Keeping Bits Safe: How Hard Can It Be?

David S. H. Rosenthal
LOCKSS Program, Stanford University Libraries, CA 94305

1 Introduction

These days, we are all data pack rats. Storage is cheap, so if there's a chance the data could possibly be useful, we keep it. And we know that storage isn't completely reliable, so we keep backup copies as well. But the more data we keep, and the longer we keep it, the greater the chance that some of it will be unrecoverable when we need it.

There is an obvious question we should be asking - how many copies in storage systems with what reliability do we need to get a given probability that the data will be recovered when we need it? This may be an obvious question to ask, but it is a surprisingly hard question to answer. Let's look at the reasons why.

To be specific, let's suppose we need to keep a petabyte for a century, and have a 50% chance that every bit will survive undamaged. This may sound like a lot of data and a long time, but there are already data collections bigger than a petabyte that are important to keep forever. The Internet Archive is already multiple petabytes.

The state of our knowledge about keeping bits safe can be summarized as:

- *The more copies the safer.* As the size of the data increases, the per-copy cost increases, reducing the number of backup copies that can be afforded.
- *The more independent the copies the safer.* As the size of the data increases, there are fewer affordable storage technologies. Thus the number of copies in the same storage technology increases, decreasing the average level of independence.
- *The more frequently the copies are audited the safer.* As the size of the data increases, the time and cost needed for each audit to detect and repair damage increases, reducing their frequency.

2 Claims

At first sight, keeping a petabyte for a century isn't hard. Storage system manufacturers make claims for their products that far exceed the reliability we need. For example, Sun claimed that their ST5800 "Honeycomb" product had a mean time to data loss (MTTDL) of 2.4×10^6 years."[1] [41]. Off-the-shelf solutions appear so reliable that backups are unnecessary.

Should we believe these claims? Where do they come from?

Before using Sun's claim for its ST5800 as an example, I should stipulate that the ST5800 was an excellent product. It represented the state of the art in storage technology, and Sun's marketing claims represent the state of the art in storage marketing. Nevertheless, Sun did not guarantee that data in the ST5800 would last 2.4×10^6 years. Sun's terms and conditions explicitly disclaim any liability whatsoever for loss of, or damage to, the data the ST5800 stores [40] whenever it occurs.

All that Sun was saying is that if you watched a large number of ST5800 systems for a long time, recorded the time at which each of them first suffered a data loss, and then averaged these times, the result would be 2.4×10^6 years. Suppose Sun watched 10 ST5800s and noticed that three of them lost data during the first year, four of them lost data after 2.4×10^6 years, and the remaining three lost data after 4.8×10^6 years, they would be correct that the MTTDL was 2.4×10^6 years. But we would not consider that a system with a 30% chance of data loss in the first year was adequate to keep a petabyte safe for a century. A single MTTDL number isn't a useful characterization of a solution.

Let's look at the slightly more scientific claim made at the recent launch of the SC5800 by the marketing department of Sirius Cybernetics[2]: "SC5800 has a MTTDL

[1] Numbers are expressed in powers-of-ten notation to help readers focus on the scale of the problems and the extraordinary level of reliability required.

[2] Purveyors of chatty doors, existential elevators and paranoid an-

of $(2.4 \pm 0.4) \times 10^6$ years". Sirius implictly assumes the failures are normally distributed and thus claims that about $2/3$ of the failures would occur between 2.0×10^6 and 2.8×10^6 years after the start of the experiment. They didn't start watching a batch of SC5800s 2.8 million years ago. So how would they know?

Sirius says they will sell 2×10^4 SC5800s per year at $\$5 \times 10^4$ each (a billion-a-year business), and they expect the product to be in the market for 10 years. The SC5800 has a service life of 10 years. So if Sirius watched their entire production of SC5800s ($\$10^{10}$ worth of storage systems) over their entire service life the experiment would end 20 years from now after accumulating about 2×10^6 system-years of data. If their claim is correct they would have about a 17% chance of seeing a single data loss event.

In other words, Sirius claims that the probability that *no SC5800 will ever lose any data* is over 80%. Or, since each SC5800 stores 5×10^{13} bytes, that there is an 80% probability that 10^{19} bytes of data will survive 10 years undamaged.

If one could believe Sirius' claim, the petabyte would look pretty safe for a century. But the claim clearly isn't based on an experiment that, even if Sirius were to do it, would not provide results for 20 years and even when it did would not validate the number in question. In fact, claims like Sun's and Sirius' are not the result of experimentation at all. No feasible experiment could validate them. They are *projections*, based on models of how components of the system such as disks and software behave.

