Durability and Availability of Erasure-Coded Systems with Concurrent Maintenance

Şuayb Ş. Arslan

"*All truths are easy to understand once they are discovered; the point is to discover them.*" G. Galilei.

Version 0.1 – April 2014

This document is written to establish some of the fundamentals of reliability theory as well as to identify the theoretical machineries for the prediction of durability/availability of the erasure-coded storage systems. Most of the contents of this document are dedicated to a review of fundamentals as well as major improved stochastic models. Yet, it also has many contributions. One of the contributions of this document is the introduction of one of the most general form of Markov models for the estimation of mean time to failure numbers. Very good approximations for the closed form solutions for this general model are investigated. Various storage configurations under different policies are compared using such advanced models. Topics that include simulation modelings for more accurate estimations are included towards the end of the document by noting the deficiencies of the simplified canonical Markov models. Throughout the document, we shall focus on concurrently maintained systems although the discussions will only slightly change for the systems repaired one device at a time. Some background on probability and coding theory is expected.

1 DEVICE RELIABILITY BASICS

When a brand new product is put into service, it performs functional operations satisfactorily for a period of time, called *useful time* period, before eventually a failure occurs and the device becomes no longer able to respond to incoming user requests. For a device component, the observed time to failure (TTF) is a continuous random variable with a probability density function $f_{TTF}(t)$, representing the lifetime of the product until the first permanent

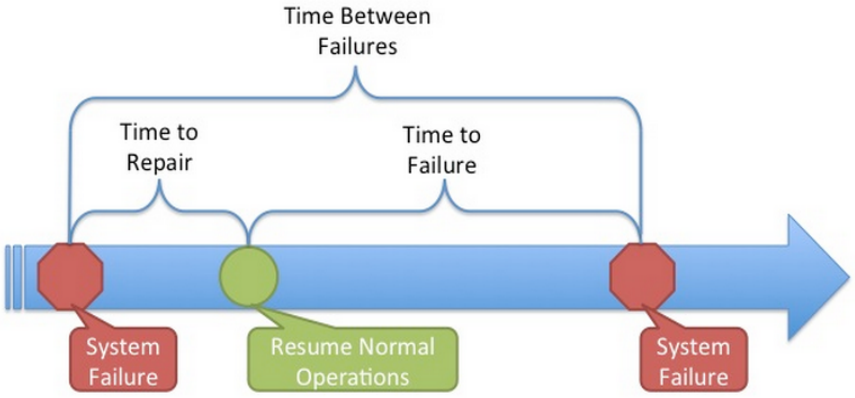

Figure 1.1: Mean time to failure and repair visualized in time domain.

failure. The probability of failure of the device can be found using the cumulative distribution function (CDF) of TTF as follows,

$$F_{TTF}(t) \triangleq Pr\{TTF \leq t\} = \int_0^t f_{TTF}(y)dy, \quad t > 0 \tag{1.1}$$

We can think of $F_{TTF}(t)$ as an *unreliability* measure between time 0 and x. On the other hand, the reliability function $R(t)$ is defined as,

$$R(t) \triangleq 1 - F_{TTF}(t) = \int_t^\infty f_{TTF}(y)dy \tag{1.2}$$

In other words, reliability is the probability of having no failures before time t and is related to CDF of TFF through Eqn. (1.2). Note that Eqn. (1.2) implies that we have $f_{TTF}(t) = -dR(t)/dx$. In a real world, it may not be possible to estimate the distribution function of TFF directly from the available physical information. A useful function for clarifying the relationship between physical modes of failure and the probability distribution of TFF is known as the *hazard rate* function or *failure rate* function, frequently denoted as $h_{TTF}(t)$. This function is defined as follows,

$$h_{TTF}(t) \triangleq \frac{f_{TTF}(t)}{R(t)} = -\frac{dR(t)}{R(t)dt} \tag{1.3}$$

Solution of the first order ordinary differential equation (1.3) yields the relationship $h_{TTF}(t) = -d(\ln(R(t)))/dt$ with the initial condition $R(0) = 1$. Note that knowing the hazard rate function is equivalent to knowing the distribution. Thus, when we talk about reliability of a system, we interchangeably use these functions to quantify it. Mean time expected until the first failure of a piece of equipment or a total data loss is one of the most popular measures of reliability. Mean time to failure (MTTF) is defined to be the expected value of the random variable TTF and is given by

$$MTTF \quad = \quad \mathbb{E}[TTF] = \int_0^\infty R(t)dt \Leftrightarrow \lim_{t\to\infty} tR(t) = 0 \tag{1.4}$$

Figure 1.2: A simple parity coding that appends a parity bit at the end of the block in order to keep the total number of 1's even.

Note that Eqn. (1.4) is true for distributions whose mean exists. For the rest of our discussions, the subscript TTF is dropped for notation simplicity.

Annualized failure rate (AFR) for a device is frequently used to estimate the failure probability of a device or a component after a full-time year use. In the conventional approach, time between the start of the operation and the point when failure happens are assumed to be independent and exponentially distributed with a constant rate λ. Therefore AFR is given by $AFR = 1 - R(t) = 1 - e^{-\lambda t}$, where $\lambda = 1/MTTF$ and x is the running time index in hours. MTTF is reported in hours and since there are 8760 hours in a year, $AFR = 1 - e^{-8760/MTTF}$. Since $8760/MTTF \ll 1$, then $AFR \approx 8760/MTTF$. Alternatively, if MTTF is expressed in years, then $AFR \approx 1/MTTF$.

In this document, if a device fails we assume that there is an external agent who can repair it. The time needed for the agent to replace or repair a failed device is also a random variable whose mean is called Mean Time To Repair (MTTR). To avoid MTTR, many companies purchase spare products/hot swaps so that a replacement can be installed quickly. Generally, however, customers inquire about the turn-around time of repairing a product, and indirectly, this would eventually fall into the MTTR category. Finally, Mean Time Between Failure (MTBF) is a reliability measure used to give the time between two consecutive failures of the same device or system component. This is the most common inquiry about a product's life span, and is important in the decision-making process of the end user. Fig. 1.1 summarizes these terms using a timeline of an operational storage system.

2 Coding Basics for Reliability

In real storage systems, the information contained in data nodes or devices are encoded to generate some form of redundancy in order to make the user data robust against device failures/defects or random/burst errors. Whenever storage nodes/devices fail, the system controller identifies those device failures and hence the failure locations are easily determined. From a coding perspective, these failures are regarded as erasures. Therefore, the process of creating this redundancy is called erasure correction coding or simply *erasure coding*. Erasure coding is used to increase the reliability of the overall storage system. Error/erasure correction coding is quite old and inclusive subject, originated almost fifty years ago. We will only consider erasure coding in the rest of this document.

Suppose that we have n devices that contain m blocks of user data. Then, we use erasure coding to generate redundancy that consists of a subdivision of that redundancy into c blocks and storing them in c different devices i.e., the total number of devices is $n = m + c$. A space optimal (also known as Maximum Distance Separable (MDS)) coding scheme allows

us to reconstruct the m device information from the information contained in any m out of n devices. Thus one can realize that MDS codes enable a regular and understandable recovery mechanism. A simple instance of MDS codes is the XOR-based *parity coding* scheme illustrated in Fig. 1.2. Using the “even" convention, the total number of binary 1's are kept even by adding one more binary digit at the end of the block. In linear algebra terms, for a given (n, m) linear code defined by the generator matrix $\mathbf{G}$, if all $m \times m$ submatrices of $\mathbf{G}$ are invertible, then this linear code can tolerate any combination of c erasures. The class of codes that has this property are called MDS codes.

We can divide erasure codes into three classes : flat MDS codes, parity–check array codes and flat XOR–based codes. Reed-Solomon codes are example of flat MDS codes while exhibiting computationally demanding implementation in practice. Most array codes are space–optimal in the number of devices and less computationally expensive than Reed-Solomon, but are typically only two or three device fault–tolerant [1], [2], [3]. Since the main purpose of this document is to present the reliability issues of these codes with respect to maintainable systems, the details of these codes are not presented. A good tutorial about erasure coding for storage applications can be found at [4].

On the other hand, most of the modern codes are flat XOR-based codes and used to protect the data in storage applications. They are based on low complexity, low locality[1], irregular erasure correction coding i.e., codes with irregular failure tolerance. Such codes are designed by considering the overall storage problem as a whole and they necessitate major changes to modeling assumptions which lead to interesting open problems regarding the reliability/durability/availablity of the storage systems. These open problems shall be covered throughout the document.

3 A Markov Reliability Model with Concurrent Maintenance

Traditionally, Markov models are used to evaluate the reliability of erasure-coded storage systems. Since we deal with system failures or total data loss scenarios, the class of Markov models we consider has absorbing states. An absorbing state is a system failure state or a total data loss state that, once entered, cannot be departed. Most models assume a RAID-like setting (i.e. MDS code – regular fault tolerance as explained previous sections), independence between failures and exponentially distributed failure/rebuild times. For this case, Markov models are perfect fit for modeling the system behavior. In fact, Markov models have provided a great deal of insight into the sensitivity of disk failure and repair on system reliability, despite these models generally capture an extremely simplistic view of an actual system.

The canonical Markov model used in storage systems is based on parity coding, which can tolerate only one device failure out of say, n storage units or devices. There are a total of three states; state n represents the state of all devices are active and operational, state $n-1$ represents the state of one failed device and state F is the state of failure because at least two

[1] In the context of erasure coding for storage, the term locality means the number of storage units or coding symbols that need to be accessed for recovering a specific unit or symbol. Locality can be considered in terms of purely coding perspective or coding+allocation strategy perspective. The context in which the locality is referred matters and may bring out different conclusions.

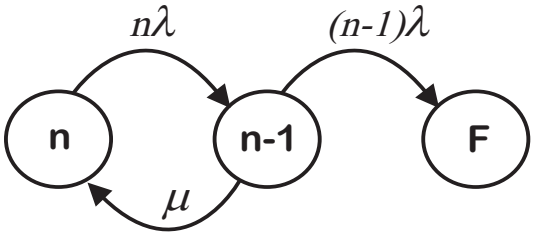

Figure 3.1: A canonical continuous time Markov model for the reliability estimation of parity-coded data.

devices are failed and data integrity is lost. It is assumed that all devices fail or are repaired at the constant rates λ and μ, respectively. State diagram of the Markov model is shown in Fig. 3.1. Let us start with the following lemma and use its result to talk about holding times i.e., the time it takes to hold in a specific state.

