Reliability of flat XOR-based erasure codes on heterogeneous devices

Kevin M. Greenan*†, Ethan L. Miller†, and Jay J. Wylie*
kmgreen@cse.ucsc.edu, elm@cs.ucsc.edu, jay.wylie@hp.com

Abstract

XOR-based erasure codes are a computationally-efficient means of generating redundancy in storage systems. Some such erasure codes provide irregular fault tolerance: some subsets of failed storage devices of a given size lead to data loss, whereas other subsets of failed storage devices of the same size are tolerated. Many storage systems are composed of heterogeneous devices that exhibit different failure and recovery rates, in which different placements—mappings of erasure-coded symbols to storage devices—of a flat XOR*-based erasure code lead to different reliabilities. We have developed redundancy placement algorithms that utilize the structure of flat* XOR*-based erasure codes and a simple analytic model to determine placements that maximize reliability. Simulation studies validate the utility of the simple analytic reliability model and the efficacy of the redundancy placement algorithms.*

1. Introduction

Erasure codes such as replication, RAID 5, and Reed-Solomon codes are the means by which storage systems are typically made reliable. Reed-Solomon codes and other maximum distance separable (MDS) codesk, provide the best tradeoff between fault tolerance and space-efficiency, but are computationally the most demanding type of erasure code. In addition to these traditional erasure codes, there are a number of proposals for novel, non-MDS erasure codes that exclusively use XOR operations to generate redundancy (e.g., [9, 10, 23]). Such XOR-based codes can be computationally more efficient than MDS codes, but offer an irregular tradeoff between performance, space-efficiency, and fault tolerance.

Methods to evaluate the space-efficiency and performance tradeoff for XOR-based codes are well understood [19, 11, 18]. However, some XOR-based erasure codes exhibit irregular fault tolerance: some subsets of failed storage devices of a given size lead to data loss, whereas other subsets of failed storage devices of the same size are tolerated. There have been many recent advances in understanding the fault tolerance [12, 23] and concomitant reliability [20, 13, 8] of such codes. However, all of these advances assume a *homogeneous* set of storage devices that all fail and recover at similar rates.

The contributions of this work are fourfold. First, we define a novel reliability problem in storage systems, the *redundancy placement problem*. Given a storage system comprised of *heterogeneous* storage devices with known failure and recovery rates, how should erasure-coded symbols be mapped to devices to maximize reliability? The redundancy placement problem is trivial for MDS codes because they exhibit regular fault tolerance—an m-tolerant MDS code will *never* lose data if m devices fail and *always* lose data if $m + 1$ devices fail—so all placements have the same reliability. For non-MDS codes though, the redundancy placement problem is non-trivial to solve. Second, we propose a simple analytic model related to mean time to data loss (MTTDL). The model is called the Relative MTTDL Estimate (RME), and it allows the *relative* reliability of different placements to be compared in a computationally efficient manner. Third, we propose two redundancy placement algorithms that use the structure of the XOR-based erasure code and the RME to determine a placement that maximizes (estimated) reliability. Fourth, we empirically demonstrate, via simulation, that the RME correctly orders different placements with regard to their reliability, and that the redundancy placement algorithms identify placements that maximize reliability.

The outline of the paper is as follows. In §2 and §3 we provide background on erasure codes, replica placement algorithms, and our prior work. We describe the RME and our redundancy placement algorithms in §4; we evaluate them in §5. We discuss the limitations of our work in §6 and then conclude in §7.

2. Background

An erasure code consists of n symbols, k of which are *data symbols*, and m of which are *parity symbols* (re-

*Hewlett-Packard Labs
†UCSC. Supported by the Petascale Data Storage Institute under Department of Energy award DE-FC02-06ER25768.

dundant symbols). We only consider *systematic* erasure codes—codes that keep the original data symbols and solely add parity symbols—because their use is generally considered a necessity to ensure good common case performance.

A maximum distance separable (MDS) code uses m redundant symbols to tolerate all erasures of size m or less [17]. Many MDS codes generate redundancy using k Galois field multiplies and $k-1$ XORs per parity symbol (e.g., Vandermonde Reed-Solomon codes). A Galois field multiplication operation can be transformed into multiple XOR operations (e.g., Cauchy Reed-Solomon codes). Parity-check array codes are another class of MDS codes that only use XOR operations to generate redundancy (e.g., RAID 4, EVENODD [3], and Row-Diagonal Parity [4]).

Hafner has categorized the construction of array codes as HoVer constructions: codes with parity symbols in both/either *Ho*rizontal and/or *Ver*tical dimensions of the array [10]. The class of codes we study are *horizontal* codes. We go beyond this, and refer to the codes we study as *flat* codes: horizontal XOR-based codes comprised of exactly one row (i.e., exactly one symbol, data or parity, per disk). Flat codes are distinct from most parity-check array codes which require multiple rows of symbols; RAID 4 is the exception because it is both a parity-check array code and a flat code.

The impact of erasure code choice on performance is a well-studied area [11]. In the grid storage community, the *read overhead* of certain classes of XOR-based erasure code is of interest. Plank et al. analyzed the *read overhead* of moderate-sized XOR-based codes using Monte Carlo methods [19] and of small-sized codes using deterministic methods [18].

A *replica placement* algorithm maps replicas to devices. Traditionally, this is done to improve performance: to reduce response time of accesses, to balance load, and for distributed caching. We exclusively consider the replica placement problem as it pertains to reliability. We use the term *redundancy placement* rather than *replica placement* because our emphasis is on the placement of erasure-coded data and parity symbols.

Our work on redundancy placement differs substantially from prior work on replica placement. In traditional RAID (erasure-coded) storage systems, many stripes of size n are placed on $N > n$ devices for performance reasons (e.g., parity declustering to reduce recovery time [1]). Thomasian and Blaum evaluate the reliability impact of various policies for mirrored disks [22], and Lian et al. [15] evaluate the difference in reliability between random and sequential placement policies for erasure-coded data. Both studies only consider homogeneous devices. In contrast, the competitive hill climbing replica placement algorithm places many distinct files, each replicated n times, on N heterogeneous servers in a manner that maximizes the availability of the least available file [5]. The Multi-Object Assignment Toolkit (MOAT) places many distinct objects, each replicated n times, on N heterogeneous devices to maximize the availability of multi-object operations in the face of correlated failures [24].

