RJ10391 (A0610-035) October 24, 2006
Computer Science

IBM Research Report

Notes on Reliability Models for Non-MDS Erasure Codes

James Lee Hafner, KK Rao
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

NOTES ON RELIABILITY MODELS FOR NON-MDS ERASURE CODES

James Lee Hafner
KK Rao

ABSTRACT: We discuss two variations on the standard model for determining the reliability (or Mean Time To Data Loss, MTTDL) for storage arrays with erasure codes. The standard model assumes the erasure code is MDS and has a certain erasure fault tolerance t. The “Hamming fault tolerance” t is one less than the Hamming distance of the code. Such codes can tolerate *all* instances of k failures for $k \leq t$ but *no* instances of more than t failures. The first variation extends the model to non-MDS codes that have resilience to some (but not all) instances of failures that exceed the Hamming fault tolerance. We say such codes have “elastic” fault tolerance. We apply this model to LDPC and WEAVER codes which have high average fault tolerance, but have very low Hamming fault tolerance. A second application of this model is to the case of multiple instances of arrays, each with an independent MDS code. Additionally, the standard model also assumes that rebuild occurs incrementally, that is, one disk at a time. We vary the model to better reflect some actual systems where rebuild is done in parallel on all failed disks.

1. Introduction

Most Markov models for reliability for storage arrays with RAID-style erasure codes assume that the code is MDS. This implies that it can tolerate all instances of some number, say t, of erasures, but it also means that it can never tolerate more than t failures. We call t the "Hamming fault tolerance" of the code, since it is derived from the Hamming distance. Codes such as LDPC codes, which are not MDS, have an "average" erasure tolerance. This means that they may only tolerate *all* instances of some small number of erasures (the true Hamming fault tolerance) but, in general, can tolerate many instances of more failures (so the average can be large). We say such codes have "elastic fault tolerance". For LDPC codes, with n data blocks and size $n+m$, this fault tolerance is usually measured by the average number of blocks (elements) that are required to recover the n data blocks if the $n+m$ blocks are read one at a time in random order. An MDS code sets this number to exactly n; for good LDPC codes, this is asymptotically n, but always greater than n.

Another issue with modeling erasure code reliability is that the reliability models for a single array, are extrapolated to more arrays by multiplying (or dividing, depending on the measure), the reliability for one array by the number of arrays. So, for example, if a single RAID array on d disks has $\text{MTTDL} = T$, then r such arrays will have $\text{MTTDL} = T/r$. We show here that this is not exactly correct, though it is very close. In fact, one can model r arrays of d disks each by a single array of size rd that has a single non-MDS erasure code with average fault tolerance. For example, two 8-drive RAID5 arrays can tolerate all instances of one failure (16 such cases, so the Hamming fault tolerance is just 1), 64 of the 120 cases of 2 drive failures (53%) but no three drive failures. Because 53% is not 50%, the MTTDL formula for this system with two arrays is not exactly 1/2 of that of a single 8-drive array. This system of two arrays is then analogous to, and can be modeled by, the same techniques as elastic fault tolerant erasure codes such as LDPC codes.

Finally, typical Markov models for RAID storage system reliability, such as in [4], assume a rebuild model of the following type: a single parameter is used to model rebuild rate of one drive at a time, regardless of the number of drives that have failed. That is, the rebuild rate is dependent on the array size, disk sizes and, typically, overall fault tolerance of the erasure code, but not on the specific failure state. In addition, they assume a compound rate from two-down to one-down to fully functional and perform the rebuild incrementally. In fact, most systems will return from the two-down state directly to the fully functional state (by rebuilding two drives in parallel) at a rate which could be the same or slightly slower than the rate from one-down to fully functional. The rate may be the same if the progress of the rebuild is gated by policy that throttles back the rebuild (of any number of drives) in order that it consumes minimal resources from the foreground process. Alternatively, the rebuild may take longer as more resources would be required for rebuilding two drives over one drive. But that rate is not necessarily compounded. For example, to rebuild one drive on a Reed-Solomon two-fault tolerant array with 8 drives, requires reading only 6 drives, and writing one. To rebuild two drives, requires again only reading 6 drives but now writing two (as well as additional computations). The rate for two-disk rebuild then may be $(6+2)/(6+1)$ times the rate for a single rebuild. We address this issue by revising the standard model. It turns out, however, that this change has essentially no effect on the MTTDL numbers, so need not be applied in general.

In the following sections, we describe the Markov models that we use to address all three of these issues. We then apply the models to determine and compare reliabilty (MTTDL) numbers with reasonable choices for the parameters.

