Differential RAID: Rethinking RAID for SSD Reliability

Asim Kadav
University of Wisconsin
Madison, WI
kadav@cs.wisc.edu

Mahesh Balakrishnan
Microsoft Research Silicon Valley
Mountain View, CA
maheshba@microsoft.com

Vijayan Prabhakaran
Microsoft Research Silicon Valley
Mountain View, CA
vijayanp@microsoft.com

Dahlia Malkhi
Microsoft Research Silicon Valley
Mountain View, CA
dalia@microsoft.com

ABSTRACT

Deployment of SSDs in enterprise settings is limited by the low erase cycles available on commodity devices. Redundancy solutions such as RAID can potentially be used to protect against the high Bit Error Rate (BER) of aging SSDs. Unfortunately, such solutions wear out redundant devices at similar rates, inducing correlated failures as arrays age in unison. We present Diff-RAID, a new RAID variant that distributes parity unevenly across SSDs to create age disparities within arrays. By doing so, Diff-RAID balances the high BER of old SSDs against the low BER of young SSDs. Diff-RAID provides much greater reliability for SSDs compared to RAID-4 and RAID-5 for the same space overhead, and offers a trade-off curve between throughput and reliability.

1. INTRODUCTION

Solid State Devices (SSDs) have emerged in the last few years as a viable secondary storage option for laptops and personal computers. Now, SSDs are poised to replace conventional disks within large-scale data centers, potentially providing massive gains in I/O throughput and power efficiency for high-performance applications. The major barrier to SSD deployment in such settings is cost [3]. This barrier is gradually being lifted through high-capacity Multi-Level Cell (MLC) technology, which has driven down the cost of flash storage significantly.

Yet, MLC devices are severely hamstrung by low endurance limits. Individual flash blocks within an SSD require expensive erase operations between successive page writes. Each erasure makes the device less reliable, increasing the Bit Error Rate (BER) observed by accesses. Consequently, SSD manufacturers specify not only a maximum BER (usually between 10^{-14} to 10^{-15} [2], as with conventional hard disks), but also a limit on the number of erasures within which this BER guarantee holds. For MLC devices, the rated *erasure limit* is typically 5,000 to 10,000 cycles per block; as a result, a write-intensive workload can wear out the SSD within months. Also, this erasure limit continues to decrease as MLC devices increase in capacity and density. As a consequence, the reliability of MLC devices remains a paramount concern for its adoption in servers [4].

In this paper, we explore the possibility of using device-level redundancy to mask the effects of aging on SSDs. Clustering options such as RAID can potentially be used to tolerate the higher BERs exhibited by worn out SSDs. However, these techniques do not automatically provide adequate protection for aging SSDs; by balancing write load across devices, solutions such as RAID-5 cause all SSDs to wear out at approximately the same rate. Intuitively, such solutions end up trying to protect data on old SSDs by storing it redundantly on other, equally old SSDs. Later in the paper, we quantify the ineffectiveness of such an approach.

We propose Differential RAID (Diff-RAID), a new RAID technique designed to provide a reliable server storage solution using MLCs. Diff-RAID leverages the fact that the BER of an SSD increases continuously through its lifetime, starting out very low and reaching the maximum specified rate as it wears out. Accordingly, Diff-RAID attempts to create an age differential among devices in the array, balancing the high BER of older devices against the low BER of young devices.

To create and maintain this age differential, Diff-RAID modifies conventional RAID in two ways. First, it distributes parity unequally across devices; since parity blocks are updated more frequently than data blocks due to random writes, this unequal distribution forces some devices to age faster than others. Diff-RAID supports arbitrary parity assignments, providing a trade-off curve between throughput and reliability. Second, Diff-RAID redistributes parity during device replacements to ensure that the oldest device in the array always holds the most parity and ages at the highest rate. When the oldest device in the array is replaced at some threshold age, its parity blocks are assigned across all the devices in the array and not just to its replacement.

Diff-RAID's ability to tolerate high BERs on aging SSDs provides multiple advantages. First, it provides a much higher degree of reliability for SSD arrays compared to standard RAID-5 or RAID-4 configurations. Second, it opens the door to using commodity SSDs past their erasure limit,

protecting the data on expired SSDs by storing it redundantly on younger devices. Third, it potentially reduces the need for expensive hardware Error Correction Codes (ECC) in the devices; as MLC densities continue to increase, the cost of masking high error rates with ECC is prohibitive. These benefits are achieved at a slight cost in throughput degradation and in device replacement complexity; these trade-offs are explored in detail below.