Replica placement algorithms place n replicas on $N > n$ devices. Our redundancy placement algorithms place n erasure-coded symbols on n heterogeneous devices. The non-MDS nature of flat XOR-based codes makes the redundancy placement problem both novel and non-trivial. There are other non-MDS XOR-based codes, e.g., Weaver codes [9].

There are many techniques beyond simple redundancy to improve storage system reliability, such as checksums, snapshots, scrubbing, auditing, and backup to tape. However, questions such as where to place backup copies or checksums to maximize reliability are outside of the scope of this work. Such questions require different models to answer that necessarily include metrics other than reliability, such as cost and performance. The work of Gaonkar et al. [7] automates the design of storage solutions that meet cost, performance, and reliability requirements. Their approach essentially solves a replica placement problem across heterogeneous tiers of storage that employ distinct reliability mechanisms.

3. Reliability of flat XOR-based codes

In this section, we describe our recent work on evaluating the reliability of flat XOR-based codes. First, we review the Minimal Erasures List (MEL), a fault tolerance metric for flat codes [23]. Second, we discuss the Fault Tolerance Vector (FTV), another fault tolerance metric for flat codes. Finally, we review the High-Fidelity Reliability (HFR) Simulator, a Monte Carlo reliability simulator, especially designed to simulate the reliability of (flat) XOR-based codes [8]. The redundancy placement algorithms we have developed use the structure of the MEL to make placement decisions. The FTV is used for comparison purposes in the evaluation section. The HFR Simulator is used to validate the efficacy of the redundancy placement algorithms.

Traditionally, the *Hamming distance* is used to describe the fault tolerance of an erasure code: a code tolerates all sets of erasures smaller than the Hamming distance. The Hamming distance completely describes the fault tolerance of MDS codes, since all erasures at or beyond the Hamming distance result in data loss. Flat erasure codes can be non-MDS: they tolerate some sets of erasures at and beyond the Hamming distance. We previously developed the Minimal Erasures (ME) Algorithm to efficiently analyze a flat erasure code and characterize its fault tolerance [23].

Consider the following definitions: A *set of erasures* is a set of erased symbols; an *erasure pattern* is a set of erasures

that results in at least one data symbol being irrecoverable; and, a *minimal erasure* is an erasure pattern in which every erasure is necessary and sufficient for it to be an erasure pattern. The ME Algorithm determines the *minimal erasures list* (MEL) of an erasure code: the list of the code's minimal erasures, which completely describes the fault tolerance of an erasure code. The MEL can be transformed into a *minimal erasures vector* MEV in which the ith element is the total number of minimal erasures of size i in the MEL. The length of the shortest minimal erasure in the MEL is the Hamming distance of the code, and so the first non-zero entry in the MEV corresponds to the Hamming distance.

We now present two flat codes in detail to provide examples of minimal erasures, the MEL, and the MEV. We denote each code by (k,m)-NAME where k is the number of data symbols, m is the number of parity symbols, and NAME is the class of the code. Every code is described by a listing of m bitmaps, one for each parity symbol, displayed as an integer. Since we only consider systematic codes, the parity symbols are $s_k, \ldots, s_{n-1}$. A bitmap describes the data symbols that participate in a parity equation and is an integer in the range $[1, \ldots, 2^k - 1]$. For instance, consider (4,4)-RAID 10 specified by the parity bitmaps $\langle 1, 2, 4, 8 \rangle$. The first parity symbol for this code, s_4, is simply a replica of s_0 and so the bitmap is 1 (i.e., $s_4 = s_0$ because $1 = 2^0$). A more complex example is (5,3)-FLAT with parity bitmaps $\langle 7, 11, 29 \rangle$. The first parity symbol for this code, s_5, has bitmap 7 because it is computed as the XOR of data symbols s_0, s_1, and s_2 (i.e., $s_5 = s_0 \oplus s_1 \oplus s_2$ because $7 = 2^0 + 2^1 + 2^2$).

The (4,4)-RAID 10 code is an example of a common RAID technique that simply replicates each data symbol. RAID 10 is a flat erasure code that tolerates any single disk failure. The MEL for the code is $\{(s_0, s_4), (s_1, s_5), (s_2, s_6), (s_3, s_7)\}$ and the MEV is $(0, 4, 0, 0)$. The MEL for (4,4)-RAID 10 is intuitive: whenever any pair of devices that store the same replicated symbol fails, data is lost. The MEV simply summarizes the count of minimal erasures of each size up to m. The MEL for (5,3)-FLAT is $\{(s_4, s_7), (s_0, s_1, s_4), (s_0, s_1, s_7), (s_0, s_2, s_6), (s_0, s_3, s_5), (s_1, s_2, s_3), (s_1, s_5, s_6), (s_2, s_4, s_5), (s_2, s_5, s_7), (s_3, s_4, s_6), (s_3, s_6, s_7)\}$, and, the MEV is $(0, 1, 10)$. This code better illustrates the irregularity that non-MDS flat codes can exhibit. There is no obvious structure or symmetry to the sets of device failures which lead to data loss.

The Fault Tolerance Vector (FTV) indicates the probability that data is lost given some number of failures. To construct the FTV, the MEL is transformed into the *erasures list* (EL). The erasures list consists of every erasure pattern for a code. The EL is a super set of the MEL, and every element in it is either a minimal erasure or a super set of at least one minimal erasure. The *erasures vector* (EV) is to the EL what the MEV is to the MEL, and is easily determined given the EL. Finally, the EV is transformed into the FTV. Let the ith entry of the EV be e_i. For a code with n symbols, the ith entry of the FTV is $e_i / \binom{n}{i}$. The FTV is the complement of the *conditional probabilities* vector described by Hafner and Rao [13].