1.1. Methodology

We used Mathematica as a tool to assist in both symbolic and numerical calculations. The methodology involved building the transition matrix $\hat{A}$ for the Markov model. Each column represented a state of the model, with the last column representing the absorbing state. Each row represented a non-absorbing state. The entries in the off-diagonal of the matrix were the transition rates between the row state and the column state. The diagonal entries were the negative of the row sum of the off-diagonals (so in $\hat{A}$ all row sums are zero). We then computed the matrix A which is the negative of $\hat{A}$ after removing the last column. The MTTDL then computed was

$$\mathrm{MTTDL}(\hat{A}) = \langle 1,0,\ldots,0\rangle A^{-1}\langle 1,1,\ldots,1\rangle^t$$

which is just the sum of the entries in the first row of A^{-1}. This can also be computed by finding y_1 in the matrix equation

$$A\langle y_1,y_2,\ldots,y_k\rangle^t = \langle 1,1,\ldots,1\rangle^t$$

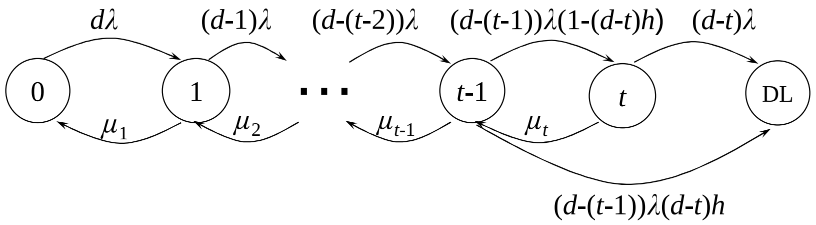
In our computations with Mathematica, both methods can be used for certain small examples, particularly the symbolic cases. However, numerically, these matrices were rather ill-conditioned. Consequently, not all computational methods ran without warnings. Ultimately, we used the second formulation above and Mathematica's `LinearSolve` function, with `Method→Multifrontal` as the most numerically stable method. All methods that ran warning-free produced identical numerical results to the accuracy reported here (there was an exception in one table entry where the default `Method` produced a value that differed from what is reported here in the 4th decimal place).

2. Basic Model

The diagram in Figure 1 shows one standard Markov model for a RAID storage system (see [4]) that has d drives and an MDS erasure code that can tolerate $t \geq 2$ failures. We modify the notation a bit from [4]. First, we suppress subscripts since we only deal with disks. Second, we use h for the "per drive" probability of uncorrectable error, whereas in the cited paper, h represented the total probability over $d-t$ drives. Finally, we model a drive rebuild rate from State k to State $k-1$ with the parameter μ_k as it may depend on the number of drives that have failed. Note that the model shows that rebuild only restores one drive at time. For MDS codes, the cost to rebuild any single drive from any state is independent of that state (e.g., in a Reed-Solomon code on $n+t$ disks and fault tolerance t, any single disk can be restored from any n disks of data, regardless of how many disks are failed). Consequently, in our numerical calculations and the rest of the paper we assume $\mu_k = \mu$, a constant, unless otherwise noted.

The parameter definitions and the numerical values we use in our results are given in Table 1. The drive reliability number is approximate: it is lower than most drive vendors claim (approximately $\mathrm{MTTF}_d = 1,000,000$hrs); however, it is higher than field data would suggest (approximately

$\text{MTTF}_d = 300000\text{hrs}$). The rebuild times seen in practice and provided by detailed modelling (see, for example, [1] with a value of 9.3hrs for Fibre Channel drives) are in the range of 8–24hrs and can depend on the drive generation, the drive interconnect and the system memory bandwidth (and to a lesser extent on the number of drives). We chose a value of 12hrs rebuild time as an approximation, so the rebuild rate in the table is $\mu = 1/12hrs$. Note that our purpose here is not so much to provide accurate numerical reliability results, but to show the relative structure of the mathematical models, so precise values are not so important.



$d\lambda$
$(d\text{-}1)\lambda$
$(d\text{-}(t\text{-}2))\lambda$
$(d\text{-}(t\text{-}1))\lambda(1\text{-}(d\text{-}t)h)$
$(d\text{-}t)\lambda$
0
1
t-1
t
DL
μ_1
μ_2
$\mu_{t\text{-}1}$
μ_t
$(d\text{-}(t\text{-}1))\lambda(d\text{-}t)h$

Figure 1: *A typical reliability model for a storage array with MDS t-fault tolerant erasure code.*

The arc from State $t-1$ to State DL shows the rate at which the system encounters a hard error during rebuild of t disks. This rate is computed as the product of the rate that a disk fails after $(d-(t-1))$ disks have failed, namely $(d-(t-1))\lambda$, and the probability of encountering a hard error when reading the necessary $(d-t)$ disks for rebuild, namely $(d-t)h$.

Label	**Description**	**Value**
λ	drive failure rate, or $1/\text{MTTF}_d$	1/500000hrs
μ	drive rebuild rate, or $1/\text{MTTR}_d$	1/12hrs
h	probability of an uncorrectable error during rebuild per drive read	C.HER
C	drive capacity	300GB
HER	hard error rate, in errors per number of bytes read	8×10^{-15}

Table 1: *Model parameters and their numerical assignments. The drive rebuild rate is assumed to be independent of number of failed drives.*

.