2. BACKGROUND

NAND-based SSDs exhibit failure behaviors which differ substantially from hard disks. Most crucially, the Bit Error Rate (BER) of modern SSDs is uniform across all disk pages, and grows strongly with the *number of updates* the SSD receives. In order to understand this behavior, we briefly overview basic features of SSDs.

SSDs are composed of *pages*, which are the smallest units that can be read or programmed (written). Pages are structured together into larger units called *blocks*, which are the smallest units that can be erased (all bits reset to 1s). Sharing erasure circuitry across multiple pages allows for higher bit densities by freeing up chip area, reducing overall cost. Bits can only be programmed in one direction, from 1s to 0s; as a result, a rewrite to a specific page requires an erasure of the block containing the page.

Modern SSDs provide excellent performance despite these design restrictions by using wear-levelling algorithms that store data in log format on flash. These wear-levelling algorithms ensure that wear is spread evenly across the SSD, resulting in a relatively uniform BER across the SSD. However, SSDs are still subject to an *erasure limit*, beyond which the BER of the device becomes unacceptably high. SSDs typically use hardware ECC to correct bit errors and extend the erasure limit; in this paper, we use the term BER to refer to the post-ECC error rate, also called the Uncorrectable Bit Error Rate (UBER), unless specified otherwise.

First-generation SSDs used SLC (Single-Level Cell) flash, where each flash cell stores a single bit value. This variant of flash has relatively high endurance limits – around 100,000 erase cycles per block – but is prohibitively expensive; as a result, it is now used predominantly in top-tier SSDs meant for high-performance applications. Recently, the emergence of MLC (Multi-Level Cell) technology has dropped the price of SSDs significantly (for example, the Intel X25-M [1] is a popular MLC SSD priced at $4 per GB at the time of writing), storing multiple bits within each flash cell to expand capacity without increasing cost. However, MLC devices have a major drawback: the endurance limit drops to around 10,000 erase cycles per block. As flash geometries shrink and MLC technology packs more bits into each cell, the price of SSDs is expected to continue decreasing; unfortunately, their pre-ECC BER is expected to increase, creating the possibility of lower erasure limits.

In this paper, we re-examine the use of standard device-level redundancy techniques in face of the unreliability characteristics of MLC devices, as they approach and cross the erasure limit. Since, as we noted above, SSDs differ from hard disks in fundamental ways, this requires a rethink of traditional redundancy solutions such as RAID.

3. WHY NOT RAID?

RAID techniques induce similar update loads on multiple disks, which in turn, may induce correlated failures in SSDs as noted above. For RAID-1, RAID-10 and RAID-01, mirrors are exposed to identical workloads and grow old together. All the devices in RAID-5 receive roughly equivalent workloads and age at similar rates. RAID-4 does impose a slightly staggered wear-out schedule; the parity device wears out at a much faster rate than the other devices for workloads dominated by random writes, and has to be replaced much before them. However, the non-parity devices in RAID-4 age at similar rates. Thus, each of these RAID options imposes a lock-step wear-out schedule on devices in the array.

As long as each SSD is replaced immediately as it hits the erasure limit, the SSD array is as reliable as an array constructed from hard disks of comparable maximum BER. However, if the SSDs are allowed to continue past their erasure limit, the reliability of the array falls drastically. Hence, when one of the devices fails, the probability of another failure during the RAID reconstruction is very high. Consequently, the overall reliability of conventional RAID arrays of SSDs varies highly, hitting a peak as multiple devices reach their erasure limit simultaneously. A more quantitative analysis of this is given in Section 5 below.

Essentially, the wear-out schedule imposed on different devices by a redundancy scheme heavily impacts the overall reliability of the array. The load balancing properties of schemes such as RAID-5 – which make them so attractive for hard disks – result in lower array reliability for SSDs. They induce a lock-step SSD wear-out schedule that leaves the array vulnerable to correlated failures.

Required is a redundancy scheme that can distribute writes unevenly across devices, creating an imbalance in update load across the array. Such a load imbalance will translate into different aging rates for SSDs, allowing staggered device wear-outs over the lifetime of the array and reducing the probability of correlated failure. This observation leads us to introduce novel RAID variants, which we collectively name Diff-RAID.

