Exploiting heterogeneity in cluster storage systems

Saurabh Kadekodi

November 16, 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Gregory R. Ganger, Co-chair
Rashmi Vinayak, Co-chair
Garth A. Gibson
Remzi Arpaci-Dusseau (University of Wisconsin-Madison)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Abstract

Large-scale storage systems include a heterogeneous mix of storage devices with different reliability, capacity and performance characteristics. This dissertation explores the benefits of explicitly factoring in heterogeneity in the characteristics of storage devices belonging to a single tier, for cheaper redundancy and more load-balanced data placement.

Data reliability in today's cluster storage systems is agnostic to the failure rate differences observed between storage devices in the same tier. Using a production dataset of over 100,000 hard disks, we show that there is a significant diversity in hard disk failure rates, as a function of make/model. We build a case for failure rate tailored redundancy, and design and prototype HeART (Heterogeneity-Aware Redundancy Tuner), an online automated framework to aid the process. HeART continuously monitors the failure rates of different disk groups, and employs robust online algorithms to safely and accurately identify redundancy schemes for each disk group, while meeting a defined reliability target. This leads to lower level of redundancy for devices with low failure rate and more redundancy for devices with high failure rate. Redundancy informed by HeART would result in 13-44% fewer disks than one-scheme-fits-all approaches in the production dataset: 33-44% compared to 3-replication and 13-19% compared to erasure codes like 6-of-9 or 10-of-14.

Technological advancements make capacity heterogeneity also prevalent, and unavoidable in large cluster storage systems. Data placement algorithms agnostic to capacity heterogeneity often cause heavy spindle imbalance across the disk fleet. This load imbalance worsens when there is data heat (accesses per unit time) heterogeneity across different clients. We propose a data placement algorithm that accounts for each dataset's current heat and expected *rate of cooling* in addition to storage device capacity heterogeneity. These simple heuristics allow for data placement that results in higher storage device utilization, lower rebalancing cost and lower tail latencies.

1 Introduction

Large cluster storage systems almost always include a heterogeneous mix of storage devices, even when using devices that are all of the same type (e.g., Flash SSDs or mechanical HDDs). Commonly, this heterogeneity arises from incremental deployment combined with per-acquisition optimization of which make/model to acquire, such as targeting the lowest cost-per-byte option available at the time. As a result, a given cluster storage system can easily include several makes/models, each in substantial quantity. In this dissertation, we explore the lost opportunities in having overlooked the heterogeneity across different generations/makes/models of storage devices of the same type, belonging to a single storage tier.

Thesis statement: *Exploiting heterogeneity in cluster storage systems can achieve lower storage cost, reduced data access latency and improved device utilization. In particular, decisions regarding the degree of replication in cluster storage systems, when informed by the storage device reliability heterogeneity—even from the same tier—can lead to lower storage cost, and, placement of data when informed by the capacity and data heat heterogeneity can achieve lower data access latency and higher device utilization.*
This dissertation supports the above claim with the following evidence:

1. Identification and evaluation of heterogeneity in real cluster storage systems
2. Enabling cost-effective data reliability via device-reliability-aware redundancy
3. Enabling cost-effective performance via capacity-aware placement

Identification and evaluation of heterogeneity in real cluster storage systems: The disk population in a particular storage tier is diverse in terms of reliability, performance and capacity. This has also been substantiated by prior studies performed across different storage clusters [23, 27]. Our observation on a 100,000+ HDD dataset from Backblaze [8] reconfirms the diversity in both reliability and capacity. We intend to enhance our reliability and capacity heterogeneity evaluation by analyzing the supercomputer disk fleet at Los Alamos National Labs (LANL) and disks belonging to filers at NetApp. The performance and data heat heterogeneity results will be performed on dis-aggregated storage clusters with known client workloads, a typical characteristic of the data center storage subsystems of internet services companies like Google, Microsoft and Facebook.

Cost-effective data reliability via device-reliability-aware redundancy (§ 2): Recent work has shown that storage device reliability varies significantly within SSDs [35]. Our observation confirms that this is true in HDDs as well, destroying the common notion of reliability homogeneity in HDDs. Redundancy level choices (e.g., degree of replication) without considering the heterogeneity in device reliability will frequently either waste capacity or fail to meet a data reliability target. For example: more redundancy is required to achieve a given degree of data reliability on devices with higher failure rates; and, of course, less redundancy is needed in the opposite case. We show that heterogeneity aware data storage that makes device-reliability-aware choices of both data placement (which devices) and redundancy level to achieve particular (possibly client specified) degrees of data reliability can reduce space utilization in cluster storage systems up to 44%.

Cost-effective performance via capacity-aware placement (§ 3): Data placement algorithms agnostic to device capacity heterogeneity can result in heavy spindle imbalance across the disk fleet [37]. Similarly, data placement that does not account for diversity in data heat (accesses per unit time) among datasets leads to high *rebalancing*. Rebalancing is the data movement done for spindle balance by moving the cold data from small disks to large disks, in order to create space for hot data on small disks. High rebalancing costs, similar to high garbage collection costs, cause high tail latencies. We propose a *capacity-aware* and *rebalance-cost-aware* data placement strategy that exploits knowledge of device capacities, space utilization, and each dataset's current data heat and expected *rate-of-cooling*. These simple heuristics allow for much better decisions regarding both initial (creation time) data placement and the additional rebalancing load.

Using the above techniques we explore the cost and performance benefits of explicitly incorporating client workload and storage device heterogeneity in data placement and redundancy decisions for storage devices belonging to a single storage tier. We focus our data-driven analysis on HDDs because they form the primary storage tier in most cluster storage systems even today. Fundamentally, our optimizations are storage device agnostic and can be applied to other storage devices with little modification.

My proposal is divided into two broad parts, techniques to exploit reliability heterogeneity (§ 2), and techniques to exploit capacity and data heat heterogeneity (§ 3). I have detailed a concrete example of tackling reliability heterogeneity on a real-world dataset throughout § 2. Research questions belonging to reliability heterogeneity that still need to be addressed are described in § 2.5. § 3 describes the ideas and process of tackling capacity and data heat heterogeneity. The concrete exploration of this topic is part of the proposed work.

2 Cluster storage systems gotta have HeART

Apart from the obvious performance and capacity differences, different disk makes/models from the same storage tier can also have substantially different reliabilities. For example, Fig. 1 shows the average *annualized failure rates ($AFRs$)* during the useful life (stable operation period) for the 6 HDD make/model-based disk groups that make up more than 90% of the cluster storage system (with 100,000+ disks) used for the Backblaze backup service [8]. The highest failure rate is over $3.5\times$ greater than the lowest, and no two are the same. Schroeder et al. [35] recently showed that different Flash SSD makes/models similarly exhibit substantial failure rate differences.

Despite such differences, redundancy settings (e.g., degree of replication in cluster storage systems) are generally configured as if all of the devices have the same reliability. Unfortunately, this approach leads to configurations that are overly resource-consuming, overly risky, or a mix of the two. For example, if the redundancy settings are configured to achieve a given data reliability target (e.g., a specific *mean time to data loss ($MTTDL$)*) based on the highest AFR of any device make/model (e.g., S-4 from Fig. 1), then too much space will be used for redundancy associated with data that is stored fully on lower AFR makes/models (e.g., H-4A). Continuing this example, our evaluations show that the overall wasted capacity can be up to 44% compared to using 3-replication for all data, and up to 19% compared to uniform use of erasure code settings

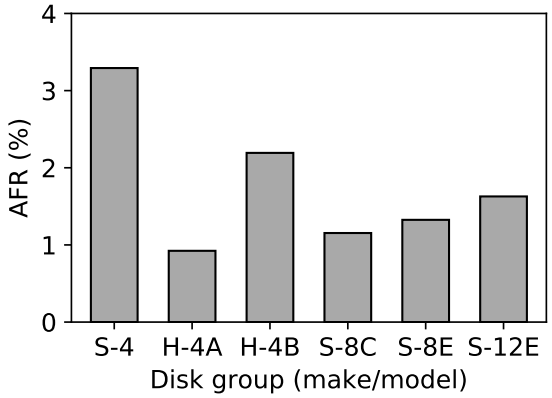


Figure 1: Annualized failure rate (AFR) for the six disk groups that make up >90% of the 100,000+ HDDs used for the Backblaze backup service [8]. Details of each disk group are given in § 2.1.

utilized in real large-scale storage clusters [17, 31]—the direct consequence is increased cost, as more disks are needed. If redundancy settings for all data are based on lower AFRs, on the other hand, then data stored fully on higher-AFR devices is not sufficiently covered and may not achieve the data reliability target, risking data loss.

This part of the dissertation presents **HeART (Heterogeneity-Aware Redundancy Tuner)**, an online tool for guiding exploitation of reliability heterogeneity among disks to reduce the space overhead (and hence the cost) of data reliability. HeART uses failure data observed over time to empirically quantify each disk group's reliability characteristics and determine minimum-capacity redundancy settings that achieve specified target data reliability levels. For the Backblaze dataset of 100,000+ HDDs over 5 years, our analysis shows that using HeART's settings could achieve data reliability targets with 13–44% fewer HDDs, depending on the baseline one-scheme-for-all settings. Even when the baseline scheme is a 10-of-14 erasure code whose overhead is already low, HeART further reduces disk space used by 16%.

Online (real-time) use of observed device reliability requires careful design. HeART uses robust statistical approaches to identify not only a steady-state AFR estimate for each disk group, but also the transitions between deployment phases: infancy→useful life→wearout, as in bathtub curve visualizations. HeART assumes that administrators have a baseline redundancy configuration that would be used in HeART's absence; that same configuration should be used for a disk group, when it is initially deployed. HeART processes failure data for that disk group, during this initial period of 3–5 months, to determine both when infancy ends and a conservative AFR estimate for the useful life period. It also suggests the most space-efficient redundancy settings supported by the storage system that will achieve the specified data reliability target.