The High-Fidelity Reliability (HFR) Simulator [8] is similar to the simulator developed by Elerath and Pecht [6]. Both are Monte Carlo reliability simulators that simulate disk failure, disk recovery, sector failure, and sector scrubbing, and both can use Weibull distributions for such failure and recovery rates. However, the HFR Simulator is *high-fidelity* in the sense that it simulates the reliability of non-MDS erasure codes that can tolerate two or more disk failures, with regard to both disk and sector failures. It is non-trivial to extend the methods of Elerath and Pecht in this manner.

The difficulty in simulating non-MDS codes is in efficiently determining if a specific set of failures leads to data loss. The HFR Simulator has two modes of bookkeeping that allow it to efficiently determine if a set of device failures leads to data loss: via the MEL, and via the FTV. Using the MEL permits the HFR Simulator to accurately determine if a specific set of failures leads to data loss. Therefore, it is the method we must use to simulate the reliability of a redundancy placement of a flat code on heterogeneous devices. The FTV is a coarse-grained metric that does not capture the details of a specific placement of symbols on heterogeneous devices; however, it describes the fault tolerance of the median placement and so is used for comparative purposes in §5.

4. Redundancy placement algorithms

We have developed two redundancy placement algorithms that identify placements of erasure-coded symbols on heterogeneous storage devices with known failure and repair rates which maximize reliability. One redundancy placement algorithm is based on brute force computation and the other is based on simulated annealing.

More formally, let S be the set of symbols in the erasure code and D be the *configuration* (set of heterogeneous devices). For a code with n symbols, $S = \{s_0, \ldots, s_{n-1}\}$ and $D = \{d_0, \ldots, d_{n-1}\}$. A placement, ρ, is a bijective function that uniquely maps each symbol in the erasure code to a single device, i.e., $\rho : S \leftrightarrow D$. The goal of the redundancy placement algorithms is to find a placement, ρ, that maximizes reliability.

4.1. Relative MTTDL estimate (RME)

We now introduce the simple analytic model that underlies both redundancy placement algorithms: the Relative MTTDL Estimate (RME). The RME can be used to compare

the reliability of different placements. It is constructed to correlate with the expected mean time to data loss (MTTDL), but it does not accurately estimate the MTTDL. The RME can be used to compare the relative merit of different placements, but not to determine if some placement meets a specific reliability requirement. The efficacy of the RME to correctly order placements by MTTDL is demonstrated in §5.

At a high level, the RME is the inverse of an estimate of the expected unavailability of a given placement. It is based on the MEL and a simple analytic device model. The MEL is a concise, exact description of a code's irregular fault tolerance. The MEL contains each minimal set of data and parity symbol failures that lead to data loss. Let $\mathrm{u}(d)$ be the expected unavailability of device d. To calculate $\mathrm{u}(d)$, the mean time to repair (MTTR) of d is simply divided by its mean time to failure (MTTF). This analytic model ignores sector failures and scrubbing, as well as the exact distribution of the device failures and repairs. Moreover, it is premised on failures being independent. The RME is calculated via the following function of the redundancy placement ρ, device unavailability u, and MEL:

$$\text{RME} = \left(\sum_{f \in \text{MEL}} \prod_{s \in f} \mathrm{u}(\rho(s)) \right)^{-1}.$$

The sum of products is inverted because RME values are values that should be maximized to improve reliability, just like MTTDL values.

The RME for the (4,4)-RAID 10 code described in §3 is as follows:

$$\begin{aligned} \text{RME} \quad = \quad & (\mathrm{u}(\rho(s_0))\mathrm{u}(\rho(s_4)) + \mathrm{u}(\rho(s_1))\mathrm{u}(\rho(s_5)) + \\ & \mathrm{u}(\rho(s_2))\mathrm{u}(\rho(s_6)) + \mathrm{u}(\rho(s_3))\mathrm{u}(\rho(s_7)))^{-1}. \end{aligned}$$

Consider a configuration in which the first 4 devices have expected device unavailability of 1.2×10^{-4} and the second 4 devices have expected device unavailability of 2.4×10^{-5}. Note that the more reliable a device is, the lower its device unavailability number, so the first 4 devices are less reliable than the second 4 devices in this configuration. Now consider two distinct placements. In the first placement, the first 4 symbols are placed on the first 4 devices, and the second 4 symbols are placed on the second 4 devices, and so the RME $= 86.8 \times 10^6$. In the second placement, the "odd symbols" (i.e., s_1,s_3,s_5, and s_7) are placed on the first 4 devices, and the "even symbols" on the second 4 devices, and so the RME $= 33.4 \times 10^6$. The first placement splits the pair of replicated symbols that occur in each minimal erasure, mapping one to a less reliable device and the other to a more reliable device. In contrast, the second placement places all of the symbols from two minimal erasures (the "odd symbols") on the less reliable devices, which, intuitively, is a less reliable placement. These RME values follow our intuition about the relative reliability of placements; this intuition is confirmed via simulation in §5.1.

There are several reasons for using the simple analytic model. First, the simplicity of the analytic device model permits efficient evaluation of the RME and so permits orders of magnitude more distinct placements to be evaluated than simulation methods in the same period of time. Second, the model only has to produce an RME that accurately orders different sets of device failures according to the likelihood that they contribute to data loss. The product of expected device unavailability accomplishes this task. Third, in a system with any redundancy, sector failures alone do not cause data loss; only multiple disk failures, or disk failures in conjunction with sector failures lead to data loss. Thus, the simple analytic model only needs to capture the reliability effects of disk failures. Fourth, we considered extending the approach of Hafner and Rao, who recently proposed a Markov model construction for XOR-based erasure codes based on homogeneous devices [13]. Extending their model to heterogeneous devices is not feasible because each device requires a distinct Markov model state per erasure pattern.