3 Models

The state of the art in this kind of modeling is exemplified by the Pergamum project at UC Santa Cruz [39]. Their model includes disk failures at rates derived from measurements [35, 30] and sector failures at rates derived from disk vendor specifications. Their system attempts to conserve power by spinning the disks down whenever possible; they make an allowance for the effect of doing so on disk lifetime but it isn't clear upon what they base this allowance. They report that the simulations were difficult:

> "This lack of data is due to the extremely high reliability of these configurations - the simulator modeled many failures, but so few caused data loss that the simulation ran very slowly. This behavior is precisely what we want from an archival storage system: it can gracefully handle many failure events without losing data. Even though we captured fewer data points for the triple inter-parity configuration, we believe the reported MTTDL is a reasonable approximation." [39]

Although the Pergamum team's effort to obtain "a reasonable approximation" to the MTTDL of their system is praiseworthy, there are a number of reasons to believe that it overestimates the reliability of the system in practice:

- The model draws its failures from exponential distributions. They thus assume that both disk and sector failures are uncorrelated, although all observations of actual failures [5, 42] report significant correlations. Correlated failures greatly increase the probability of data loss [6, 13].
- Other than a small reduction in disk lifetime from each power-on event, they assume that failure rates observed in always-on disk usage translate to their mostly-off environment. A study [43] published after their paper reports a quantitative accelerated life test of data retention in almost-always-off disks. It shows that some of the 3.5" disks anticipated by the Pergamum team have data life dramatically worse in this usage mode than 2.5" disks using the same head and platter technology.
- They assume that disk and sector failures are the only failures contributing to the system failures, although a study [17] shows that other hardware components contribute significantly.
- They assume that their software is bug-free, despite several studies of file and storage implementations [20, 14, 31] that uniformly report finding bugs capable of causing data loss in all systems studied.
- They also ignore all other threats to stored data [34] as possible causes of data loss. Among these are operator error, insider abuse and external attack. Each of these has been the subject of anecdotal reports of actual data loss.

What can models like this tell us? Their results depend on both:

- the details of the simulation of the system being studied which, one hopes, accurately reflect its behavior, and
- the data used to drive the simulation which, one hopes, accurately reflect the behavior of the system's components.

Under certain conditions, it is reasonable to use these models to compare different storage system technologies. The most important condition is that the models of

droids to the nobility and gentry of this galaxy [1].

the two systems use the same data. A claim that modeling showed system A to be more reliable than system B when the data used to model system A had much lower failure rates for components such as disk drives would not be credible.

These models may well be the best tools available to evaluate different techniques for preventing data loss, but they aren't good enough to answer our question. We need to know the *maximum* rate at which data will be lost. The models assume things, such as uncorrelated errors and bug-free software, that all real-world studies show are false. The models exclude most of the threats to which stored data is subject. And in those cases where similar claims, such as those for disk reliability [35, 30], have been tested they have been shown to be optimistic. The models thus provide an estimate of the *minimum* data loss rate to be expected.

4 Metrics

Even if we believed the models, the MTTDL number doesn't tell us how much data was lost in the average data loss event. Is petabyte system A with a MTTDL of 10^6 years better than a similar size system B with a MTTDL of 10^3 years? If the average data loss event in system A loses the entire petabyte, where the average data loss event in system B loses a kilobyte, it would be easy to argue that system B was 10^9 times better.

Mean time to data loss is not a useful metric for how well a system stores bits through time, because it relates to time but not to bits. Nor is the Unrecoverable Bit Error Rate (UBER) typically quoted by disk manufacturers; it is the probability that a bit will be read incorrectly irrespective of how long it has been sitting on the disk. It relates to bits but not to time. Thus we see that we lack even the metric we would need to answer our question.

Let us over-simplify the problem to get a clearer picture. Suppose we had eliminated all possible sources of correlated data loss, from operator error to excess heat. All that remained would be "bit rot", a process that randomly flips the bits the system stores with a constant small probability per unit time. In this model we can treat bits like radioactive atoms, so that the time after which there is a 50% probability that a bit will have flipped is the "bit half-life".

The requirement of a 50% chance that a petabyte will survive for a century translates into a bit half-life of 8×10^{17} years. The current estimate of the age of the universe is 1.4×10^{10} years, so this is a bit half-life approximately 6×10^7 times the age of the universe.