Naturally, the useful life period does not last forever. HeART continues to process failure data for each disk group, automatically identifying the onset of the wearout period. At this point, a transition to more conservative redundancy (e.g., the original baseline), and possibly decommisioning, is warranted. Importantly, HeART distinguishes between anomalous failure occurrences (e.g., one-time device-independent events, like a power surge, in which many devices fail together) and true changes in the underlying AFR.

This work makes four primary contributions. First, it highlights an often overlooked aspect of device heterogeneity (reliability) that should be exploited in cluster storage systems, and

quantifies potential cost-and/or-reliability benefits. Second, it confirms the above observation and quantification with analysis of multi-year reliability data from a sizable cluster storage deployment (Backblaze), showing up to 13–44% reduction in the overall number of disks needed to achieve target data reliability. Third, it describes an online tool (HeART) that automatically determines per-disk-group useful life AFRs and durations, and identifies the right redundancy scheme settings for each. Fourth, it shows that 's algorithms are effective with data from large-scale production cluster (Backblaze) and are able to expose the expected capacity savings opportunities without compromising data reliability.

2.1 Having HeART can make you rich

This section builds a case for HeART by showing the benefits of using different redundancy schemes for disk groups exhibiting different reliability characteristics in the same commercially used cluster storage system. We evaluate our case by quantifying the space overhead reduced by adopting the different redundancy schemes.

2.1.1 The Backblaze dataset

Our analysis is based on an open source dataset from a data backup organization, Backblaze [8]. This dataset comprises of over 5 years of disk reliability statistics belonging to a production cluster storage system with over 100,000 HDDs.

We use the standard metric, *annualized failure rate (AFR)*, to describe a disk's fail-stop rate [13, 36]. We calculate AFR using the following formula:

$$AFR\ (\%) = \frac{f_d}{n_1 + n_2 + \ldots + n_d} \times 365 \times 100 \tag{1}$$

where f_d is the number of disks failed in d days and n_i is the number of disks operational during day i.

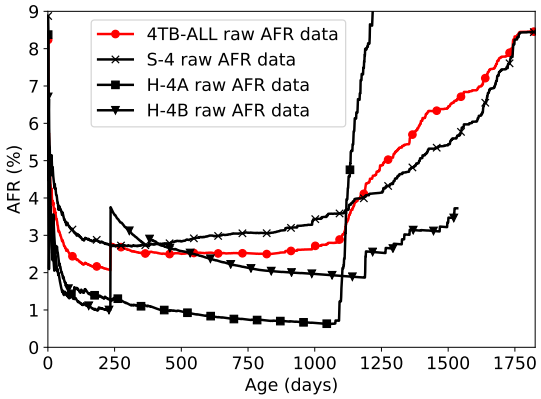


Figure 2: AFR comparison between all 4TB disks grouped together, and disk groups broken down by make/model. The AFR differences in make/model-based grouping enables HeART to perform finer grained optimization leading to higher benefits.

Make/Model	Disk group shorthand	# of disks	Oldest disk age
Seagate ST4000DM000	S-4	37015	5 yrs
HGST HMS5C4040ALE640	H-4A	8715	4.77 yrs
HGST HMS5C4040BLE640	H-4B	15048	4.2 yrs
Seagate ST8000DM002	S-8C	9885	1.99 yrs
Seagate ST8000NM0055	S-8E	14395	1.2 yrs
Seagate ST12000NM0007	S-12E	21581	8 mts

Table 1: The disk groups identified from the Backblaze dataset for reliability heterogeneity analysis. The disk group shorthand above is used to represent the respective makes/models throughout the paper. Each disk group has $\approx$10,000 or more disks.

2.1.2 Disk group formation and varying AFRs

In order to effectively exploit heterogeneity in AFRs of different disk groups, we need to categorize the disks using some parameter that potentially groups disks with similar AFRs together, and has substantially different AFRs across groups. Whichever way we choose to group the disks, in order to gain statistical confidence in the AFR value, we need to ensure that each disk group has a sizeable population. Our definition of a sizeable population is approximately 10,000 or more disks, similar to population sizes in previous reliability studies [24].

There are multiple possible disk groups. Among possible options are grouping by capacity, make/model, usage (e.g., I/O rates per unit time) or operational conditions (e.g., by chassis placement or temperature). Unfortunately we do not have access to the operational conditions or the usage patterns. Therefore, we can only analyze the categorization on the basis of make/model or capacity.

Fig. 2 shows the AFR by considering all 4TB disks as one disk group (red curve with circular marks) and the AFRs of the three make/models of 4TB disks as individual disk groups (black curves). We see significant differences between AFRs when disks are categorized by make/model, suggesting that grouping by capacity is insufficient. HeART groups by make/model. Table 1 shows the six make/model disk groups that make up over 90% of the Backblaze deployment, with their population size, the age of their oldest disk, and the shorthand names we will use throughout the paper.

AFR variation over time. AFR values of each disk group varies over the lifetime of disks. It is well known that the AFR values over a disk's lifetime follow a *bathtub curve* [15, 16, 45] as shown in Fig. 3a.

Fig. 4 shows the AFR behavior versus age of the six disk makes/models. The three disks in the top row clearly exhibit all three phases of the bathtub curve.[1] This is because the oldest disks from the S-4, H-4A and H-4B disk groups in the dataset are old enough to have entered their wearout stages. Since the deployment of S-8C, S-8E and S-12E disks has been more recent, these disk groups have ended their infant mortality, but are yet to enter their wearout stages.

[1]Fig. 4c corresponding to disk H-4B does not completely conform to the bathtub shape. We will discuss this case later in detail.

2.1.3 Quantifying the benefit of varying AFRs

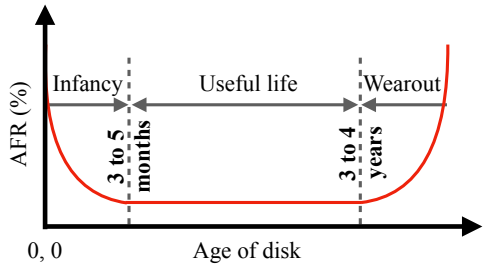
Our goal is to reduce the disk space usage by tailoring the overhead of the redundancy scheme employed to the failure rate of a disk group during its useful life period. We parameterize a redundancy scheme using two parameters n and k, and call it a (n, k) scheme. For any replication based scheme, $k = 1$ and n represents the total number of replicas. For any erasure coding based scheme, k represents the number of data chunks and $(n - k)$ is the number of parity chunks, thus resulting in n chunks in total [2]. For an (n, k) redundancy scheme, the storage overhead is given by $\frac{n}{k}$.

HeART achieves reduction in storage space consumed by explicitly factoring in the widely varying range of AFR values in deciding the appropriate redundancy scheme for each disk group. Based on the canonical bathtub curve (Fig. 3a), and the AFR curves shown in Fig. 4, we conclude that the safest phase to apply lower redundancy (without risking not meeting the reliability target) during a disk group's lifetime is in its useful life period. In Fig. 3b, we show the abstract timeline of a disk group, where r denotes the redundancy scheme applied in each phase. Since all cluster storage systems today use some form of redundancy scheme whose resilience is acceptable to them, we assume that to be the *default* redundancy scheme. Since infancy and wearout periods have higher AFR compared to useful life, for every disk group, the default redundancy scheme should be used for all infancy and wearout periods. This is shown as r_{def} in Fig. 3b. Thus, HeART suggests lower redundancy as compared to the default scheme only during the useful life period, during which AFR values are relatively stable.

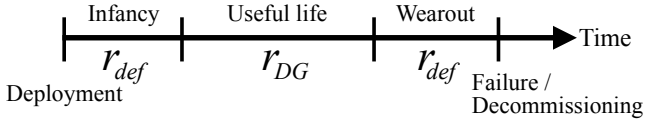
We quantify reliability using the standard reliability metric, *mean time to data loss* ($MTTDL$), which is calculated based on two rates – *mean time to failure* ($MTTF$) and *mean time to repair* ($MTTR$) [26]. In large-scale cluster storage systems, the time it takes to detect that a disk has failed (approximately 15 minutes) dominates the time taken to reconstruct all of its data [17, 20]. This decouples the $MTTR$ of a disk from its features such as storage capacity or make/model, making $MTTDL$ proportional only to $MTTF$ (which is directly computed using its AFR), while allowing us to treat $MTTR$ as a constant.

As soon as a disk group enters its useful life period, HeART chooses a redundancy scheme (r_{DG}) that meets the following conditions:

[2] Although the description of the notation applies only to "systematic" codes, and most of the codes employed in storage systems are indeed systematic, HeART also works with storage systems employing non-systematic codes.

