Reliability Challenges for Storing Exabytes

Ahmed Amer
Santa Clara University
Santa Clara, CA
aamer@scu.edu

Darrell D.E. Long
University of California
Santa Cruz, CA
darrell@cs.ucsc.edu

Thomas Schwarz, S.J.
Universidad Católica del Uruguay
Montevideo, Uruguay
tschwarz@ucu.edu.uy

***Abstract*—As we move towards data centers at the exascale, the reliability challenges of such enormous storage systems are daunting. We demonstrate how such systems will suffer substantial annual data loss if only traditional reliability mechanisms are employed. We argue that the architecture for exascale storage systems should incorporate novel mechanisms at or below the object level to address this problem. Our argument for such a research focus is that focusing solely on the device level will not scale, and in this study we analytically evaluate how rapidly this problem manifests.**

I. Introduction

Fontana, Decad and Hetzler calculate in 2013 that it would require a capitalization of \$100B in order to replace HDD enterprise applications with NAND flash [4]. We can safely assume that disks will continue to be the mainstay of mass storage systems. According to the road map of the storage industry, disk drives will reach a capacity of 12 TB in 2015 [5]. If the storage utilization of a disk is on average 80%, and if we devote 4% of the available space to store parity data, then we need $113,778$ 12 TB-disks per exabyte of storage. This scale will pose new challenges in reliability, administration, and power consumption. Here, we concentrate on the reliability implications of exascale storage.

At failure rates observed in today's data centers, an exascale file system will suffer on average more than one disk failures per hour. While the overall data loss rate might be low, its economic costs will vary according to the type and value of data stored. For example, a simple data warehouse might be filled with consumer data that retains its value even if it loses a substantial portion of the data stored, while losing one disk sector's worth of data might destroy the capacity of researchers to evaluate an experiment at CERN, necessitating a tremendously expensive repetition of that experiment. The stakes are high. As Keeton and colleagues state bluntly: "Losing information when a storage device or data center fails can bring a company to its knees — or put it out of business altogether" [11].

In the following, we assume a large-scale storage solution that uses storage bricks, self-contained storage units with a reasonably large number of disks. Storage bricks seem to be the correct level for storage units, since dealing with individual disks is much more prone to error. A technician who removes a good (instead of a failed) disk drive from a storage system will cause the system to go into degraded mode, reconstructing data and possibly finding that it is now impossible to do so. A brick contains tens of such disks and is managed as a unit. If a brick is removed or temporarily disconnected, the system has lost access to a large part of its data, but if the brick is reconnected, the data becomes accessible again. The ease of management makes an organization based on bricks so attractive and justifies our considering only this type of architecture for a storage system.

A storage brick stores data redundantly and can recover from failures internally. We assume such bricks are organized as a fully declustered RAID Level 5 or Level 6 disk array, with several distributed spare disks. If a brick has suffered several failures, its data will be moved to other bricks and the entire brick removed.

We calculate data loss rates for these standard storage bricks, and find that the data loss rates are acceptable for some applications but not necessarily for all. We present here an argument for data protection at a higher level in the exa scale storage system. The alternative would be to improve the system reliability by increasing individual node reliability. This would contradict the character of storage bricks. Instead, we can use semantic information for higher valued data to use intelligent placement on top of the device level, or knowledge of the underlying brick arrangements below this level to inform placement according to environmental conditions.

Below we consider only two causes for dataloss, namely full disk failure and latent disk errors. To this, future storage systems architects will also have to add losses due to operational errors, physical brick failure, losses due to networking outages, and generic catastrophes such as flooding.

II. Disk Failures

In 2015, we can estimate disk capacities to be about 12 TB, while the access rate for disks would be about 200 MB/s. We organize the data using *storage bricks* as the building blocks. Each storage brick is a fully declustered disk array that provides network access and is self-organizing, self-configuring, self-tuning, self-healing, and self-managing [1], [6], [7].

There are two main sources of data loss in magnetic disk storage, catastrophic loss of a disk and latent errors. An error in the latter category affects a small number of blocks but does not affect other blocks on the disk. It is detected only when trying to access a block. Disk scrubbing can be used to detect latent errors before they can cause a data recovery operation to fail [17], whereas intra-disk redundancy masks latent errors

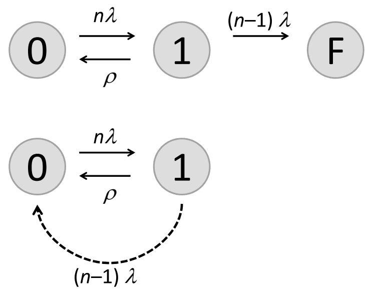


Fig. 1. Markov models for a fully declustered RAID Level 5

completely [9], and idle read-after-write can detect latent write errors shortly after they occur [12].

We first concentrate on evaluating reliability numbers of a storage block by only considering full disk failures for the moment. We use Markov models to assess the data protection of such storage bricks.

We first consider a storage brick that is organized as a completely declustered RAID Level 5 with some spare disks. We assume that all disks suffer data loss at a constant *failure rate*. We also assume that disks fail independently. The upper half of Figure 1 gives a Markov model for our fully declustered storage block with n disks, a failure rate λ, and a repair rate ρ. The model consists of two non-failure states. State 0 models the storage brick without any failures and State 1 with one failed disk and recovering the lost data. State F is the failure state. In State 1, the brick discovers the failed disk and begins reconstructing the data in it to available spare space.