This bit half-life requirement clearly shows how difficult the problem we have set ourselves is. Suppose we want to know whether a system we are thinking of buying is good enough to meet the 50% chance of keeping a petabyte for a century. Even if we are sublimely confident that every source of data loss other than "bit rot" has been totally eliminated, we still have to run a benchmark of the system's bit half-life to confirm that it is longer than 6×10^7 times the age of the universe. And this benchmark has to be complete in a year or so; it can't take a century.

So we take 10^3 systems just like the one we want to buy, write a petabyte of data into each so we have an exabyte of data altogether, wait a year, read the exabyte back and check it. If the system is just good enough, we might see 5 bit flips. Or, because "bit rot" is a random process, we might see more, or less. We would need either a lot more than an exabyte of data or a lot more than a year to be reasonably sure that the bit half-life was long enough for the job. But even an exabyte of data for a year costs 10 times as much as the system we want to buy.

What this thought-experiment tells us is that we are now dealing with such large numbers of bits for such a long time that we are never going to know whether the systems we use are good enough:

- The known causes of data loss are too various and too highly correlated for models to produce credible projections.
- Even if we ignore all those causes, the experiments that would be needed to be reasonably sure random "bit rot" is not significant are too expensive, or take too long, or both.

5 Measuring Failures

It wasn't until 2007 that researchers started publishing studies of the reliability that actual large-scale storage systems were delivering in practice. Enterprises such as Google [9] and institutions such the Sloan Digital Sky Survey [37] and the Large Hadron Collider [8] were collecting petabytes of data with long-term value that had to remain on-line to be useful. The annual cost of keeping a petabyte on-line was more than a million dollars [27]. It is easy to see why questions of the economics and reliability of storage systems became the focus of researchers' attention.

Papers at the 2007 FAST conference used data from NetApp [35] and Google [30] to study disk replacement rates in large storage farms. They showed that the manufacturer's MTTF numbers were optimistic. Subsequent analysis of the NetApp data [17] showed that all other components contributed to the storage system failures, and that:

> 'Interestingly, [the earlier studies] found disks are replaced much more frequently (2–4 times) than vendor-specified [replacement rates]. But as this study indicates, there are

other storage subsystem failures besides disk failures that are treated as disk faults and lead to unnecessary disk replacements." [17]

Two studies, one at CERN [18] and one using data from NetApp [5], greatly improved on earlier work using data from the Internet Archive [6, 36]. They studied *silent data corruption* in state-of-the-art storage systems; events in which the content of a file in storage changes with no explanation or recorded errors.

The NetApp study looked at the incidence of silent storage corruption in individual disks in RAID arrays. The data was collected over 41 months from NetApp's filers in the field, covering over 1.5×10^6 drives. They found over 4×10^5 silent corruption incidents. More than 3×10^4 of them were not detected until RAID restoration and could thus have caused data loss despite the replication and auditing provided by NetApp's row-diagonal parity RAID [11].

The CERN study used a program that wrote large files into CERN's various data stores, which represent a broad range of state-of-the-art enterprise storage systems (mostly RAID arrays), and checked them over a period of 6 months. A total of about 9.7×10^{16} bytes was written and about 1.92×10^8 bytes was found to have suffered silent corruption, of which about 2/3 was persistent; re-reading did not return good data. In other words, about 1.2×10^{-9} of the data written to CERN's storage was permanently corrupted within six months. We can place an upper bound on the bit half-life in this sample of current storage systems by assuming that the data was written instantly at the start of the 6 months and checked instantly at the end; the result is 2×10^8 or about 10^{-2} times the age of the universe. Thus to reach the petabyte for a century requirement we would need to improve the performance of current enterprise storage systems by a factor of at least 10^9.

6 Tolerating Failures

Despite the manufacturer's claims, current research shows that state-of-the-art storage systems fall so many orders of magnitude below our bit preservation requirements that we cannot expect even dramatic improvements in technology to fill the gap. Maintaining a single replica in a single storage system is not an adequate solution to the bit preservation problem.

Practical digital preservation systems must therefore:

- Maintain more than one copy by *replicating* their data on multiple, ideally different, storage systems.
- Audit or (*scrub*) the replicas to detect damage, and repair it by overwriting the known-bad copy with data from another.

The more replicas and the more frequently they are audited and repaired the longer the bit half-life we can expect. This is, after all, the basis for the backups and checksums technique in common use. In fact, current storage systems already use techniques like this internally, for example in the form of RAID [29]. Despite this the bit half-life they deliver is inadequate. Unfortunately adding the necessary inter-storage-system replication and scrubbing is expensive.