The transition matrix for this model is given by

$$\begin{pmatrix} -a_0 & a_0 & & & & & 0 \\ \mu_1 & -(\mu_1+a_1) & a_1 & & & & 0 \\ & \ddots & \ddots & \ddots & & & \vdots \\ & & \mu_{t-2} & -(\mu_{t-2}+a_{t-2}) & a_{t-2} & & 0 \\ & & & \mu_{t-1} & -(\mu_{t-1}+a_{t-1}) & a_{t-1}(1-b_t) & a_{t-1}b_t \\ & & & & \mu_t & -(\mu_t+a_t) & a_t \end{pmatrix} \tag{2.1}$$

where $a_k = (d-k)\lambda$ and $b_k = (d-k)h$, for $k = 0,\ldots,t$.

Setting $\mu_k = \mu$ and solving this model (as in [6]), we derive the formula

$$\text{MTTDL} = \frac{\mu^t M_t(\lambda/\mu, d, h)}{d(d-1)\cdots(d-t)\lambda^t(\lambda+h\mu)}, \tag{2.2}$$

where $M_t(x,d,h)$ is a polynomial of degree t, with constant term 1 that is also linear in h. Here are a few examples:

$$\begin{aligned} M_2(x,d,h) &= 1 + x((2d-2) + h(d-1)(d-2)) \\ &\quad + x^2((3d^2-6d+2) - hd(d-1)(d-2)) \\ M_3(x,d,h) &= 1 + x((2d-3) + h(d-2)(d-3)) \\ &\quad + x^2(3(d-1)(d-2) + h(2d-1)(d-3)) \\ &\quad + x^3(2(d^2-3d+1)(2d-3) - hd(d-1)(d-2)(d-3)). \end{aligned}$$

In general, the coefficient of x in these polynomials is given by

$$(2d-t) + h(d-(t-1))(d-t)$$

and the coefficient of x^t is given by

$$P_t(d) - hd(d-1)\cdots(d-t)$$

where $P_t(d)$ is a (complicated) polynomial of degree t in d. It may be possible to determine recursive formulas for M_t but we have not done so yet.

Typically, h is many orders of magnitude smaller than d; this means that $M_t(\lambda/\mu,d,h) \approx M_t(\lambda/\mu,d,0)$, which simplifies the formulation of the result. Also, as in [4], $\mu \gg \lambda$; this implies that $M_t(\lambda/\mu,d,h) \approx 1$ and so the above formula simplifies further to

$$\text{MTTDL} \approx \frac{\mu^t}{d(d-1)\cdots(d-t)\lambda^t(\lambda+h\mu)}. \tag{2.3}$$

3. Non-MDS codes with elastic fault tolerance

In this section we generalize the basic model to cover the case where the erasure code can tolerate many (but not necessarily) all instances of k erasures for $k = 1, \ldots, t$ where t is the limit after which no erasures can be tolerated. We call this t the "upper threshold fault tolerance". (If the code has n data inputs and codeword size $n+m$, then necessarily $t \le m$. We do not need this fact explicitly, but it is useful when trying to determine the parameters in the model.) The Markov model for this system is given by the diagram in Figure 2; we describe the transition rates σ_k and δ_k below and as before, for the numerical calculations we set $\mu_k = \mu$.

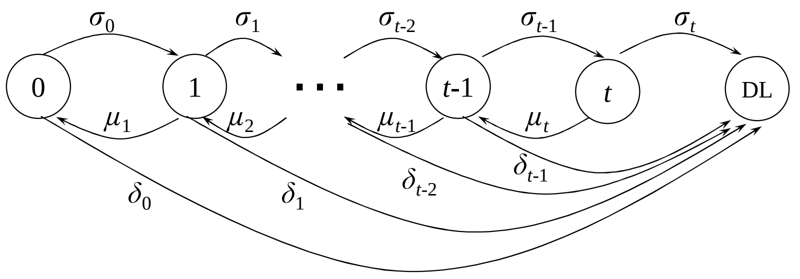


Figure 2: *Reliability model for a storage array with non-MDS erasure code.*

Let p_k be the probability that the erasure code can tolerate one more disk failure, given that it has already tolerated k failures (and so is in rebuild mode). The number t no longer represents the Hamming fault tolerance of the code, but the upper threshold fault tolerance, or the maximal number of faults for which the erasure code can tolerate at least one instance of that many failures. For $0 \le k \le t-1$, the transition rates are given by

$$\delta_k = (d-k)\lambda \left\{(1-p_k) + p_k(1-p_{k+1})(d-(k+1))h\right\} \qquad (3.1)$$

$$\sigma_k = (d-k)\lambda p_k \left\{1-(1-p_{k+1})(d-(k+1))h\right\} \qquad (3.2)$$

and $\delta_t = 0$, $\sigma_t = (d-t)\lambda$. Note that $\sigma_k + \delta_k = (d-k)\lambda = a_k$.