Let p_0 and p_1 be the probabilities of the model being in States 0 and 1. The Kolmogorov differential equations for p_0 and p_1 are

$$\begin{aligned} p_0'(t) &= -n\lambda p_0 + \rho p_1 \\ p_1'(t) &= n\lambda p_0 - ((n-1)\lambda + \rho)\, p_1. \end{aligned}$$

The solution is

$$\begin{aligned} p_0(t) &= \frac{A\cosh\left(\frac{Dt}{2}\right) + (\rho-\lambda)\sinh\left(\frac{Dt}{2}\right)}{A\exp(\frac{1}{2}t((2n-1)\lambda+\rho))} \\ p_1(t) &= \frac{2\lambda n\sinh\left(\frac{1}{2}tD\right)}{A\exp(\frac{1}{2}t((2n-1)\lambda+\rho))} \\ A &= \sqrt{\lambda^2 + 2(2n-1)\lambda\rho + \rho^2} \\ D &= \sqrt{\lambda^2 - 2\lambda\rho + 4n\lambda\rho + \rho^2}. \end{aligned}$$

We now derive an expression for the repair time $1/\rho$. Since there are k data blocks in a reliability stripe, the reconstruction of a single block on a failed disk requires reading k blocks and writing one spare block with the reconstructed data. We can assume a data layout that distributes this work evenly to all the disks in the brick [2], [8], [15], [16]. With reasonable accuracy, we can assume a constant bandwidth of b bytes per second for reads and writes. The size of the disks is S. It then takes S/b to read or write a full disk. However, disks are usually not full. If ϕ is the proportion of the failed disk that is used, then the time rewrite data to a replacement drive shrinks to $\phi S/b$. Assume that we allocate a portion ψ of the available bandwidth to recovery operations. Recovering one disk involves reading from k disks and writing to one disk. This work is evenly divided over the $n-1$ remaining disks, so that the time for repair is given by

$$R = \frac{k+1}{n-1} \times \frac{\phi S}{\psi b}.$$

For example, if $S = 12$ TB, $b = 200$ MB/s , $\phi = 0.80$ (disks are 80% full), and $\psi = 0.5$ (we use half the disk bandwidth for recovery), then $R = 27.962(k+1)/(n-1)$ h. Our calculation uses a fixed repair time. In practice, recovery operations are opportunistic and less aggressive in order to not drive up the disk utilization unnecesasrily. The repair time would then be distributed and depend on the utilization of the brick by normal operations. As R is much smaller than $1/\lambda$, it turns out that assuming an exponential repair rate in our Markov model does not noticeably distort the results.

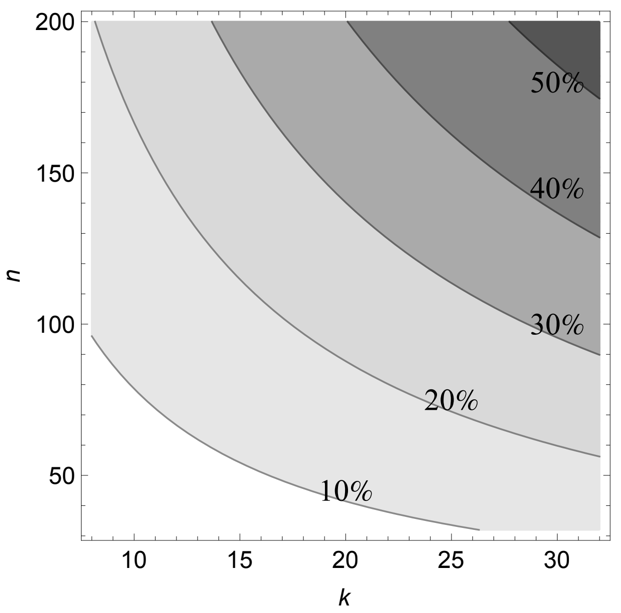


Fig. 2. Contour plot of the five year failure rate of a storage brick with n disks, reliability stripe size k and disk failure rate of 1 per $100,000$ hours.

With these solutions, we can calculate the 5-year survival rate of a RAID Level 5 storage brick. We assume that $\psi = 0.5$ of the bandwidth is reserved for recovery and that disks are $\phi = 0.8$ full. We assume a disk failure rate during the first five years of $1/100000$ per hour. We plot the result in Figure 2. We can see that the reliability of these bricks is quite low. This is due in great part to the large size of the disks and the low ratio of bandwidth to capacity.