4.2. Brute force algorithm

The brute force redundancy placement (BF-RP) algorithm evaluates the RME for all possible placement and identifies the placement with the largest RME as the best placement. The RME is a simple equation that can be evaluated efficiently. Calculating an RME value requires $|\text{MEL}|$ additions and less than $m \times |\text{MEL}|$ multiplications. Consider the calculation of the RME for (4,4)-RAID 10 given above. It required four additions because $|\text{MEL}| = 4$, and four multiplications because each of the four minimal erasures consists of exactly two symbols. Since all minimal erasures consist of m or fewer symbols, each product requires $m - 1$ or fewer multiplications.

For a code with n symbols and n distinct devices, there are $n!$ possible placements to evaluate. Given the efficiency of the RME calculation, it is feasible to evaluate the RME for every possible placement for small codes. For example, in §5 the BF-RP algorithm is used to find the best placement for some codes with $n = 12$. Each such execution of the BF-RP performs $12! = 479001600$ RME calculations to determine the best placement.

4.3. Simulated annealing algorithm

For large codes, the factorial number of distinct placements make it is infeasible to apply the BF-RP algorithm. The best placement for a code maximizes the RME value. Therefore, the problem of finding the best placement can be understood as an optimization problem. Unfortunately, the nonlinear structure of the RME equation—all of the terms

in the summation are products of variables to be assigned via the optimization—precludes linear optimization techniques.

Fortunately, there are many non-linear optimization techniques. An approach that requires little work, in terms of formulating constraint equations, is simulated annealing [14]. This made simulated annealing, a stochastic optimization technique, appealing as the first optimization approach for us to evaluate. Simulated annealing uses randomization to find a solution; however, there is a chance that the solution found is not globally optimal.

The simulated annealing redundancy placement (SA-RP) algorithm takes an MEL and a configuration of devices as input. The SA-RP algorithm is initialized with a randomly selected placement. Each *step* in SA-RP is based on a random number of random swaps of mappings in the current placement. As the algorithm proceeds, the number of random swaps performed at each step decreases. This is the manner in which we capture the "cooling" aspect of simulated annealing, in which randomness is reduced over time so that some locally optimal placement is settled upon. In SA-RP, we include parameters to backtrack if a step that decreased the RME does not lead to a larger RME after some number of additional steps. The SA-RP algorithm is invoked multiple times, while keeping track of the best RME value found over different invocations. Because each invocation is initialized with a different random placement, repeated invocations find distinct locally maximal placements (RME values).

Unfortunately, simulated annealing does not lend itself to many practical rigorous statements about the quality of solution found. However, our empirical results indicate that the SA-RP algorithm quickly produces good solutions.

5. Evaluation

To evaluate the BF-RP and SA-RP algorithms, we consider configurations that have devices with failure models between two bounds. The first device failure model is based on that used by Elerath and Pecht (cf. Table 2 in [6]). Disk failures are distributed according to a Weibull distribution with parameters $\gamma = 0$, $\eta = 500000$, and $\beta = 1.12$. (Note that we "rounded up" the η parameter of 461386 hours used by Elerath and Pecht.) Disk recoveries are distributed according to a Weibull distribution with $\gamma = 6$, $\eta = 12$, and $\beta = 2$. We refer to the first device as the 500k device because its expected MTTF is 500 thousand hours. The 500k device is the most reliable device we consider in the evaluation. We refer to the least reliable device as the 100k device. The 100k device differs from the 500k device only in its MTTF: $\eta = 100000$ instead of $\eta = 500000$. To calculate the RME, only the MTTF for disk failure and the MTTR for a recovery is used. The HFR Simulator uses the specified Weibull distributions to simulate the MTTDL. All simulations are based on devices that exhibit only disk failures and recoveries; sector failures are not included in this evaluation.

There are two types of heterogeneous configurations we evaluate. *Bimodal configurations* consist of only two types of devices: 100k devices and 500k devices. For example, an 8-disk 3-bimodal configuration consists of 3 100k devices and 5 500k devices. *Uniform configurations* consist of one 100k device and one 500k device; the remaining devices have MTTF values uniformly distributed between $\eta = 100000$ and $\eta = 500000$. For example, an 8-disk uniform configuration consists of one device with each of the following η values: 100000, 157000, 214000, 271000, 328000, 385000, 442000, 500000. We evaluate 8-, 12-, and 20-disk configurations.

Table 1 lists the flat codes analyzed by the redundancy placement algorithms. The MEV and FTV for each code is listed. All of the codes have a Hamming distance of 2. The MEL is used to calculate the RME and so is more useful than the MEV for understanding the results in this section. The MEL for the (4,4)-RAID 10 and (5,3)-FLAT are given in §3. The MEL of the (6,2)-FLAT is $\{(s_0, s_1), (s_2, s_3), (s_2, s_6), (s_3, s_6), (s_4, s_5), (s_4, s_7), (s_5, s_7)\}$. The MEL for the larger codes is too verbose to list. The FTV is used for comparison purposes because the reliability simulated based on the FTV approximates the median reliability over all possible placements.

The specific flat codes listed in Table 1 were selected because, for the given values of k and m, they are the most fault tolerant flat codes [23]. One is the (4,4)-RAID 10 which was selected because it has a familiar structure. The specific values of k and m were selected so that all of the codes have a Hamming distance of 2. It takes many CPU days for the HFR Simulator to simulate a single data loss event for more fault-tolerant codes so we restricted the Hamming distance to ensure that the results of the redundancy placement algorithms could be validated via simulation.

Beyond the flat codes, we also included MDS codes with Hamming distance two to provide context. The placement of such codes does not affect their reliability because all sets of device failures of Hamming distance size or greater lead to data loss.