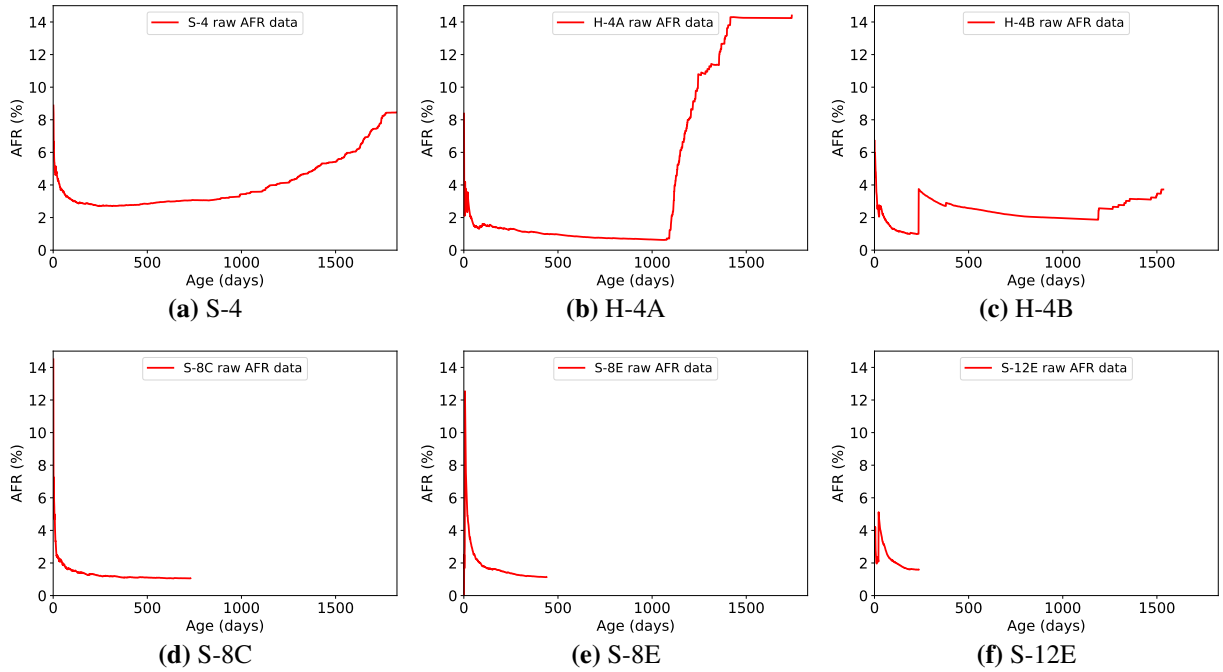


(a) The canonical bathtub curve used to represent disk failure characteristics.



(b) An abstract timeline of a disk group from deployment to failure or decommissioning with the three distinct phases. The notation below denotes the fault tolerance scheme employed during the respective phase.

Figure 3: Canonical bathtub curve and the different redundancy schemes applied during different lifetime phases of any disk.



(a) S-4 **(b)** H-4A **(c)** H-4B

(d) S-8C **(e)** S-8E **(f)** S-12E

Figure 4: Cumulative raw AFR versus age (in days) for all six disk groups being analyzed from the Backblaze dataset.

1. is as reliable as r_{def}, i.e. $MTTDL^{r_{DG}} \geq MTTDL^{r_{def}}$
2. tolerates at least as many failures as r_{def}

According to condition 1 above, we need to set a target $MTTDL$ in order to compare the resilience of different redundancy schemes. Although prior studies have shown $MTTDL$ targets to be as low as 10,000 years [30], in order to ensure that we do not regress on reliability that disks in our dataset can currently offer, we set the target $MTTDL$ to be $MTTDL$ of the default redundancy scheme applied on the disk group with the highest AFR. S-4 is the disk group with highest useful life AFR in the Backblaze dataset (refer Fig. 1). Therefore, for every default redundancy scheme, we will use S-4's $MTTDL$ for that scheme as the target $MTTDL$.

We start by estimating the space reduction for the canonical redundancy scheme: 3-replication. Recall that, under the $(n,\ k)$ notation introduced above, 3-replication is denoted as the $(3, 1)$. Under 3-replication, the $3.5\times$ difference in AFR between disk groups S-4 and H-4A causes a $45\times$ difference in their $MTTDL$s. Thus, we can choose a weaker redundancy scheme (a scheme with lower storage overhead $\frac{n}{k}$), so long as conditions 1 and 2 above are fulfilled.

Multiple redundancy schemes can achieve same or similar $MTTDL$ values. These schemes differ in their stripe width (n) or number of parity chunks per stripe ($n-k$) or both. It is also well known that codes with longer stripe widths achieve the same $MTTDL$ with low space overhead compared to shorter codes. At the same time, long codes suffer from much higher network I/O traffic along with delays because of stragglers, since many more disks have to be accessed when performing reconstruction of failed data [20, 29]. Therefore, we only highlight cost reductions for codes which are at most $2\times$ the stripe width of the default redundancy scheme. For storage systems that can tolerate straggler delays or high network traffic, allowing longer codes might yield even more benefit.

Disk groups		r_{def} = (3, 1)		r_{def} = (9, 6)		r_{def} = (14,10)	
DG	AFR	r_{DG}	**Cost↓**	r_{DG}	**Cost↓**	r_{DG}	**Cost↓**
S-4	3.29%	(3, 1)	**NA**	(9, 6)	**NA**	(14, 10)	**NA**
H-4A	0.92%	(6, 4)	**50%**	(18, 15)	**20%**	(28, 24)	**16%**

Table 2: A sample of the estimated savings achievable through HeART. S-4 disks cannot show benefit because we restrict them to use the default redundancy scheme (r_{def}).

For r_{def} = 3-replication, we can tune the redundancy on H-4A disks to $(6, 2)$ (our limit in terms not going beyond $2\times$ the stripe width of r_{def}) and achieve an $MTTDL$ which is still over $2\times$ that of S-4's $MTTDL$. 3-replication has a storage overhead which is $2\times$ that of $(6, 2)$, causing a 50% reduction in disk space. All other disk groups have AFRs between S-4's AFR and H-4A's AFR, and therefore should expect to see maximum space savings up to 50%.

We will now highlight the space reduction when erasure coding schemes are used as the default. We exemplify using $(9, 6)$ and $(14, 10)$ schemes since they are currently employed in large scale data centers [17, 28, 31] and open source systems like HDFS [11]. For $(9, 6)$ as the default scheme, the $MTTDL$ differences between S-4 and H-4A disks is over $100\times$. Similar to 3-replication, the high AFR differences allow us to tune redundancy to as high as our $2\times$ stripe width limit, i.e. $(18, 15)$ leading to an overall space reduction of 20%. Similarly, when using $(14, 10)$ as the default scheme, the $MTTDL$ differences between S-4 and H-4A is over $500\times$. This again allows us to choose the longest code for H-4A when $r_{def} = (14, 10)$, i.e. $(28, 24)$, providing a space reduction of 16%. The cost reductions are summarized in Table 2.

In personal communication, a very large internet services company revealed that they try very hard to keep the free space in their storage system below 5% in order to minimize operating costs. In light of this information, space savings of 20% and 17% when using erasure codes, essentially for free, builds a solid case for factoring in device reliability heterogeneity in informing choice of redundancy schemes.

2.2 The ways of the HeART

This section describes the challenges, design and implementation of HeART. We also quantify the cost reductions achieved by HeART for the Backblaze dataset.

2.2.1 Identifying challenges in incorporating reliability heterogeneity

There are several challenges in taking the idea presented in § 2.1 to practice.

Challenge 1: Function online and be quick. In making our case for HeART, we made use of the complete failure information (i.e. the full bathtub curve) for the disk groups. This helped in clearly identifying the 3 stages of a disk group's lifetime and how AFR varies in each of the stages. However, in practice, AFR values for disk groups deployed in cluster storage systems can only be known in an online fashion (i.e. as a continuous stream of reliability data, as it is generated). Furthermore, the crux of our cost reduction comes from quickly tuning the redundancy scheme as soon as we are confident of a disk group having entered its useful life period. Thus, our first challenge in building HeART is that it needs to function in an online

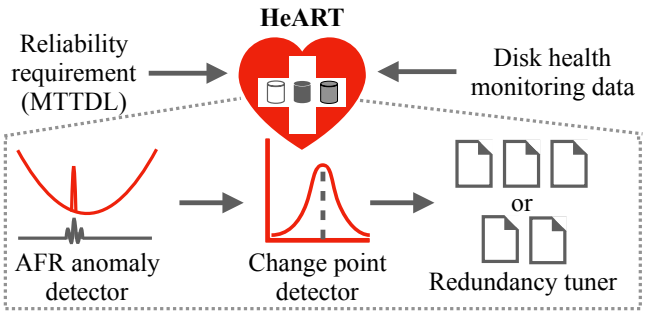


Figure 5: Schematic diagram of HeART. Components include an anomaly detector, an online change point detector and a redundancy tuner.

fashion taking continuous stream of disk health data as input and can quickly react to the changes in the failure rate.

Challenge 2: Be accurate. It is important to correctly identify the three different stages of the bathtub curve for each disk group (recall Fig. 3a). If we are hasty in declaring the end of the infancy period or lax in identifying end of useful life, we might not meet the reliability target because of having tuned the redundancy to a relatively low failure rate during the useful life period. In contrast, if we are too lax about declaring end of infancy or too hasty in declaring onset of the wearout stage, the opportunity of cost reduction will diminish.

Challenge 3: Filter-out anomalies. Anomalous events like power outages, natural disasters or human error can cause large numbers of disks to fail at once. It is important to distinguish between an accidental rise in AFR due to such anomalous events versus the rise in AFR due to onset of the wearout stage. Our third challenge is to perform AFR anomaly detection to avoid prematurely declaring end of useful life, consequently reducing the window of opportunity for cost reduction. At the same time, HeART needs to exercise caution so as to not treat a genuine rise in AFR as an anomaly, which risks missing reliability targets.

High-level description of HeART. We address these challenges in HeART by designing it as shown in Fig. 5. HeART assumes the existence of a periodic disk health assessment mechanism already in place, which indeed exists in most large-scale cluster storage systems (according to personal communication with data center operators of large internet services companies). Throughout infancy, the default redundancy scheme (r_{def}) is used to protect the data stored on a disk group. Every periodic capture of disk health is passed through an *anomaly detector*. Following an anomaly check, the cumulative AFR of every disk group is passed through a *change point detector*, which checks if a transition to different phase of life has occurred. Once the change point detector announces start of the useful life period, HeART suggests a new redundancy mechanism for the useful life of the disk group (r_{DG}). The AFR at the end of infancy padded with a tunable buffer is used to calculate $MTTDL^{r_{DG}}$. The buffer is introduced to tolerate the fluctuation of AFR during the useful life period. Details of the buffer and its effect on useful life are explained in § 2.3.3. We call this as AFR_{DG}, or the *determined useful life* AFR. HeART keeps checking for anomalies and change points throughout the useful life period as well. When the change point detector marks the end of useful life, HeART raises an alert to reset the redundancy scheme to r_{def} to handle the increased AFR during wearout, as was handled in the absence of HeART.