There are two scenarios that can lead from a non-data loss State k to State DL. First, from State k where k disks have failed, a transition may occur because a $k+1$st disk fails *and* the code cannot tolerate this additional failure. This contributes a term $(d-k)\lambda(1-p_k)$ since $(d-k)\lambda$ is the rate that another disk may fail and $(1-p_k)$ is the probability that, given the system has survived k disk losses, it *cannot* tolerate another. This conditional probability reflects the present state at State k. Second, from State k, it is possible to arrive at State DL because (a) another disk fails, (b) the erasure code in principle can tolerate this failure, (c) a hard error occurs during rebuild and (d) the erasure code *cannot* tolerate this hard error (essentially equivalent to yet another disk failure that cannot be tolerated). This contributes a term $(d-k)\lambda p_k(1-p_{k+1})(d-(k+1))h$. These two terms explain (3.1).

In order for the system to transition from State k to State $k+1$ (and not incur a data loss), it must be the case that (a) a disk fails, (b) the erasure code can tolerate this additional failure and (c) during rebuild, either no hard error occurs that the code cannot tolerate *or* a hard error does occur but the code *can* tolerate it. The first item contributes a factor $(d-k)\lambda$. The second item contributes a factor p_k and the third item contributes $(1-(d-(k+1))h)+p_{k+1}(d-(k+1))h = \{1-(1-p_{k+1})(d-(k+1))h\}$.

Observe that for an MDS code with fault tolerance exactly t, then $p_k = 1$ for $k \leq t-1$ and $p_k = 0$ for $k \geq t$. In this case, the model reduces to the model presented above in Section 2. Note that in our model, $p_t = 0$.

The transition matrix for this model is given by

$$\begin{pmatrix} -a_0 & \sigma_0 & & & & & \delta_0 \\ \mu_1 & -(\mu_1+a_1) & \sigma_1 & & & & \delta_1 \\ & \ddots & \ddots & \ddots & & & \vdots \\ & & \mu_{t-2} & -(\mu_{t-2}+a_{t-2}) & \sigma_{t-2} & & \delta_{t-2} \\ & & & \mu_{t-1} & -(\mu_{t-1}+a_{t-1}) & \sigma_{t-1} & \delta_{t-1} \\ & & & & \mu_t & -(\mu_t+a_t) & a_t \end{pmatrix} \tag{3.3}$$

where $a_k = (d-k)\lambda$ and δ_k and σ_k are given by (3.1) and (3.2), respectively. This is very similar to that given in (2.1). The main diagonal of the transition matrix is unchanged. The upper diagonal is now replaced by σ_k, which reflects the probabilities of surviving an extra disk loss without a hard error (as described above) and generalizes the term $a_{t-1}(1-b_t)$ in the next to the last row and column of (2.1). The last column (except for last row) is replaced by δ_k and generalizes the element in the last column, next to last row.

The formula for the solution can be given explicitly, but it is quite complicated. However, we apply the model to various examples in each of the next subsections.

3.1. Computing the conditional probabilities

To the compute the conditional probabilities p_k defined as the probability that the erasure code can tolerate an additional disk failure given that it has failed and tolerated k disk losses, we proceed as follows. As above, let d be the number of drives in the array. Let n_k be the total number of possible k disk failure instances, so that $n_k = \binom{d}{k}$. Let s_k be the number of such k-disk failure instances which the erasure code can tolerate. Then q_k, the *unconditional* probability that the code can tolerate k failures, is given by

$$q_k = \frac{s_k}{n_k}.$$

To compute the conditional probabilities, p_k, we use the formula $\Pr(X|Y) = \Pr(X \cap Y)/\Pr(Y)$. If $X = X_k$, the event of surviving k faiulres, then $\Pr(X_{k+1} \cap X_k) = \Pr(X_{k+1})$, since a system cannot survive $k+1$ failures without also surviving every instance of k failures that it contains. Hence, with $q_0 = s_0 = 1$, we have $p_k = q_{k+1}/q_k$, or, by the above,

$$p_k = \frac{s_{k+1}}{\binom{d}{k+1}} \Bigg/ \frac{s_k}{\binom{d}{k}} = \frac{s_{k+1}(k+1)}{s_k(d-k)}.$$

To compute s_k, we simply test a specific erasure code to determine what k disk failure combinations it can tolerate (we do with with simple tests on the generator matrix as in [3]).

3.2. LDPC code example

The general model of reliability given here is ideally suited to understanding the reliability of LDPC codes as they might be applied to storage arrays. For more complex systems, for example, distributed systems, the model given here is insufficient since there are many more components in such systems (nodes, networks, switches, etc.) and these need to modeled more carefully. However, the notions described here can be (and should be) extended to the more complex systems.

For this section, we give one example – that taken from the RAID tutorial given by Plank [5]. Plank's example LDPC code has generator matrix:

$$\left(\begin{array}{cccccccccccccccc|cccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&1&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1
\end{array}\right). \tag{3.4}$$

By examining this matrix, it is not hard to see that the Hamming fault tolerance of this code is 1 (that is, it can tolerate all instances of 1 fault, but not all 2-fault instances). For example, the first row implies that the first data element appears only in the first parity, hence the loss of that row and that parity is a data loss event. Clearly, it has maximum fault tolerance at most 4 since there is one data element that touches all four parity.