While the very real possibility of data loss at all is disconcerting, at each episode, the brick will only lose some of its data. The economical costs of each data loss will depend on the amount of data that is lost, the impact on the usability of the remaining data, and the economic value of the data that becomes unusable. With these values, we can make decisions on the provisioning of a storage system. We therefore want to

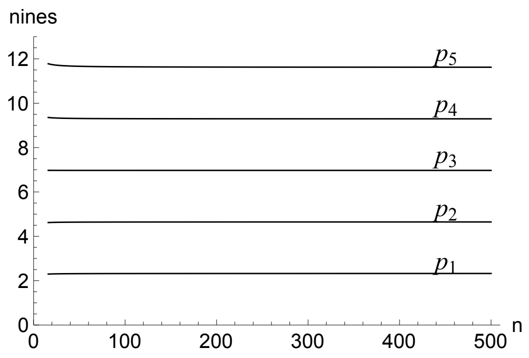


Fig. 3. Stable state probabilities p_i for being in a state with i failed disks for $i = 1, 2, 3, 4, 5$. We give $-\log_{10}(p_i$ for $\lambda = 1/100,000$, $k = 16$, $\psi = 0.5$, and $\phi = 0.8$.

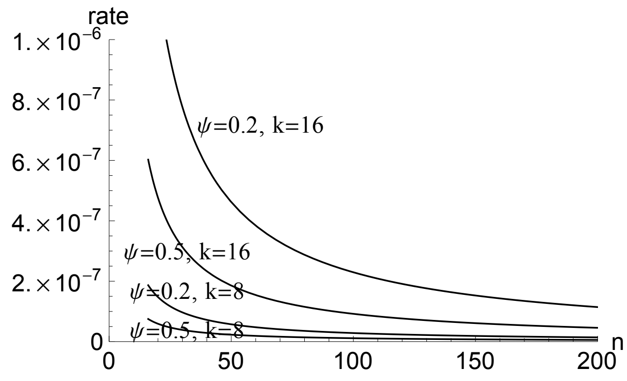


Fig. 4. Dataloss rates for a RAID Level 5 storage brick

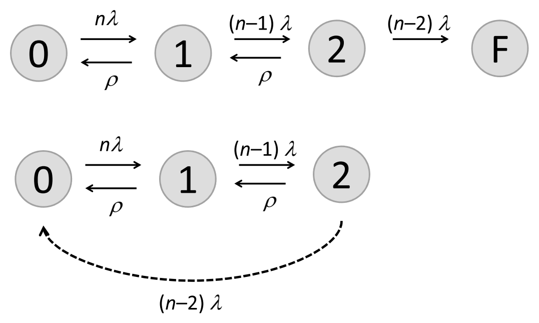
calculate the data loss rate of an individual storage brick. To this end, we change the Markov model to the ergodic model presented in the lower part of Figure 1. The new model does away with the Failure State and replaces the transition from State 1 to the Failure State with a transition to State 0. This transition models a data loss, after which a new brick replaces the old one "magically" with all the data that could be rescued from its predecessor's catastrophic failure. Our assumption is conservative in that it can only underestimate data loss rates. The new model is simpler and we can calculate equilibrium probabilities for being in either of State 0 or State 1. If we denote the probability of being in State i with p_i, then we have

$$\begin{aligned} p_0 + p_1 &= 1 \\ n\lambda p_0 - (\rho + (n-1)\lambda)p_1 &= 0 \end{aligned}$$

with solution

$$\begin{aligned} p_0 &= 1 - \frac{n\lambda}{(2n-1)\lambda + \rho} \\ p_1 &= \frac{n\lambda}{(2n-1)\lambda + \rho}. \end{aligned}$$

The failure transition is taken with rate $p_1(n-1)\lambda$. This number constitutes the rate at which these bricks suffer data loss [19].


$$\begin{tikzcd} 0 \arrow[r, shift left, "n\lambda"] & 1 \arrow[l, shift left, "\rho"] \arrow[r, shift left, "(n-1)\lambda"] & 2 \arrow[l, shift left, "\rho"] \arrow[r, "(n-2)\lambda"] & F \end{tikzcd}$$

$$\begin{tikzcd} 0 \arrow[r, shift left, "n\lambda"] & 1 \arrow[l, shift left, "\rho"] \arrow[r, shift left, "(n-1)\lambda"] & 2 \arrow[l, shift left, "\rho"] \arrow[ll, bend left, dashed, "(n-2)\lambda"] \end{tikzcd}$$

Fig. 5. Markov models for a fully declustered RAID Level 6

If during a rebuild, a second disk fails, some data will be lost. We now calculate the average amount of data lost. The first disk has already had some of its data recovered to spare space. We call the proportion not yet recovered α. A block in this part of the disk cannot be reconstructed if one of the k blocks in the same reliability stripe is located on the second failed disk. This happens with probability $k/(n-1)$. The first failed disk loses $\phi\alpha k/(n-1)$ of its data. However, only $(k-1)/k$ of this data is user data, so that the total amount of user data lost is $\alpha\phi(k-1)/(n-1)$. Similarly, a block on the second failed disk is lost if it shares a reliability stripe with an as yet unreconstructed block in the first failed disk. Therefore, this disk loses $\phi\alpha(k-1)/(n-1)$ of its user data as well. In total, we lose $2\phi\alpha(k-1)/(n-1)$ user data, and on average $\phi(k-1)/(n-1)$ data, since α has mean $1/2$.