All MTTDL values in this section are measured in hours. The HFR Simulator is used to produce all of the MTTDL values [8]. Except where noted, MTTDL values in tables and annotated on histograms are based on simulations of 1000 data loss events. Each histogram consists of fifty bins equally sized from the shortest MTTDL to the longest MTTDL. The MTTDL values for data points in histograms are based on 100 data loss events and so exhibit greater variance.

code	Minimal Erasures Vector (MEV)	Fault Tolerance Vector (FTV)	parity bitmaps
(6,2)-FLAT	$(0, 7)$	$(0, 0.25,\ \ 1)$	$\langle 15, 51\rangle$
(5,3)-FLAT	$(0, 1, 10)$	$(0, 0.036, 0.29, 1)$	$\langle 7, 11, 29\rangle$
(4,4)-RAID 10	$(0, 4, 0, 0)$	$(0, 0.14,\ \ 0.43, 0.77, 1)$	$\langle 1, 2, 4, 8\rangle$
(10,2)-FLAT	$(0, 18)$	$(0, 0.27,\ \ 1)$	$\langle 127, 911\rangle$
(9,3)-FLAT	$(0, 5, 34)$	$(0, 0.076, 0.38, 1)$	$\langle 31, 227, 365\rangle$
(17,3)-FLAT	$(0, 19, 162)$	$(0, 0.10,\ \ 0.43, 1)$	$\langle 1023, 31775, 105699\rangle$
(16,4)-FLAT	$(0, 5, 80, 315)$	$(0, 0.026, 0.15, 0.48, 1)$	$\langle 511, 7711, 26215, 43691\rangle$

Table 1. Flat erasure codes.

code	100k	500k
(6,2)-FLAT	3.99×10^7	9.66×10^8
(5,3)-FLAT	2.88×10^8	6.89×10^9
(4,4)-RAID 10	6.59×10^7	1.83×10^9
(7,1)-MDS	1.01×10^7	2.55×10^8
(10,2)-FLAT	1.54×10^7	3.89×10^8
(9,3)-FLAT	5.28×10^7	1.40×10^9
(11,1)-MDS	4.06×10^6	1.03×10^8
(17,3)-FLAT	1.44×10^7	3.55×10^8
(16,4)-FLAT	5.42×10^7	1.32×10^9
(19,1)-MDS	1.55×10^6	3.60×10^7

Table 2. MTTDL of homog. config. in hours.

Table 2 lists MTTDL values of all the codes evaluated in this section in homogeneous configurations. Two configurations are listed: one based on 100k devices and the other based on 500k devices. Obviously, 500k homogeneous configurations are more reliable than 100k homogeneous configurations. The MTTDL of all the heterogeneous configurations fall between the MTTDL of these two homogeneous configurations.

5.1. Eight-disk configurations

In this section we exhaustively evaluate the three flat codes of size 8 on various configurations. First, consider the 4-bimodal distribution. Figures 1, 2, and 3 respectively show MTTDL histograms for (4,4)-RAID 10, (5,3)-FLAT, and (6,2)-FLAT. These histograms are constructed by simulating the MTTDL of the $8! = 40320$ possible placements.

Each histogram is annotated with a vertical line. The vertical line corresponds to the MTTDL for the FTV. The FTV is described in §3, it estimates the MTTDL of the median placement. In these figures, the FTV MTTDL is indeed near the median MTTDL over all possible placements.

Each histogram is also annotated with a series of lines labeled with integers. These lines are related to RME cal-

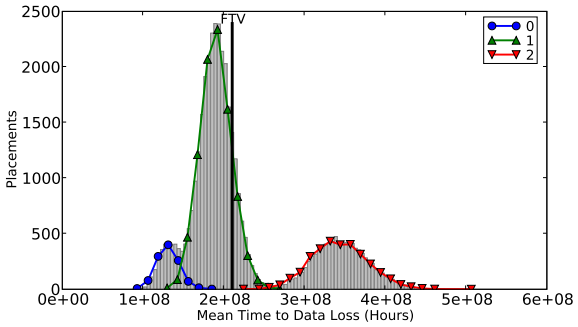

Figure 1. (4,4)-RAID10, 4-bimodal.

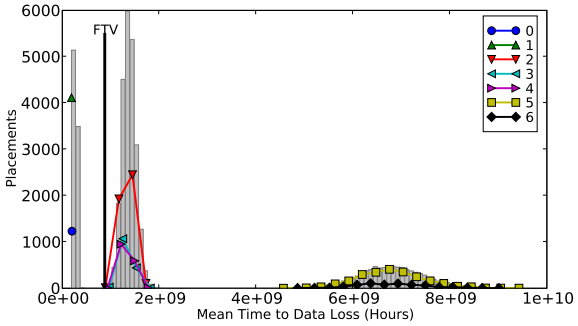

Figure 2. (5,3)-FLAT, 4-bimodal.

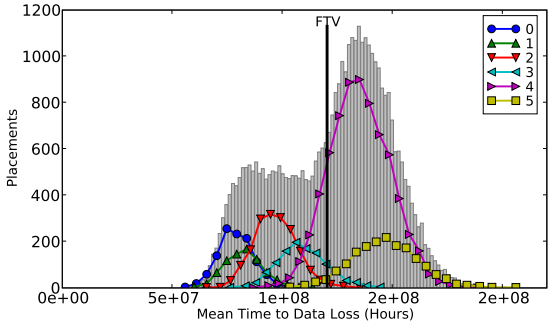

Figure 3. (6,2)-FLAT, 4-bimodal.

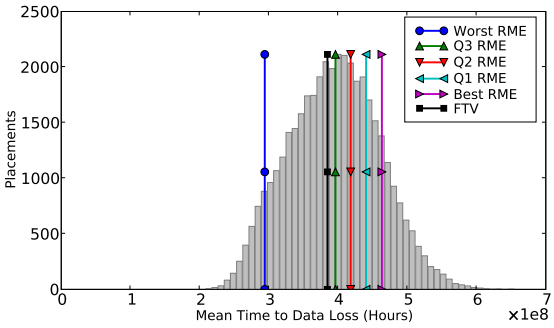

Figure 4. (4,4)-RAID10, uniform.

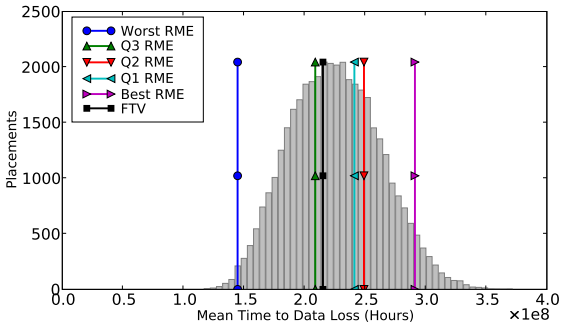

Figure 6. (6,2)-FLAT, uniform.

