

Reliability for Networked Storage Nodes

KK Rao, James Lee Hafner and Richard A. Golding
IBM Almaden Research Center

kkrao@us.ibm.com, hafner@almaden.ibm.com, rgolding@us.ibm.com

Abstract

High-end enterprise storage has traditionally consisted of monolithic systems with customized hardware, multiple redundant components and paths, and no single point of failure. Distributed storage systems realized through networked storage nodes offer several advantages over monolithic systems such as lower cost and increased scalability. In order to achieve reliability goals associated with enterprise-class storage systems, redundancy will have to be distributed across the collection of nodes to tolerate both node and drive failures. In this paper, we present alternatives for distributing this redundancy, and models to determine the reliability of such systems. We specify a reliability target and determine the configurations that meet this target. Further, we perform sensitivity analyses where selected parameters are varied to observe their effect on reliability.

1. Introduction

High-end enterprise storage systems currently deployed in production environments have traditionally been monolithic systems – so-called ‘big iron’ with several symmetrical multiprocessors, multiple internal fabrics and large cache memories. These systems are expensive – requiring customized hardware and multiple redundant components and paths to ensure that there is no single point of failure. In contrast, achieving scalability through distributed storage systems is becoming increasingly prevalent in research and development [4], [8], [12], and, to some extent, in commercial deployments [9]. A significant aspect of distributed systems is the ability to use common building blocks across a wide range of storage requirements: from a few terabytes to the scale of petabytes. This translates into several advantages: lower cost due to economies of scale, reduced number of inventory types, commonality of software across the product line, and so on.

The distributed storage system in this paper is modeled after the Collective Intelligent Bricks project in IBM Research [14]. The storage system consists of several bricks where each brick or node is a sealed unit consisting of a controller, power supply, networking interfaces and disk drives. Several components in the node represent single points of failure. In order to achieve reliability goals associated with high-end enterprise-class storage systems, redundancy has to be distributed across the collection of nodes to tolerate both drive and node failure. In this paper, we will model the reliability of such

a system and look at the alternatives for distributing redundancy between the nodes in order to meet reliability goals of large-scale enterprise systems.

The goal of this paper is to view redundancy requirements from a storage viewpoint. We assume that there is enough redundancy in switches and links so that reliability is limited by storage nodes and drives; that is, the interconnect fabric and topology is not a constraining factor in determining the overall reliability of the system. This is typically the case with [14]. A detailed reliability analysis of such a system is described in [3].

We describe the different configurations for achieving reliability in distributed storage systems, in Section 3. In Section 4, we describe the models used to obtain reliability for these configurations. The implication of distributing data across such a system and its impact on reliability is presented in Section 5. Section 6 presents a baseline reliability analysis. We analyze the sensitivity of the reliability of some of the configurations to several parameters, in Section 7. A discussion of this analysis, the insights gained from it and results for a specific reliability target are presented in Section 8.

2. Related Work

Trivedi [13] covers reliability analysis and in particular, the use of continuous-time Markov chains with absorbing states for crash failures. The modeling and analyses presented in this paper are based on this work. Xin *et al.* [15] present reliability for large distributed systems but do not consider node failures. Also, while uncorrectable sector errors are dealt with through a scheme of signatures, the reliability improvements through the use of this scheme are not characterized. Saito *et al.* [12] describe building enterprise disk arrays using commodity components. However, the reliability analysis does not address uncorrectable sector errors. Snappy Disk and Petal [8] represent shared-disk, shared-metadata systems and partitioned-disk, partitioned-metadata systems respectively. The availability analysis presented in [8] is intended only to gain insights into the factors affecting availability rather than to derive accurate predictions. Plank *et al.* [10] describe LDPC erasure codes in the context of peer-to-peer and widely distributed storage systems, and Goodson *et al.* [5] describe erasure-coded storage that tolerates Byzantine failures. Both these papers focus on the algorithms for erasure coding for distributed storage nodes and do not address the reliability analysis needed to ensure that erasure coded distributed storage will meet required reliability goals. [1], [2], [6]

and [7] describe codes that tolerate two or more erasures. Some of these codes are candidates to realize the fault tolerant configurations described below.