In our calculations, we neglected the effect of additional disk failures when one disk failure is already present. Our approximation is justified by a calculation that shows that being in a state with 2, 3, 4, ... failures occurs with an exponentially decreasing probability. We solved an extension of the Markov model with more failure states (using our standard assumptions of disk failure rate of one per $100,000$ hours, $\psi = 0.5$ and $\phi = 0.8$) to calculate the steady state probabilities p_i that the storage brick has i failures. Figure 3 illustrates the results. There, we give the negative decadic logarithm of the probabilities. For example, $p_1 \approx 10^{-2.1}$. The result show us that the steady state probabilities only depend on n for small values, and even then, not in a dramatic way. As we can see, the probabilities fall in this typical case by a factor of 100. Therefore, the third failure will not add more than about 1% to the dataloss rate as calculated above. The fourth, fifth, ... failure can therefore be completely neglected.

Our model describes behavior in steady-state, but of course a new storage brick starts out without any failures, and is not in this steady-state. We compared the failure behavior of a fresh brick with an existing one according to the approximation by steady state, and found that the failure rates were too close to make any appreciable difference for data loss rates. For brevity, we do not include these results.

We now turn our attention to RAID Level 6. Figure 5 shows the usual Markov model followed by the ergodic change obtained by changing the transition to the Failure State to

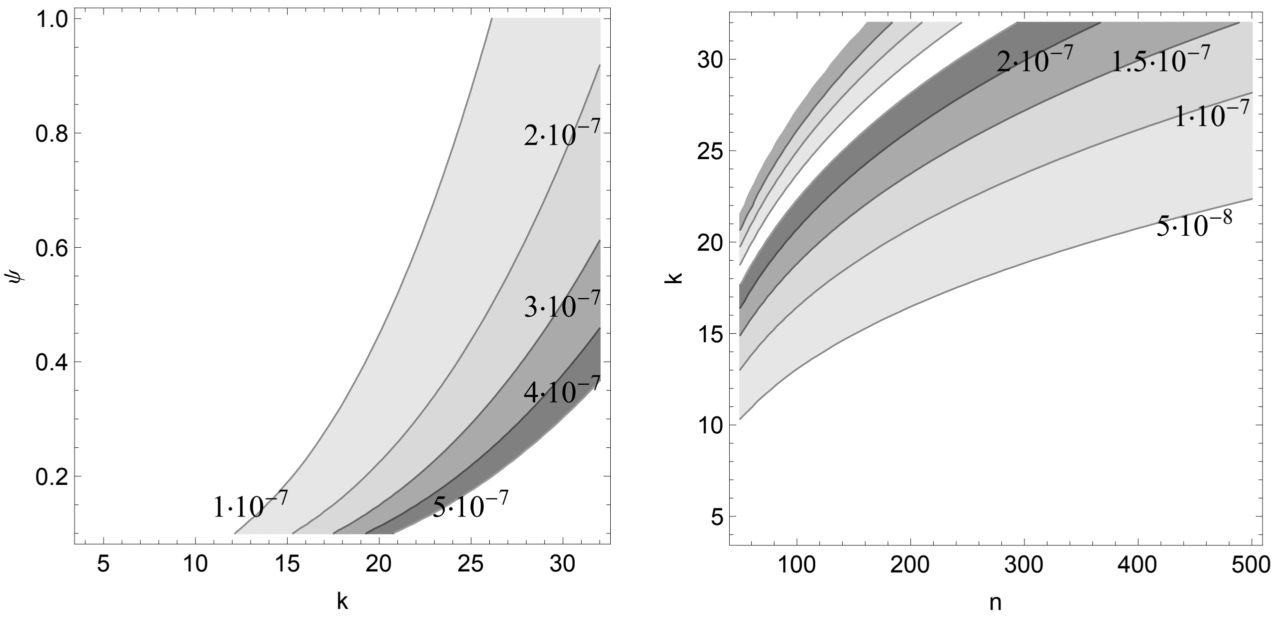


Fig. 6. Contour graph of the dataloss rates for a RAID Level 5 storage brick with $\lambda = 1/100,000h$, utilization $\phi = 80\%$, and $n = 200$ (left) and with $\lambda = 1/100,000h$, utilization $\phi = 80\%$, and $\psi = 0.5$ (right)

a transition to the Initial State. Now, we have three states representing fully functioning, fully declustered, RAID Level 6 arrays. First, there is the initial State 0 with no failed disks. In State 1, there is a single disk failure, which is being repaired at rate ρ. We use the same convention as before for repair transitions. Strictly speaking, the "repaired" disk array has one less fully functioning disk. In this scenario, we assume that we are continuing to repair the first failed disk using all the bandwidth allotted to repair from all remaining disks in the array. Therefore, we continue to have a repair rate ρ for the state transition to State 1. The equilibrium conditions of the state probabilities p_0 (being in State 0), p_1 (being in State 1), and p_2 (being in State 2) for the ergodic model are

$$\begin{aligned} p_0 + p_1 + p_2 &= 1 \\ n\lambda p_0 &= \rho p_1 + (n-2)\lambda p_2 \\ \rho p_1 + (n-1)\lambda p_1 &= \rho p_2 + n\lambda p_0 \end{aligned}$$

