk drive would be scrubbed no more than 3 times per year. Under opportunistic scrubbing, disks are only scrubbed when they are already on because of another data access, reducing the switches between power-on and off.

When no scrubbing is done ("no-scrub" in Figure 6), there is a great deal of data loss over 100 years. Since block errors accumulate as time goes by, it is likely the block errors will prevent data recovery when an entire disk fails. The number of data losses drops greatly when disk drives are scrubbed under various schemes, demonstrating the importance of scrubbing on system reliability.

We observed that, in most configurations, the occurrence of data loss decreases when the rate of user data accesses decreases, except for the opportunistic scrubbing when data access becomes very infrequent for the two-way mirroring redundancy configuration. This is because the chance for disk scrubbing decreases as we only execute scrubbing when data gets accessed. We can see that random scrubbing performs the worst of the three scrubbing schemes and that the opportunistic scheme provides high reliability when data access is relatively frequent, but the number of data losses increases when data is accessed infrequently. We found that the likelihood of data loss is a little higher under RAID 5 as compared to two-way mirroring when disk scrubbing is used because of the difference in storage efficiency of two-way mirroring and RAID 5. Our current comparisons are based on a fixed *number* of disk drives rather than on fixed *user capacity*; fewer disk drives are required for RAID 5 than two-way mirroring to provide the same usable capacity. We plan to compare these two assumptions in our future work.



(a) Two-way mirroring.



(b) RAID 5 redundancy.

Figure 6. Occurrences of data loss over a 100 year period for different access frequencies under various data scrubbing schemes.

7.3. Disk Scrubbing Frequency

We have shown that opportunistic scrubbing is the most attractive among the three mechanisms when data access is relatively frequent. For systems where data is infrequently accessed, we must power disks on periodically to scrub them in addition to doing scrubbing when the drive is accessed normally, resulting in a scrubbing process that is a mix of the opportunistic and deterministic schemes. In such

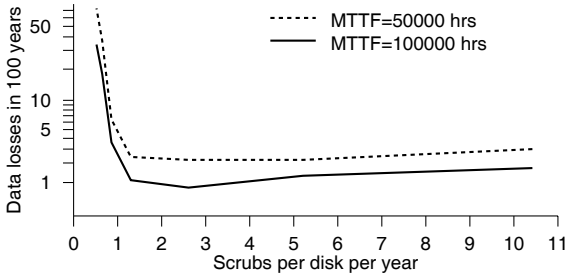


Figure 7. The number of data losses over 100 years under opportunistic disk scrubbing with different scrub rates for two different disk MTTF values.

a system, how frequently should drives be scrubbed? The chance of data loss in a large-scale archival system depends heavily on the rate at which disk drives are scrubbed. More frequent scrubbing promptly detects errors at the cost of long power-on hours and more power cycles. In our simulations, we fixed the data access rate at one terabyte per 24 hours, which implies that each disk drive would be accessed three times per year on average.

In Figure 7, we show the number of data losses with the varied scrubbing rate in 100 years. We also compare two types of drives with MTTF of 100,000 and 50,000 hours. We see a sharp drop in the likelihood of data loss when a disk drive gets scrubbed at least once per year. When the rates of data access and disk scrubbing are equivalent on a disk drive, *i. e.*, three times per year, the chance to of data loss is fairly low: it occurs about one time for disks with 100,000 hour MTTF and two times for those with 50,000 hour MTTF. Interestingly, we did not observe a further decrease in data loss as we increased the scrubbing frequency from three to eleven times per year; rather, a slight *increase* in data loss was noticed. This phenomenon comes from the POH (power-on-hour) effect on drive reliability. Aggressive scrubbing requires more power cycles, adversely affecting drive reliability. For systems designers, it is important to consider the effect of power-on/off and to measure tradeoffs between too little and too much scrubbing when they make the decision of the scrubbing frequency.