2008 cost figures from the San Diego Supercomputer Center[3] show that maintaining a single on-line copy of a petabyte for a year then cost about $\$1.05 \times 10^6$. A single near-line copy on tape cost about $\$4.2 \times 10^5$ a year. These costs decrease with time, albeit not as fast as raw disk costs. The British Library estimates a 30% per annum decrease. Assuming that this rate continues for at least a decade, if you can afford about 3.3 times the first year's cost to store an extra replica for a decade, you can afford to store it indefinitely. So, adding a second replica of a petabyte on disk would cost about $\$3.5 \times 10^6$ and on tape would cost about $\$1.4 \times 10^6$. Adding cost to a preservation effort to increase reliability in this way is a two-edged sword; doing so necessarily increases the risk that preservation will fail for economic reasons.

Further, without detailed understanding of the rates at which different mechanisms cause loss and damage, it still isn't possible to answer the question we started with, and know how many replicas would make us as safe as we need to be, and thus the cost of the necessary replication. At small scales the response to this uncertainty is to add more replicas, but as the scale increases this rapidly becomes unaffordable.

Replicating among identical systems is much less effective than replicating among diverse systems. Identical systems are subject to common mode failures, for example caused by a software bug in all the systems damaging the same data in each. On the other hand, purchasing and operating a number of identical systems will be considerably cheaper than operating a set of diverse systems.

Each replica is vulnerable to loss and damage. Unless they are regularly audited they contribute little to increasing bit half-life. The bandwidth and processing capacity needed to scrub the data are both costly, and adding these costs increases the risk of failure. Custom hardware [25] could compute the SHA-1 [28] checksum of a petabyte of data in a month, but doing so requires impressive bandwith - the equivalent of three gigabit Ethernet interfaces running at full speed the entire month. User access to data in long-term storage is typically infrequent; they are therefore rarely architected to provide such high-bandwidth read access. System cost increases rapidly with I/O bandwidth, and the additional accesses

[3] 2007 figures are in [27]

to the data (whether on disk or on tape) needed for scrubbing themselves potentially increase the risk of failure.

The point of writing software that reads and verifies the data systems store in this way is to detect damage and exploit replication among systems to repair it, thereby increasing bit half-life. How well can we do this? RAID is an example of a software technique of this type applied to disks. In practice, the CERN study [18] looking at real RAID systems from the outside showed a significant rate of silent data corruption, and the NetApp study [5] looking at them from the inside showed a significant rate of silent disk errors that would lead to silent data corruption. A study [20] of the full range of current algorithms used to implement RAID found flaws leading to potential data loss in all of them. Both this study, and another from IBM [16], propose improvements to the RAID algorithms but neither claim that they can eliminate silent corruption, or even accurately predict its incidence:

> "while we attempt to use as realistic probability numbers as possible, the goal is not to provide precise data loss probabilities, but to illustrate the advantage of using a model checker, and discuss potential trade-offs between different protection schemes." [20]

Thus, although inter-system replication and scrubbing are capable of decreasing the incidence of data loss, they cannot eliminate it completely. And the replication and scrubbing software itself will contain bugs that can cause data loss. It must be doubtful that we can implement these techniques well enough to increase the bit half-life of systems with an affordable number of replicas by 10^9.

7 Magic Media

Considering the difficulties facing disk drive technology [12], the reliability they achieve is astonishing, but it clearly isn't enough. News sites regularly feature stories reporting claims that some new storage medium has solved the problem of long-term data storage. Stone DVDs [23] claimed to last 1000 years were a recent example. These claims should be treated as skeptically as those of Sun and other storage system manufacturers. It may well be that the media in question are more reliable than their competitors, but as we have seen raw media reliability is only a part of the story. Our petabyte would be a stack of 2×10^5 stone DVDs. A lot can happen to a stack that big in 100 years. Truly magic media that were utterly reliable would make the problems better, but they would not make them go away completely.

I remember magnetic bubble memory, so I have a feeling of *deja vu*, but it is starting to look possible that flash memory, or possibly more exotic solid-state technologies such as memristors or phase change memory, may supplant disks. There is a lot to like about these technologies for long-term storage, but will they improve storage reliability?

Year	Seconds
1990	240
2000	720
2006	6450
2009	8000
2013	12800

Table 1: The time to read an entire disk of various generations.