Clearly, $t \le 20 - 16$ for this code (by a simple dimensionality argument). By a straightforward calculation, the counts s_k can be computed as

$$\{s_0, s_1, s_2, s_3, s_4\} = \{1,\ 20,\ 185,\ 969,\ 2515\},$$

out of $\{1, 20, 190, 1140, 4845\}$ possibilities, respectively, and $s_k = 0$ for $k \ge 5$. Consequently, the unconditional probablities q_k for this code are

$$\begin{aligned}\{q_0, q_1, q_2, q_3, q_4\} &= \{1,\ \tfrac{20}{20},\ \tfrac{185}{190},\ \tfrac{969}{1140},\ \tfrac{2515}{4845}\}\\ &= \{1.0,\ 1.0,\ 0.97,\ 0.85,\ 0.52\}\end{aligned}$$

and the conditional probabilities p_k are given by

$$\begin{aligned}\{p_0, p_1, p_2, p_3, p_4\} &= \{1, \tfrac{37}{38}, \tfrac{323}{370}, \tfrac{10060}{16473}\} \\ &= \{1.0,\ 0.97,\ 0.87,\ 0.61,\ 0.0\}\end{aligned}$$

The WEAVER codes [2] also have elastic fault tolerance. We consider here four WEAVER codes with Hamming fault tolerance 1, 2, 3 and 4. The WEAVER parameter sets we use are $\{1\}$, $\{1{,}2\}$, $\{2{,}3{,}4\}$ and $\{1{,}2{,}3{,}6\}$, respectively. Among the many choices, these have the smallest maximal value in each set, so should have better elastic fault tolerance (because their localization property is optimal). The last code of 4-fault tolerance may be the one used by Plank [5], though for his calculations, the specific choice did not matter.

On 20 drives, the counts s_k for each of these WEAVER codes are given in Table 2 (the last row gives the total number of failure combinations). It seems remarkable that these codes, though they have relatively small Hamming fault tolerance, have such high survival rates even when half the drives are lost.

		Number of disk failures k										
params	**FT**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\{1\}$	1	1	20	170	800	2275	4004	4290	2640	825	100	2
$\{1,2\}$	2	1	20	190	1100	4225	11044	19440	21960	14300	4200	246
$\{1,2,4\}$	3	1	20	190	1140	4785	14544	31940	48960	48040	24460	3166
$\{1,2,3,6\}$	4	1	20	190	1140	4845	15444	37780	70120	93080	75320	19084
n_k		1	20	190	1140	4845	15504	38760	77520	125970	167960	184756

Table 2: *WEAVER code survival counts on 20 drives. The last row, n_k, indicates the total number of failure combinations.*

	Hamming fault tolerance t			
$\mu = 1/8$	**1**	**2**	**3**	**4**
LDPC	2.07×10^{7}			
WEAVER	5.17×10^{6}	5.39×10^{10}	5.61×10^{14}	6.82×10^{18}
MDS	5.45×10^{5}	1.89×10^{9}	6.96×10^{12}	2.72×10^{16}
$\mu = 1/12$				
LDPC	2.06×10^{7}			
WEAVER	5.16×10^{6}	3.58×10^{10}	2.48×10^{14}	2.09×10^{18}
MDS	5.43×10^{5}	1.26×10^{9}	3.08×10^{12}	8.03×10^{15}

Table 3: *Comparison of* MTTDL *of an LDPC on 20 disks (with Hamming fault tolerance* 1 *and average fault tolerance 3.81) versus WEAVER and MDS codes on 20 disks with Hamming fault tolerance $t = 1,2,3,4$.*

Table 3 shows the MTTDL for the LDPC code above, the four WEAVER codes and MDS codes on 20 drives with Hamming fault tolerance between 1 and 4. We do not specify a particular MDS code here, since all MDS codes on 20 drives with a specified fault tolerance will have the same conditional probabilites (1 up to fault tolerance) and so the same MTTDL. For these calculations, we used the values for the parameters in Table 1 and add an additional table for $\mu = 1/8$hrs. The results indicate that the LDPC code, though it has high average fault tolerance, does not have the system reliability of either an MDS or a WEAVER two-fault tolerant code (the LDPC code is two or three orders of magnitude less). However, in spite of the fact that the LDPC code has Hamming fault tolerance equal to one, it has two orders of magnitude better reliability than a 1-fault tolerant MDS code (e.g., RAID5). The WEAVER code has better fault tolerance than an equivalent MDS code, by one or two orders of magnitude across all the fault tolerance levels. It should be noted, however that the storage efficiency of the WEAVER codes is exactly 50%, and of the MDS codes is 95%, 90%, 85% and 80%, respectively. The LDPC code has storage efficiency 80%. So the extra reliability comes at a cost of storage efficiency.