3. Redundancy Configurations

As mentioned earlier, a node consists of a controller card, network interfaces, a collection of disk drives and associated power supplies. Apart from the disk drives and the network interfaces, all other major components are not duplicated. Thus, the node is inherently unreliable as the failure of any one of these components will result in node failure. Hence, in order to build a highly reliable storage system out of a collection of such nodes, redundancy will have to be distributed through the collection.

We will look at two dimensions to realizing redundancy in the collection of nodes: redundancy within nodes to tolerate internal drive failures, and redundancy across nodes to tolerate entire node failures. Within the nodes, we will employ three possible configurations: no internal RAID, RAID 5 and RAID 6, which tolerate 0, 1 and 2 drive failures respectively. (The term RAID 5 is well understood; we use the term RAID 6 to denote any scheme that can tolerate two drive failures.) We will achieve redundancy across nodes by applying three types of erasure codes between them: codes that can tolerate 1, 2 and 3 node failures respectively. The three node configurations and three erasure code types between nodes yield a total of 9 combinations between them.

We assume that the nodes in this system are enclosed entities that are not amenable to service actions. This implies a fail-in-place philosophy where failed components within a node are not replaced. Specifically, in the case of failure of one or more disk drives within a node, the node will continue to operate with a reduced set of disks until either all disks fail or some other critical component fails rendering the node unusable. For the case of nodes with internal RAID (RAID 5 or RAID 6) we will assume that on a drive failure, data is re-striped removing the failed drive from the array, thereby restoring redundancy at the end of this operation. The resulting loss in capacity is adjusted against the spare capacity, as described below.

The fail-in-place service model implies that storage capacity is initially over-provisioned so that loss in capacity with subsequent failures can be tolerated. The over-provisioned storage capacity is either sufficient to deal with expected failures over the operational life of the installation, or spare nodes are added at appropriate times – e.g. when spare capacity drops below predetermined thresholds.

4. Reliability Models

In this reliability analysis, we are primarily interested in preventing data loss. Consequently, to compare redundancy configurations, we use the expected number of data loss events per unit time as a measure of reliability. We believe the expected number of data loss events per unit time is a metric that is easier to comprehend and

relate to than the more traditional Mean Time to Data Loss – MTTDL. We will use Markov models to determine MTTDL, and use it to obtain the expected number of data loss events per year.

We look at three types of failures that could lead to data loss: an uncorrectable read error from a disk drive, a failure of a disk drive and a failure of a node. A data loss event occurs when the above failures occur in a combination that cannot be handled by the data protection scheme used in the system. For example, in the simpler case of a single controller, a RAID 5 array can tolerate a single failure (a disk failure or an uncorrectable read error). When a drive in the RAID 5 array fails and the array is rebuilding to a spare or a replacement drive, if either a second drive fails or an uncorrectable read error occurs on any of the remaining drives, then a data loss event occurs. Clearly, the failure of the second drive results in data loss of a larger scale than the uncorrectable read error, but either failure results in some data loss.

We will assume that the failures are independent events. Clearly, certain types of failures such as drive failures within a batch of drives could be correlated failures. However, in our experience, we have seen that correlated failures do not result in orders of magnitude differences between observed and predicted reliability numbers. Furthermore, the modeling presented in this paper can be extended to correlated failures.

With respect to an uncorrectable read error, we assume it can result in a data loss event only if the array is in a critical state (e.g. a single drive failure in a RAID 5 array or two drive failures in a RAID 6 array) and cannot tolerate any further errors. We believe this is a reasonable assumption because as long as the array has not lost drives up to the fault tolerance of its RAID scheme, the recovery from an uncorrectable read error just requires reading from some or all of the remaining drives and regenerating the data item that encountered the read error. For a RAID 5 array, the conditions under which this recovery can fail are 1) if a corresponding sector in another strip on the same stripe encounters an uncorrectable read error, or 2) if another drive fails during this recovery. (For arrays with higher fault tolerance, additional failures must occur in order for the recovery to fail.) We believe that these conditions are extremely low probability occurrences (for example, in condition 2, the recovery time for a sector error is very small) and can be ignored.