The remainder of this section describes our approach to addressing the above mentioned

challenges. While other options may perform even better, we leverage established tools and algorithms from the online algorithms and time-series analysis literature; our evaluations indicate that they are effective. We show the efficacy of HeART using the Backblaze dataset in § 2.3.

2.2.2 Online anomaly detection

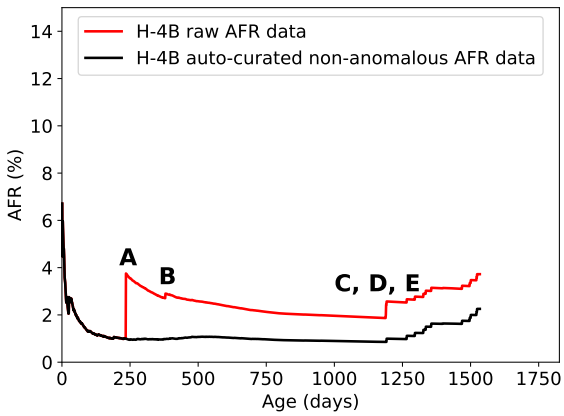
Incidents like losing power to a rack of disks, a natural disaster, or an accident, cause a large number of failures resulting in a sudden rise in AFR. Such bulk failures are typically handled by defining appropriate failure domains and placing data spread across the failure domains [28, 29]. Such failures are not reflective of the true rise in AFR because of wearout, and therefore HeART considers these incidents as anomalies. It is important to note that the benefits we extract from exploiting the reliability heterogeneity are proportional to the length of the useful life period, and therefore prematurely announcing wearout stage due to an anomaly would significantly diminish achievable gains.

We use the H-4B disks as a motivating example for anomaly detection (shown in Fig. 6a). The raw data (red) curve shows that just after a few days into its useful life, there are large spikes in the AFR curve for drives that are about 235 days old (point A) and 380 days old (point B). Further along, we observe three more spikes that are in succession for disks that are about 1200 days old (points C, D and E). Failures corresponding to points A and B are all caused because of 322 drives failing on one particular date. Here, failure of disks of two different ages correspond to a failure event on the same day because these disks were deployed on different dates. Fig. 6b shows the total number of disks running and number of disk failures of H-4B by date. The left y-axis shows the cumulative disks of H-4B running on each day. The steps in the black curve show the incremental deployments of H-4B. The right y-axis shows the number of H-4B disks failing on each day (red curve). The tallest red spike in Fig. 6b corresponds to points **A** and **B** from Fig. 6a. Points C and D occurred because of disks failing on different days. In the absence of anomaly detection, HeART would have incorrectly concluded that the disk group's wearout stage has begun as early as point A. The details of the anomaly detector used are described in § 2.3.1.

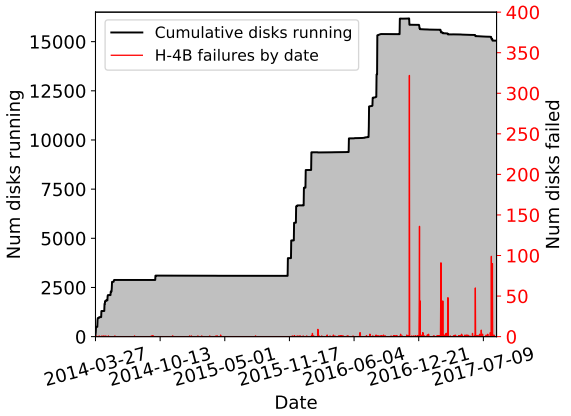
2.2.3 Online change point detection

We refer to the transition of a stage in the lifetime of a disk group as a *change point*. There are two major change points for each disk group; end of infant mortality stage and the onset of the wearout stage. The following subsections describe our method of identifying the two change points, while the details of the change point detector used are described in § 2.3.1.

Onset of useful life period: HeART uses prior studies about infant mortality in HDDs along with change point detection to decide a disk group's end of infancy. Prior studies performed on the Google and EMC disk fleets [23, 27] have shown that infant mortality lasts for approximately one quarter. Therefore, in order to be conservative, HeART exempts the first quarter from being assessed for end of infant mortality. Since disk reliability data is collected periodically, each time data is collected after the first 90 days, we run change point detection on the AFR curve generated by a sliding window of the past 30 days. HeART declares end of infancy if the last change point marked by the detector is over 30 days old, and the failure rate during the last 30 days



(a) Raw and HeART-curated AFR curves for the H-4B disk group. Five spikes in AFR (points A–E) which correspond to failures part of three anomalous bulk failure events are automatically filtered out by HeART.



(b) Total number of disks and number of disk failures by date for H-4B disks. The step-wise jumps in the black curve represent incremental deployments. The largest red spike constitute the disks that failed on a particular day causing anomalies A and B in Fig. 6a.

Figure 6: Anomaly detection in HeART by disk age and by date

is relatively constant. The days between consecutive change points is called a *segment*. More precisely, HeART declares end of infancy when the difference between the observed maximum and minimum AFR values in the most recent segment is less than a certain threshold T_{flat}. T_{flat} is the threshold for *flatness* and is a tunable parameter in HeART. Sensitivity to T_{flat} is described in § 2.3.3. Note that HeART takes a conservative approach in declaring the onset of useful life period of a disk group to ensure that it is highly confident about reducing redundancy for data stored on that disk group.

End of useful life period: Being lax in declaring the end of useful life period (i.e., onset of wearout) can risk in HeART not meeting the intended reliability target. Hence, HeART takes a conservative approach and marks the end of useful life for the first AFR observed that is greater than the determined useful life AFR. Since HeART checks for anomalous AFR fluctuations before checking for change points, if the anomaly detection phase does not filter out an increase in AFR, HeART assumes it to be a true increase in AFR. Thus, here too HeART takes a conservative approach and errs on the side of exiting the useful life period early and reverting to the default redundancy scheme.

2.3 Measuring HeART

This section describes implementation details of various components that make up HeART and presents an evaluation of HeART on the Backblaze dataset.

2.3.1 Implementation of the components

Our current implementation of HeART leverages existing, standard algorithms for anomaly detection and change point detection. Employing more sophisticated algorithms might lead to even better results.

Anomaly detector: For anomaly detection, our current implementation of HeART uses the RRCF algorithm [4] exposed by Amazon's data analytics service offering called Kinesis [3]. [3] The anomaly detector acts on a reliability data stream made available by the disk health monitoring system. The output from the anomaly detector is also a data stream containing anomaly scores produced by the RRCF algorithm. Potential anomalies identified by RRCF have a higher anomaly score than data that the algorithm considers non-anomalous. RRCF generates the anomaly score based on how different the new data is compared to the recent past. For consistency with change point detection, we provide the window size of the recent past to be one month. If the anomaly score is above a certain threshold, HeART considers that snapshot of reliability data as anamolous. RRCF advises to only consider the highest anomaly scores as true anomalies [4]. The anomaly score threshold is a tunable parameter in HeART. Lowering the score makes HeART more sensitive to fluctuations in AFRs.

Change point detector: Our current implementation of HeART uses a standard window-based change point detection algorithm the which compares the discrepancy between adjacent *sliding windows* within the AFR curve to determine if a change point has been encountered. In particular, we employ ***Ruptures*** library for online change point detection [42, 43]. We set the sliding window size to one month, because AFRs at a lower granularity than a month are jittery.

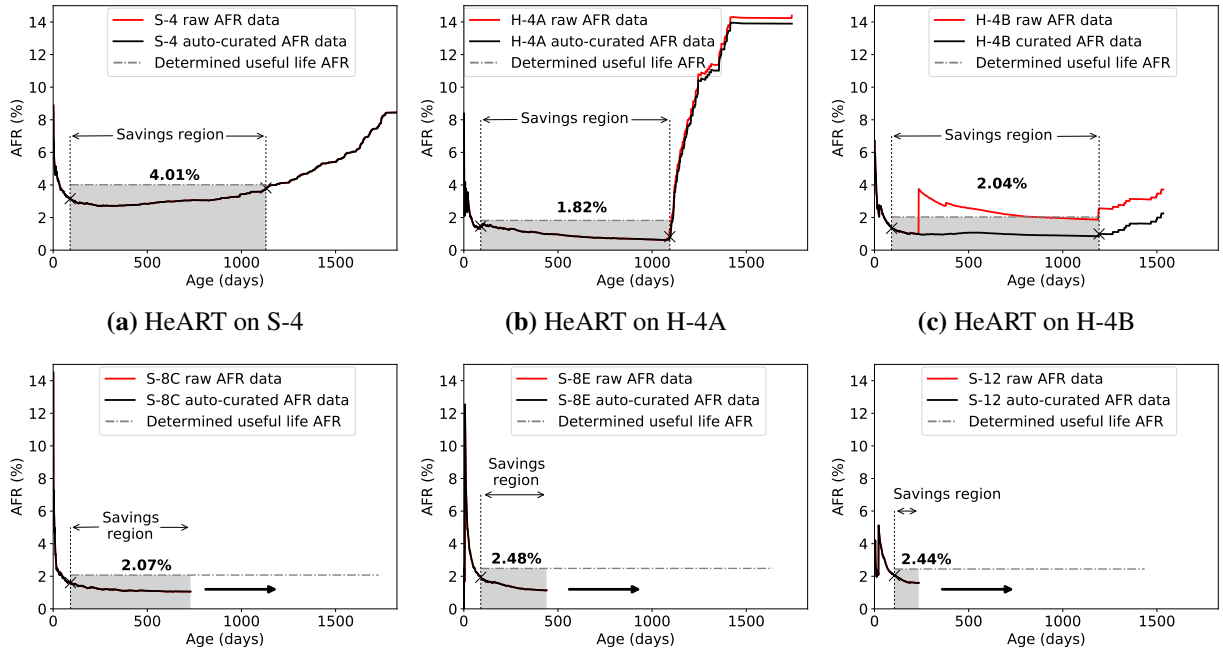
2.3.2 Evaluation on the Backblaze dataset