The solutions are

$$\begin{aligned} p_0 &= \frac{(n-2)(n-1)\lambda^2 + (n-2)\lambda\rho + \rho^2}{(2-6n+3n^2)\lambda^2 + 2(n-1)\lambda\rho + \rho^2} \\ p_1 &= \frac{n\lambda(n-2+\rho)}{(2-6n+3n^2)\lambda^2 + 2(n-1)\lambda\rho + \rho^2} \\ p_2 &= \frac{(n-1)n\lambda^2}{(2-6n+3n^2)\lambda^2 + 2(n-1)\lambda\rho + \rho^2}. \end{aligned}$$

Each repair still involves reading k disks and writing an additional disk. Thus, the repair times are still the same as calculated for RAID Level 5, giving us the repair rate

$$\rho = \frac{n}{k+1}\frac{\psi b}{\phi S}.$$

Data loss is described by a transition from State 2 to State 0 and occurs with rate $p_2(n-2)\lambda$. At the moment of the transition, we have three failed disks. This happens if the recovery of data on the first disk has not yet finished. The times of the second and third failure are during the repair time of the first disk. Let t_0 be the time of the first failure and let $1/\rho$ be the repair time of the first disk. We can assume that the other two failures are uniformly and independently distributed in the interval $[t_0, t_0 + 1/\rho]$. The second failure is the first of the two failure events and happens on average $1/3$ into the interval, and the second $2/3$ into the interval. Therefore, on average the first disk has already been $2/3$ recovered.

A block that has not been recovered on the first disk cannot be reconstructed if it its reliability stripe contains blocks on the second and the third failed disks. This happens with probability

$$q = \binom{n-3}{k-3}\binom{n-1}{k-1}^{-1} = \frac{(k-1)(k-2)}{(n-1)(n-2)}.$$

We lose a block on the first disk with probability $q/3$. We lose data in a block in the second disk if its reliability stripe contains a block on the first failed disk that has not yet been reconstructed, and a block on the third failed disk. This again gives a loss probability for the block of $q/3$. The same argument holds for the third disk. We lose a block on one of the failed disks with probability $q/3$ and have in total a loss rate of blocks of q.

The rate of data loss with disk size S is therefore given by

$$\frac{(k-2)(k-1)\lambda^3 n}{\frac{5.7\times10^{23}(n-1)^2\psi^2}{(k+1)^2S^2\phi^2} + \frac{1.5\times10^{12}\lambda(n-1)^2\psi}{(k+1)S\phi} + \lambda^2(3(n-2)n+2)}.$$

The constants in this formula arise from using absolute numbers for the disk size and the bandwidth. We can see

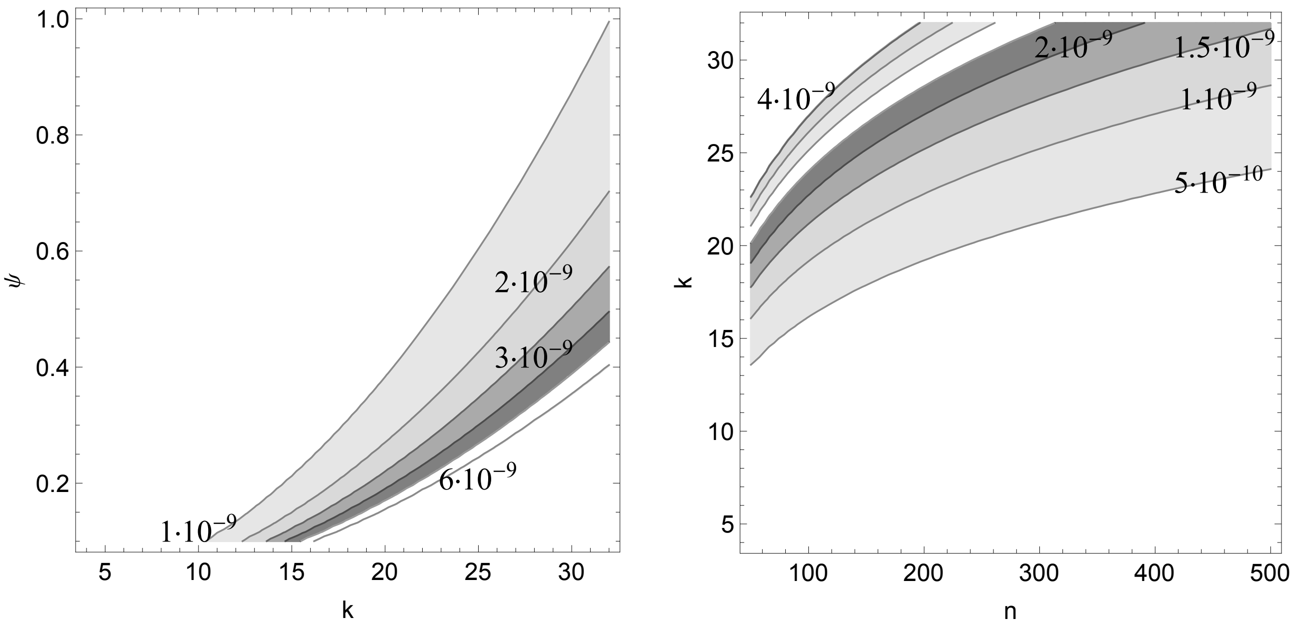


Fig. 7. Contour graph of the dataloss rates for a RAID Level 6 storage brick with $\lambda = 1/100,000h$, utilization $\phi = 80\%$, and $n = 200$ (left) and with $\lambda = 1/100,000h$, utilization $\phi = 80\%$, and $\psi = 0.5$ (right)

by comparing Figures 6 and 7 that the additional parity per reliability stripe of declustered RAID Level 6 brings a 100-fold decrease in data loss rates.