We make another key assumption – that a background process is periodically scrubbing all the drives, handling uncorrectable media errors that may have developed through normal operations. This process occurs while all drives are available and redundancy information can be used to correct such errors. Thus, for data loss calculations, we only consider the contribution to media errors through read operations required for rebuilding arrays – and not the read operations in response to normal requests.

To describe our modeling methodology, we illustrate the technique for a RAID 5 disk array. Figure 1 shows the

Markov model for a RAID 5 array with mean time to failure of the disk drives $MTTF_d$ and mean time to repair (rebuild) a drive failure $MTTR_d$.

State 0 corresponds to a fully operational state. State 1 corresponds to a drive failure that will not experience an uncorrectable error during the rebuild. State 2 represents a data loss state – either due to a second drive failure or due to an uncorrectable error during rebuild. The parameters are:

- $d =$ number of drives in the array
- $\lambda_d =$ drive failure rate $= 1/MTTF_d$
- $\mu_d =$ drive rebuild rate $= 1/MTTR_d$
- $h =$ probability of an uncorr. error during rebuild
 $= (d-1) \cdot C \cdot HER$
- $C =$ drive capacity
- $HER =$ disk hard error rate expressed in hard errors per number of bytes read

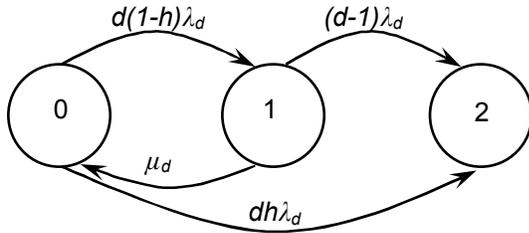


Figure 1. Markov Model for a RAID 5 array

It should be noted that the above model mathematically represents the reliability for a RAID 5 array and does not directly correspond to the functional states of the array (fully operational, rebuilding, etc.). The intuition behind the model in Figure 1 is as follows: h is the probability that the rebuild operation following a drive failure will encounter an uncorrectable error. As the basic drive failure rate is $d\lambda_d$, the term $dh\lambda_d$ represents the rate at which drive failures result in data loss due to uncorrectable errors during rebuild. Other representations such as the one shown in Figure 2 are also possible. There is no significant difference in the results from either of these representations.

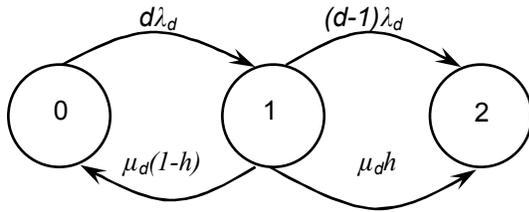


Figure 2. Alternate Markov Model for a RAID 5 array

Throughout this paper, we will use the representation of Figure 1. The methodology to solve a Markov model with absorbing states is described in [13]. Typically, $\mu_d \gg \lambda_d$. Solving this model for $MTTDL$ gives

$$MTTDL = \frac{(2d-1-dh)\lambda_d + \mu_d}{d(d-1)\lambda_d^2 + d\lambda_d\mu_d h}$$

$$\approx \frac{\mu_d}{d(d-1)\lambda_d^2 + d(d-1)\lambda_d\mu_d \cdot C \cdot HER}$$

4.1. Node Set and Redundancy Set

We introduce the concepts of node set and redundancy set for a storage system made up of networked storage nodes as shown in Figure 3. Data “objects” are stored across multiple nodes in such a system in order to meet requirements such as reliability (the focus of this paper) and performance. Each data object constitutes exactly one ‘stripe’ of data – that is, the redundancy elements (parity) can be computed entirely from this data. For a given data object, the set of nodes that contain the data and its corresponding redundancy (parity) elements constitutes a redundancy set. The node set is the set of all the nodes that have some redundancy set relationships with each other. We assume that data and parity are evenly distributed across all the nodes in the node set. Thus, each node has one or more redundancy set relationships with every other node in the node set. A storage system consists of one or more node sets. The total number of redundancy sets of size R in a node set of size N is given by the combinations $\binom{N}{R}$.