Lemma 1: *Let $X_1,\ldots,X_n$ be independent exponentially distributed random variables with rates λ_i for $i = 1,\ldots,n$. Then, the distribution function of* $\min\{X_1,\ldots,X_n\}$ *is exponential with rate $\sum_i \lambda_i$ and the probability that the minimum is X_s, $s \in \{1,\ldots,n\}$ is given by $\lambda_s/\sum_i \lambda_i$.*

PROOF: Let us consider the probability,

$$Pr\{\min\{X_1,\ldots,X_n\} > t\} = \prod_i Pr\{X_i > t\} = e^{-(\sum_i \lambda_i)t} \tag{3.1}$$

which means that the cumulative distribution function of $\min\{X_1,\ldots,X_n\}$ is that of an exponential distribution with rate $\sum_i \lambda_i$. One can realize that it is not the case with $\max\{X_1,\ldots,X_n\}$. Furthermore,

$$\begin{aligned}
Pr\{X_s \text{ is the minimum}\} &= Pr\{X_s < X_j \text{ for } j \neq s\} && (3.2)\\
&= \int_0^\infty Pr\left\{X_s < X_j \text{ for } j \neq s | X_s = t\right\} \lambda_s e^{-\lambda_s t} dt && (3.3)\\
&= \int_0^\infty \lambda_s e^{-\lambda_s t} \prod_{j\neq s} e^{-\lambda_j t} dt = \lambda_s \int_0^\infty e^{-(\sum_j \lambda_j)t} && (3.4)\\
&= \lambda_s/(\lambda_1 + \cdots + \lambda_n) && (3.5)
\end{aligned}$$

as desired. □

Since in each state, the label represents the number of devices failing with the same rate λ, the transition out of state n happens with rate $n\lambda$. Using the result of lemma 1, we can conclude that the holding time Y_n is exponentially distributed with rate $n\lambda$. Similarly, the holding time in state $n-1$, Y_{n-1} is exponentially distributed with rate $(n-1)\lambda+\mu$. The probability that the system transitions from state $n-1$ to state F (failed state) with probability $p_F = (n-1)\lambda/((n-1)\lambda+\mu)$.

Traditional storage system reliability analysis uses a metric called the mean time to data loss (MTTDL). MTTDL is the expected (average) time to enter state F. MTTDL is not a direct measure of reliability but is an average based on the reliability, given by $\int_0^\infty R(t)dt$. MTTDL might be very hard to compute for most of the stochastic models that describe the real life

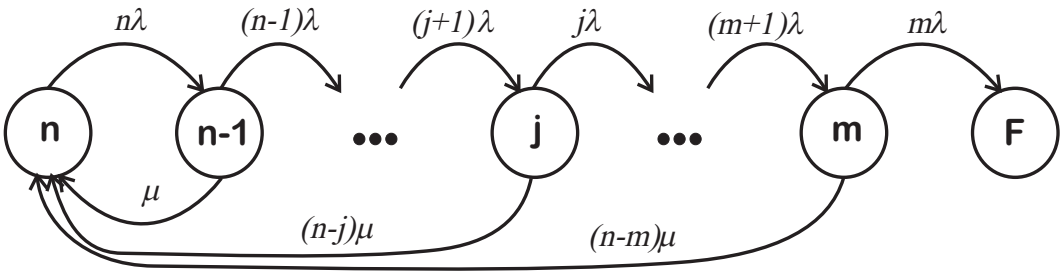

Figure 3.2: A generalized continuous time Markov model for the reliability estimation of (n, m)–coded data.

failure phenomenon. Fortunately, there are explicit mathematical methods to compute the MTTDL for simpler Markov models such as given in Fig. 3.1.

Let us assume we visit state $n-1$ repeatedly K many times before the whole system ends up in state F. Thus the total time before data loss is given by

$$Y_{DL} = \sum_{i=1}^{K} (Y_n^{(i)} + Y_{n-1}^{(i)}) \tag{3.6}$$

where $Y_n^{(i)}$ is the i-th holding time. By definition, MTTDL is $\mathbb{E}[Y_{DL}]$. Conditioning on $K = k$ and summing over all possibilities (unconditioning over the distribution of K – the distribution of K turns out to be Geometric) will yield

$$\begin{aligned}
\mathbb{E}[Y_{DL}] &= \sum_{k=1}^{\infty} \left(\sum_{i=1}^{k} (\mathbb{E}Y_n^{(i)} + \mathbb{E}Y_{n-1}^{(i)}) \right) p_F (1-p_F)^{k-1} && (3.7)\\
&= \sum_{k=1}^{\infty} \left(\frac{k}{n\lambda} + \frac{k}{(n-1)\lambda + \mu} \right) p_F (1-p_F)^{k-1} && (3.8)\\
&= \frac{2n\lambda + \mu - \lambda}{n\lambda((n-1)\lambda + \mu)} \mathbb{E}[K] = \frac{2n\lambda + \mu - \lambda}{n\lambda((n-1)\lambda + \mu)} \frac{(n-1)\lambda + \mu}{(n-1)\lambda} && (3.9)\\
&= \frac{\mu + \lambda(2n-1)}{\lambda^2 n(n-1)} && (3.10)
\end{aligned}$$

Note that probability distribution of Y_{DL} is not exponential yet its mean is easy to compute. In fact, the probability distribution of Y_{DL} is a sum of exponentials, but hard to compute it in a closed form. Later, we will see good approximations for Y_{DL} yield a sum of exponentials argument as well. Let us generalize the Markov model in Fig. 3.1 by letting the system correct c device failures with m data units or devices i.e., $n = m + c$. The previous parity-coded system has $c = 1$ and $m = n - 1$. Furthermore, let us assume that concurrent repairs are made in each state so that we have the Markov model shown in Fig. 3.2. Particularly for this type of generalization, there are Laplace transform-based techniques to compute the MTTDL.

The reliability function $R(t)$ is the probability of being in any state except the state F before time t. In otherwords,

$$R(t) = P_n(t) + P_{n-1}(t) + \cdots + P_{m+1}(t) + P_m(t) \tag{3.11}$$

where $P_j(t)$ is the probability of being in state j at time t. Furthermore, Laplace transform of the reliability function $R(t)$ for $t \geq 0$ is given by

$$R^*(s) = \int_0^\infty R(t)e^{-st}dt \tag{3.12}$$

From equation (3.12), we recognize that MTTDL is an aggregate measure of reliability and is given by the evaluation of transformed function at $s = 0$, i.e.,

$$\begin{aligned} MTTDL = R^*(0) = \int_0^\infty R(t)dt &= \int_0^\infty P_n(t)dt + \cdots + \int_0^\infty P_m(t)dt & (3.13)\\ &= P_n^*(0) + P_{n-1}^*(0) + \cdots + P_m^*(0) & (3.14) \end{aligned}$$

Using some probability theory background, let us write down the conventional Kolomogorov equations in time domain for this model as follows [5],

$$\begin{aligned} P_n'(t) + P_n(t).n\lambda &= P_{n-1}(t).\mu + P_{n-1}(t).2\mu + \cdots + P_{n-1}(t).(n-m)\mu \\ &\cdots \\ P_j'(t) + P_j(t).(j\lambda + (n-j)\mu) &= P_{j-1}(t).(j+1)\lambda \\ &\cdots \\ P_F'(t) &= P_m(t).m\lambda \end{aligned}$$

and the corresponding transform domain equations shall be,

$$\begin{aligned} sP_n^*(s) - P_n(0) + P_n^*(s).n\lambda &= P_{n-1}^*(s).\mu + P_{n-2}^*(s).2\mu + \cdots + P_m^*(s).(n-m)\mu \\ &\cdots \\ sP_j^*(s) - P_j(0) + P_j^*(s).(j\lambda + (n-j)\mu) &= P_{j-1}^*(s).(j+1)\lambda \\ &\cdots \\ sP_F^*(t) - P_F(0) &= P_m^*(s).m\lambda \end{aligned}$$

with the initial conditions $P_n(0) = 1$ and $P_j(0) = 0$ for $j < n$. These linear set of equations can be put in a matrix form easily as follows,

$$\begin{pmatrix} s+n\lambda & -\mu & -2\mu & \dots & -(n-m)\mu & 0 \\ -n\lambda & s+(n-1)\lambda+\mu & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & -m\lambda & s \end{pmatrix} \begin{pmatrix} P_n^*(s) \\ P_{n-1}^*(s) \\ \vdots \\ P_F^*(s) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and can be expressed in a matrix notation

$$\mathbf{AP} = \mathbf{N}^{(0)}$$

from which the inversion of the matrix ($\mathbf{A}^{-1}$) and multiplication from right yields the solution for $\mathbf{P}$ i.e., $P_j^*(s)$ for $j \in \{n, n-1, \dots, m, F\}$ where $\mathbf{N}^{(l)} = [0\ 0\ \dots 0\ 1\ 0 \dots\ 0\ 0]$ and $(l+1)$th vector entry is unity. Of course for large n, efficient methods to invert the matrix might be needed.

Once we find $P_j^*(s)$, using inverse Laplace transform, we can obtain $P_j(t)$ and eventually $R(t)$. For example in [6], MTTDL for $c=1$, $c=2$ and $c=3$ are calculated explicitly and they are given by

$$\begin{aligned}
MTTDL_{c=1} &= \frac{\mu+\lambda(2m+1)}{\lambda^2 m(m+1)} \\
MTTDL_{c=2} &= \frac{2\mu^2+\mu\lambda(5m+6)+\lambda^2(3m^2+6m+2)}{\lambda^3 m(m+1)(m+2)} \\
MTTDL_{c=3} &= \frac{6\mu^3+\mu^2\lambda(17m+33)+\mu\lambda^2(14m^2+47m+33)+2\lambda^3(2m^3+9m^2+11m+3)}{\lambda^4 m(m+1)(m+2)(m+3)}
\end{aligned}$$

Since we are interested in the first column of the matrix inverse, MATLAB/Mathematica symbolic inverse computation capability may be very useful to calculate explicit expressions for reasonable choices of c. As can be seen $MTTDL_{c=1}$ is the same as the equation in (3.10) we previously derived. We observe that for large c, MTTDL expressions get quite complicated. In conventional storage systems however, $\omega=\mu/\lambda$ ratio is very large. Therefore, simplifications to MTTDL can be performed using the assumption $\omega=\mu/\lambda \gg 1$. In [7], an approximate expression for the generic MTTDL is derived based on the assumption that ω ratio is very large. The approximate expression is quite neat and given by

$$MTTDL_c \approx \frac{\omega^c}{\lambda(n-c)\binom{n}{c}} \tag{3.15}$$

Unfortunately, even with this assumption the expressions for $R(t)$ for large values of c gets very complicated. In [6], good approximations for $R(t)$ are given for any m and $c=0,1,2$ and 3 as follows

$$\begin{aligned}
R_{c=0}(t) &= e^{-m\lambda t} && (3.16)\\
R_{c=1}(t) &\approx C_{1,m,0}e^{-t/MTTDL_{c=1}} + C_{1,m,-\mu}e^{-t(\mu+\lambda(2m+1))} && (3.17)\\
R_{c=2}(t) &\approx C_{2,m,0}e^{-t/MTTDL_{c=2}} + C_{2,m,-\mu}e^{-t(\mu+\lambda(2m+3))} + C_{2,m,-2\mu}e^{-t(2\mu+\lambda m)} && (3.18)\\
R_{c=3}(t) &\approx C_{3,m,0}e^{-t/MTTDL_{c=3}} + C_{3,m,-\mu}e^{-t(\mu+\lambda(2m+5))} \\
&\quad + C_{3,m,-2\mu}e^{-t(2\mu+\lambda(m+1))} + C_{3,m,-3\mu}e^{-t(3\mu+\lambda m)} && (3.19)
\end{aligned}$$

where the coefficients of the exponentials are

$$C_{c,m,0} = 1 + \sum_{i=1}^{c} \frac{1}{i} m \binom{m+c}{c} \omega^{-c-1} \tag{3.20}$$

and

$$\begin{aligned}
&C_{1,m,-\mu} = -\frac{\lambda^2 m(m+1)}{\mu^2}, C_{2,m,-\mu} = -\frac{\lambda^3 m(m+1)(m+2)}{\mu^3}, C_{3,m,-\mu} = -\frac{\lambda^4 m(m+1)(m+2)(m+3)}{2\mu^4}, \\
&C_{2,m,-2\mu} = -\frac{\lambda^3 m(m+1)(m+2)}{4\mu^3}, C_{3,m,-2\mu} = -\frac{\lambda^4 m(m+1)(m+2)(m+3)}{4\mu^4}, \\
&C_{3,m,-3\mu} = -\frac{\lambda^4 m(m+1)(m+2)(m+3)}{18\mu^4}.
\end{aligned}$$

Through a tedious algebra, these expressions can be generalized to any m and c, although the original work has not derived those expressions [6]. Exact closed form expressions for the general case is an open problem.