4. DESIGN

The goal of Diff-RAID is to use device-level redundancy to mask the high BERs of aging SSDs. The basic idea is to store data redundantly across devices of different ages, balancing the high BER of old SSDs past their erasure limit against the low BER of young SSDs. Diff-RAID implements this idea by creating and maintaining age differentials within a RAID array.

We measure the age of a device by the average number of cycles used by each block in it, assuming a perfect wear-levelling algorithm that equalizes this number across blocks. For example, if each block within a device has used up roughly 7,500 cycles, we say that the device has used up 7,500 cycles of its lifetime.

Diff-RAID controls the aging rates of different devices in the array by distributing parity blocks unevenly across them. Random writes cause parity blocks to attract much more

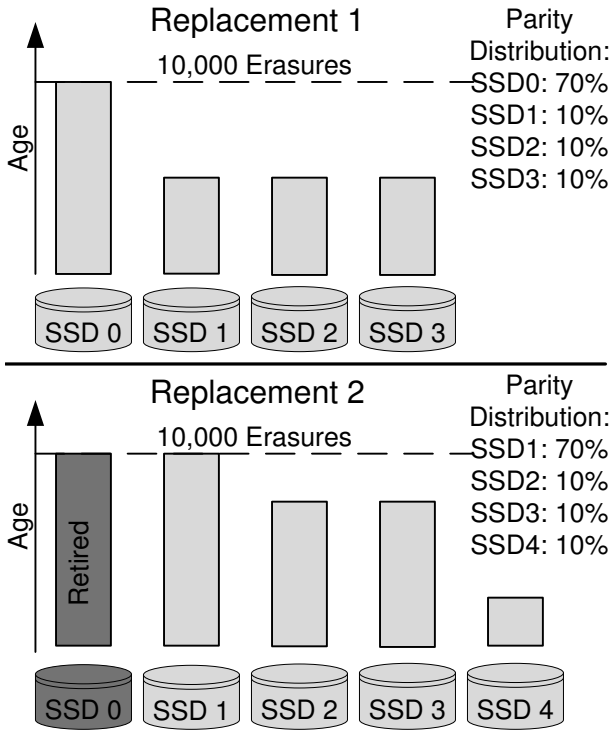

Figure 1: Age distribution of the array during its first two device replacements; age-driven shuffling ensures that SSD-1 holds 70% of the parity after the first replacement.

write traffic than data blocks; on every random write to a data block, the corresponding parity block has to be updated as well. As a result, an SSD with more parity receives a higher number of writes and ages correspondingly faster.

We represent parity assignments with n-tuples of percentages; for example, $(40, 15, 15, 15, 15)$ represents a 5-device array where the first device holds 40% of the parity and the other devices store 15% each. An extreme example of uneven parity assignment is RAID-4, represented by $(100, 0, 0, 0, 0)$ for 5 devices, where the first device holds all the parity. At the other extreme is RAID-5, represented by $(20, 20, 20, 20, 20)$ for 5 devices, where parity is distributed evenly across all devices.

For a workload consisting only of random writes, it is easy to compute the relative aging rates of devices for any given parity assignment. For an n device array, if a_{ij} represents the ratio of the aging rate of the ith device to that of the jth device, and p_i and p_j are the percentages of parity allotted to the respective devices, then:
$a_{ij} = \frac{p_i*(n-1)+(100-p_i)}{p_j*(n-1)+(100-p_j)}$

In the example of 5-device RAID-4, the ratio of the aging rate of the parity device to that of any other device would be $\frac{100*4+0}{0*4+100} = 4$; in other words, the parity device ages four times as fast as any other device. Since RAID-4 is an extreme example of uneven parity assignment, it represents an upper bound on the disparity of aging rates in an array; all other parity assignments will result in less disparity, with RAID-5 at the other extreme providing no disparity.

Ideally, we would like the aging rate of a device to be proportional to its age, so that the older a device, the faster it ages. The intuition here is simple: the presence of an old SSD makes the array more unreliable, and hence we want to use up its erase cycles rapidly and remove it as soon as possible. Conversely, young SSDs make the array more reliable, and hence we want to keep them young as long as possible. If two SSDs are at the same age, we want to age one faster than the other in order to create an imbalance in ages.