Identifying useful life period. Fig. 7 shows the results from HeART running on all 6 disk groups of the Backblaze dataset. HeART accurately identifies the infancy, useful life and wearout stages of S-4, H-4A and H-4B disk groups shown in Figs. 7a, 7b and 7c respectively. In case of S-8C, S-8E and S-12E disk groups (Fig. 7d, 7e and 7f), HeART identifies the end of infancy and correctly shows that they are still in their useful life. The width of the shaded region of each disk group highlights the "savings region", i.e. the useful life period determined by HeART for which HeART employs a potentially lower redundancy scheme. The height of shaded region in Fig. 7 denotes the AFR values protected by the useful life AFR value determined by HeART for that disk group. It is important to note that even though Fig. 7 shows cumulative AFR behavior, HeART performs anomaly detection and online change point detection on AFRs calculated using monthly sliding windows. Thus, not only is the cumulative AFR always inside the shaded region, but the instantaneous failure rate for any 30-day period is also less than the determined AFR value. In fact, the first rise in the instantaneous failure rate is what determines the end of the useful life period.

In contrast to S-4 (Fig. 7a), H-4A (Fig. 7b) and H-4B (Fig. 7c) disk groups have a sudden occurrence of their respective wearout stages. The quick reactivity requirement explained in § 2.2.1 comes into effect for these disk groups. How quickly HeART reacts to changes in the AFR is determined by how quickly failure data is provided to HeART. Since Backblaze maintains daily snapshots of disk health, the quickest reaction to an increased failure rate is on the day that the failures occur. In our evaluation, HeART successfully identifies the increased AFR on the very day it was provided with the increased AFR data.

Anomaly detection. As explained in § 2.2.2, the anomaly detector successfully detects five

[3] We use Amazon's service so as to avoid re-implementing a state-of-the-art algorithm.



(a) HeART on S-4 **(b)** HeART on H-4A **(c)** HeART on H-4B

(d) HeART on S-8C **(e)** HeART on S-8E **(f)** HeART on S-12E

Figure 7: HeART in action on all disk groups from the Backblaze dataset, showing successful identification of infant mortality, useful life and wearout stages and automatic removal of anomalies.

anomalies in the lifetime of H-4B disks. Additionally, two anomalies are also detected for the H-4A disks. Correctly identifying anomalous events increased the useful life period of H-4B disks by over $5\times$. In the absence of anomaly detection, the end of useful life period would have been incorrectly identified at age 235 days (shown by point A in Fig. 6a).

Cost savings per disk group. Table 3 summarizes the cost savings of employing disk group specific redundancy in their respective useful lifespans. Similar to § 2.1, we restrict the stripe width (n) of the optimized code to at most twice the stripe width of the default redundancy scheme (r_{def}).

It is important to note that the useful life AFRs determined by HeART are higher than the useful life AFRs shown in Fig. 1. This is because HeART chooses to be conservative in determining a useful life AFR value to ensure that reliability targets are comfortably met and to elongate the length of the useful life period to maximize benefits. Recall from § 2.1, that HeART adds a buffer above the useful life AFR determined at the end of infancy (an additional 25% by default, but is tunable).

We start by examining the cost reduction for the canonical redundancy scheme: 3-replication. We see that HeART achieves $33 - 44\%$ savings across disk groups. For example, consider H-4A disks which have the lowest determined useful life AFR (consistent with our observation from Fig. 1). Recall from § 2.1 that in order to ensure that we do not regress on reliability that disks in our dataset can currently offer, we set the target $MTTDL$ to be $MTTDL$ of the default redundancy scheme applied on the disk group with the highest AFR. S-4 is the disk group with highest useful life AFR. Using 3-replication as the default scheme for S-4 (and consequently our $MTTDL$ target), HeART achieves up to 44% space reduction on H-4A disks by using the

Disk groups		r_{def} = (3, 1)				r_{def} = (9, 6)			
DG	AFR	$MTTDL^{3-rep}$	r_{DG}	$MTTDL^{r_{DG}}$	**Cost↓**	$MTTDL^{(9,6)}$	r_{DG}	$MTTDL^{r_{DG}}$	**Cost↓**
S-4	4.01%	$3.18E+6$	(3, 1)	$3.18E+6$	**NA**	$5.52E+7$	(9, 6)	$5.52E+7$	**NA**
H-4A	1.82%	$3.40E+7$	(5, 3)	$3.40E+6$	**44%**	$1.30E+9$	(17, 14)	$6.88E+7$	**19%**
H-4B	2.04%	$2.42E+7$	(4, 2)	$6.04E+6$	**33%**	$8.24E+8$	(16, 13)	$5.70E+7$	**17%**
S-8C	2.07%	$2.31E+7$	(4, 2)	$5.78E+6$	**33%**	$7.77E+8$	(15, 12)	$7.17E+7$	**16%**
S-8E	2.48%	$1.34E+7$	(4, 2)	$3.36E+6$	**33%**	$3.77E+8$	(13, 10)	$6.65E+7$	**13%**
S-12E	2.44%	$1.41E+7$	(4, 2)	$3.53E+6$	**33%**	$4.02E+8$	(13, 10)	$7.09E+7$	**13%**

Disk groups		r_{def} = (14, 10)			
DG	AFR	$MTTDL^{(14,10)}$	r_{DG}	$MTTDL^{r_{DG}}$	**Cost↓**
S-4	4.01%	$6.07E+9$	(14, 10)	$6.07E+9$	**NA**
H-4A	1.82%	$3.15E+11$	(28, 24)	$6.24E+10$	**16%**
H-4B	2.04%	$1.78E+11$	(25, 21)	$3.15E+10$	**14%**
S-8C	2.07%	$1.66E+11$	(25, 21)	$2.88E+10$	**14%**
S-8E	2.48%	$6.71E+10$	(28, 23)	$9.75E+9$	**13%**
S-12E	2.44%	$7.28E+10$	(28, 23)	$1.08E+10$	**13%**

Table 3: Disk space saved by HeART by tuning the redundancy in the useful life of a disk group according to the observed disk group-specific AFRs. The units for $MTTDL$s is years. The cost savings are calculated for 3 default schemes: 3-replication, (9,6) erasure code and (14,10) erasure code. We limit stripe widths to be at most twice the length of the default redundancy scheme.

code $(5, 3)$. The rest of the disks also optimize over 3-replication by using $(4, 2)$ and achieving a space reduction of 33% in their respective useful lives.

Using $(9, 6)$ as the default redundancy scheme for S-4 disks, we notice a space reduction of 19% on H-4A disks by using the $(17, 14)$ redundancy scheme. Other disks also optimize over $(9, 6)$, but end up using shorter code lengths compared to H-4A disks because they have relatively higher determined AFR values. Nevertheless, all disk groups achieve a space reduction of 13% and above.

Finally, we measure HeART's performance when using $(14, 10)$ erasure code as the default redundancy scheme. $(14, 10)$ has a low storage overhead (1.4), which is the lowest compared to 3-replication and $(9, 6)$, making it harder to find codes that meet the target $MTTDL$ and reduce overhead. Despite these constraints, we achieve a 16% space reduction for H-4A disks and again, a minimum improvement of 13% for all other disk groups. In fact, we hit our $2\times$ stripe-length limit for H-4A disks with a $5\times$ higher $MTTDL$ value suggesting further improvement if we allow even longer codes.

Overall cost reduction. To highlight the overall cost reduction achieved across all six disk models during their useful lives, we show the capacity-weighted cost savings in Fig. 8. The cost reduction using allowing a maximum stripe width of $2\times$ the default redundancy scheme given an overall benefit of slightly under 30% for the entire cluster. This graph also contains the S-4 disks which have the highest population and were restricted to use only the default redundancy scheme. The overall cost reductions for having erasure codes as the default redundancy scheme is around 10%. Fig. 8 also shows the achievable savings if longer codes are allowed (specifically, up to $4\times$ the default stripe width). We see that HeART achieves 43% savings overall when the default scheme is 3-replication, and between 15-20% savings overall when using the popular

erasure coding schemes.

2.3.3 Sensitivity analysis

Several configuration parameters govern the behavior of HeART, but most of them are dependent on the ready-made tools we have used for different components of our system (e.g., the threshold for anomaly scores when using RRCF for anomaly detection). However, two fundamental parameters are independent of which anomaly detector or change point detector is used.

Before going into the details about the two parameters, we note that the modulation of both the parameters only has an effect on the gains that our optimization can yield. *Neither of them affects correctness* of HeART, or protection of data in any way. This allows operators of cluster storage operators to start with conservative values, observe the AFR behavior of their disks and accordingly choose apt values to minimize their costs without risking not reaching their reliability target.