3.1 Durability and MTTDL

One of the important metrics related to the theory of reliability is *durability*. This metric is defined to be the duration of time the system is able to provide access to the user data or entity. If the data is protected by a redundancy mechanism i.e., replication or erasure correction coding, durability refers to the permanent loss of the encoded/replicated user data rather than the permanent loss of parity or replica blocks. The unit of durability is usually expressed in terms of days or years. In industry terminology however, durability is expressed in terms of nines (9's). This refers to the probability of seeing no error during the operation of the system for the first few number of years.

As mentioned before, the aggregate nature of the MTTDL reveals only very little information about the reliability/durability of the overall system. Exact calculations for the time to data loss distribution will yield the actual information about the reliability of the system, yet it may be cumbersome to compute it. We have seen that $R(t)$ is not exponentially distributed in general. However for simplicity let us assume it is distributed exponentially with the rate $1/MTTDL$. The number of 9's is then given by $\left\lfloor \log_{10}\left(\frac{1}{1-R(t)}\right)\right\rfloor$ where $R(t) = e^{-t/MTTDL}$ is the reliability/durability function and $t \geq 0$ is the time expressed in units of MTTDL (usually in hours). For example, for a system with MTTDL of $2,500,000$ hours, and an operating time of interest of 1 year (8760 hours), we have the durability number computed as follows,

$$R(8760) = e^{-8760/2500000} = 0.9965 \rightarrow \text{two 9's} \tag{3.21}$$

which means the system will operate without a failure with probability 0.9965 for the first year of use at a 100% duty cycle. Next, this simplistic view will be compared to the approximations given in the previous section for a replication redundancy scheme. We shall see later in the document that the exponential approximation is pretty good for durability computations.

3.2 A case study: Replication v.s. Erasure Correction

Data storage systems use different redundancy schemes to prevent data loss that can occur because of multiple concurrent[2] device failures. Replication is one of the widely used schemes where each data block is replicated and the replicas are stored in different nodes to improve the chances that at least one replica survives when multiple storage nodes fail. In a replication scheme, $m = 1$ and $c = 1,2,\ldots$. Table 3.1 illustrates that for large ω, approximations given for $MTTDL$ and the time to failure distribution being exponential are quite accurate, particularly when the quantities are expressed in terms of nines.

Let us increase the number of data devices from $m = 1$ to $m = 100$ and recalculate these values for $c = 1,2,3$. The results are shown in Table 3.2. As can be seen, approximations are still mostly accurate, if not, they are an underestimator.

[2]Here the failures need not to fail at exactly the same point in time. We rather refer to the extra failures as concurrent failures before the repair of the failed devices is completed.

Fail/Repair rates		$c=1$		$c=2$		$c=3$	
λ	μ	$e^{-t/MTTDL_1}$	$R_{c=1}(t)$	$e^{-t/MTTDL_2}$	$R_{c=2}(t)$	$e^{-t/MTTDL_3}$	$R_{c=3}(t)$
$1/200K$	$1/24$	4	4	8	8	12	12
$1/500K$	$1/24$	5	5	9	9	14	14
$1/1.2M$	$1/24$	6	6	11	11	15	15
$1/200K$	$1/240$	3	3	6	6	9	9
$1/500K$	$1/240$	4	4	7	7	11	11
$1/1.2M$	$1/240$	5	5	9	9	12	12

Table 3.1: Number of Nines for 3 and 4-replicated systems per archive.

Fail/Repair rates		$c=1$		$c=2$		$c=3$	
λ	μ	$e^{-t/MTTDL_1}$	$R_{c=1}(t)$	$e^{-t/MTTDL_2}$	$R_{c=2}(t)$	$e^{-t/MTTDL_3}$	$R_{c=3}(t)$
$1/200K$	$1/24$	1	1	3	3	5	5
$1/500K$	$1/24$	2	2	4	4	7	7
$1/1.2M$	$1/24$	2	2	5	5	8	8
$1/200K$	$1/240$	0	0	1	1	2	3
$1/500K$	$1/240$	1	1	2	2	4	4
$1/1.2M$	$1/240$	1	1	3	3	5	6

Table 3.2: Number of Nines for Erasure-coded systems per archive.

3.3 Discussion on the accuracy of MTTDL

Although MTTDL is a useful tool for making relative comparisons, it is meaningless measure for the absolute measurements [8]. In addition, given the unrealistic assumptions used to derive closed-form expressions, it became a subject of question recently. A system designer may be interested in the probability of failure for the first few years instead of the mean time to failure. The aggregate nature of the MTTDL usually conveys very limited information to the customer. In fact, $R(t)$, also known as durability, is what most of the customers are interested in.

Traditional reliability models were constructed with four simplifying assumptions: the only failures are whole-device failures, the use of single-disk fault-tolerant codes such as parity coding, devices fail at a constant rate, devices are repaired at a constant rate. While traditional Markov models enable quick analytic reliability estimates of single-disk fault-tolerant systems, they do not extend well to multi-disk fault-tolerant systems, the inclusion of sector failures and they do not compensate for time dependence in failure and repair.

There are several problems with these set of assumptions of a simple Markov model. For example, many models do not account for sector failures, do not accurately model device rebuild processes and assume that all devices exhibit the same failure/rebuild rates. In addition, extension to a general model that accounts for m-erasure correction is not straightforward. Canonical Markov models have memoryless interarrival times, yet most of the time, the storage components already in the rebuilt process are assumed to start repair from the beginning if another storage component fails. These considerations led research community to think of more advanced modeling techniques to estimate the reliability more accurately.

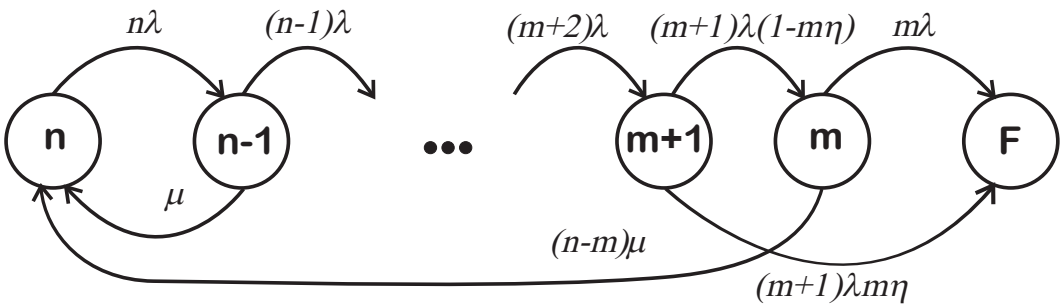

Figure 3.3: A continuous time Markov model for the reliability estimation of (n,m)–coded data incorporates the hard errors in $MTTDL$ estimation.

Most of them emphasize the hardness of deriving analytical results and encourage Monte Carlo type simulations for accurate estimations [10].

3.4 Arguments in favor of MTTDL and Advanced models

Although the inaccuracies of the previous calculations of MTTDL are plenty, this notorious reliability metric, based on exponential failure and repair times, has been shown to be insensitive to the actual distribution of failure/repair times as long as the constituent storage devices have MTTF being much larger than their MTTR [9] and operate independent of eachother. This result essentially implies that the system MTTDL computations of previous sections will not be affected if the device failure distributions were changed from an exponential to a some other distribution with the same mean. Given that, there have been few attempts to make the original Markov model more realistic. We shall review them in this section and propose the most general form of the canonical model for reliability estimations.

3.4.1 Incorporating a hard error (uncorrectable error)

This requirement is observed to be necessary when the system operates in the critical mode i.e., a state in which one more device failure leads to total system crash and/or data loss. This requirement imposes a slight change in our Markov model as shown in Fig. 3.3. In this model, η represents the probability of seeing an uncorrectable error per device read during the rebuilt process. Let `UCER` denote the uncorrectable error rate of the device (such as 10^{-15}, expressed in terms of errors per number of bytes or bits read), η is given by $\texttt{device capacity} \times \texttt{UCER}$. Although this is the published expression [11], it comprises an inaccuracy that $\texttt{device capacity} \times \texttt{UCER}$ can be greater than 1. However, η appears to be a probability term and must satisfy $0 \leq \eta \leq 1$. Thus, a more accurate expression can be given as $\eta = 1-(1-\texttt{UCER})^{\texttt{device capacity}}$.

The transition from state $m+1$ to state F is introduced to model the rate at which the system encounters an uncorrectable error while reading and rebuilding t device failures. Based on the analysis given in [11], the rate is computed as the product of the rate that a disk fails when $(m+1)$ devices are available, i.e., $(m+1)\lambda$ and the probability of encountering an uncorrectable error when reading m devices for rebuild (Note here that we assume conventional

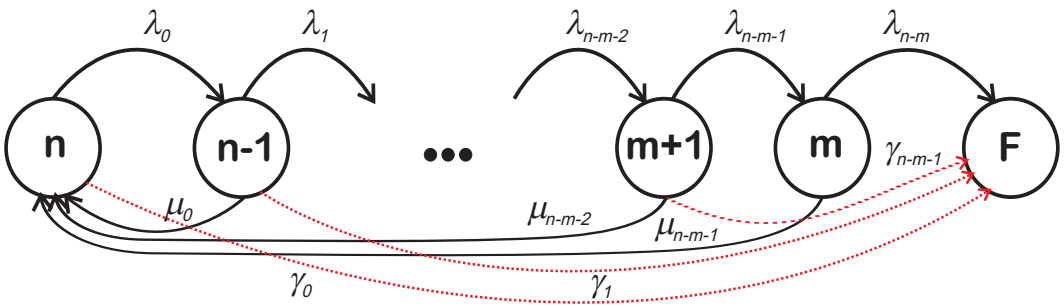

Figure 3.4: A generalized continuous time Markov model that can be used for the reliability estimation of (n,m)–coded data that incorporates the irregular fault tolerance as well as the hard errors in $MTTDL$ estimation.