Our results show that even RAID Level 5 already does a reasonable job protecting against the effects of disk failure at very reasonable hardware costs. The declustered architecture distributes the recovery operation in the degraded mode over many disks, leading to quick recovery from failure and the brick spends less time in the vulnerable state. The second parity drive in the RAID Level 6 architecture reduces the dataloss rate by about 100 times. In absolute numbers, the system still loses data. A typical 1 exabyte system will lose on the order of 0.5 GB/h on average, or about 4.4 TB of data per year, due to full disk failures. To protect against this loss, each brick could have one additional parity drive, but the cost-benefit relationship seems hard to justify.

III. Latent Disk Errors

Latent Sector Errors (LSE) are a reasonably frequent occurrence in disk systems. Bairavasundaram [3] observed a large set of disks and discovered that 3.5% of disks developed latent errors over 32 months. Rozier and colleagues [13] discuss a similar failure type called Undetected Disk Errors (UDE) that can result in users receiving faulty data, and concluded that additional protection mechanisms might be warranted. If we assume the existence of such a protection mechanism, then we can convert such unidentified disk errors into latent sector errors.

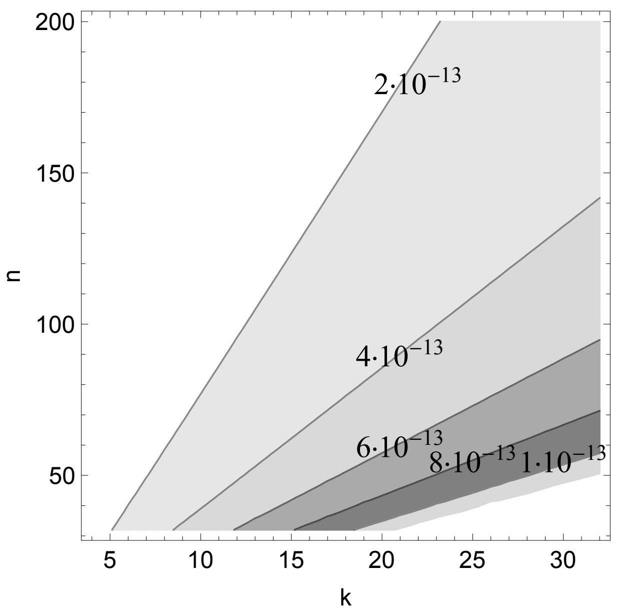
The genesis of LSE has, to our knowledge, not been exhaustively researched. Failure mechanisms include mechanical errors during writes (such as exceeding flight height of the head over the platter, or servo errors) and defects or

$$\begin{tikzcd} \text{OK} \arrow[r, shift left, "\gamma"] & \text{F} \arrow[l, shift left, "\sigma"] \end{tikzcd}$$

Fig. 8. Markov model for LSE generation and removal

contamination of the recording medium on the platters. Iliadis and colleagues [10] assume in their analysis that LSE only appear as the results of writes. Writes might very well account for the vast majority of LSE, but we do not know this for certain. Schroeder and colleagues [14] used the same data set as Bairavasundaram to obtain correlational data on LSE and found variance over time, usage, and age of the disk. They also saw strong interdependence of LSE. Nevertheless, we will assume here a simple constant rate of LSE that would be realistic assuming the use of scrubbing and an infra-disk protection scheme.

In the absence of a better understanding, we assume that LSE are created by a failure process with rate γ. LSE are detected by the failure of a normal read or during a deliberate scrubbing process. They are then fixed, and thereby removed from the system, using the redundancy in the reliability stripe. If removals happen at rate σ, then we can use the simple ergodic Markov model in Figure 8 to calculate the probability that a disk has a LSE to be $\gamma/(\gamma+\sigma)$. For example, if 3.5% of all disks develop one or more LSE over 32 months [3], and we scrub a disk completely every 2 weeks, about 0.05% of all disks have an LSE. If we assume that user reads are as effective as scrubbing at discovering LSE, then this percentage decreases to 0.025%. LSE seem to be highly correlated, and we assume that each incident of LSE affects on average B sectors. An reasonable value for B drawn from observation is approximately 30.