Flatness parameter (T_{flat}): T_{flat} is used to deduce the end of the infant mortality period. As mentioned in § 2.2.3, the end of infancy is defined as the first $30+$ day segment beyond the first quarter such that the difference between maximum and minimum observed AFR is below the threshold T_{flat}. Thus, T_{flat} essentially determines the flatness in the AFR curve for a given period. Currently we define T_{flat} to be 0.5. A larger T_{flat} value will reduce the length of infancy until it reaches 90 days, beyond which, it will have no effect. A lower T_{flat} will enforce a stricter flatness criteria typically causing end of infancy to be declared late. Ending infancy sooner potentially causes HeART to choose a larger value as the determined useful life AFR. This, in turn causes HeART to choose a stronger redundancy scheme (with higher space overhead) compared to one that we would have chosen with the determined AFR value derived as a result of a lower T_{flat} value. This reduces the achievable savings within the useful life period of the disk group. As a tradeoff, we get a larger useful life period with a larger T_{flat}, since not only does infancy end sooner, but also the onset of wearout stage is postponed since the increased useful life AFR now has higher tolerance to AFR variances throughout the useful life. Fig. 9 shows the effect of varying T_{flat} on the non-anomalous AFR curve of H-4B disks. We show the results for two different values $T_{flat} = 0.3$ and $T_{flat} = 0.5$ for clarity. When T_{flat} was set to 0.3, we can see HeART declaring end of infancy at close to 100 days. Despite the buffer added to the determined useful life AFR at the end of infancy, the fluctuation in monthly AFRs caused a spike on day 394 to rise above the determined AFR, causing HeART to announce end of useful life. In contrast, when $T_{flat} = 0.5$, infancy was declared to end on day 91, and the determined

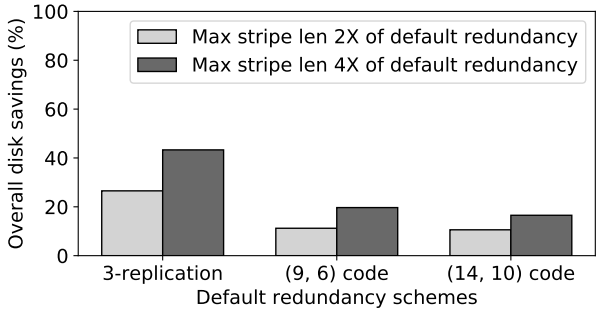


Figure 8: Overall space reduction by HeART on the Backblaze dataset.

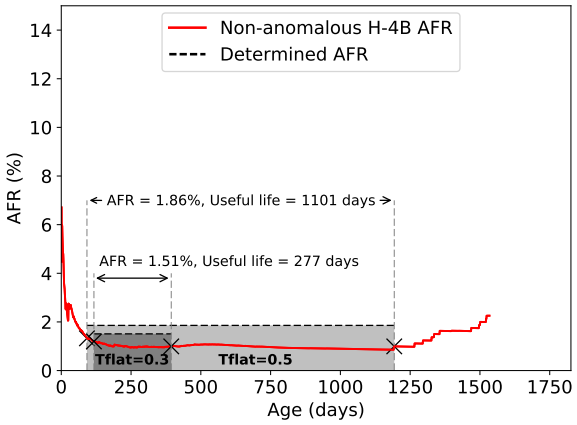


Figure 9: The effect of varying T_{flat} (AFR flatness threshold) on the H-4B disk group's AFR curve. Larger T_{flat} implies a higher useful life AFR along with a larger useful life period. The default value for T_{flat} in HeART is 0.5.

AFR value was high enough to tolerate the spike on day 394, increasing the useful life period by a huge amount.

Useful life AFR buffer: The next parameter is the AFR buffer used to determine the useful life AFR at the end of infancy. Currently the useful life AFR is determined as the maximum AFR value from the segment that identified end of infancy *plus an additional buffer*, which is a tunable parameter. The choice of the buffer value has similar tradeoffs as the flatness parameters discussed above. A high buffer value implies a more conservative approach to setting the determined useful life AFR. This will most likely prolong the useful life period, but restricts the tuning of the redundancy scheme due to the high useful life AFR value determined (and thus reducing benefits). In contrast, setting a low buffer value will possibly shorten the useful life period but allow more cost reductions during the useful life. Operators can set the buffer based on AFR fluctuations observed in their storage systems which can stem anywhere from workload patterns to operational conditions.

2.4 HeART-less alternatives (related work)

Ursa Minor [1] identified the heterogeneity in the data, and made a case for heterogeneity in storage clusters for better accommodating the diversity in data, rather than having a "one-size-fits-all" approach of monolithic disk-arrays. The rest of the related work is classified into disk reliability studies that identify reliability heterogeneity, techniques to predict disk failures using reliability data, and systems that automate redundancy scheme selection.

Numerous studies have been conducted in the past for characterizing disk failures [9, 14, 19, 21, 23, 26, 27, 32, 33, 34, 38]. Among the studies conducted on large production systems, Shah and Elerath [14, 38], Pinheiro et al. [27] and Ma et al. [23] independently verify that failure rates are highly correlated with disk manufacturers. These studies were conducted on the NetApp, Google and EMC disk fleets respectively. Schroeder and Gibson also conducted a similar reliability study on disks from high performance computing environment [33], not only highlighting reliability heterogeneity between disks deployed across systems, but also pointing out that disk datasheet reliability is very different from reliability observed in the field. Recently, Schroeder et al. [35] highlighted the heterogeneity in the reliability of different SSD technologies from four

different manufacturers. Schroeder et al. [32] also published the heterogeneity of partial disk failures (sector errors) across makes/models for the NetApp disks.

There have been numerous works that predict disk failures. [18, 25, 40, 44, 46]. Among the more recent ones, Mahdisoltani et al. [24] use machine learning techniques to predict occurrence of partial disk errors using S.M.A.R.T. data. Anantharaman et al. [6] use random forests and recurrent neural networks to predict remaining useful life for HDDs. Both these studies were performed on the Backblaze dataset.

Thereska et al. [41] built a self-prediction capability in cluster storage systems to assist in making informed redundancy and data placement decisions by answering *what-if* questions. It differs from HeART in that it does not perform and adapt to online analysis of reliability characteristics. The reliability metrics need to be known in advance. In designing for disasters [22], the authors built an optimization framework that automatically provided data dependability solutions to protect against site-level disasters by using information like workload patterns, and cost of recovery; it also assumes reliability characteristics are provided. Tempo [39] is a system that proactively creates replicas to ensure high durability in wide-area network distributed systems. It does this economically by allowing the user to specify a maximum maintenance bandwidth, and its design revolves around the efficient use of a distributed hash table. Carbonite [12] is another internet-scale replication algorithm that ensure high data durability in response to transient and permanent failures.

2.5 Bolstering HeART (proposed work)

2.5.1 Addressing capacity concerns between transitions

On its face, HeART's redundancy scheme transitions appear to require massive redistributions, all at once. Not only would this be a load spike concern, not assuaged by the "not much load over the lifetime" argument, but it also potentially creates a capacity concern: a bulk transition from r_{DG} to the less space-efficient r_{def}, at the end of the useful life period, could require more space than is available. Fortunately, we do not expect this issue to arise in practice, as disks of a disk group are deployed over time rather than all at once (see, e.g., Fig. 6b). Additional disk space is also created when disks transition from using r_{def} to using r_{DG}. Since end of useful life is determined based on deployment age, rolling deployment will mean rolling wearout. Capacity exhaustion due to any given transition (of one subset of disks from one disk group to another) should not be as large a driver of slack capacity requirements, in practice, as other sources of variability (e.g., user demand). Furthermore, transitions from r_{def} to r_{DG} at the onset of useful life period can be gradually executed as this only reduces the achieved capacity savings and does not affect correctness or reliability guarantees.

We believe that the right way to view HeART's transitions, in practice, is of individual disks rather than the disk group as a whole. A newly deployed disk is part of a default group, initially, which uses r_{def}. If it matches a particular disk group (e.g., same make/model), then it is moved to that disk group at the end of its infancy period. At that point, default group data is migrated from it, and disk group data is migrated to it. When a disk reaches the empirically-learned wearout age for its disk group, it is moved in the opposite direction. Naturally, disks that have reached wearout are strong targets for decommissioning, as well, since their AFRs are on the rise.

2.5.2 Cheaper data redistribution techniques

Many cluster storage systems include data redistribution mechanisms to deal with planned decommissioning and capacity/load balancing. Use of HeART will also require their use for transitioning from the default redundancy scheme (r_{def}) to a disk-group-specific scheme (r_{DG}), after infancy, and back again upon onset of wearout. Although at worst, it is two transcoding (read, re-encode, write) cycles of the data over the multi-year deployment time of each disk, there are ways to mitigate this cost.

Transcoding by code thinning. Popular erasure codes are also *maximum distance separable (MDS)* codes. These codes benefit from being able to reconstruct data from any *k* data chunks from a stripe of size *n*. Using this property, one technique to reduce storage overhead in useful life is by simply deleting excess parity chunks (such that required reliability is maintained). This optimization obviates the transcoding step required at the start of useful life, reducing the overall transcoding load by half, but still requires the transcoding at the onset of the wearout stage. Furthermore, this step assumes that r_{def} has enough slack in order to withstand thinning. We aim to identify stripe widths that can tolerate thinning in a practical setting.

Disks can change failure families, its data need not. As a consequence of rolling deployments, disks entering useful life coincide with other disks exiting useful life. Furthermore, there are also new disks being introduced regularly. Thus, rather than transcoding the data of the disks entering useful life, we can simply move the data on those disks to disks that are in their infancy, or wearout stages, assuming data copy is less expensive than transcoding.

We aim to quantify whether the copying data can totally eliminate transcoding for real world datasets. Furthermore, we would also like to understand a bound (or an approximation) on the size of incremental deployments as a function of the total disk population that would allow copying to eliminate transcoding.

2.5.3 Reliability of stripes spanning multiple disk groups

In the absence of HeART, a dis-aggregated storage system (separate disk and compute pools—a common architecture in large cluster storage systems today) tends to place a stripe of width n across *any n disks*, uniformly at random. By explicitly differenciating between disk group reliability, HeART disallows this freedom, since different stripes would then end up with different amount of reliability. Therefore, the current evaluation of HeART calculates the reliability of a stripe assuming it has been stored entirely within one disk group. This restriction unfortunately lends itself to correlated failures that might arise from disk group specific phenomena (e.g. buggy firmware, or collocated deployments), along with the requirement of sufficiently large disk groups.