MDS codes, which may require many device reads), i.e., $P_{UCER} = m\eta$. Similar to our previous argument, this product can assume values greater than one. A more accurate expression shall be given by

$$P_{UCER} = 1 - (1-\eta)^m \tag{3.22}$$

An approximate solution for the MTTDL (without the corrections given for η and P_{UCER} i.e., $\eta = \texttt{device capacity} \times \texttt{UCER}$ and $P_{UCER} = m\eta$) is given in [11] for one-device-at-a-time repair strategy. Using similar arguments we can easily extend those results to concurrent repairs and obtain the following expression for $\eta \ll n$ and $\omega = \mu/\lambda \gg 1$,

$$MTTDL_c \approx \frac{c!w^c}{n(n-1)\ldots(n-c)(\lambda+\eta\mu)} \tag{3.23}$$

3.4.2 Modifications for Irregular Codes Or multiple regular codes

In addition to previous improvements, let us generalize the model to cover the class of more popular *irregular* erasure correction codes. By “irregular", we mean the modern non-MDS erasure correction codes that are constructed based on bipartite graphs and has irregular fault tolerance. Most prominent ones of this class are XOR-based LDPC or fountain codes that possess good reparability properties with simple encoding and decoding operations.

Let us introduce one of the most general Markov model and its application to reliability analysis of systems which are protected by irregular codes. Note that few generalizations are made in [11], but not to that level shown in Fig. 3.4. Furthermore, some special cases of this model is considered and analyzed such as in [12]. In this model, device failures happen one at a time with rate λ_i while due to irregular fault tolerance of the erasure correction code, it is possible to go from any state except state m to fail state with some rate γ_i. Also, as mentioned in the past research that Markov models assume rebuild clock is ignored. In other words, each Markov model resets the rebuild time for all disks being rebuilt whenever another disk fails during rebuild. The model of concurrent rebuilds considered in this document adopts a rebuild policy that restarts the rebuild of all failed disks each time a disk fails. Alternatively,

one could estimate the remaining repair time at each state by steadily increasing the rate of ongoing rebuild operations as more failures occur. Thus, in a general model we should allow different rebuild rates μ_i to meet this goal of better modeling the real world.

With regard to this general model, there are three questions to be explored,

- Given the nature of the irregular code and its fault tolerance, how do we efficiently compute rates $\{\lambda_i, \gamma_i\}$ to be able to use with this model?
- Given the nature of rebuilds and associated system format/structure, how do we estimate rates $\{\mu_i\}$ to be able to use with this model?
- Is there a closed form expression for MTTDL? and $R(t)$? for any c and m.

Let us start with the first question. This question is in fact answered in [11]. Given the irregular code and the symbol allocation policy (such as CRUSH [13]), let us denote the probability p_k that the overall system can tolerate one more device failure provided that it has already tolerated k failed devices. There are two scenarios that can lead from an operational (survival) state to failure (absorbing) state. First of all, at the j-th state ($n \geq j \geq m$) one more device fails and the correction code cannot tolerate for this loss and the state change ends up with the failure state. The rate of this happening is $j\lambda(1-p_{n-j})$, direct multiplication follows from basic Markov models in probability theory. Another scenario that ends up in data loss is that an extra device fails, the erasure code is able to tolerate. Yet, during the rebuild process a hard error (uncorrectable error) is encountered and the erasure code cannot tolerate this additional error. The rate of this happening is given by $j\lambda p_{n-j}(1-p_{n-j+1})(j-1)\eta$ which can be obtained using conditional probability arguments. Ofcourse, here we assume all the operational $j-1$ devices are used/accessed in the rebuilt process which may not necessarily be the case for regenerating erasure codes [17]. Thus, the expression can be modified based on the properties of the erasure code used. Finally since $\lambda_{n-j} + \gamma_{n-j} = j\lambda$, for $n \geq j \geq m+1$ we have the following

$$\gamma_{n-j} = j\lambda\left((1-p_{n-j}) + p_{n-j}(1-p_{n-j+1})(j-1)\eta\right) \tag{3.24}$$

$$\lambda_{n-j} = j\lambda - \gamma_{n-j} \tag{3.25}$$

with $\lambda_{n-m} = (n-m)\lambda$ and $\gamma_{n-m} = 0$.

Here, we still need to explain how to obtain conditional probabilities p_ks. Let us denote the unconditional probability q_k that the overall system can tolerate k device failures out of $\binom{n}{k}$ total number of possible instances. Suppose that s_k of these possibilities can be tolerated by the system, then we have $q_k = s_k \Big/ \binom{n}{k}$. Suppose that a $k+1$ pattern of failures the system can tolerate, than any failure subset of size k of these $k+1$ failures can be tolerated. This means that $p_k = q_{k+1}/q_k$, or more explicitly,

$$p_k = \frac{s_{k+1}}{\binom{n}{k+1}} \Big/ \frac{s_k}{\binom{n}{k}} = \frac{s_{k+1}(k+1)}{s_k(n-k)} \tag{3.26}$$

where s_ks are usually found through an elaborate test of the overall system (erasure code performance/properties plus the symbol allocation strategies) protected by the irregular erasure

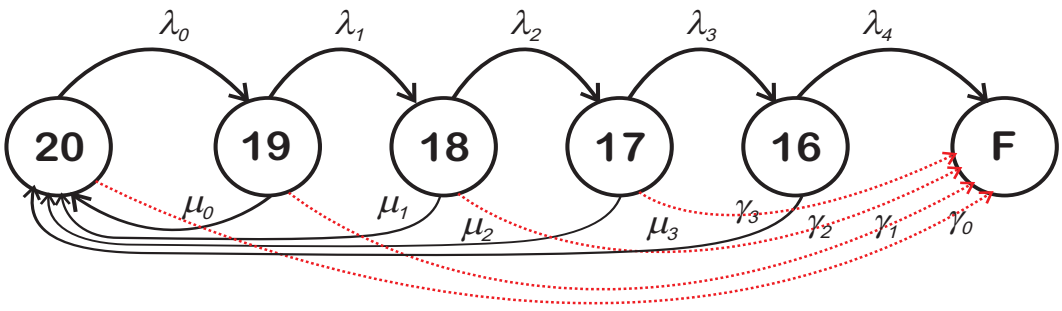

Figure 3.5: A generalized Markov model used for the reliability estimation of two (10,8) RAID 6 arrays.

code. Independent of the allocation policy, studies like [14] investigate fault tolerance metrics such as minimum erasure patterns for any linear block code. The results of such studies can be useful for the computation of durability numbers of erasure-coded storage systems, as we shall see shortly.

For the second question, we can simply assume $\mu_i = \mu$ (namely a homogenous repair process) where the rebuild time for $n-i$ devices is the same as the rebuild times for a single derive. Ofcourse more elaborate approaches will increase the accuracy in the final durability results. In another system, concurrent failure repair may imply $\mu_i = (i+1)\mu$ (namely progressive repair) because each device is repaired with the same rate μ. Thus, the choice for repair rates are a strong function of the overall system implementation.

As for the third question, we can use Mathematica or Matlab symbolic toolbox to calculate closed form expressions for small values of m and c. The derivation of a closed form expression for any m and c can be significant for two main reasons. First, we can model the durability modeling for long block length irregular codes (whose block sizes are practically designed to be long compared to that of conventional algebraic codes). Secondly, we can model multiple (in fact a vast array of) short length MDS-protected RAID type systems.

Let us give an example for the latter: consider two 2 out-of 10 ($m = 8$, $c = 2$) RAID6 systems that are used to store user data with $n = 20$. We notice that

- Every zero-device failure can be tolerated ($q_0 = 1$).
- Every one-device failure (each $\binom{20}{1} = 20$ possibilities) can be tolerated ($q_1 = 1$).
- Every two-device failures (each $\binom{20}{2} = 190$ possibilities) can be tolerated ($q_2 = 1$).
- Only 900 three-device failures out-of $1140 = \binom{20}{3}$ can be tolerated ($q_3 = 0.7895$).
- Only 2025 three-device failures out-of $4845 = \binom{20}{4}$ can be tolerated ($q_4 = 0.4180$).
- No device failure larger than four can be tolerated ($q_k = 0$ for $k > 4$).

from which we deduce $\{p_0, p_1, p_2, p_3, p_4\} = \{1, 1, 0.7895, 0.5294, 0\}$. The Markov model shown in Fig. 3.6 is used to estimate the reliability with the following set of parameters,

Fail/Repair rates		General Markov (HR)		General Markov (PR)		Conventional Method (PR)	
λ	μ	MTTDL	Durability 9's	MTTDL	Durability 9's	MTTDL	Durability 9's
$1/200K$	$1/24$	1.035×10^9	5	1.1×10^9	5	1.93×10^{10}	6
$1/500K$	$1/24$	6.9×10^9	5	7.1×10^9	5	3.01×10^{11}	7
$1/1.2M$	$1/24$	4.1×10^{10}	6	4.13×10^{10}	6	4.17×10^{12}	8

Table 3.3: Number of nines for two RAID6 systems with $m = 8$ and $c = 2$. HR: Homogenous Repair i.e., $\mu_i = \mu$, PR: Progressive Repair i.e., $\mu_i = (i+1)\mu$.

- $\gamma_0 = n\lambda\left((1-p_0) + p_0(1-p_1)(n-1)\eta\right) = 0$,
  $\lambda_0 = n\lambda - \gamma_0 = n\lambda = 20\lambda$.
- $\gamma_1 = (n-1)\lambda\left((1-p_1) + p_1(1-p_2)(n-2)\eta\right) = 0.2105 \times (n-1)(n-2)\lambda\eta = 72\lambda\eta$,
  $\lambda_1 = (n-1)\lambda - \gamma_1 = 19\lambda - 72\lambda\eta$.
- $\gamma_2 = (n-2)\lambda\left((1-p_2) + p_2(1-p_3)(n-3)\eta\right) = 3.79\lambda - 33.4\lambda\eta$,
  $\lambda_2 = (n-2)\lambda - \gamma_2 = 18\lambda - 3.79\lambda + 33.4\lambda\eta = 14.21\lambda + 33.4\lambda\eta$.
- $\gamma_3 = (n-3)\lambda\left((1-p_3) + p_3(1-p_4)(n-4)\eta\right) = 8\lambda - 144\lambda\eta$
  $\lambda_3 = (n-3)\lambda - \gamma_3 = 17\lambda - 8\lambda + 144\lambda\eta = 9\lambda + 144\lambda\eta$.
- $\gamma_4 = 0$
  $\lambda_4 = (n-4)\lambda = 16\lambda$.

Let us further assume we use disks of size $1TB$ as our storage devices with 10^{-15} uncorrectable error rate i.e., $\eta = 10^{-3}$. The results for MTTDL as well as durability in terms of $9's$ are obtained using symbolic computations through MATLAB and are evaluated/presented in Table 3.3. Closed form expressions for this general case, m and c to allow efficient computation will be given in the next section.