To understand this intuition better, consider a conventional RAID-4 array. With a workload of random writes, the data devices in the array will be at approximately 2,500 erase cycles each when the parity device hits 10,000 cycles and is replaced. After three such replacements of the parity device, the data devices will all be at 7,500 cycles, resulting in an aging and unreliable array.

Consequently, on every device replacement, we shuffle the logical positions of the devices in the array to order them by age, before applying the parity assignment to them. In the simple example of RAID-4, each time a parity device is replaced, the oldest of the remaining data devices becomes the new parity device (breaking ties arbitrarily), whereas the new replacement device becomes a data device.

Figure 1 illustrates this scheme for a 4-device array with a parity assignment of $(70, 10, 10, 10)$; by the formula given previously, this assignment results in the first device aging twice as fast as the other three devices. As a result, when SSD-0 reaches 10,000 erase cycles, the other three SSDs are at 5,000 cycles. When SSD-0 is replaced, SSD-1 is assigned 70% of the parity blocks, and the other SSDs (including the new one) are assigned 10% each.

Interestingly, if we continue this age-driven shuffling over multiple replacements for a given parity assignment, the device ages observed at replacement time eventually converge to a stationary distribution. Figure 2 plots the distribution of device ages at replacement time for a 5-device array with a parity assignment of $(100, 0, 0, 0, 0)$; essentially, this is RAID-4 with the parity shifting to the oldest remaining device on each replacement. After a small number of replacements, the ages of the devices at replacement time converge so that the oldest remaining device is always at 5,000 erase cycles, with the other devices at 3,750, 2,500 and 1,250 cycles respectively. We can calculate the convergent age distribution for any parity assignment — we omit details for lack of space.

Diff-RAID provides a trade-off between reliability and throughput. The Diff-RAID parity assignment corresponding to RAID-4 provides the least throughput – with the single parity device acting as a bottleneck – but the highest reliability, and the one corresponding to RAID-5 provides the highest throughput but the least reliability. Since Diff-RAID supports any arbitrary parity assignment, administrators can choose a point on the trade-off curve that fits their requirements. Potentially, Diff-RAID itself could adaptively choose the most reliable parity assignment for a given throughput

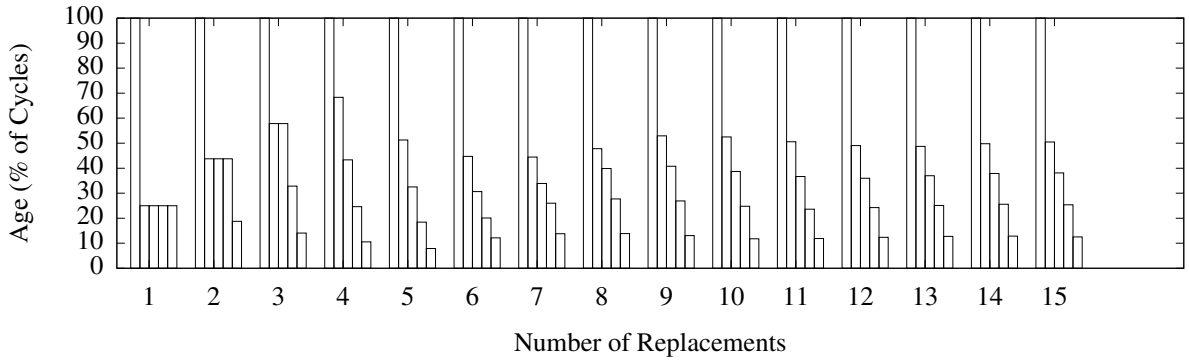

Figure 2: Age distribution of devices for Diff-RAID parity assignment $(100, 0, 0, 0, 0)$ (i.e., RAID-4 with age-driven shuffling on device replacements). Each set of bars represents the age distribution of devices in the array when the oldest SSD wears out.

level as the write performance required by the workload fluctuates.

When a device is replaced and a new device takes its place, age-driven shuffling of the parity assignments has to occur. One implementation option involves treating a device replacement as a failure and triggering the RAID reconstruction process to rebuild the contents of the replaced SSD on a new device, but modifying the reconstruction logic so that it interchanges parity blocks and data blocks on different SSDs to achieve the required parity assignment over a shuffled ordering of the devices. We expect to explore different implementation options that perform device replacement without triggering RAID reconstruction logic, as well.