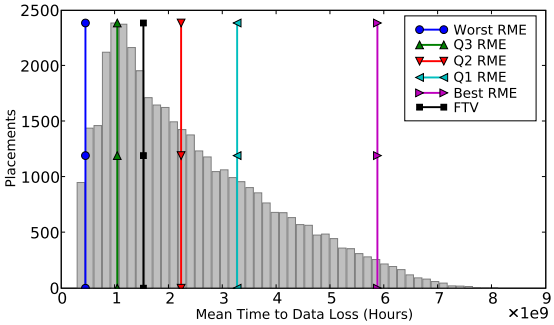

Figure 5. (5,3)-FLAT, uniform.

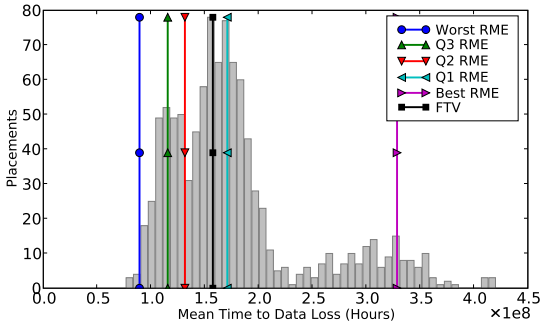

Figure 7. (9,3)-FLAT, 6-bimodal.

culations. For each of these codes, the BF-RP algorithm is used to determine the RME of each distinct placement. We were surprised to discover that for each of these codes, only a small number of distinct RME values were produced. From this, we hypothesized that there are *isomorphic placements*, i.e., different placements that have the same RME and MTTDL.

Each line on each histogram is effectively a sub-histogram for an isomorphic class of placements. The integer labels on the classes are in order of RME value, so line 0 has a lower RME than line 1. Note that we expected there to be no more than $\binom{8}{4} = 70$ distinct RME values for the 4-bimodal configuration because the first four and last four devices are identical. Figures 1, 2, and 3 show that there are respectively 3, 7, and 6 isomorphic classes.

To better understand isomorphic placements, consider (4,4)-RAID 10. The following are example placements for each isomorphic placement class: $0 : (s_1, s_3, s_5, s_7, s_0, s_2, s_4, s_6)$, $1 : (s_0, s_1, s_2, s_4, s_3, s_5, s_6, s_7)$, and $2 : (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$. The first four symbols in each placement is on a 100k device, and the second four symbols are on a 500k device. We already discussed the placements for classes 0 and 2 in §4.1. The placement for class 1 is consistent with the prior discussion: one pair of replicated symbols is on the 100k devices and so we expect the MTTDL to fall between class 0 (two pairs of replicated symbols on 100k devices) and class 2 (no pairs of replicated symbols on 100k devices).

The ordering of sub-histograms in each experiment strongly support our hypothesis that the RME correctly orders different placements with regard to reliability. The spread within each sub-histogram is due to statistical variance; remember that each simulation to produce a histogram data point was run for only 100 iterations. Because the MEL for (5,3)-FLAT and (6,2)-FLAT contain more entries than that of (4,4)-RAID 10, they each have more isomorphic placement classes. The distribution of isomorphic placement classes is interesting: the density of placements in the median classes appears to be greater than in the "best" class. This suggests that good placements are less common.

Now consider the uniform configuration instead of the 4-bimodal configuration. Figures 4, 5, and 6 respectively show MTTDL histograms for (4,4)-RAID 10, (5,3)-FLAT, (6,2)-FLAT. The FTV MTTDL is annotated on these histograms. Sub-histograms for isomorphic placement classes are not presented. The uniform configuration leads to too many such classes to illustrate. To be more precise, there are respectively 105, 840, and 280 distinct classes. These values are all much lower than the $8! = 40320$ potential distinct RME values, and so these results also support

the idea of isomorphic placement classes. Instead of sub-histograms, a vertical line is shown for a placement from each of the following isomorphic placement classes: Worst RME, Q3 (third quartile) RME, Q2 (second quartile) RME, Q1 (first quartile) RME, and Best RME.

Our hypothesis is that the MTTDL of the placement from the Worst RME class would be less than that of the placement from the Q3 RME class, and so on. The results support this hypothesis. One exception is the results for Q1 RME and Q2 RME for (6,2)-FLAT which are out of order. The difference between the MTTDL results for Q1 RME and Q2 RME is small though; we conclude that the Q1 RME and Q2 RME placements simply have quite similar reliabilities.

We expect the FTV MTTDL to provide an estimate of the median placement, because it is computed using the FTV, a probability vector derived from the MEV. As a consequence the FTV MTTDL should align with the Q2 RME MTTDL. Our results support this hypothesis, allowing us to use the FTV MTTDL as a reference when comparing the reliability of placements.

Table 3 summarizes results for all of the bimodal configurations and the uniform configuration. For each code, the FTV MTTDL and the Best RME MTTDL are listed. In all cases, the Best RME MTTDL is better than that of the FTV MTTDL.

All of these experiments support our hypothesis that the RME can be used to identify placements of erasure-coded symbols that maximizes reliability. The reliability difference between the worst placements and best placements range from from around a factor of two for (4,4)-RAID 10 to an order of magnitude for (5,3)-FLAT. Table 3 shows that across all configurations, differences between the FTV MTTDL and the Best RME MTTDL range from no difference to a factor of six.