8. Conclusions and Future Work

We studied the impact of disk scrubbing on a large archival storage system, showing that scrubbing is essential to long-term data survival. We established a methodology to model disk scrubbing and showed by example that disk scrubbing is an important technique to protect against the impact of block failures. We examined three different disk scrubbing techniques—random, deterministic, and opportunistic—and showed via both modeling and simulation that opportunistic scrubbing is the most attractive scheme because it does not power on disk drives solely to check them, instead using "normal" power on periods to scrub the drives. We also explored the frequency of disk scrubbing and showed that the power-on-hour effect has a significant impact on overall reliability in large archival storage systems.

Our research has established that disk scrubbing is an important tool for system designers as petabyte-scale long-term archives begin to use disks rather than tapes. However, there is still much to be done in exploring the use of disk scrubbing for long-term archiving. For example, are there additional scrubbing policies that may do even better than the three simple policies we have proposed? Are there storage system changes, such as different file systems or redundancy techniques, that might make scrubbing easier or more efficient? Additionally, our analysis has been hampered by the lack of exact numbers; we hope that this study gives impetus to a further analysis of disk drives in various settings. By developing more detailed models, validating them with practical experiences, and developing new techniques for better, more efficient disk scrubbing, we hope to provide long-term archive designers with the tools they need to ensure that data is never lost because of media or device failure.

References

[1] D. Anderson, J. Dykes, and E. Riedel. More than an interface–SCSI vs. ATA. In *Proceedings of the 2003 Conference on File and Storage Technologies (FAST)*, San Francisco, CA, Mar. 2003.

[2] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (SC '02)*, Nov. 2002.

[3] J. G. Elerath and S. Shah. Disk drive reliability case study: dependence upon head fly-height and quantity of heads. In *Proceedings of the 2003 Annual Reliability and Maintainability Symposium*, pages 608–612. IEEE, 2003.

[4] L. Hellerstein, G. A. Gibson, R. M. Karp, R. H. Katz, and D. A. Patterson. Coding techniques for handling failures in large disk arrays. *Algorithmica*, 12:182–208, 1994.

[5] R. J. Honicky and E. L. Miller. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *Proceedings of the 18th International Parallel & Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, Apr. 2004. IEEE.

[6] H. H. Kari. *Latent Sector Faults and Reliability of Disk Arrays*. PhD thesis, Helsinki University of Technology, 1997.

[7] H. H. Kari, H. Saikkonen, and F. Lombardi. On the methods to detect latent sector faults of a disk subsystem. In *1st*

International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '93), pages 317–322, Jan. 1993.

[8] W. Litwin, R. Mokadem, and T. Schwarz. Disk backup through algebraic signatures in scalable and distributed data structures. In *Proceedings of the 5th Workshop on Distributed Data and Structures (WDAS 2003)*, Thessaloniki, Greece, 2003.

[9] W. Litwin and T. Schwarz. Algebraic signatures for scalable, distributed data structures. In *Proceedings of the 20th International Conference on Data Engineering (ICDE '04)*, pages 412–423, Boston, MA, 2004.

[10] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software—Practice and Experience (SPE)*, 27(9):995–1012, Sept. 1997. Correction in James S. Plank and Ying Ding, Technical Report UT-CS-03-504, U Tennessee, 2003.

[11] T. Schwarz. Verification of parity data in large scale storage systems. In *Parallel and Distributed Processing Techniques and Applications (PDPTA '04)*, Las Vegas, NV, June 2004.

[12] Seagate. Estimating drive reliability in desktop computers and consumer electronic systems. http://www.digit-life.com/articles/storagereliability/, 2004.

[13] Self-Monitoring, Analysis and Reporting Technology (SMART) disk drive monitoring tools. http://sourceforge.net/projects/smartmontools/.

[14] T. Suel, P. Noel, and D. Trendafilov. Improved file synchronization techniques for maintaining large replicated collections over slow networks. In *Proceedings of the 20th International Conference on Data Engineering (ICDE '04)*, pages 153–164, Boston, MA, 2004.

[15] Q. Xin, E. L. Miller, T. J. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. Reliability mechanisms for very large storage systems. In *Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 146–156, Apr. 2003.