As can be seen changing the repair strategy has only minor effect on the MTTDL results. Also included are the set of results using the basic model for each RAID6 array and compute the MTTDL using Equation (3.15). It assumes exponential data loss probability distribution and thus computes the MTTDL of the two RAID6 arrays based on a simple algebraic addition of data loss rates. This conventional scheme does not take into account the uncorrectable errors as well. As expected, the results with conventional scheme show more optimistic MTTDL and durability numbers.

For an arbitrary number of MDS-protected arrays of devices, it turns out that there is a closed form expression for s_k. In other words, let us assume we have π arrays each having n devices c of which are redundant for failure recovery. The total number of choosing k failed devices out of πn is given by $\binom{\pi n}{k}$. Let us assume further that there are r_j number of arrays with j device failures with $0 \le j \le c$. The general expression for s_k can be given by [11]

$$s_k = \sum_{\substack{r_0+r_1+\cdots+r_c=\pi \\ \sum_{i=0}^{c} i r_i = k}} \binom{\pi}{r_0, r_1, \ldots, r_c} \prod_{i=0}^{c} \binom{n}{i}^{r_i} \tag{3.27}$$

This expression follows because

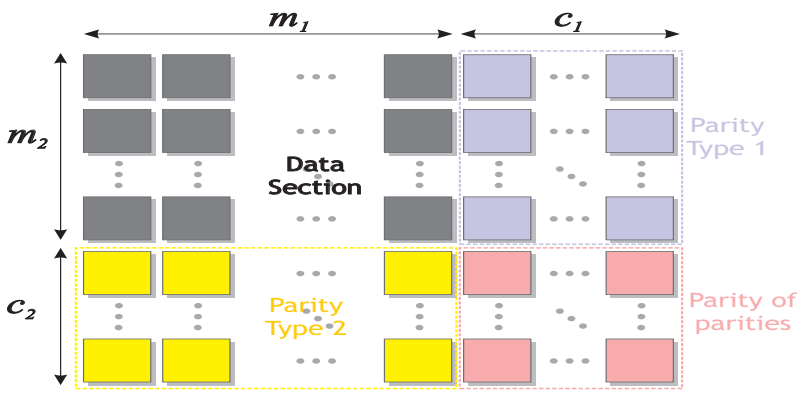

Figure 3.6: A two-dimensional RAID scheme where $m_1 m_2$ data blocks are protected by a total of $c_1 k_2 + c_2 k_1 + c_1 c_2$ parity blocks.

- the multinomial coefficient counts the number of combinations we can distribute i failures over r_i arrays etc.
- Given that r_i arrays with i failures, there are $\binom{n}{i}^{r_i}$ ways for distributing those failures,

with the constraint that total number of arrays must be π and failures be k. Multinomial theorem is a generalization of the binomial theorem. This generalized version implies that we have

$$(x_0 + x_1 + \cdots + x_c)^{\pi} = \sum_{r_0 + r_1 + \cdots + r_c = \pi} \binom{\pi}{r_0, r_1, \ldots, r_c} \prod_{i=0}^{c} x_i^{r_i} \tag{3.28}$$

If we replace x_i with $\binom{n}{i} x^i$, we shall have

$$\left(1 + \binom{n}{1} x + \binom{n}{2} x^2 + \cdots + \binom{n}{c} x^c\right)^{\pi} = \sum_{r_0 + r_1 + \cdots + r_c = \pi} \binom{\pi}{r_0, r_1, \ldots, r_c} \prod_{i=0}^{c} \binom{n}{i}^{r_i} x^{\sum_{i=0}^{c} i r_i} \tag{3.29}$$

from which we notice that if we set $\sum_{i=0}^{c} i r_i = k$, we can realize that s_k is the coefficient in this expansion for x^k, i.e.,

$$s_k = coef\left(\left(\sum_{i=0}^{c} \binom{n}{i} x^i\right)^{\pi}, x^k\right) \tag{3.30}$$

Therefore, we have

$$p_k = \frac{coef\left(\left(\sum_{i=0}^{c} \binom{n}{i} x^i\right)^{\pi}, x^{k+1}\right)(k+1)}{coef\left(\left(\sum_{i=0}^{c} \binom{n}{i} x^i\right)^{\pi}, x^k\right)(\pi n - k)} \tag{3.31}$$

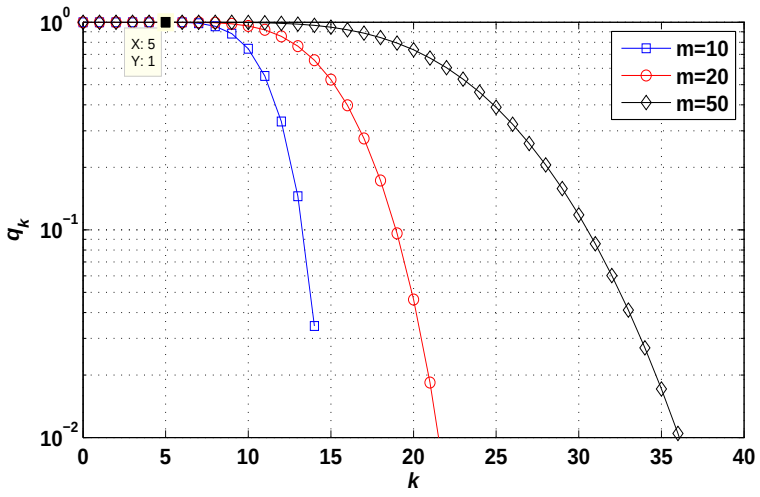

Figure 3.7: q_k values as a function of k and m for a RAID61 configuration.

3.4.3 2D STORAGE ARRAYS USING MDS CODES

Another interesting case is the use of MDS codes in the context of two dimensional storage arrays. In the past, particular class of two-dimensional RAID schemes are shown to provide better reliability against single dimensional counterparts [15]. However, no general treatment has been conducted. There are recent studies like [16], they only consider the mean time to the first failure of multidimensional RAID type storage systems using certain assumptions like independence. Here our consideration is more general for two-dimensional arrays and aims at finding reliability estimates about MTTDL using the general Markov model. For this, we need to determine s_ks and error tolerability of two dimensional storage arrays. We assume that each horizontal and vertical array component are protected by an MDS code with c_1 and c_2 parity blocks, respectively. A simple bounded distance decoder is assumed. Decoding can start either in horizontal or vertical direction wherever appropriate. For notation simplicity let $n_1 = m_1 + c_1$ and $n_2 = m_2 + c_2$. We have the following conjecture for s_k if $k \leq (c_1+1)(c_2+1)+\min\{c_1,c_2\}$

$$s_k = \begin{cases} \binom{n_1 n_2}{k} & \text{if } k \leq c_1 c_2 + c_1 + c_2 \\ \binom{n_1 n_2}{k} - \binom{n_1}{c_1+1}\binom{n_2}{c_2+1}\binom{n_1 n_2-(c_1+1)(c_2+1)}{k-(c_1+1)(c_2+1)} & \text{if } (c_1+1)(c_2+1) \leq k \leq (c_1+1)(c_2+1)+\min\{c_1,c_2\} \end{cases}$$

The exact expressions for s_k for $k > (c_1+1)(c_2+1)+\min\{c_1,c_2\}$ gets more complicated and the closed form expressions for any k is an open problem. Let us consider an example that is a special case of the 2D storage arrays using MDS codes.

Example 1: *A mirrored (n_1, m_1)-coded array ($c_1 = n_1 - m_1$ parities),* is a special case of a 2D storage array where the vertical encoding is nothing but a single parity generation though information duplication. For example RAID 51 and RAID 61 systems are special cases of this example. In order for a column to fail, both of the copies must be unavailable/failed. For a given k device failures in the 2D array, there are at most $\lfloor k/2 \rfloor$ column failures. Since the

horizontal array can tolerate c_1 device failures at most, maximum number of column failures can not exceed $\min\{\lfloor k/2 \rfloor, c_1\}$ in order for the overall system be able to recover the data. For a given $0 \leq j \leq \min\{\lfloor k/2 \rfloor, c_1\}$ column failures, we realize that equation (3.30) can be used with $k = k-2j$, $n = 2$ and $r = n_1$ to calculate number of tolerable cases for each j. Note also that each case constitute a disjoint set. Since we have $\binom{n_1}{j}$ possible selections of these columns, we have

$$s_k = \sum_{j=0}^{\min\{\lfloor k/2 \rfloor, c_1\}} \binom{n_1}{j} coef\left((1+2x)^{n_1}, x^{k-2j}\right) \mathbf{1}_{y \leq n_1 + j}(k) \tag{3.32}$$

where $\mathbf{1}_{y \leq A}(y)$ is the indicator function that evaluates to 1 if $y \leq A$. This function is needed in our expression because for each term in the summation, we assume there are j column failures and if $k-2j > n_1 - j$ or $k > n_1 + j$, it would mean that there are at least $j+1$ column failures, which is contrary to our conditional statement that we have j column failures.

In Fig. 3.7, the computation for q_k is shown for RAID61 ($c_1 = 2$) configuration with $m_1 = 10, 20, 50$ and $n_2 = 2$ as a function of k. Notice that for each case if $k \leq c_1 c_2 + c_1 + c_2 = 5$, all combinations of k failed devices are tolerable. It might be interesting for example to drive a similar expression for $n_2 = 3$ while we keep $c_2 = 1$ by using parity coding. Even in this simple extension, the expression can get pretty complex. We also note that the precise definition of the decoding algorithm across all devices also play a key role for the calculation of s_k for this particular configuration. For instance, vertical and horizontal decoding iterations can be allowed until the decoding can no longer improve the failure statistics.

Exercise 1: Using the result of the above example, compute the MTTDL estimates of RAID51, RAID6 and RAID61 for comparison. Use $\lambda = 1/200K$ and $\mu = 1/24$ in your computations.

Using this result, a generalization to any n_2 with $m_2 = 1$ and $c_2 = n_2 - 1$ can easily be made. The result of this generalization gives a closed form expression as indicated in the following lemma.

Lemma 2: *For a 2D array of storage devices with only one constraint that $m_2 = 1$, the closed form expression for s_k is of the form*

$$s_k = \sum_{j=0}^{\min\{\lfloor k/n_2 \rfloor, c_1\}} \binom{n_1}{j} coef\left(\left((1+x)^{n_2} - x^{n_2}\right)^{n_1}, x^{k-n_2 j}\right) \mathbf{1}_{y \leq n_1 + c_2 j}(k) \tag{3.33}$$

PROOF: Proof of this lemma is pretty straightforward using the previous arguments of the example where $c_2 = 1$ and $n_2 = 2$. □

The closed form expressions for s_k without the constraint of lemma 2 becomes more complicated. This is in fact the same interesting combinatorial problem, mentioned earlier. Although many studies have been conducted regarding ball and bins argument in the past to be used to calculate such quantities, with the specific decoding principle we assume in this framework (iterative both in horizontal and vertical directions), there appears to be no solution for the general case published yet.