3.3. Multiple Independent Arrays

As discussed in the introduction, when extrapolating reliability from one array to many arrays, the typical approach is to simply divide (or multiply depending on the metric) by the number of arrays. However, this is not entirely accurate, though it is a good approximation in practice as we now show. To see the model difference heuristically, consider two RAID5 arrays on 8 disks each. Nominally, each has Hamming fault tolerance one and with two such arrays, the expected extrapolation factor is exactly 2. Clearly, this *system* can tolerate all instances of one disk failure. However, it can tolerate 64 of the 120 possible instances of two drive failures (or 53%), that is, more than half. (It cannot tolerate any three failures so it has upper threshold fault tolerance 2.) This suggests that the MTTDL of this system may be somewhat better than exactly half that for a single array of 8 disks. It also shows that our “elastic fault tolerance” model applies directly.

To apply our model, we need the survival counts. Suppose we have a system of r arrays, and each array has d disks with an MDS t-fault tolerant erasure code. This means that each array can independently tolerate all instances of t disk failures but no instances of $t+1$ failures (though the system as a whole certainly can). Furthermore, it is clear that the system cannot tolerate any combination of $tr+1$ disks failures as that would imply at least $t+1$ on at least one array.

So, rd plays the role of d (number of disks) and tr plays the role of t (upper threshold fault tolerance) in the model. Consequently, $n_k = \binom{rd}{k}$. The survival counts s_k are given by

$$s_k = \sum_{\substack{r_0+r_1+\cdots+r_t=r \\ 0r_0+1r_1+2r_2+\cdots+tr_t=k}} \binom{r}{r_0, r_1, \ldots, r_i} \prod_{i=1}^{t} \binom{d}{i}^{r_i}. \tag{3.5}$$

The expression before the product is the multinomial coefficient, which counts the number of combinations of r rolls of a $t+1$ sided dice. This equals the number of ways we can distribute 0 failures on r_0 arrays each, 1 failure on r_1 arrays, etc., up to t failures on r_t array. The total number of arrays must be r and the total number of failures must be k. This explains the conditions on the sum. Given that there are r_i arrays with i failures, there are $\binom{d}{i}^{r_i}$ ways those failures can be distributed on those arrays, which accounts for the product term in each summand. Another way to determine this value is as the coefficient of x^k in the expansion of

$$\left(1+\binom{d}{1}x+\binom{d}{2}x^2+\cdots+\binom{d}{t}x^t\right)^r.$$

Table 4 gives a summary of the MTTDL for various configurations of drives and array sizes and Hamming fault tolerance (with λ, h and $\mu_k = \mu$ as in Table 1). The first number in each cell is the MTTDL number computed with the parameter values from Table 1. The second number in each cell is the r times the ratio of the 1-array case to the r-array case, that is, comparing the traditional model to this more accurate model. Over this broad range of parameters, the traditional model, though quite close to the more accurate model, does not in fact agree with it.

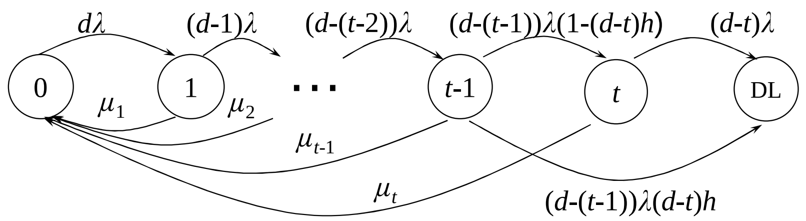
		number of arrays r					
$t = 1$		**1**	**2**	**4**	**8**	**16**	**32**
d	**4**	1.7192×10^7	8.5952×10^6	4.2968×10^6	2.1476×10^6	1.0730×10^6	5.3567×10^5
		1.0000	0.9999	0.9997	0.9993	0.9986	0.9970
	8	3.6847×10^6	1.8420×10^6	9.2066×10^5	4.5998×10^5	2.2964×10^5	1.1447×10^4
		1.0000	0.9998	0.9994	0.9987	0.9972	0.9942
	12	1.5635×10^6	7.8153×10^5	3.9055×10^5	1.9506×10^5	9.7309×10^4	4.8436×10^4
		1.0000	0.9997	0.9992	0.9980	0.9958	0.9913
	16	8.601×10^5	4.2989×10^5	2.148×10^5	1.0723×10^5	5.3458×10^4	2.6571×10^4
		1.0000	0.9996	0.9989	0.9974	0.9945	0.9886
$t = 2$							
d	**4**	3.5816×10^{11}	1.7904×10^{11}	8.9485×10^{10}	4.4708×10^{10}	2.2320×10^{10}	1.1125×10^{10}
		1.0000	0.9998	0.9994	0.9986	0.9971	0.9940
	8	2.5588×10^{10}	1.2788×10^{10}	6.3889×10^9	3.1895×10^9	1.5898×10^9	7.9003×10^8
		1.0000	0.9995	0.9987	0.9972	0.9941	0.9880
	12	6.5145×10^9	3.2550×10^9	1.6256×10^9	8.1088×10^8	4.0357×10^8	1.9992×10^8
		1.0000	0.9993	0.9981	0.9958	0.9912	0.9820
	16	2.5598×10^9	1.2787×10^9	6.3833×10^8	3.1817×10^8	1.5811×10^8	7.8079×10^7
		1.0000	0.9991	0.9975	0.9944	0.9882	0.9761
$t = 3$							
d	**4**	1.4923×10^{16}	7.4592×10^{15}	3.7274×10^{15}	1.8615×10^{15}	9.2864×10^{14}	4.6218×10^{14}
		1.0000	0.9997	0.9991	0.9980	0.9957	0.9911
	8	2.1323×10^{14}	1.0654×10^{14}	5.3209×10^{13}	2.6543×10^{13}	1.3210×10^{13}	6.5443×10^{12}
		1.0000	0.9993	0.9982	0.9959	0.9913	0.9821
	12	3.0159×10^{13}	1.5064×10^{13}	7.5187×10^{12}	3.7463×10^{12}	1.8602×10^{12}	9.1726×10^{11}
		1.0000	0.9990	0.9972	0.9938	0.9869	0.9732
	16	8.2043×10^{12}	4.0964×10^{12}	2.0434×10^{12}	1.0170×10^{12}	5.0380×10^{11}	2.4726×10^{11}
		1.0000	0.9986	0.9963	0.9916	0.9825	0.9644