Again, we don't know the answer yet. Despite flash memory's ubiquity, it isn't even clear yet how to measure its UBER:

> "UBER values can be much better than 10^{-15} but UBER is a strong function of program/erase cycling and subsequent retention time, so UBER specifications must be coupled with maximum specifications for these quantities." [26]

In other words, it depends how you use it, which doesn't appear to be the case for disk. Flash memory used for long-term data storage, which is written once and read infrequently, should in principle perform very well. And the system-level effects of switching from hard disk to flash can be impressive:

> "FAWN [Fast Array of Wimpy Nodes] couples low-power embedded CPUs to small amounts of local flash storage, and balances computation and I/O capabilities to enable efficient, massively parallel access to data. ... FAWN clusters can handle roughly 350 key-value queries per Joule of energy – two orders of magnitude more than a disk-based system" [3]

Fast CPUs, fast RAM and fast disks all use lots of power, so the low power draw of FAWN is not a surprise. But the high performance comes from another aspect of disk evolution. Table 1 shows how long it would take to read the whole of a state-of-the-art disk of various generations.

Disks have been getting bigger but they haven't been getting equivalently faster. This is to be expected, the data rate depends on the inverse of the diameter of a bit, but the capacity depends on the inverse of the area of a bit. FAWN nodes can read their entire contents very quickly, useful for scrubbing as well as answering queries.

This is all encouraging, but once it became possible to study the behavior of disk storage at a large scale it

became clear that system-level reliability fell far short of the media specifications. This should make us cautious about predicting a revolution from flash or any other new storage technology.

8 Economics

Ever since Clayton Christensen published *The Innovator's Dilemma* [10] it has been common knowledge that disk drive cost per byte halves every two years. So you might argue that you don't need to know how many copies you need to keep your data safe for the long term, you just need to know how many you need to keep it safe for the next few years. After that, you can keep more copies.

In fact, what has been happening is that the capacity at constant cost has been doubling every two years, which isn't quite the same thing. As long as this exponential grows faster than you generate new data, adding copies through time is a feasible strategy.

Alas, exponential curves can be deceiving. Moore's Law has continued to deliver smaller and smaller transistors. However, a few years ago it effectively ceased delivering faster and faster CPU clock rates. It turned out that, from a business perspective, there were more important things to spend the extra transistors on than making a single CPU faster. Like putting multiple CPUs on a chip.

At a recent Library of Congress meeting, Dave Anderson of Seagate warned [4] that something similar is about to happen to hard disks. Technologies (HAMR and PMR) are in place to deliver the 2013 disk generation, i.e. a consumer 3.5¨ drive holding 8TB. But the business case for building it is weak. The cost of the transition to PMR in particular is daunting [24]. Laptops, netbooks and now tablets are destroying the market for the desktop boxes that 3.5¨ drives go into. And very few consumers fill up the 2009 2TB disk generation, so what value does having an 8TB drive add? Let alone the problem of how to back up an 8TB drive on your desk! What is likely to happen, indeed is already happening, is that the consumer market will transition rather quickly to 2.5" drives. This will eliminate the high-capacity $100 3.5¨ drive, since it will no longer be produced in consumer quantities. Consumers will still buy $100 drives, but they will be 2.5¨ and have perhaps 1/3 the capacity. For a while the $/byte curve will at best flatten, and more likely go up. The problem this poses is that large-scale disk farms are currently built from consumer 3.5¨ drives. The existing players in the market have bet heavily on the exponential cost decrease continuing; if they're wrong it will be disruptive.

9 The Bigger Picture

Our inability to compute how many backup copies we need to achieve a reliability target is something we're just going to have to live with. In practice, we aren't going to have enough backup copies, and stuff will get lost or damaged. This should not be a surprise, but somehow it is. The fact that bits *can* be copied correctly leads to an expectation that they always *will* be copied correctly, and then to an expectation that digital storage will be reliable. There is an odd cognitive dissonance between this and people's actual experience of digital storage, which is that loss and damage are routine occurrences [22].

The fact that storage isn't reliable enough to allow us to ignore the problem of failures is just one aspect of a much bigger problem looming over computing as it continues to scale up. Current long-running Petascale high performance computer applications require complex and expensive checkpoint and restore schemes because the probability of a failure during execution is so high that restarting from scratch is infeasible. This approach will not scale to the forthcoming generation:

> "... it is anticipated that Exascale systems will experience various kind of faults many times per day. It is also anticipated that the current approach for resilience, which relies on automatic or application level checkpoint-restart, will not work because the time for checkpointing and restarting will exceed the mean time to failure of a full system. ...
>
> Some projections estimate that, with the current technique, the time to checkpoint and restart may exceed the mean time to interrupt of top supercomputers before 2015. This not only means that a computation will do little progress; it also means that fault-handling protocols have to handle multiple errors – current solutions are often designed to handle single errors." [7]

Just as with storage, the numbers of components and interconnections are so large that the incidence of failures is significant. And the available bandwidths relatively so low that recovering from the failures is time-consuming enough that multiple failure situations have to handled. There is no practical, affordable way to mask the failures from the applications. Application programmers will need to pay much more attention to detecting and recovering from errors in their environment. To do so they will need both the APIs and the system environments implementing them to become much more failure-aware.