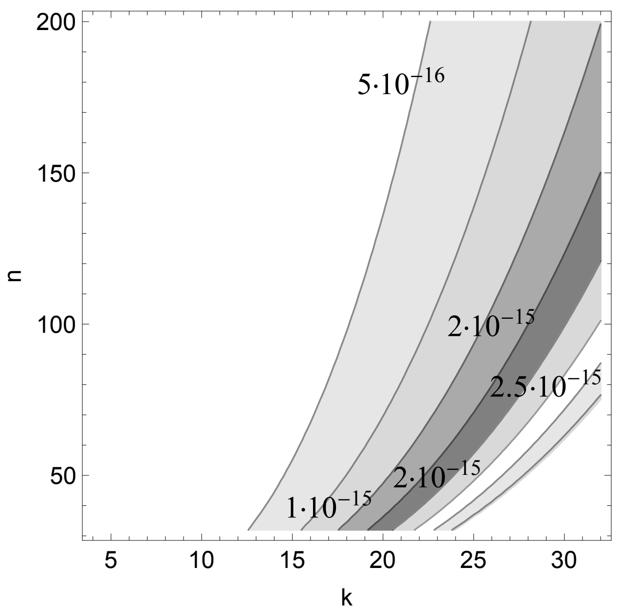


Fig. 9. Contour graph for the data loss rate due to irrecoverable LSE. We assume 12 TB drives with 4KB sectors, a disk utilization of 80%, and an average number of $B = 30$ sectors affected. The left figure is for RAID Level 5 while the right for RAID Level 6.

A fully declustered RAID Level 5 storage brick with n disks suffers a complete disk failure at a rate of $n\lambda$, where λ is the average failure rate of disks during their economic life span. After a failure, an LSE cannot be recovered if it is on a sector that shares its reliability stripe with the failed disk. This happens to $(k-1)/(n-1)$ of all sectors. In order to lose user data, a sector needs to be actually storing valid user data. Since the disk is ϕ full, and since $(k-1)/k$ of its contents are user data (and not parity data), and since B/T of the sectors have an LSE (B average number of sectors affected and T the total number of sectors on a disk), we lose

$$\frac{\gamma}{\gamma+\sigma} \times \frac{k-1}{n-1} \times \phi \times \frac{k-1}{k} \times \frac{B}{T}$$

of sectors due to irrecoverable LSE. This is also the dataloss rate due to irrecoverable LSE. As Figure 9 shows, the dataloss rate seems acceptable. Unfortunately, it does not represent the effects of latent sector errors. A single unreadable sector renders a complete file unusable, and the failure of this file might make even more information unusable. Depending on the type of user data stored, the effects of LSE might need to be multiplied with a factor that could easily reach 10^5, in which case the loss rates would not be acceptable any longer for many types of data.

In a RAID Level 6 array, a sector with an undiscovered error is not recoverable, if two disks have failed and its reliability stripe contains both failed disks. The latter happens with probability $\binom{k-3}{n-3}/\binom{k-1}{n-1} = \frac{(k-1)(k-2)}{(n-1)(n-2)}$. To obtain the rate at which a second disk failure happens, we can reuse the calculation of the previous section for RAID Level 5. Accordingly, with repair rate ρ, the second failure happens at rate

$$\tau = \frac{n\lambda}{(2n-1)\lambda+\rho} \times (n-1)\lambda.$$

where the first factor gives the probability that the system is in a vulnerable state with one disk failure, and the second factor gives the rate at which the second disk fails. We recall that, on average, half of the sectors in the first disk are already recovered elsewhere. To have contained useful data, the sector needs to have been used (probability ϕ) and containing user data (probability $(k-2)/k$). This gives the loss rate as

$$\frac{1}{2} \times \frac{(k-1)(k-2)}{(n-1)(n-2)} \times \tau \times \phi \times \frac{k-2}{k} \times \frac{B}{T}.$$

As Figure 9 shows, the loss rate due to LSE falls by a factor of about 100. Whether this is acceptable, depends on the data stored, since a single sector failure can result in much more data becoming unusable. Since the dataloss rate due to latent sector error can be controlled by scrubbing and is considerably lower than the dataloss rate due to complete disk failure, the important factor in assessing its impact is this magnification factor of a sector failure. A single failed sector usually renders a file unusable, but that file is typically only part of a system of resources. This magnification effect depend entirely on the nature of the data stored in the failed sector. For most types of data a factor of 2×10^5 separating a typical dataloss rate due to disk errors from the rate due to latent sector errors is high enough that we can consider the former (rate of loss due to disk error) to represent the critical rate in the system. We close this discussion with the *caveat* that disk failures and LSE are not the only causes for dataloss, and that eventually, factors such as environmental effects (*e.g.*, power loss, overheating, fire) and operational errors begin to become more important. Assessing those is beyond the capacity of a general analysis.

IV. Conclusions

The overall data loss for an exabyte filesystem will remain high even if we upgrade all the underlying storage components to more reliable declustered RAID-6 schemes. While such dataloss may be acceptable in usage scenarios that allow the reconstruction of missing data (such as scientific simulations and similar HPC applications), or in cases where the economic losses from losing data are negligible, it is a growing restriction on the feasibility and usability of exascale storage systems. There are simply some applications where data loss is a failure of the system. Based on our analytical results we can conclude that simply focusing on making individual storage bricks more reliable is not a solution to this problem. We contend that this demands a broadening of the solution space being investigated and we offer two directions in which we will find such solutions. The first is a focus on the data, the second is a focus on the infrastructure.

A system can be made effectively more reliable when given a better understanding of data being protected, and a better understanding of what constitutes a failure to the users of the system. For example, to permanently lose a source file from a code project will likely render the entire project unusable. To lose a second source file from the same project does not effectively change this state of affairs for the affected project, but to lose a second source file from a different project is a far more serious problem. This argues that to build a more reliable system, it would be worthwhile to incorporate semantic knowledge of the overlying data to inform placement decisions. In this manner the failure of one storage brick could be contained to the smallest number of overlying data sets. This is a similar approach to that pursued by the Perses project [18].