3.4.4 XOR-BASED ERASURE CODES

For an arbitrary erasure code – possibly non-MDS protected array of devices, the computation of s_k is challenging and is a strong function of the code's graphical construction. Even without the consideration of coded symbol allocation methodology, identification of code fault tolerance might be quite complex to compute. For example a methodology is presented for a linear code (an XOR-based code) constructed using a systematic binary generator matrix in [18] used for erasure correction. The same algorithm turns out to be applicable to non-systematic codes as well. The claims of that paper are based on the decodability of the code, i.e., the study assumes an optimal decoder. In fact, such a fault tolerance profile is a function of the encoding and decoding algorithms that the code is used with.

For a complete characterization of the fault tolerance of an XOR-based erasure code, every set of erasure patterns must be enumerated. Unfortunately, there are exponentially many such erasure patterns to enumerate. Instead of finding all erasure patterns that lead to decoder failure, *minimal erasures* are found to characterize the fault tolerance of the erasure code. A minimal erasure is a set of erasures that leads to irrecoverable data loss and in which every erasure is both necessary and sufficient for this to be so. Furthermore, an enumeration of all minimal erasures are termed as *minimal erasures list* (MEL). The MEL characterizes completely the fault tolerance of an erasure code while having relatively small size relative to all erasure patterns causing a data loss. An algorithm[3] is presented in [18] for efficiently determining the MEL of a linear XOR-based erasure code defined by a binary generator matrix. Given the MEL of an (n,k) linear XOR-based code, the minimum erasure vector (MEV) is of size $n-k$ in which ith entry counts the number of minimum erasure patterns of weight i for $1 \leq i \leq n-k$. It is easy to realize that MEV can be instrumental for the computation of s_k.

Example 2: Let us consider the following binary generator matrix for a $(8,4)$ systematic XOR-based linear code,

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Using the ME algorithm, we can compute the MEV = $\{0,0,4,5\}$. From this, we can see that $s_3 = \binom{8}{3} - 4 = 52$ and through simple enumeration $s_4 = \binom{8}{4} - 25 = 45$ and $s_k = 0$ for all $k \geq 5$.

4 MTTDL FOR THE GENERALIZED MARKOV MODEL

In this section, we will give the exact as well as approximate expressions for the MTTDL of the most general Markov model introduced in the previous section. As we mentioned before, for large values of c, m and large volumes of storage arrays make the conventional computations useless. Closed form expressions are golden in that sense to predict future reliability estimates for long and large size of storage arrays as well as gigantic arrays of data coded by irregular modern coding techniques.

[3] A Matlab implementation of this algorithm can be found at http://www.suaybarslan.com/mea.txt

4.1 Efficient Computation of MTTDL

Following our previous construction, we have the transform domain Kolomogorov equations put into a coefficient matrix $\mathbf{A}$ as shown below.

$$\mathbf{A} = \begin{bmatrix} s+\lambda_0+\gamma_0 & -\mu_0 & -\mu_1 & \dots & -\mu_{c-1} & 0 \\ -\lambda_0 & s+\lambda_1+\mu_0+\gamma_1 & 0 & \dots & 0 & 0 \\ 0 & -\lambda_1 & s+\lambda_2+\mu_1+\gamma_2 & \dots & 0 & 0 \\ 0 & 0 & -\lambda_2 & \dots & 0 & 0 \\ \vdots & \vdots & & \ddots & 0 & 0 \\ 0 & 0 & \dots & s+\lambda_{c-1}+\mu_{c-2}+\gamma_{c-1} & 0 & 0 \\ 0 & 0 & \dots & -\lambda_{c-1} & s+\lambda_c+\mu_{c-1} & 0 \\ -\gamma_0 & -\gamma_1 & \dots & -\gamma_{c-1} & -\lambda_c & s \end{bmatrix}$$

As can be seen, the inversion of $\mathbf{A}$ might be quite challenging using symbolic software toolboxes if the size of the matrix is large. Instead, neat closed form expressions are prefered using the special structure of the coefficient matrix. Let us start by stating the key result of this section without proof. For a given number of redundant c units/devices and $x = 0, 1, \dots, c$, let us define the following *key rate vector* with the corresponding entries,

$$\mathbf{\Lambda}_x^{(c)} \triangleq \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{c-1} \\ \lambda_c \end{bmatrix} + \mathbf{V}_x^{(c)} \begin{bmatrix} \gamma_0 \\ \mu_0+\gamma_1 \\ \mu_1+\gamma_2 \\ \vdots \\ \mu_{c-2}+\gamma_{c-1} \\ \mu_{c-1} \end{bmatrix}$$

where

$$\mathbf{V}_x^{(c)} = \begin{bmatrix} \mathbf{0}_x & \mathbf{0}_{x\times c+1-x} \\ \mathbf{0}_{c+1-x\times x} & \mathbf{I}_{c+1-x} \end{bmatrix}$$

and $\mathbf{I}_{c+1-x}$ and $\mathbf{0}_x$ are identity and all-zero matrices, respectively. Let us assume $\Lambda_x^c(j)$ to denote the $(j+1)$-th entry of the vector $\mathbf{\Lambda}_x^{(c)}$ for $0 \le j \le c$. Then, we have the following transform domain expressions evaluated at $s = 0$,

$$P_{m+c-x}^*(0) = \frac{1}{\phi_c(0)} \prod_{\substack{j=0 \\ j\neq x}}^{c} \Lambda_x^c(j) \tag{4.1}$$

where the denominator can be computed recursively by

$$\phi_t(0) = \prod_{i=0}^{t} \lambda_i + (\mu_{t-1}+\lambda_t)\left[\phi_{t-1}(0) - \prod_{i=0}^{t-1}\lambda_i + \gamma_{t-1}\left(\prod_{i=0}^{t-2}(\gamma_i+\lambda_i)+\xi_t\right)\right] \tag{4.2}$$

for $1 \le t \le c$ with the initial condition $\phi_0(0) = \lambda_0$. Here, we can show that $\xi_1 = \xi_2 = 0$ and $\xi_3 = \gamma_0\mu_0$. The exact expressions for $\xi_{\{t>3\}}$ get more complicated for large t. However, we

realize that ξ_ss are relatively small compared to the rest of the expression given for $\phi_t(0)$. Thus, a good approximation is

$$\phi_t(0) \approx \prod_{i=0}^{t} \lambda_i + (\mu_{t-1} + \lambda_t) \left[\phi_{t-1}(0) - \prod_{i=0}^{t-1} \lambda_i + \gamma_{t-1} \left(\prod_{i=0}^{t-2} (\gamma_i + \lambda_i) \right) \right] \tag{4.3}$$

Using the above arguments, we can find a closed form expression for $MTTDL_c$ expressed as follows,

$$MTTDL_c = \sum_{x=0}^{c} P^*_{m+c-x}(0) = \frac{1}{\phi_c(0)} \sum_{x=0}^{c} \prod_{\substack{j=0 \\ j \neq x}}^{c} \Lambda^c_x(j) \tag{4.4}$$

The equation (4.4) will become approximate (in fact an overestimator) if we use equation (4.3) to approximate $\phi_c(0)$. For example, we executed the actual computation and the approximation for $\lambda = 1/200K$, $\mu = 1/24$, $m = 8$, $\eta = 10^{-3}$, $c = \{1,2,3,4,5\}$ and $r = \{1,2,3\}$ the error due to approximation is observed to be less than 10^{-6}. We project to find tight bounds on the evolution of ξ_t and predict the maximum error that can be inserted into our computations. For a special case, ξ_t can be expressed in a closed form as stated in the following lemma.

Lemma 3: *For $t \geq 4$ and $x \leq c-2$, if multiple of MDS protected arrays or a XOR-based erasure coded storage array can tolerate any x or less device failures, we have $\gamma_0 = 0, \ldots, \gamma_{x-2} = 0$ and therefore*

$$\xi_t \begin{cases} = \gamma_{t-3}\mu_{t-3} \prod_{i=0}^{t-4} \lambda_i & \text{if } t \leq x+2 \\ > \gamma_{t-3}\mu_{t-3} \prod_{i=0}^{t-4} \lambda_i & \text{Otherwise.} \end{cases} \tag{4.5}$$

PROOF: Proof of this lemma will be provided with when we establish the the evolution of ξ_t as a function of failure and repair rates of the underlying Markovian process. □

Note that this result implies that if evey combination of $c-2$ or more device failures are tolerable given the coding algorithm and allocation policy, ξ_t has a simple closed form and thus exact expression can be calculated for $\phi_c(0)$ given by the recursive relation for $t = 1, \ldots, c$

$$\phi_t(0) = \prod_{i=0}^{t} \lambda_i + (\mu_{t-1} + \lambda_t) \left[\phi_{t-1}(0) - \prod_{i=0}^{t-1} \lambda_i + \gamma_{t-1} \left(\prod_{i=0}^{t-2} (\gamma_i + \lambda_i) + \gamma_{t-3}\mu_{t-3} \prod_{i=0}^{t-4} \lambda_i \right) \right] \tag{4.6}$$

and the MTTDL expression become exact. Otherwise, our computations shall generate an upper bound for the actual MTTDL number (see why this is?).

Beauty of these type of closed form expressions is that, one can deduce the reliability of large set of erasure coded arrays. For example, we can easily compute that 125 of RAID6 arrays with $m = 8$ data devices (A total of $1PB$ user data stored on 1250 $1TB$ disk devices) has a durability number of 3. This shows that parallel functioning conventional RAID schemes for data protection is quite unreliable as the scale of the stored data expands to PB ranges. If we reduce $m = 8$ to $m = 7$, allowing a total of $875TB$ user data storage in the same storage system increases the durability number to 6.

Number of failed symbols/blocks		0	1	2	3	4	5	6
Generic MDS Code	Recoverability (%)	100	100	100	100	100	100	100
	Avg. read overhead	1.0	1.61	2.22	2.83	3.44	4.06	4.67
Basic Pyramid Code (BPC)	Recoverability (%)	100	100	100	100	100	94.12	59.32
	Avg. read overhead	1.0	1.28	1.56	1.99	2.59	3.29	3.83
Generalized Pyramid Code (GPC)	Recoverability (%)	100	100	100	100	100	94.19	76.44
	Avg. read overhead	1.0	1.28	1.56	1.99	2.59	3.29	4.12
GPC w/o global symbols	Recoverability (%)	100	100	100	100	97.94	88.57	65.63
	Avg. read overhead	1.0	1.28	1.56	1.87	2.32	2.93	3.85

Table 4.1: A generic (18, 12) MDS code and few Pyramid codes with the associated recoverability/efficieny characteristics as given in [19]

4.2 Generalized Model applied to Pyramid Codes

An interesting case study would be to apply the generalized Markov model of the previous section to one of the modern erasure codes such as Pyramid Codes of Microsoft Azure Storage [19]. Pyramid codes are designed to improve the recovery performance for small-scale device failures and have been implemented in archival storage [20]. Pyramid codes are constructed from standard MDS codes by constructing newer parity symbols from already existing parities in order to trade-off the *recoverability*, *coding overhead* and the *average read overhead*, which are important parameters to optimize for a storage application. Let us use a (16, 12) MDS code as the basis for (18, 12) set of pyramid codes given in table 4.1. These recoverability and read overhead values are computed and presented in [19]. First, we notice that recoverability values divided by 100 gives us q_i values for each one of the pyramid codes. The metric, *average read overhead*, represents the average number of extra device reads as an overhead in order to access each data block. Let us consider an example of one block failure in the (18,12) MDS code to show how this metric is computed. If the failure is a redundant block (6/18 chance), then the data blocks can be accessed directly, so the average read overhead is 1. Otherwise, the failure shall be a data block (12/18 chance), then the read overhead is twelve for the failed data block and one for the rest of the eleven data blocks. Hence, the average read overhead is (12+11)/12. Altogether, the average read overhead is $1 \times 6/18 + (12+11)/12 \times 12/18 \approx 1.61$.