10 API Enhancements

Storage APIs are starting to move in this direction. Recent interfaces to storage services [2] allow the applica-

Digest	Match	No Match
Unchanged	Data OK	Data bad
Changed	Deliberate alteration	Data and/or digest bad

Table 2: The four cases of message digest comparison.

tion's write call to provide not just a pointer to the data and a length, but optionally also the application's message digest of the data. This allows the storage system to detect whether the data was damaged during its journey from the application to the device, or while it was sitting in the storage device, or being copied back to the application. Recent research has shown that the memory buffers [44] and data paths [17] between the application and the storage devices contribute substantially to errors.

Let's take Amazon S3's REST API [2] as an example to show that, while these developments are welcome, they are far from a panacea. The PUT request supports an optional (and recommended) Content-MD5 header containing the application's digest of the data. The responses to most requests, including PUT, include an ETag header with the service's MD5 of the object. The application can remember the digest it computed before the PUT and, when the PUT returns, verify that the service's digest matches.

Doing so is a wise precaution, but all it really tells the application is that the service knows what the application thinks is the correct digest. The service knows this digest, not because it computed the digest of the correct data, but because the application provided it in the Content-MD5 header. A malign or malfunctioning service could respond correctly to PUT and HEAD requests by remembering the application's digest, without ever storing the data or computing its digest.

The application could try to detect a malign or malfunctioning service by using a GET to obtain the stored data, computing the digest (a) of the returned data, and comparing that with (b) either the digest in the response's ETag header, or with the digest it computed before the original PUT and remembered (which should be the same). It might seem that there are two cases; if the two message digests match then the data is OK[4], otherwise it isn't. There are actually four cases, as shown in Table 2, depending on whether the digest (b) is unchanged or not. The four cases illustrate two problems:

- The bits forming the digest are no different from the bits forming the data; neither are magically incorruptible. A malign or malfunctioning service could return bad data with a digest in the ETag header that matched the data but was not the digest originally computed. Applications need to know whether the digest has been changed. A system for doing so without incorruptible storage is described in [15].

- Given the pricing structure for cloud storage services such as Amazon S3, it is too expensive to extract the entire data at intervals to confirm that it is being stored correctly. Some method in which the service computes the digest of the data is needed, but simply asking the service to return the digest of a stored object is not an adequate check [33]. The service must be challenged to *prove* that its object is good. The simplest way to do this is to ask the service to compute the digest of a nonce (a random string of bits) and the object; because the service cannot predict the nonce a correct response requires access to the data *after* the request is received. Systems using this technique are described in [21] and [38].

Early detection is a good thing; the shorter the time between detection and repair the smaller the risk that a second error will compromise the repair. But detection is only part of the solution; the system also has to be able to repair the damaged data. It can do so only if it has replicated the data elsewhere, and some deduplication layer has not optimized away this replication.

11 Conclusion

It would be nice to end on an up-beat note, describing some technological fix that would allow applications to ignore the possibility of failures in their environment, and specifically in long-term storage. Unfortunately, in the real world, failures are inevitable. As systems scale up they become more frequent. Even throwing money at the problem can only reduce the incidence of failures, not exclude them entirely. Applications in the future will need to be much more aware of, and careful in responding to, failures.

The high-performance computing community accurately describes what needs to be done:

> "We already mentioned the lack of coordination between software layers with regards to errors and fault management. Currently, when a software layer or component detects a fault it does not inform the other parts of the software running on the system in a consistent manner. As a consequence, fault-handling actions taken by this software component are hidden to the rest of the system. ... In an ideal wor[l]d, if a software component detects a potential error, then the information should propagate to other components that may be affected by the error

[4] Assuming the digest algorithm hasn't been broken, not a safe assumption for MD5 [19]

or that control resources that may be responsible for the error." [7]

In particular, as regards storage, APIs should copy Amazon's S3 by providing optional data integrity capabilities that allow applications to perform end-to-end checks. These APIs should be enhanced to allow the application to provide an optional nonce that is prepended to the object data before the message digest reported to the application is computed. This would allow applications to exclude the possibility that the reported digest has been remembered rather than re-computed.