Table 4: MTTDL *comparison for multi-arrays versus a single array. Each cell contains the computed* MTTDL *according to the multi-array model and the ratio* $r * \mathrm{MTTDL}(1)/\mathrm{MTTDL}(r)$.

4. Rebuild-in-parallel

In the two models above, the rebuild occurs incrementally, from k failed disks to $k-1$ failed disks. This does not necessarily model actual system behaviors. There are systems which apply the rebuild in this manner in order to reduce vulenerability of a system as quickly as possible, especially, if the rebuild time for more than one drive is significantly more than that for a single drive. However, many other systems implement rebuild of multiple drives in parallel; that is, they go from a multi-failure state back to the errror-free state directly. Figure 3 shows a model that more closely resembles a standard implementation for an MDS code where rebuild returns to the fully operational state from any given failure state (except the DL state). Here, μ_k is the rate at which the system can rebuild from k failures to fully functional; that is, rebuild k failed disks simultaneously. The same variation can be applied to the model in Figure 2 but we do not do so here.

It can be argued that in most systems and depending on system resources, the rebuild time for t drives is essentially equal to the rebuild times for 1 drive. In particular, this will happen unless there are a huge number of drives and the memory or disk interconnect bandwidth becomes a bottleneck. Consequently, for the rest of this discussion, we assume that $\mu_k = \mu$, a constant. (This is also a reasonable assumption for the purposes of this paper, since, as noted above, we are not concerned with extremely accurate reliability predictions, but on relative comparison of different models).



$d\lambda$
$(d\text{-}1)\lambda$
$(d\text{-}(t\text{-}2))\lambda$
$(d\text{-}(t\text{-}1))\lambda(1\text{-}(d\text{-}t)h)$
$(d\text{-}t)\lambda$
0
1
t-1
t
DL
μ_1
μ_2
$\mu_{t\text{-}1}$
μ_t
$(d\text{-}(t\text{-}1))\lambda(d\text{-}t)h$

Figure 3: *Revised reliability model for single array, with MDS t-fault tolerant code, with rebuild proceeding in parallel directly back to "zero" loss state.*

The transition matrix is given by

$$\begin{pmatrix} -a_0 & a_0 & & & & & 0 \\ \mu_1 & -(\mu_1+a_1) & a_1 & & & & 0 \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \mu_{t-2} & & -(\mu_{t-2}+a_{t-2}) & & a_{t-2} & & 0 \\ \mu_{t-1} & & & & -(\mu_{t-1}+a_{t-1}) & a_{t-1}(1-b_t) & a_{t-1}b_t \\ \mu_t & & & & & -(\mu_t+a_t) & a_t \end{pmatrix} \tag{4.1}$$

where, as before, $a_k = (d-k)\lambda$ and $b_t = (d-t)h$.

Note that this differs from the basic model in Figure 1 by left shifting all the μ_k terms on the lower diagonal to the first column. The solution, when $\mu_k = \mu$, a constant, is given by the formula

$$\text{MTTDL} = \frac{\mu^t \widetilde{M}_t(\lambda/\mu, d, h)}{d(d-1)\cdots(d-t)\lambda^t(\lambda+h\mu)}, \tag{4.2}$$

where $\widetilde{M}_t(x,d,h)$ is a polynomial of degree t in x, is linear in h and has constant term equal to 1. That is, it has a form very similar to (2.2), up to first order terms. The denominators of the two models are the same. The algebraic difference between the numerators of the two MTTDLs is given by

$$\lambda\mu^{t-1}Q_{t-2}(\lambda/\mu, d, h)$$

where $Q_{t-2}(x,d,h)$ is a polynomial of degree $t-2$ in x, is linear in h and has constant term given by

$$(t-1)(d-t/2)-h(d-(t-1))(d-t).$$

Consequently,

$$\mathrm{MTTDL}_{\mathrm{parallel}} = \mathrm{MTTDL}_{\mathrm{incr}}\left(1+O\left(\frac{dt\lambda}{\mu}\right)\right)$$

With $\mu \gg \lambda$, we get the same approximation result as in (2.3).