5.2. Twelve-disk configurations

For 12-disk configurations, it is not feasible to evaluate every possible placement via simulation, but it is feasible to do so via the RME metric. We ran the BF-RP algorithm for the (9,3)-FLAT, and (10,2)-FLAT codes for all possible bimodal configurations and the uniform configuration. We also ran the SA-RP algorithm on these configurations. We run the SA-RP algorithm for a total of 1000000 steps; if the RME does not improve in 25 steps, the placement reverts to the last best placement for this execution; if the best RME placement does not improve in 1000 steps, a new execution is initialized with a random placement. In all cases, the SA-RP algorithm identified a placement from the same isomorphic placement class as the BF-RP algorithm (i.e., its RME is the same as the Best RME).

To determine the quality of the placements selected by the BF-RP and SA-RP algorithms, we simulated the Best RME MTTDL and the FTV MTTDL for a subset of configurations. The results are listed in Table 4. In most cases, the MTTDL of the placement with the Best RME is significantly better than that of the FTV. For the 9-bimodal configuration, the MTTDL values for (10,2)-FLAT are effectively the same.

From the BF-RP results, we also can identify the Worst RME, Q3 RME, Q2 RME, and Q3 RME placements. We simulated the MTTDL of these placements as well as 1000 random placements to generate low fidelity histograms. An example of such a histogram for (9,3)-FLAT in the 6-bimodal configuration is given in Figure 7. The histograms further support the hypothesis that the RME metric correctly orders placements by reliability.

5.3. Twenty-disk configurations

For 20-disk configurations, it is infeasible to evaluate every possible placement via simulation or the RME metric. Instead, we use the SA-RP algorithm to identify an Approximate Best RME placement for these configurations. We ran the SA-RP algorithm for the (17,3)-FLAT, and (16,4)-FLAT codes for all of the bimodal configurations and the uniform configuration. The SA-RP algorithm is run in the same manner as for the 12-disk configurations.

To determine the quality of the placements selected by the SA-RP algorithm, we simulated the MTTDL of the Approximate Best RME placement found by SA-RP and compare it with the FTV MTTDL for a subset of configurations. The results are listed in Table 5. In all cases, the Approximate Best RME MTTDL is significantly better than the FTV MTTDL.

6. Discussion

The redundancy placement algorithms based on the RME effectively find reliable placements. However, we have not characterized how sensitive the redundancy placement algorithms are to different failure models. Specifically, we do not have a good characterization of the conditions necessary for the RME to correctly order placements by their reliability.

We believe that extensive simulation will permit us to do such characterization. Unfortunately, the HFR Simulator is currently too slow to run the potentially millions of analyses necessary to do such characterization. The existence of isomorphic placement classes suggests an avenue for speeding up the redundancy placement algorithms. If we identify the sets of symbols that are *equivalent*, i.e., that if swapped yield a new placement in the same isomorphic class, then both the BF-RP and SA-RP algorithms can operate over a much smaller state space. Such a speedup may facilitate better RME characterization.

The specific device models used in the evaluation are based on the distributions that Elerath and Pecht used [6]. We believe that the models of Elerath and Pecht are as good

	(4,4)-RAID 10		(5,3)-FLAT		(6,2)-FLAT		
configuration	FTV	Best RME	FTV	Best RME	FTV	Best RME	(7,1)-MDS
1-bimodal	8.60×10^8	8.39×10^8	3.30×10^9	6.90×10^9	4.98×10^8	6.20×10^8	1.19×10^8
2-bimodal	4.74×10^8	5.97×10^8	1.94×10^9	6.53×10^9	2.94×10^8	3.74×10^8	6.71×10^7
3-bimodal	3.01×10^8	4.35×10^8	1.23×10^9	6.40×10^9	1.72×10^8	2.54×10^8	4.41×10^7
4-bimodal	1.69×10^8	3.49×10^8	9.24×10^8	6.37×10^9	1.33×10^8	1.47×10^8	2.96×10^7
5-bimodal	1.51×10^8	1.81×10^8	6.07×10^8	6.62×10^9	8.76×10^7	9.67×10^7	2.05×10^7
6-bimodal	1.09×10^8	1.19×10^8	4.53×10^8	6.89×10^9	6.50×10^7	7.42×10^7	1.61×10^7
7-bimodal	8.29×10^7	8.42×10^7	3.40×10^8	1.35×10^9	4.99×10^7	5.18×10^7	1.26×10^7
uniform	4.34×10^8	4.88×10^8	1.56×10^9	6.11×10^9	2.34×10^8	2.79×10^8	5.60×10^7

Table 3. MTTDL of 8-disk configurations in hours.

	(9,3)-FLAT		(10,2)-FLAT		
configuration	FTV	Best RME	FTV	Best RME	(11,1)-MDS
3-bimodal	3.45×10^8	7.88×10^8	9.83×10^7	1.31×10^8	3.13×10^7
6-bimodal	1.57×10^8	3.30×10^8	4.25×10^7	5.14×10^7	1.27×10^7
9-bimodal	8.81×10^7	1.06×10^8	2.57×10^7	2.54×10^7	5.75×10^6
uniform	2.70×10^8	5.63×10^8	8.01×10^7	9.59×10^7	2.77×10^7

Table 4. MTTDL of 12-disk configurations in hours.

as any currently available. Recently published analyses of failure data [21, 16, 2] will hopefully result in better failure models. We expect that such models will change the MTTDL values, but not the placement that is most reliable. The RME is based on the assumption that failures are independent. The RME equation may have to change if significant correlation is found in failure models.

When we developed the RME metric, we assumed that sector failures would have a secondary effect on placement decisions and so could be excluded from the RME metric. We have some initial results for the RME metric in systems with sector failures. For codes with a Hamming distance greater than 2, the RME still correctly order placements by reliability. For codes with a Hamming distance of 2, data loss events are dominated by single-disk, single-sector failures. For such codes, if every symbol occurs in at least one minimal erasure of size two, e.g., like (6,2)-FLAT, then placement had little affect on overall reliability. Whereas, for (5,3)-FLAT, only the symbols s_4 and s_7 occur in a minimal erasure of size 2, and so placements based on the RME maximize reliability.