In fact, we can find the average read overhead for a generic (n, k) MDS code when we have j failures using the following generalized expression[4]

$$\Phi_j = \sum_{i=0}^{j} \frac{(ik + k - i)\binom{n-k}{j-i}\binom{k}{i}}{k\binom{n}{j}} \tag{4.7}$$

Although the average overhead is not the only metric effecting the repair process, for simplicity we assume repair rates to be inversely proportional to that metric. Let us define

$$\overline{\mu}_j \triangleq \delta\mu \ln((j+1)\Phi_{j+1}) \tag{4.8}$$

[4]This expression can easily be proved by Induction and thus the proof is omitted for space.

Failure/Nominal Repair rates		$\lambda = \frac{1}{200K}$	$\lambda = \frac{1}{500K}$	$\lambda = \frac{1}{1.2M}$
Generic MDS Code	MTTDL ($hours$)	2.2e+15	6.4e+17	1.3e+20
	Durability ($nines$)	11	13	15
Basic Pyramid Code (BPC)	MTTDL ($hours$)	1.3e+17	5.2e+18	1.7e+20
	Durability ($nines$)	13	14	16
Generalized Pyramid Code (GPC)	MTTDL ($hours$)	1.32e+17	5.26e+18	1.76e+20
	Durability ($nines$)	13	14	16
GPC w/o global symbols	MTTDL ($hours$)	1.83e+14	3e+15	4.1e+16
	Durability ($nines$)	10	11	12

Table 4.2: MTTDL for various erasure codes.

where μ is the nominal rate of the repair per device and δ is a constant used to model the relative bandwidth constraint (with respect to an MDS code, for an MDS code it is normalized to $\delta = 1$) to reflect on the repair rates based on average read overhead metric. This formulation assumes an inverse exponential relationship between the nominal repair rate and the average read overhead. More field data is needed to confirm such a relationship.

Let χ_i be the average read overhead using one of the pyramid codes when i devices fail. A closed form expression for χ_i for a given pyramid code with any desired level of hierarchy is not derived in the original paper and is an interesting open problem. We shall use the computed values of χ_i in [19] for our MTTDL computation. Using the generalized Markov model of the previous section, let us further assume a homogenous repair strategy is used in the system i.e., $\mu_j = \frac{\overline{\mu}_j}{\ln((j+1)\chi_{j+1}))}$. Some results are shown in table 4.2 using a nominal repair rate $\mu = 1/168$ (1 week mean repair time), $\eta = 10^{-3}$ and $\delta = 20$. We observe that basic and generalized pyramid codes provide better durability numbers thanks to their efficient repair mechanisms. As λ gets close to zero, the frequency of repairs go down and hence the advantage of pyramid codes diminish. This can be observed with the MTTDL results given for $\lambda = 1/1.2M$. Another interesting observation is that given these computation parameter settings, global symbols are quite crucial for pyramid codes for maintaining a desired level of durability.

5 Availability

One of the other important metrics of Reliability is *availability*, defined as the fraction of time the system is able to provide access to the data through some redundancy scheme. Availability deals more with temporary inaccessibility when enough number of devices are unable to respond for the reconstruction of the data. The lifetime of a device is made up of periods of availability and unavailability. Device *uptime* means the time while the device is operational/online whereas the *downtime* of the device refers to the period in which the device is unresponsive/offline. When the device is offline, the data it contains is temporarily unavailable and does not participate in the system unless it becomes online again. Devices can become unavailable for a variety of reasons. For example, a storage device, node or networking switch can be overloaded; the operating system may crash or restart; the controller

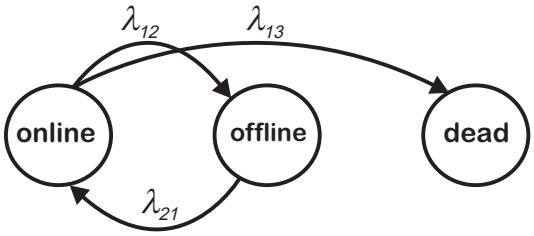

Figure 5.1: Markov chain for availability of a device/node.

may experience a hardware error; or the whole cluster of storage devices could be brought down for maintenance. The vast majority of such unavailability events are transient and do not result in permanent data loss [21].

One of the simplest definitions for availability can be given by the ratio of the expected uptime divided by the sum of the expected up and down times, i.e.,

$$p_A \triangleq \frac{\mathbb{E}uptime}{\mathbb{E}downtime + \mathbb{E}uptime} \tag{5.1}$$

If we assume uptimes and downtimes to be exponentially distributed, the underlying random process becomes a Markovian, the state diagram of which can be shown as in Fig. 5.1. Let us say the mean uptime and downtime of a device are given by t_{up} and t_{down}. Device availability is given by $p_A = t_{up}/(t_{up} + t_{down})$. Furthermore if the MTTF $(1/\lambda)$ [5] of the device is sufficiently large with respect to mean uptime and down times ($\lambda t_{up} > \lambda t_{down} \gg 1$), we can show that [22], the transition rates of the continuous Markov model are

$$\lambda_{12} = \frac{p_A - \lambda t_{up}}{t_{up} p_A}, \;\; \lambda_{13} = \frac{\lambda}{p_A}, \;\; \lambda_{21} = \frac{1}{t_{down}}. \tag{5.2}$$

Note that if the device never goes offline, i.e., $p_A = 1$ or $t_{down} = 0$, we have $\lambda_{13} = \lambda = 1/t_{up}$. Moreover, using the result of Lemma 1, we can conclude that the probability of a transition from state **online** to the state **dead** is given by

$$p_{13} = \frac{\lambda_{13}}{\lambda_{12} + \lambda_{13}} = \frac{\lambda t_{up}}{p_A} = \lambda(t_{up} + t_{down}) \tag{5.3}$$

which will be useful for next section's discussion.

5.1 Availability with Timeouts

Although the state diagram is clear, the system cannot distinguish between a data/parity device that is dead, and one that is merely offline. In both cases, the device is unresponsive and there is no way of knowing whether it is going to be back online again. This detection problem leads to an uncertainty about when to initiate the repair process. One of the mechanisms to decide when to trigger a repair is to wait some period of time (called the *timeout*) for

[5] Here we implicitly assumed that device lifetime is exponentially distributed. In fact it can be shown that this is the case if $\lambda t_{down} \ll 1$ [22].

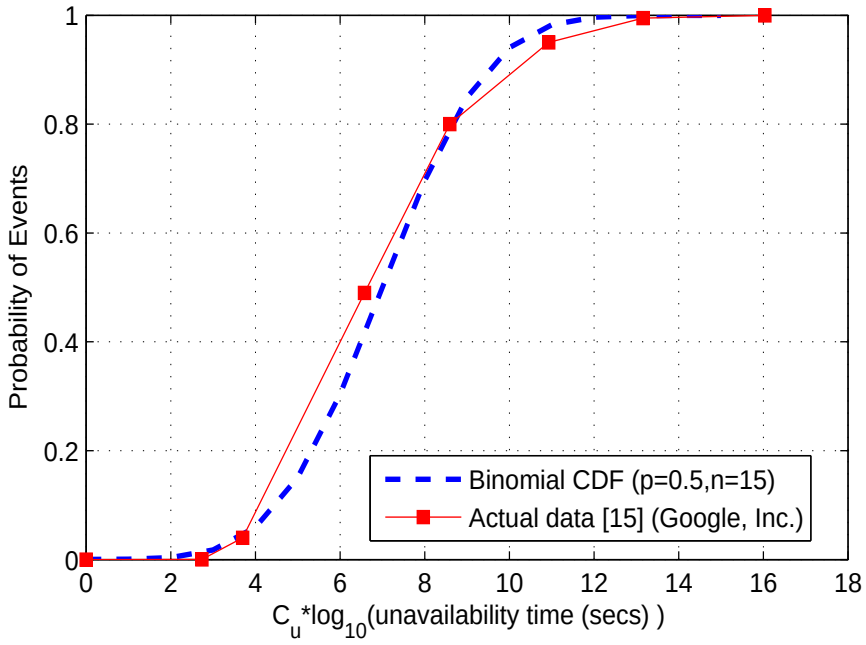

Figure 5.2: Cumulative distribution function of the duration of node unavailability periods versus the binomial fit.

the offline/dead replica to return to the online state. If the device returns online before the timeout occurs, no repair is necessary; otherwise, the system assumes that it is dead, and initiates a repair process. Choosing a timeout period depends on how aggressively the system wants to repair. The timeout period must be optimized for a given system to prevent from inefficient use of system resources (for example unnecessary repairs).

The following discussion is given in [22] and will be restated here as it provides us useful information about the actual repair process with timeouts. They introduced the system parameter α such that the timeout period shall be given by αt_{down}. The parameter α allows the system to trade off how aggressively it responds to the potential loss of a data/parity versus how many unnecessary repair initiations are made.

Let Y_α denote the time that passes between the data/parity generation, and leave of the online state without return i.e., either the instant of irrecoverable error or a period of time exceeding the timeout. From the arguments of [22], we have the following lemma.

Lemma 4 [22]: *The expected value of the random variable Y_α is given by*

$$\mathbb{E}Y_\alpha = \frac{(1-p_{13})(1-e^{-\alpha})\left[t_{up} + t_{down}\left(1 - \frac{\alpha e^{-\alpha}}{1-e^{-\alpha}}\right)\right]}{p_{13} + (1-p_{13})e^{-\alpha}} + t_{up} \tag{5.4}$$

PROOF: Proof of this lemma is given in [22]. □

A general practice of choosing an appropriate α can be given by solving the following

equation given λ, t_{up} and t_{down}

$$\frac{\lambda(1-p_{13})(1-e^{-\alpha})\left[t_{up}+t_{down}\left(1-\frac{\alpha e^{-\alpha}}{1-e^{-\alpha}}\right)\right]}{p_{13}+(1-p_{13})e^{-\alpha}} = 1+\lambda(1-\alpha)t_{down}-p_{13} \tag{5.5}$$

where we can notice that solving this nonlinear equation is nothing but finding the point that satisfies $\mathbb{E}Y_\alpha + \alpha t_{down} = 1/\lambda$ i.e., expected time to timeout intersects with the linear line $y = 1/\lambda$. This in turn means the repair process is initiated and hopefully generates exactly one new replica at every device lifetime. The same study also presents results regarding the insertion of memory to the overall repair process. Apparently, taking into account which devices or nodes are subject to repair, and when they return to online state, unnecessary redundancy can be eliminated to save the system resources. Interestingly though, when α is chosen meticulously it is shown that with-memory repair is not significantly better than without-memory repair.