We can overcome this restriction by viewing different disk groups as different fault domains (analogous to power domains) and identify placement strategies of stripes across multiple disk groups. We intend to explore the analytical and practical aspects of placement across disk groups for a real world dataset, taking into account rolling deployments, required data redistribution and cross-group $MTTDL$ calculation.

2.5.4 Disk group formation

Alternate disk groupings. The formation of disk groups in HeART is by make/model. This is not a fundamental choice, but rather based on the dataset we are analyzing. The Backblaze dataset does only has information about disk makes/models, and not about their operational conditions or usage patterns, which might potentially be better grouping strategies. Researchers at Los Alamos National Labs (LANL) have identified disk failure rates being correlated with their placement in the chassis. By getting access to such data, we intend to identify alternate grouping strategies and evaluate how well HeART performs.

Containing the number of disk groups. Since HeART relies on statistically significant number of disks to be a part of one disk group, which allows us to be confident on the nature of its bathtub curve, we need to be careful to avoid having small disk group sizes. One strategy to ensure large enough groups is to coalesce "similar" groups to form larger groups.

We intend to describe thresholds under which coalescing disk groups is viable. We will also address questions about the level of redundancy to be set in the coalesced disk group compared to the individual disk groups such that each constituent disk group meets the target reliability.

3 Incorporating capacity and client data heat heterogeneity (proposed work)

Continuous technological advancements cause new storage devices to have different characteristics compared to previously deployed storage devices, especially in terms of storage capacity, with newer disks in the same storage tier usually having larger capacity. In contrast, there is hardly any throughput improvement across HDDs of different makes/models/generations. Fig 10 shows the differences in the disk capacities of the Backblaze disk groups analyzed as a part of HeART, the inverse of which is roughly the throughput per unit capacity. Similar to AFR differences between the disk groups, the difference between throughput per unit capacity between the disk group with the smallest capacity and the disk group with the largest capacity is $3\times$ in the Backblaze dataset.

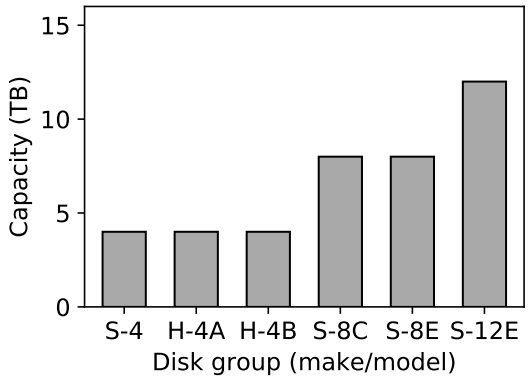


Figure 10: Capacity differences for the 6 disk groups from Backblaze that we analyzed in HeART. The inverse of this graph is the throughput per unit capacity under the assumption of throughput homogeneity over the disk fleet.

Problems with data placement being agnostic to capacity heterogeneity: Data placement strategies agnostic to capacity differences, e.g. uniformly random selection of disks from among all devices to store a stripe, or round-robin data placement are simple and stateless (hence efficient), but will often lead to smaller disks filling up faster than their larger counterparts. Assuming the same average read request rate per data chunk (which we will show to be false, aggravating the problem even further), the larger disks receive a much higher request rate compared to smaller disks. Since there is negligible difference in the throughput per unit capacity across different disks, larger HDDs observe long queues of pending requests, causing high average latencies and unbearable tail latencies.

Problems with data placement being agnostic to data heat heterogeneity: The largest cluster storage systems today belong to large internet service organizations. The clients of these storage systems aren't end users. They are in fact applications used by end users like email applications, video streaming applications, social media applications, etc. Therefore, the workload patterns experienced by these storage systems stem from a handful of clients (applications) and are well understood.

These clients' data size and data access patterns are very different from one another. Thus, we cannot treat all data chunks (independent of client) to have the same characteristics. Failing to account for differences in data heat during data placement might cause multiple very hot data chunks to be placed on the same disk, causing a lot of contention on that disk. On the other hand, placing mostly cold data on some disk might end up under-utilizing the throughput of that disk. This spindle imbalance caused by data heat imbalance stems from placement algorithms being agnostic to data heat, even when all disks have the same capacity. Similar to ignoring capacity heterogeneity, ignoring data heat heterogeneity also results in higher latencies.

In real world deployments that display capacity heterogeneity and data heat heterogeneity, both the above mentioned problems occur, leading to poor throughput and intolerable latencies. In this part of the dissertation, we strengthen the data placement algorithm by using an additional per-application heuristic; its data's *rate of cooling*. In § 3.3, we reason how rate of cooling knowledge in conjunction with data heat heterogeneity and capacity heterogeneity can enable the design of rebalance-friendly data placement strategies with reduced data access latency and equal data heat distribution across the disk fleet.

3.1 Background

Serenyi et al. [37] described an algorithm to incorporate disk capacity heterogeneity in data placement decisions for Google data centers. Their algorithm divided the data according to data heat (read request rate per unit time) into two groups: hot data and cold data, and chose to have a fixed sized hot data layer on each disk, while filling the rest of the disk with cold data. Several implicit assumptions are made in this algorithm:

1. *Data heat doesn't change over time*, i.e. data that is hot remains hot, and cold data never becomes hot again.
2. *All hot pieces of data cool at the same rate*, creating a balanced load throughout the disk

fleet.

3. When equally divided among all disks, the *hot data per disk is smaller than the capacity of the smallest disk*.

These assumptions may be too restrictive in practice. Microsoft Azure's storage system team described heavy fluctuation in data heat among their clients. One of Azure's largest clients had 47% of its read requests for new data within the first 24 hours of its creation. This violates the above assumptions and highlights the following properties: *new data is hot data* and *data heat changes over time*. The above mentioned algorithm fails to incorporate the dynamic nature of data heat, and therefore is not friendly to the amount of data rebalancing that might be needed compared to the naive uniformly random data placement.

3.2 Measuring efficacy of data placement algorithms

We identify the following metrics to measure how well a data placement algorithm performs:

3.2.1 Minimum data *rebalancing*

Rebalancing is the background process of copying data from one disk to another in order to either make space for newer data, or to have better load distribution. In a disk fleet consisting of devices with highly varying capacities and performance characteristics, data rebalancing is inevitable. But, since data rebalancing amounts to additional work (analogous to write amplification), a good data placement strategy would *minimize data rebalancing*.

3.2.2 Equal data heat distribution

In order to ensure equal disk utilization, the amount of hot data in each disk has to be roughly equal. The hot data layer has to be equal, and not proportional to the disk capacity because as mentioned before, the throughput of HDDs with different capacities is still the same. Therefore, a good data placement algorithm will try to ensure *equal data heat distribution*.

3.2.3 Minimum computation in the critical path

Previous general purpose data placement approaches have involved solving expensive optimization problems in order to realize possible data placement solutions [2, 5, 7, 10]. Today, with aggressive latency requirements for both reads and writes, a practical data placement algorithm will try and *minimize the computation in the critical path*.

3.3 A case for incorporating dataset *rate of cooling*

In addition to knowing the heat of a client's data, we also inherently have the information of its rate of cooling. A dataset's rate of cooling can be easily calculated by checking its data heat at different times since its creation. Since there are a handful of clients and workload patterns almost always have periodic patterns (i.e. diurnal, weekly, etc.) we can use past information of workload characteristics to approximate the future behaviour of data from that client.

We motivate the case for incorporating a dataset's rate of cooling with an example. Suppose we have two disks (say A and B) of different capacities having data with equal heat. Using the static algorithm mentioned above, a new data chunk which is hot is equally likely to be stored in either disk A or B. Suppose we also had the information of the data chunk's rate of cooling. The following two scenarios are possible:

- In case it cooled quickly, writing the data on the large capacity disk would be better compared to writing it on the small capacity disk because the rebalancing algorithm would end up moving it in the near future to the large capacity disk, to make space for new data on the small capacity disk.
- In case it cooled slowly, placing it on the small capacity disk would be better since it would utilize the small capacity disk for a longer amount of time.

Thus, incorporating rate of cooling enables us to make a better placement decision from the point of view of both *minimizing data rebalancing* and *improving disk utilization*.

3.4 Evaluating the rate of cooling approach

Similar to the data driven approach employed in evaluating reliability heterogeneity, the true evaluation of the impact of rate of cooling can only happen with real data.

3.4.1 Measuring rate of cooling

Measuring the rate of cooling from the data heat has to be performed at a granularity that is small enough to capture changes in heat, but large enough to not get influenced by jitter. Identifying the mechanism to measure rate of cooling is the first step in realizing our optimization.

3.4.2 Measuring optimal placement with perfect rate of cooling info

The notion of optimal data placement would be possible only with perfect knowledge of the rate of cooling. Unfortunately perfect rate of cooling information is only available post-facto. But, in order to understand the upper bound of the benefits that we can expect from our approach, we believe it is important to understand the amount of data rebalancing that is obviated, and extent to which disk utilization is enhanced with perfect knowledge of rate of cooling. Therefore, our baseline would be the spindle utilization and amount of rebalancing required under exact knowledge of each data chunk's initial heat and rate of cooling.

3.4.3 Measuring practical benefits with limited info of past rate of cooling

A more realistic understanding of the achievable benefits is by predicting the future rate of cooling based on the data heat observed in the past. Due to periodic workload patterns and temporal locality, we expect the prediction of the recent future based on the recent past to be fairly accurate. We propose measuring the benefits for varying windows of recency along with the extent to which the temporal effect assists in our confidence of predicting future rate of cooling.