5. SIMULATION RESULTS

In this section, we show through simulation that Diff-RAID provides much better reliability for SSD arrays than conventional RAID. Also, we show that Diff-RAID can push inexpensive, low-end SSDs to twice their rated erasure limit without compromising the overall reliability of the array. We model the Bit Error Rate of an individual SSD using data in a study published by Intel and Micron [2]. That paper contains raw (pre-ECC) BERs for four MLC devices up to 10,000 erase cycles, of which one device is rated at 5,000 cycles. We use the raw BER curve of this 5,000 cycle device as a starting point, assume 2-bit ECC, and modify the resulting UBER data slightly to obtain a smooth curve that hits an UBER of 10^{-14} at 5,000 cycles, comparable to the UBER of SATA hard disks.

We quantify the reliability of a RAID array by considering a common failure mode — a disk fails in the array, and during reconstruction an uncorrectable bit error is encountered elsewhere in the array, resulting in the loss of data. For a 6-disk RAID-5 array of 80 GB SATA disks (with UBER equal to 10^{-14}), the probability of data loss given a disk failure is around 3%, which remains fairly constant throughout the lifetime of the array. In other words, for every 100 instances of a disk failure in such an array, roughly 3 will experience some data loss.

Figure 3 illustrates the difference between conventional RAID-5 and RAID-4 used with SSDs and a Diff-RAID parity assignment corresponding to RAID-4. The parity assignment used by Diff-RAID is $(1, 5, 5, 5, 5, 5)$ — effectively, the only difference between conventional RAID-4 and the Diff-RAID version is the age-driven shuffling of parity on each device replacement. Conventional RAID-5 causes all SSDs age in lock-step fashion, and conventional RAID-4 does so with the data devices; as a result, the probability of data loss on an SSD failure climbs to almost 1 for both solutions as the array ages, and periodically resets to almost zero whenever all SSDs are replaced simultaneously. In contrast, Diff-RAID never experiences a data loss probability of more than 17%, and once the age distribution converges, it consistently maintains a data loss probability under 4%, which is comparable to the 3% data loss probability of a SATA disk array. In this simulation, we assume that the oldest device in the array fails initially, and compute the probability of data loss in the remaining devices; if any device is allowed to fail initially with equal probability, the RAID-5 curve remains unchanged, the RAID-4 curve does not change significantly, and the Diff-RAID curve is bounded at 35%.

Figure 4 shows the space of possible Diff-RAID configurations for a 6-device array. Each point in this graph represents a different parity assignment; on the x-axis is the throughput of the solution as a fraction of the total array throughput, and on the y-axis is the probability of data loss upon a device failure. As expected, the parity assignment corresponding to RAID-4 is on one extreme of the trade-off curve, providing high reliability but very low throughput, and the assignment corresponding to RAID-5 is at the opposite end, with high throughput but very low reliability.

Sequential Writes: So far, we have assumed a workload consisting only of random writes. Sequential writes that update the entire stripe can potentially reduce the reliability of a Diff-RAID array, since they reduce the ability of parity blocks to imbalance write load. When we ran the experiment shown in Figure 3 with a workload consisting of 10% sequential writes, the worst case unreliability for Diff-RAID in the initial period climbed to 23%; after convergence, the unreliability stays under 8%. With a workload where sequential writes comprise 50% of all writes, Diff-RAID is still bounded at 25% unreliability after convergence, providing 4 times the reliability of conventional RAID-4 and RAID-5. In