Acknowledgements

Grateful thanks are due to Eric Allman, Kirk McKusick, Jim Gettys, Tom Lipkis, Mark Compton, Petros Maniatis and the late Jim Gray. Some of this material was originally presented at iPRES 2008 and subsequently published in the Intl. J. Digital Curation [32].

Biography

David Rosenthal has been an engineer in Silicon Valley for a quarter of a century, including as a Distinguished Engineer at Sun Microsystems and employee #4 at Nvidia. For the last decade he has been working on the problems of long-term digital preservation under the auspices of the Stanford Library.

References

[1] Douglas Adams. The Hitch-Hiker's Guide to the Galaxy, 1978. British Broadcasting Corp.

[2] Amazon. Amazon S3 API Reference. `http://docs.amazonwebservices.com/AmazonS3/latest/API/`, March 2006.

[3] David G. Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. FAWN: a fast array of wimpy nodes. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, New York, NY, USA, 2009. ACM.

[4] Dave Anderson. Hard Drive Directions. `http://www.digitalpreservation.gov/news/events/other_meetings/storage09/docs/2-4_Anderson-seagate-v3_HDtrends.pdf`, September 2009.

[5] Lakshmi Bairavasundaram, Garth Goodson, Bianca Schroeder, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. An Analysis of Data Corruption in the Storage Stack. In *Proceedings of 6th USENIX Conf. on File and Storage Technologies*, 2008.

[6] Mary Baker, Mehul Shah, David S. H. Rosenthal, Mema Roussopoulos, Petros Maniatis, TJ Giuli, and Prashanth Bungale. A Fresh Look at the Reliability of Long-term Digital Storage. In *Proceedings of EuroSys2006*, Leuven, Belgium, April 2006.

[7] Franck Cappello, Al Geist, Bill Gropp, Sanjay Kale, Bill Kramer, and Marc Snir. Toward Exascale Resilience. Technical Report TR-JLPC-09-01, INRIA-Illinois Joint Lab. on Petascale Computing, July 2009.

[8] CERN. Worldwide LHC Computing Grid. `http://lcg.web.cern.ch/LCG/`, 2008.

[9] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Grube. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the 7th Usenix Symp. on Operating System Design and Implementation*, pages 205–218, 2006.

[10] Clayton M. Christensen. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press, June 1997.

[11] Peter Corbett, Bob English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar. Row-Diagonal Parity for Double Disk Failure Correction. In *3rd Usenix Conference on File and Storage Technologies*, San Francisco, CA, March 2004.

[12] Jon Elerath. Hard-Disk Drives: The Good, the Bad, and the Ugly. *Comm. ACM*, 52(6), June 2009.

[13] Jon G. Elerath and Michael Pecht. Enhanced Reliability Modeling of RAID Storage Systems. In *DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 175–184, Washington, DC, USA, 2007. IEEE Computer Society.

[14] Dawson Engler. A System's Hackers Crash Course: Techniques that Find Lots of Bugs in Real (Storage) System Code. In *Proceedings of 5th USENIX Conf. on File and Storage Technologies*, San Jose, CA, February 2007.

[15] Stuart Haber and W. Scott Stornetta. How to Timestamp a Digital Document. *Journal of Cryptology: the Journal of the International Association for Cryptologic Research*, 3(2):99–111, 1991.

[16] J. L. Hafner, V. Deenadhayalan, W. Belluomini, and K. Rao. Undetected disk errors in RAID arrays. *IBM J. Research & Development*, 52(4/5), September 2008.

[17] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky. Are Disks the Dominant Contributor for Storage Failures? A Comprehensive Study of Storage Subsystem Failure Characteristics. In *Proceedings of 6th USENIX Conf. on File and Storage Technologies*, 2008.

[18] Peter Kelemen. Silent Corruptions. In *8th Annual Workshop on Linux Clusters for Super Computing*, 2007.

[19] Vlastimil Klima. Finding md5 collisions - a toy for a notebook. Cryptology ePrint Archive, Report 2005/075, 2005. `http://eprint.iacr.org/2005/075`.

[20] Andrew Krioukov, Lakshmi N. Bairavasundaram, Garth R. Goodson, Kiran Srinivasan, Randy Thelen, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Parity Lost and Parity Regained. In *Proceedings of 6th USENIX Conf. on File and Storage Technologies*, 2008.

[21] Petros Maniatis, Mema Roussopoulos, TJ Giuli, David S. H. Rosenthal, Mary Baker, and Yanto Muliadi. Preserving Peer Replicas By Rate-Limited Sampled Voting. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 44–59, Bolton Landing, NY, USA, October 2003.