7. Conclusions

We introduced the novel *redundancy placement problem* in which a mapping, called a *placement*, of the symbols in a flat XOR-based code onto a set of heterogeneous storage devices with known failure and recovery rates must be found to maximize reliability. To solve this problem, we developed the Reliability MTTDL Estimate (RME), a simple model based on estimated device unavailability and the Minimal Erasures List (MEL), a concise characterization of the fault tolerance of an XOR-based code. We developed two redundancy placement algorithms, the BF-RP algorithm based on brute force computation, only suitable for small redundancy placement problems, and the SA-RP based on simulated annealing, and suitable for larger problems. Simulation results based on the High-Fidelity Reliability (HFR) Simulator provide extensive empirical evidence that the RME correctly orders different placements for a given code by MTTDL. Additional simulation results suggest that the placements found by the SA-RP algorithm are significantly more reliable than the median placement. The results of BF-RP algorithm lead us to realize the existence of *isomorphic placements*, sets of placements which have the same MTTDL.

References

[1] G. A. Alvarez, W. A. Burkhard, and F. Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *ISCA-1997: 24th Annual International Symposium on Computer Architecture*, pages 62–72, Denver, CO, June 1997. ACM.

	(17,3)-FLAT		(16,4)-FLAT			
configuration	FTV	Approx. Best RME	FTV	Approx. Best RME	(19,1)-MDS	(18,2)-MDS
5-bimodal	8.94×10^7	1.29×10^8	3.59×10^8	1.39×10^9	9.63×10^6	1.50×10^{10}
10-bimodal	4.36×10^7	5.02×10^7	1.89×10^8	1.33×10^9	4.48×10^6	5.71×10^9
15-bimodal	2.33×10^7	2.61×10^7	8.72×10^7	2.85×10^8	2.53×10^6	1.94×10^9
uniform	8.49×10^7	9.62×10^7	3.38×10^8	8.71×10^8	8.49×10^6	1.27×10^{10}

Table 5. MTTDL of 20-disk configurations in hours.

[2] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An analysis of latent sector errors in disk drives. *SIGMETRICS Perform. Eval. Rev.*, 35(1):289–300, 2007.

[3] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Trans. Comput.*, 44(2):192–202, 1995.

[4] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. Row-diagonal parity for double disk failure correction. In *FAST-2004: 3rd USENIX Conference on File and Storage Technologies*, pages 1–14. USENIX Association, March 2004.

[5] J. R. Douceur and R. P. Wattenhofer. Optimizing file availability in a secure serverless distributed file system. In *Symposium on Reliable Distributed Systems*. IEEE, 2001.

[6] J. F. Elerath and M. Pecht. Enhanced reliability modeling of raid storage systems. In *DSN-2007*, pages 175–184. IEEE, June 2007.

[7] S. Gaonkar, K. Keeton, A. Merchant, and W. H. Sanders. Designing dependable storage solutions for shared application environments. In *DSN-2006: The International Conference on Dependable Systems and Networks*, pages 371–382. IEEE, June 2006.

[8] K. M. Greenan and J. J. Wylie. High-fidelity reliability simulation of erasure-coded storage. Technical Report (to appear), Hewlett-Packard Labs.

[9] J. L. Hafner. WEAVER Codes: Highly fault tolerant erasure codes for storage systems. In *FAST-2005: 4th USENIX Conference on File and Storage Technologies*, pages 212–224. USENIX Association, December 2005.

[10] J. L. Hafner. HoVer erasure codes for disk arrays. In *DSN-2006: The International Conference on Dependable Systems and Networks*, pages 217–226. IEEE, June 2006.

[11] J. L. Hafner, V. Deenadhayalan, T. Kanungo, and K. Rao. Performance metrics for erasure codes in storage systems. Technical Report RJ–10321, IBM, August 2004.

[12] J. L. Hafner, V. Deenadhayalan, K. Rao, and J. A. Tomlin. Matrix methods for lost data reconstruction in erasure codes. In *FAST-2005: 4th USENIX Conference on File and Storage Technologies*, pages 183–196. USENIX Association, December 2005.

[13] J. L. Hafner and K. Rao. Notes on reliability models for non-MDS erasure codes. Technical Report RJ–10391, IBM, October 2006.

[14] S. Kirkpatrick, C. Gelatt Jr.., and M. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598:671–680, May 1983.

[15] Q. Lian, W. Chen, and Z. Zhang. On the impact of replica placement to the reliability of distributed brick storage systems. In *ICDCS 2005: Proceedings of the 25th International Conference on Distributed Computing Systems*, pages 187–196. IEEE, 2005.

[16] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *FAST-2007: 5th USENIX Conference on File and Storage Technologies*. USENIX Association, 2007.

[17] J. S. Plank. Erasure codes for storage applications. Tutorial slides, presented at *FAST-2005: 4th Usenix Conference on File and Storage Technologies*, December 2005.

[18] J. S. Plank, A. L. Buchsbaum, R. L. Collins, and M. G. Thomason. Small parity-check erasure codes - exploration and observations. In *DSN-2005: The International Conference on Dependable Systems and Networks*, pages 326–335. IEEE, July 2005.

[19] J. S. Plank and M. G. Thomason. A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In *DSN-2004: The International Conference on Dependable Systems and Networks*, pages 115–124. IEEE, June 2004.

[20] K. Rao, J. L. Hafner, and R. A. Golding. Reliability for networked storage nodes. In *DSN-2006: The International Conference on Dependable Systems and Networks*, pages 237–248. IEEE, June 2006.

[21] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *FAST-2007: 5th USENIX Conference on File and Storage Technologies*, pages 1–16. USENIX Association, 2007.

[22] A. Thomasian and M. Blaum. Mirrored disk organization reliability analysis. *IEEE Trans. Comput.*, 55(12):1640–1644, 2006.

[23] J. J. Wylie and R. Swaminathan. Determining fault tolerance of XOR-based erasure codes efficiently. In *DSN-2007*, pages 206–215. IEEE, June 2007.

[24] H. Yu, P. B. Gibbons, and S. Nath. Availability of multi-object operations. In *NSDI-2006: Proceedings of the Symposium on Networked Systems Design and Implementation*, May 2006.