4 Timeline

- **Nov, Dec 2018:** Addressing capacity concerns (§ 2.5.1), cheap data redistribution (§ 2.5.2).
- **Dec 2018:** Proposal.
- **Jan, Feb 2019:** NetApp and LANL reliability data collection and analysis.
- **Mar, Apr 2019:** Alternate disk group formations (§ 2.5.4), stripes spanning multiple disk groups (§ 2.5.3).
- **Apr 2019, May 2019:** Write second reliability heterogeneity paper with new data addressing proposed work in HeART.
- **Summer 2019:** Microsoft / Facebook reliability and rate of cooling data collection and preliminary evaluation including building rate of cooling simulator (§ 3.4.2).
- **Aug, Sep 2019:** Rate of cooling evaluation (§ 3.4), write paper on rate of cooling for FAST 2020.
- **Jul, Aug, Sep, Oct, Nov, Dec 2019:** Write thesis.
- **Dec 2019:** Defend.

References

[1] ABD-EL-MALEK, M., COURTRIGHT II, W. V., CRANOR, C., GANGER, G. R., HENDRICKS, J., KLOSTERMAN, A. J., MESNIER, M. P., PRASAD, M., SALMON, B., SAMBASIVAN, R. R., ET AL. Ursa minor: Versatile cluster-based storage. In *USENIX File and Storage Technologies (FAST)* (2005).

[2] ALVAREZ, G. A., BOROWSKY, E., GO, S., ROMER, T. H., BECKER-SZENDY, R., GOLDING, R., MERCHANT, A., SPASOJEVIC, M., VEITCH, A., AND WILKES, J. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems (TOCS)* (2001).

[3] AMAZON. Kinesis. `https://aws.amazon.com/kinesis/`.

[4] AMAZON. Robust random cut forest. `https://docs.aws.amazon.com/kinesisanalytics/latest/sqlref/sqlrf-random-cut-forest.html`.

[5] AMIRI, K., AND WILKES, J. Automatic design of storage systems to meet availability requirements. *Technical Report HPL-SSP-96-17* (1996).

[6] ANANTHARAMAN, P., QIAO, M., AND JADAV, D. Large scale predictive analytics for hard disk remaining useful life estimation. In *2018 IEEE International Congress on Big Data (BigData Congress)* (2018), IEEE.

[7] ANDERSON, E., SPENCE, S., SWAMINATHAN, R., KALLAHALLA, M., AND WANG, Q. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems (TOCS)* (2005).

[8] BACKBLAZE. Disk reliability dataset. `https://www.backblaze.com/b2/hard-drive-test-data.html`.

[9] BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. An analysis of latent sector errors in disk drives. In *ACM SIGMETRICS Performance Evaluation Review* (2007), ACM.

[10] BOROWSKY, E., GOLDING, R., MERCHANT, A., SCHREIER, L., SHRIVER, E., SPASOJEVIC, M., AND WILKES, J. Using attribute-managed storage to achieve qos. In *Building QoS into distributed systems*. Springer, 1997.

[11] BORTHAKUR, D. Hdfs and erasure codes (hdfs-raid). `https://wiki.apache.org/hadoop/HDFS-RAID`, 2009.

[12] CHUN, B.-G., DABEK, F., HAEBERLEN, A., SIT, E., WEATHERSPOON, H., KAASHOEK, M. F., KUBIATOWICZ, J., AND MORRIS, R. Efficient replica maintenance for distributed storage systems.

[13] COLE, G. Estimating drive reliability in desktop computers and consumer electronics systems. *Seagate Technology Paper TP* (2000).

[14] ELERATH, J. Hard-disk drives: The good, the bad, and the ugly. *Communication of ACM 13, 7* (2009).

[15] ELERATH, J. G. Afr: problems of definition, calculation and measurement in a commercial environment. In *Reliability and Maintainability Symposium, 2000. Proceedings. Annual* (2000), IEEE.

[16] ELERATH, J. G. Specifying reliability in the disk drive industry: No more mtbf's. In *Reliability and Maintainability Symposium, 2000. Proceedings. Annual* (2000), IEEE.

[17] FORD, D., LABELLE, F., POPOVICI, F. I., STOKELY, M., TRUONG, V.-A., BARROSO, L., GRIMES, C., AND QUINLAN, S. Availability in globally distributed storage systems. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2010).

[18] HAMERLY, G., ELKAN, C., ET AL. Bayesian approaches to failure prediction for disk drives.

[19] HEIEN, E., KONDO, D., GAINARU, A., LAPINE, D., KRAMER, B., AND CAPPELLO, F. Modeling and tolerating heterogeneous failures in large parallel systems. In *ACM / IEEE High Performance Computing Networking, Storage and Analysis (SC)* (2011), ACM.

[20] HUANG, C., SIMITCI, H., XU, Y., OGUS, A., CALDER, B., GOPALAN, P., LI, J., YEKHANIN, S., ET AL. Erasure coding in windows azure storage. In *USENIX Annual Technical Conference (ATC)* (2012), Boston, MA.

[21] JIANG, W., HU, C., ZHOU, Y., AND KANEVSKY, A. Are disks the dominant contributor for storage failures?: A comprehensive study of storage subsystem failure characteristics. *ACM Transactions on Storage (TOS)* (2008).

[22] KEETON, K., SANTOS, C. A., BEYER, D., CHASE, J. S., WILKES, J., ET AL. Designing for disasters. In *USENIX File and Storage Technologies (FAST)* (2004).

[23] MA, A., TRAYLOR, R., DOUGLIS, F., CHAMNESS, M., LU, G., SAWYER, D., CHANDRA, S., AND HSU, W. Raidshield: characterizing, monitoring, and proactively protecting against disk failures. *ACM Transactions on Storage (TOS)* (2015).

[24] MAHDISOLTANI, F., STEFANOVICI, I., AND SCHROEDER, B. Proactive error prediction to improve storage system reliability. In *USENIX Annual Technical Conference (ATC)* (2017).

[25] MURRAY, J. F., HUGHES, G. F., AND KREUTZ-DELGADO, K. Hard drive failure prediction using non-parametric statistical methods. In *Proceedings of ICANN/ICONIP* (2003).

[26] PATTERSON, D. A., GIBSON, G., AND KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). In *ACM International Conference on Management of Data (SIGMOD)* (1988), ACM.

[27] PINHEIRO, E., WEBER, W.-D., AND BARROSO, L. A. Failure trends in a large disk drive population. In *USENIX File and Storage Technologies (FAST)* (2007).

[28] RASHMI, K., SHAH, N. B., GU, D., KUANG, H., BORTHAKUR, D., AND RAMCHANDRAN, K. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster. In *USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)* (2013).

[29] RASHMI, K., SHAH, N. B., GU, D., KUANG, H., BORTHAKUR, D., AND RAMCHANDRAN, K. A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers. *ACM SIGCOMM Computer Communication Review* (2014).

[30] SAITO, Y., FRØLUND, S., VEITCH, A., MERCHANT, A., AND SPENCE, S. Fab: building distributed enterprise disk arrays from commodity components. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (2004), ACM.

[31] SATHIAMOORTHY, M., ASTERIS, M., PAPAILIOPOULOS, D., DIMAKIS, A. G., VADALI, R., CHEN, S., AND BORTHAKUR, D. Xoring elephants: Novel erasure codes for big data. In *Proceedings of the VLDB Endowment* (2013), VLDB Endowment.

[32] SCHROEDER, B., DAMOURAS, S., AND GILL, P. Understanding latent sector errors and how to protect against them. *ACM Transactions on storage (TOS)* (2010).

[33] SCHROEDER, B., AND GIBSON, G. A. Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you? In *USENIX File and Storage Technologies (FAST)* (2007).

[34] SCHROEDER, B., AND GIBSON, G. A. Understanding failures in petascale computers. In *Journal of Physics: Conference Series* (2007), IOP Publishing.

[35] SCHROEDER, B., LAGISETTY, R., AND MERCHANT, A. Flash reliability in production: The expected and the unexpected. In *USENIX File and Storage Technologies (FAST)* (2016).

[36] SEAGATE. Hard disk drive reliability and MTBF / AFR. `http://knowledge.seagate.com/articles/en_US/FAQ/174791en`.

[37] SERENYI, D. From GFS to COLOSSUS: Cluster-level Storage @ Google. `http://www.pdsw.org/pdsw-discs17/slides/PDSW-DISCS-Google-Keynote.pdf`, 2017.

[38] SHAH, S., AND ELERATH, J. G. Disk drive vintage and its effect on reliability. In *Relia-*

bility and Maintainability, 2004 Annual Symposium-RAMS (2004), IEEE.

[39] SIT, E., HAEBERLEN, A., DABEK, F., CHUN, B.-G., WEATHERSPOON, H., MORRIS, R. T., KAASHOEK, M. F., AND KUBIATOWICZ, J. Proactive replication for data durability.

[40] STROM, B. D., LEE, S., TYNDALL, G. W., AND KHURSHUDOV, A. Hard disk drive reliability modeling and failure prediction. *IEEE Transactions on Magnetics* (2007).

[41] THERESKA, E., ABD-EL-MALEK, M., WYLIE, J. J., NARAYANAN, D., AND GANGER, G. R. Informed data distribution selection in a self-predicting storage system. IEEE.

[42] TRUONG, C., OUDRE, L., AND VAYATIS, N. A review of change point detection methods. In *arXiv:1801.00718v1 [cs.CE]* (2018).

[43] TRUONG, C., OUDRE, L., AND VAYATIS, N. ruptures: change point detection in python. In *arXiv:1801.00826v1 [cs.CE]* (2018).

[44] WANG, Y., MA, E. W., CHOW, T. W., AND TSUI, K.-L. A two-step parametric method for failure prediction in hard disk drives. *IEEE Transactions on industrial informatics* (2014).

[45] YANG, J., AND SUN, F.-B. A comprehensive review of hard-disk drive reliability. In *Reliability and Maintainability Symposium, 1999. Proceedings. Annual* (1999), IEEE.

[46] ZHAO, Y., LIU, X., GAN, S., AND ZHENG, W. Predicting disk failures with hmm-and hsmm-based approaches. In *Industrial Conference on Data Mining* (2010), Springer.