Table 5 shows that, numerically, the MTTDL is essentially the same between the two models. The parameters λ and h are fixed as in Table 1. We vary the number of drives in the array and the mean time to rebuild at 8hrs and 12hrs; these are the two "tunable" parameters (MTTF_d and h are determined by the inherent characteristics of the drives). There are two subtables, one each for $t=2$ and $t=3$. Clearly, though the models and the closed form solutions are different, the impact on the MTTDL results is minimal. In each cell, the top number is the MTTDL for the traditional model, the middle number is the MTTDL for this revised model and the bottom number is the ratio.

	number of drives d				
$\mu = 1/8$	**8**	**12**	**16**	**20**	**24**
$t=2$	3.8505×10^{10} 3.8509×10^{10} 0.9999	9.8024×10^{9} 9.8041×10^{9} 0.9998	3.8515×10^{9} 3.8524×10^{9} 0.9998	1.8922×10^{9} 1.8927×10^{9} 0.9997	1.0659×10^{9} 1.0663×10^{9} 0.9997
$t=3$	4.8130×10^{14} 4.8140×10^{14} 0.9998	6.8071×10^{13} 6.8094×10^{13} 0.9997	1.8516×10^{13} 1.8525×10^{13} 0.9995	6.9565×10^{12} 6.9605×10^{12} 0.9994	3.1723×10^{12} 3.1745×10^{12} 0.9993
$\mu = 1/12$					
$t=2$	2.5588×10^{10} 2.5592×10^{10} 0.9998	6.5145×10^{9} 6.5162×10^{9} 0.9997	2.5598×10^{9} 2.5607×10^{9} 0.9997	1.2577×10^{9} 1.2582×10^{9} 0.9996	7.0853×10^{8} 7.0890×10^{8} 0.9995
$t=3$	2.1323×10^{14} 2.1329×10^{14} 0.9997	3.0159×10^{13} 3.0174×10^{13} 0.9995	8.2043×10^{12} 8.2099×10^{12} 0.9993	3.0825×10^{12} 3.0852×10^{12} 0.9991	1.4058×10^{12} 1.4073×10^{12} 0.9989

Table 5: *Comparison of basic rebuild-incremental model to the revised rebuild-in-parallel model. Each cell contains the* MTTDL *for the basic rebuild-incremental model, the* MTTDL *for the rebuild-in-parallel model and the ratio.*

5. Summary

In this paper, we discussed two variations on the basic model used to measure reliability (or MTTDL) for storage arrays with MDS erasure codes. The first variation extends the traditional model to non-MDS erasure codes with elastic fault tolerance by reflecting the fact that a non-MDS code can recover from some (but not all) instances of multiple failures, more than that determined by the Hamming fault tolerance. We applied this model to two examples: (a) an LDPC code and

WEAVER codes compared to the MDS model and (b) multiple independent arrays. In the first example, we showed the the LDPC code's MTTDL does not reflect the large average fault tolerance; in the second example, we showed that the traditional method of treating each array independently is a very good approximation to the more accurate model given here. The second model variation reflected the fact that most storage systems rebuild all failed disks simultaneously and restore to the "fully functional" state in one transition; that is, they do not incrementally restore disks. The numerical results indicate that this variation has little effect on the predicted MTTDL.

References

[1] A. Dholakia, E. Eleftheriou, X.-Y. Hu, I. Iliadis, J. Menon, and KK Rao. Analysis of a new intra-disk redundancy scheme for high reliability raid storage systems. Technical Report RZ 3652, IBM Research, Zurich, Switzerland, 2005.

[2] J. L. Hafner. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *Proceedings of the Fourth USENIX Conference on File and Storage Technologies*, pages 211–224, San Francisco, CA USA, December 2005.

[3] J. L. Hafner, V. Deenadhayalan, KK Rao, and J. A. Tomlin. Matrix methods for lost data reconstruction in erasure codes. In *Proceedings of the Fourth USENIX Conference on File and Storage Technologies*, pages 183–196, San Francisco, CA USA, December 2005.

[4] KK Rao, J. L. Hafner, and R. Golding. Reliability for networked storage nodes. In *Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN'06)*, pages 237–246, Philadephia, PA USA, June 2006.

[5] J. Plank. Erasure codes for storage applications, December 2005. `http://www.cs.utk.edu/˜plank/plank/papers/FAST-2005.html`.

[6] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley, 2nd edition, 2001.