5.2 Case Study: Google unavailability data

The data gathered from tens of Google storage cells, each with 1000 to 7000 storage nodes, over a one year period [21] suggests that less than 10% of unavailability events last longer than 15 minutes. Let us express the unavailability event duration in units of seconds, and use a mapping of the duration metric using $C_u \times \log_{10}(\text{“unavailability event duration"})$, where C_u is an appropriate constant. This change appropriately allows us to fit a binomial CDF to the real data as shown in Fig. 5.2. The binomial distribution has parameters $p = 0.5$ and $n = 15$ to match the data with $C_u = 3.7$. Although such is given for node unavailability in [21], we expect similar trends with device/disk unavailabilities.

Using the binomial distribution approximation based on the Google's data, t_{down}, the mean downtime can be found to be around 0.03 hours. It is also reported in [21] that they typically wait 15 minutes (0.25 hours) before commencing recovery of data on unavailable nodes. This implies $\alpha = 8.\overline{3}$. We can either set an AFR i.e., using an exponential time distribution, λ or the uptime t_{up}. It is usually more reasonable to measure t_{up} and through solving the equation (5.5), λ can be calculated. With this value and the estimated repair rates, MTTDL computations can be performed as previously discussed.

From Google's paper, it is convenient to assume an AFR value of %4 (also reported to be more realistic observed in many data centers and field measurements [21]) and calculate the t_{up} as the standard uptime measurements are not reported in the paper. Using these assumptions, we solve the equation (5.5) for t_{up} to be 218,089 hours. This implies $\lambda_{13} = 0.99584$ which is close to one. Ofcourse, unavailability of devices can dramatically vary depending on the environment and correlation factor between other functional system components. True values shall change our calculations but definitely not the methodology presented here.

6 Advanced Modeling

In our previous considerations, the set of extensions and Markov analysis associated with these models can be argued to capture a simplistic view of an actual storage system, partic-

ularly while the size and the scale of such systems are ever growing. Although, the previous discussions are quite improved versions of the available analytical methods using real data (this makes them great machineries for back-of-the-envelope comparisons), they can still be criticized due to the underlying model assumptions such as constant failure and repair rates. Unfortunately though, the inclusion of time dependence into reliability models increases complexity and can prohibit analytic reliability estimates. In addition, we have observed that extending the canonical model to multi-disk fault-tolerant systems leads to issues in modeling rebuild due to memoryless property of underlying exponential failure/repair time distribution.

6.1 Incorporating sector errors

Previous models given for incorporating hard errors into the Markov model suffers from representing accurately the latent sector errors which are found to be more frequent than unrecoverable hard bit read errors [23]. In the literature, there have been two ways to model the latent sector failure/scrubs. One is to treat them as disk failures or model them as a separate random processes. Here we present the latter approach which is based on regular scrubbing to remedy the latent sector errors. Given a fixed scrub period for a given sector, denoted as T_S, load on a given sector, l, probability of a sector error due to a write operation, P_w and the ratio of write requests in the total system load, r_w, the probability of an unrecoverable error on a given sector at an arbitrary time t (not time dependent) is given by [24]

$$P_S = \left(1 - \frac{1 - e^{-lT_S}}{lT_S}\right) P_w r_w \tag{6.1}$$

In the same study, a time-dependent probability is derived to be of the form

$$P_S(t) = \left(1 - e^{-l(t \mod T_S)}\right) P_e \tag{6.2}$$

where P_e is the single sector failure probability. Unfortunately, incorporating the load term l, into such a calculation to come up with a proper sector failure model remains an open problem. P_S can be used to calculate the likelihood of a latent sector failure on a disk. If a disk has S sectors, then the probability that one or more latent sector errors are present at an arbitrary point in time is $P_{LSE} = 1 - (1 - P_S)^S$. Given the fact that sector errors are more dominant type of failures relative to unrecoverable bit errors, P_{LSE} can be used in place of P_{UCER} given in the Markov model that incorporates the hard error in the MTTDL calculations (See equation 3.22). Although such a change shall create more accurate estimates, it does not account for the critically exposed region of the rebuild process or the number of sector errors [10].

6.2 Simulators

With all the extensions of the canonical Markov model, the remaining critical problem with previously discussed Markov models is the time dependent failure and repair rates.

In addition, they are argued not to model well multi-disk fault tolerance, irregular fault-tolerance and sector errors. Due to these reasons, the most effective way remaining is simulation. Standard simulation tools handles time dependence pretty effectively, however obtaining a result in a reasonable amount of time is non-trivial (mostly due to rare events). Standard simulation tools use two methodologies to evaluate the reliability of an erasure coded system [25]. One of them is based on the estimation of unreliability function $U(t)$ and the other is based on MTTDL. Let T_F be the random variable characterizing the failure time, T_i be the stopping time of the ith iteration and $\mathbf{1}_{T_F \leq t}(t)$ be the indicator function, using a law of large numbers argument we have

$$\hat{U}(t) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{T_F \leq t}(t) \tag{6.3}$$

$$MTTDL = \frac{1}{N} \sum_{i=1}^{N} T_i \tag{6.4}$$

where the simulation is assumed to carry out N consecutive iterations. As previously noted, aforementioned simulation methods are inefficient when evaluating highly fault tolerant systems. The major drawback to this type of simulation is the amount of time required to get an accurate result when the probability of failure is extremely low. The main approach to circumvent this situation is to increase the frequency of rare events after some threshold number of failures occur in the standard simulation tool. This technique is generally called *Importance Sampling* in literature [26]. Main objective is to increase the probability of seeing data loss event while keeping the variance (system) reduced after the threshold is reached.

There have been many more advanced simulation-based reliability estimators, renowned by their high fidelity book keeping (tracking which disk and sector failures have occurred, and efficiently determining if the failures constitute a data loss event) features. One of them is High-Fidelity Reliability (HFR) Simulator. For more information and the validation routines, the reader is referred to the reference [10].

REFERENCES

[1] 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.

[2] A. G. Tomislav, G. S. Kleiman, J. Leong, S. Sankar, P. Corbett and B. English, “Row-diagonal parity for double disk failure correction," *In USENIX Conference on File and Storage Technologies,* 2004.

[3] C. Huang and L. Xu. “STAR: an effcient coding scheme for correcting triple storage node failures," *In FAST'05: Proceedings of the 4th conference on USENIX Conference on File and Storage Technologies,* Berkeley, CA, USA, 2005.

[4] J. S. Plank and C. Huang, “Tutorial: Erasure Coding for Storage Applications," *Slides presented at FAST–2013: 11th Usenix Conference on File and Storage Technologies,* Feb, 2013.

[5] M. Rausand and A. Høyland, “System Reliability Theory", John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd ed., 2004.

[6] W. Burkhard and J. Menon, “Disk Array Storage System Reliability", *In proceedings of the international Symposium on Fault-tolerant computing,* pp. 432–441, 1993.

[7] W. A. Burkhard and P. D. Stojadinovic, “Storage-Efficient Reliable Files", *In Proceedings of the Winter 1992 USENIX Conference,* pp. 69–77, San Francisco, January 1992.

[8] K. M. Greenan, J. S. Plank, and J. J. Wylie, “Mean time to meaningless: MTTDL, Markov models, and storage system reliability," *in Proc. of the USENIX Workshop on Hot Topics in Storage and File Systems* (HotStorage), pp. 1–5., 2010.

[9] V. Venkatesan, I. Iliadis, “A general reliability model for data storage systems," *In Proc. 9th IntŠl Conference on Quantitative Evaluation of Systems (QEST 2012),* pp. 209–219, 2012.

[10] K. Greenan, “Reliability and power-effciency in erasure-coded storage systems," PhD thesis, University of California, Santa Cruz, Dec. 2009.

[11] J. L. Hafner and KK Rao, “Notes on reliability models for non-MDS erasure codes," *Technical Report RJ–10391*, IBM, Oct. 2006.

[12] J.-F. Pâris, T. Schwarz, S.J., A. Amer and D. D. E. Long, “Highly Reliable Two-Dimensional RAID Arrays for Archival Storage," *Proc. 31st Int. Performance of Computers and Communication Conf.,* pp. 324–331, Dec. 2012.

[13] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn. CRUSH: “Controlled, scalable, decentralized placement of replicated data," *In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC'06)*, Tampa, FL, Nov. 2006.

[14] J. J. Wylie and R. Swaminathan, “Determining Fault Tolerance of XOR-based Erasure Codes Efficiently," DSN-2007: *IEEE International Conference on Dependable Systems and Networks,* Edinburgh, Scotland, Jun., 2007.

[15] J.-F. Pâris, A. Amer, and T. J. E. Schwarz, “Low-Redundancy Two Dimensional RAID Arrays," *Proc. 2012 Int. Conf. on Computing, Networking and Communications, Data Storage Technology and Applications Symp.*, pp. 507–511, Jan.–Feb. 2012.

[16] S. S. Arslan, “Redundancy and Aging of Efficient Multidimensional MDS Parity-Protected Distributed Storage Systems," *IEEE Trans. Device and Materials Reliability*, Vol. 14, No. 1, pp. 275–285, Mar. 2014.

[17] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems," *IEEE Trans. Inf. Theory,* vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[18] J. J. Wylie and R. Swaminathan, "Determining Fault Tolerance of XOR-based Erasure Codes Efficiently," *DSN–2007: The International Conference on Dependable Systems and Networks, IEEE,* Edinburgh, Scotland, June, 2007.

[19] C. Huang, M. Chen, and J. Li, “Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," *ACM Transactions on Storage,* 9(1):1–28, Mar. 2013.

[20] A. Wildani, T. J. E. Schwarz, E. L. Miller, and D. D. Long, “Protecting against rare event failures in archival systems," *In Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'09)*, pp. 1–11, London, UK, Sept. 2009.

[21] Daniel Ford, Franis Labelle, Florentina I. Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlna, “Availability in Globally Distributed Storage Systems," *In Proceedings of the 9th Symposium on Operating Systems Design and Implementation*, (OSDI'10), Vancouver, Canada, Oct. 2010.

[22] S. Ramabhadran, and J. Pasquale, “Analysis of the durability of replicated distributed storage systems," *UCSD Technical Report CS2007–0900*, 2007.

[23] L.N. Bairavasudaram, “Characteristics, impact, and tolerance of partial disk failures. Diss," The University of Wisconsin, USA. Madison: ProQuest, UMI Dissertations Publishing, 2008.

[24] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, “Disk scrubbing versus intra-disk redundancy for high-reliability raid storage systems," *In SIGMETRICS'08: Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems,* pp. 241–252, New York, NY, USA, 2008.

[25] V.F. Nicola, P. Shahabuddin, and M.K. Nakayama, “Techniques for fast simulation of models of highly dependable systems," *IEEE Transactions on Reliability*, 50(3):246–264, Sept. 2001.

[26] Marvin K. Nakayama and Perwez Shahabuddin, “Quick simulation methods for estimating the unreliability of regenerative models of large, highly reliable systems," *Probab. Eng. Inf. Sci.,* 18(3):339–368, 2004