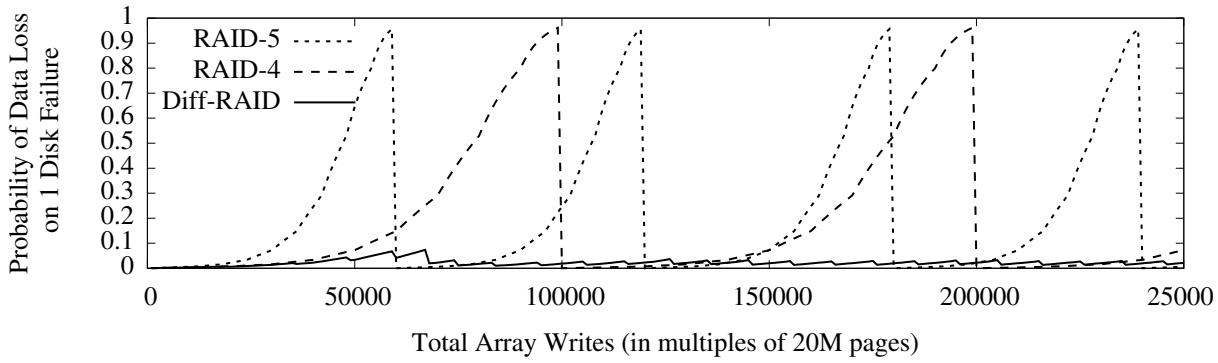

Figure 3: RAID-5 and RAID-4 versus Diff-RAID for a 6-device array. Diff-RAID is configured to use a RAID-4 parity assignment – $(100, 0, 0, 0, 0, 0)$ – but differs by performing age-driven shuffling on device replacements.

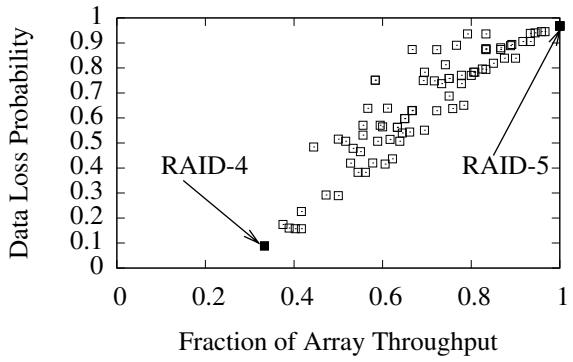

Figure 4: Throughput and Reliability at different Diff-RAID parity assignments for a six device RAID array.

the worst case where all writes are sequential, Diff-RAID degrades to provide reliability identical to conventional RAID.

6. FUTURE WORK

We are currently implementing Diff-RAID as a software RAID driver. We hope that such an implementation will provide insight into the end-to-end performance of SSD-based storage stacks. With commodity RAID controllers – or software RAID running on conventional machines – it is possible that the RAID logic is the bottleneck for the random write throughput of the array. In this case, we will not see the throughput reliability trade-off described in Figure 4; instead, the throughput of all configurations will lie under an upper threshold.

We expect the Diff-RAID implementation to have a policy component that determines which parity assignment to use. As mentioned previously, the policy could select the most reliable configuration below the application's throughput requirement; alternately, it could factor in the maximum throughput possible with the hardware setup. Also, Diff-RAID could allow the application to specify a threshold level of data loss probability and select the highest throughput configuration under that threshold.

While this paper focuses on parity-based RAID, other schemes which use mirroring (such as RAID-1 or RAID-10) are vulnerable to correlated aging as well. Diff-RAID does not immediately generalize to mirroring, since it uses the higher write load of parity blocks to age SSDs differentially. Different techniques are required to combat correlated aging in mirrored RAID setups.

In general, the large-scale introduction of SSDs into clustered settings poses many interesting questions. System administrators may be required to track the age of individual devices and define proactive replacement policies. Also, considering a larger pool of SSDs – as opposed to a single array – may allow for solutions that are more effective at creating age differentials.

7. CONCLUSION

Diff-RAID is a new RAID variant designed specifically for SSDs. It distributes parity unevenly across the array to force devices to age at different rates, and redistributes this parity on each device replacement. By doing so, Diff-RAID maintains an age differential between devices in the RAID array, balancing the high BER of aging devices against the low BERs of younger devices and reducing the chances of correlated failures. Compared to conventional RAID-5 or RAID-4, Diff-RAID provides a higher degree of reliability for SSDs for the same space overhead, and provides a trade-off curve between throughput and reliability. It can potentially be used to operate SSDs past their erasure limit, as well as reduce the need for expensive ECC within SSDs.

8. REFERENCES

[1] Intel Corporation. Intel X18-M/X25-M SATA Solid State Drive. `http://download.intel.com/design/flash/nand/mainstream/mainstream-sata-ssd-datasheet.pdf`.

[2] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill. Bit Error Rate in NAND Flash Memories. In *IRPS 2008: IEEE International Reliability Physics Symposium*, 2008.

[3] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In *EuroSys 2009*. ACM, 2009.

[4] D. Roberts, T. Kgil, and T. Mudge. Integrating NAND Flash Devices onto Servers. *Commun. ACM*, 52(4):98–103, 2009.