A system could also be made more reliable when given a better understanding of the the infrastructure upon which it is built. While device failures are often analyzed as being independent events, this is likely not true at exascales. As such, rather than a blind insistence on building using ever more reliable individual components, we advocate the building of more reliable architectures that can inform reliable data placement based on the physical nature of the underlying infrastructure. An event that can result in multiple drive failures, for example, is the physical damage of a single rack or its interconnection interfaces. A system that disperses data across physical groupings would therefore decrease the likelihood of such correlated failures affecting data that has not been protected via a parity or replication scheme.

Simply building exascale storage systems as a conglomeration of more reliable individual nodes will not scale. Treating the data upon it as equally important series of anonymous bytes, or treating the infrastructure below it as an independent sea of identical unrelated bricks, are implied by such a device-centric approach. As such considerations can be addressed by a software layer at or below the object level, or at or above the volume level, means that we have opportunities to broaden reliable storage research and build more reliable systems.

References

[1] M. Abd-El-Malek, W. Courtright II, C. Cranor, G. Ganger, J. Hendricks, A. Klosterman, M. Mesnier, M. Prasad, B. Salmon, R. Sambasivan *et al.*, "Ursa minor: versatile cluster-based storage," in *Conference on File and Storage Technologies*, 2005, pp. 59–72.

[2] G. Alverez, W. Burkhard, L. Stockmeyer, and F. Cristian, "Declustered disk array architectures with optimal and near-optimal parallelism," in *Proceedings, 25th International Symposium on Computer Architecture*. IEEE, 1998, pp. 109–120.

[3] L. Bairavasundaram, A. Arpaci-Dusseau, R. Arpaci-Dusseau, G. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, pp. 1–28, 2008.

[4] R. Fontana, G. Decad, and S. Hetzler, "The impact of areal density and millions of square inches (MSI) of produced memory on petabyte shipments for tape, NAND flash, and HDD storage class," in *Proceedings, 29th IEEE Conference on Massive Data Storage*, 2013.

[5] R. E. Fontana, S. R. Hetzler, and G. Decad, "Technology roadmap comparisons for tape, HDD, and NAND flash: implications for data storage applications," *IEEE Transactions on Magnetics*, vol. 48, no. 5, pp. 1692–1696, 2012.

[6] S. Frølund, A. Merchant, Y. Saito, S. Spence, and A. Veitch, "Fab: enterprise storage systems on a shoestring," in *Hot Topics in Operating Systems*, 2003, pp. 133–138.

[7] J. Gray, "Storage bricks have arrived," in *Invited Talk at the First USENIX Conference on File And Storage Technologies (FAST02)*, 2002.

[8] M. Holland and G. Gibson, "Parity declustering for continuous operation in redundant disk arrays," in *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 1992, pp. 23–35.

[9] I. Iliadis, R. Haas, X. Hu, and E. Eleftheriou, "Disk scrubbing versus intra-disk redundancy for high-reliability raid storage systems," in *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. ACM, 2008, pp. 241–252.

[10] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk scrubbing versus intradisk redundancy for raid storage systems," *ACM Transactions on Storage*, vol. 7, no. 2, pp. 5:1–5:42, Jul. 2011.

[11] K. Keeton, C. Santos, D. Beyer, J. Chase, and J. Wilkes, "Designing for disasters," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 59–62.

[12] A. Riska and E. Riedel, "Idle read after write: IRAW," in *USENIX 2008 Annual Technical Conference*. USENIX Association, 2008, pp. 43–56.

[13] E. Rozier, W. Belluomini, V. Deenadhayalan, J. Hafner, K. Rao, and P. Zhou, "Evaluating the impact of undetected disk errors in raid systems," in *IEEE/IFIP International Conference on Dependable Systems & Networks (DNS), 2009*. IEEE, 2009, pp. 83–92.

[14] B. Schroeder, S. Damouras, and P. Gill, "Understanding latent sector errors and how to protect against them," *ACM Transactions on storage (TOS)*, vol. 6, no. 3, p. 9, 2010.

[15] T. Schwarz and W. Burkhard, "Almost complete address translation (ACATS) disk array declustering," in *Proceedings of the Eighth IEEE Symposium on Parallel and Distributed Processing*, 1996, pp. 324–331.

[16] T. Schwarz, J. Steinberg, and W. Burkhard, "Permutation development data layout (PDDL)," in *Proceedings of the 5th International Symposium on High-Performance Computer Architecture*. IEEE, 1999, pp. 214–217.

[17] T. Schwarz, Q. Xin, E. Miller, D. Long, A. Hospodor, and S. Ng, "Disk scrubbing in large archival storage systems," in *Proceedings of the IEEE 12th Annual International Symposium onn Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*. IEEE, 2004, pp. 409–418.

[18] A. Wildani, E. Miller, I. Adams, and D. Long, "Perses: Data layout for low impact failures," SSRC, UCSC, Tech. Rep., 2012.

[19] Q. Xin, E. Miller, T. Schwarz, D. Long, S. 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*. IEEE, 2003, pp. 146–156.