[22] Cathy Marshall. "It's like a fire. You just have to move on": Rethinking Personal Digital Archiving. In *6th USENIX Conference on File and Storage Technologies*, Berkeley, CA, 2008. USENIX.

[23] Lucas Mearian. Start-up claims its DVDs last 1,000 years. *Computerworld*, November 2009.

[24] Chris Mellor. Drive suppliers hit capacity increase difficulties. *The Register*, July 2010.

[25] H. E. Michail, A. P. Kakarountas, G. Theodoridis, and C. E. Goutis. A low-power and high-throughput implementation of the SHA-1 hash function. In *Proceedings of the 9th WSEAS International Conference on Computers*, 2005.

[26] Neal Mielke, Todd Marquart, Ning Wu1, Jeff Kessenich, Hanmant Belgal, Eric Schares, Falgun Trivedi, Evan Goodness, and Leland R. Nevill. Bit Error Rate in NAND Flash Memories. In *46th Annual International Reliability Physics Symposium*, pages 9–19. IEEE, 2008.

[27] Richard L. Moore, Jim D'Aoust, Robert H. McDonald, and David Minor. Disk and Tape Storage Cost Models. In *Archiving 2007*, May 2007.

[28] National Institute of Standards and Technology (NIST), Washington, D.C., USA. *Federal Information Processing Standard Publication 180-1: Secure Hash Standard (SHA-1)*, April 1995.

[29] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 109–116, Chicago, IL, USA, June 1988.

[30] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz Andre Barroso. Failure Trends in a Large Disk Drive Population. In *Proceedings of 5th USENIX Conf. on File and Storage Technologies*, San Jose, CA, February 2007.

[31] Vijayan Prabhakaran, Nitin Agrawal, Lakshmi Bairavasundaram, Haryadi Gunawi, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. IRON File Systems. In *Proceedings of the 20th Symposium on Operating Systems Principles*, 2005.

[32] David S. H. Rosenthal. Bit Preservation; A Solved Problem? *Intl. J. Digital Curation*, 1(5), 2010.

[33] David S. H. Rosenthal. LOCKSS: Lots Of Copies Keep Stuff Safe. In *NIST Digital Presebvation Interoperability Framework Workshop*, March 2010.

[34] David S. H. Rosenthal, Thomas S. Robertson, Tom Lipkis, Vicky Reich, and Seth Morabito. Requirements for Digital Preservation Systems: A Bottom-Up Approach. *D-Lib Magazine*, 11(11), November 2005.

[35] Bianca Schroeder and Garth Gibson. Disk failures in the real world: What Does an MTTF of 1,000,000 Hours Mean to You? In *Proceedings of 5th USENIX Conf. on File and Storage Technologies*, San Jose, CA, February 2007.

[36] T. Schwarz, M. Baker, S. Bassi, B. Baumgart, W. Flagg, C. van Imngen, K. Joste, M. Manasse, and M. Shah. Disk Failure Investigations at the Internet Archive. In *Work-in-Progress Session, NASA/IEEE Conf. on Mass Storage Systems and Technologies*, 2006.

[37] SDSS. The Sloan Digital Sky Survey. `http://www.sdss.org/`, May 2008.

[38] Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, and Ram Swaminathan. Auditing to Keep Online Storage Services Honest. In *HOTOS XI, 11th Workshop on Hot Topics in Operating Systems*, May 2007.

[39] Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, and Kaladhar Voruganti. Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-Based Archival Storage. In *Proceedings of 6th USENIX Conf. on File and Storage Technologies*, 2008.

[40] Sun Microsystems. Sales Terms and Conditions, Section 11.2. `http://store.sun.com/CMTemplate/docs/legal_terms/TnC.jsp#11`, December 2006.

[41] Sun Microsystems. ST5800 presentation. Sun PASIG Meeting, June 2008.

[42] N. Talagala. *Characterizing Large Storage Systems: Error Behavior and Performance Benchmarks*. PhD thesis, CS Div., Univ. of California at Berkeley, Berkeley, CA, USA, October 1999.

[43] Paul Williams, David S. H. Rosenthal, Mema Roussopoulos, and Steve Georgis. Predicting the Archival Life of Removable Hard Disk Drives. In *Archiving 2008*, June 2008.

[44] Yupu Zhang, Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. End-to-end Data Integrity for File Systems: A ZFS Case Study. In *8th USENIX Conference on File and Storage Technologies*, Berkeley, CA, 2010. USENIX.