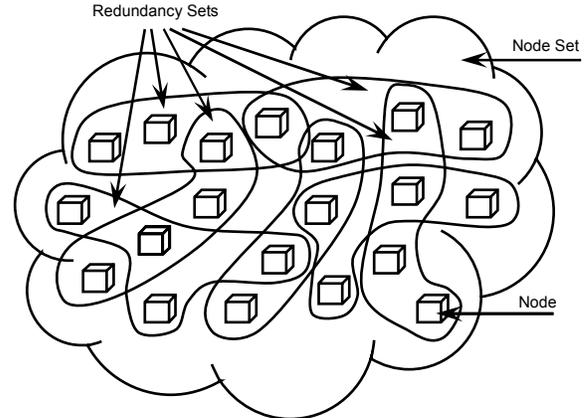


Figure 3. Node Sets and Redundancy Sets

The even distribution of data implies that the failure domain is the entire node set and not just individual redundancy sets. For example, in a redundancy scheme that tolerates only a single failure, when such a failure has occurred and is being recovered from, a failure of any second node in the node set will result in data loss.

We will present the modeling for systems where the nodes have internal RAID in section 4.2, and the modeling for nodes without internal RAID in section 4.3.

4.2. Nodes with Internal RAID

For a system in which the nodes have internal RAID, we use hierarchical Markov models to obtain $MTTDL$. We represent the RAID array internal to a node in a Markov model and obtain array failure rates from it. We then use these failure rates in a higher level Markov model representing the redundancy arrangement between nodes. Note that we assume that the nodes are not amenable to service actions, and that on a drive failure, the array is re-stripped to remove the failed drive from the array. Consequently, the μ_d term as depicted in Figure 1 is the

array re-stripe rate and not the array rebuild rate. We already obtained the *MTTDL* for a RAID 5 array as:

$$MTTDL \approx \frac{\mu_d}{d(d-1)\lambda_d^2 + d(d-1)\lambda_d\mu_d \cdot C \cdot HER}$$

We define array failure as the failure of disk drives beyond the fault tolerance provided by the RAID scheme. From the above, we obtain λ_D , the rate of array failure and λ_S , the rate of a sector error during a re-stripe. These are:

$$\lambda_D(RAID5) \approx \frac{d(d-1)\lambda_d^2}{\mu_d}$$

$$\lambda_S(RAID5) \approx d(d-1)\lambda_d \cdot C \cdot HER$$

Below is the Markov model for a RAID 6 array.

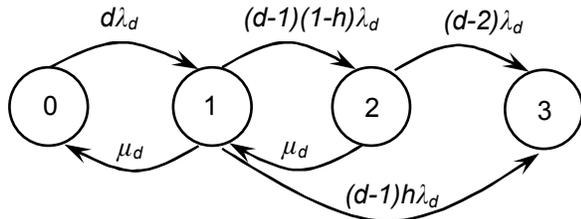


Figure 4. Markov Model for a RAID 6 array

State 0 is when the array is fully operational; state 1 is when a single drive has failed; state 2 corresponds to a second drive failure that will not experience an uncorrectable read error during rebuild; and state 3 represents a data loss state, either due to triple drive failure or an uncorrectable error when rebuilding with 2 drives failed. Solving this model for *MTTDL* (see [13]) gives

$$MTTDL \approx \frac{\mu_d^2}{d(d-1)(d-2)\lambda_d^3 + d(d-1)(d-2)\lambda_d^2\mu_d \cdot C \cdot HER}$$

Correspondingly, we obtain λ_D and λ_S as

$$\lambda_D(RAID6) \approx \frac{d(d-1)(d-2)\lambda_d^3}{\mu_d^2}$$

$$\lambda_S(RAID6) \approx \frac{d(d-1)(d-2)\lambda_d^2 \cdot C \cdot HER}{\mu_d}$$

We will use these rates in the higher level model for the erasure codes between nodes. Figure 5 shows the Markov model for nodes with internal RAID (either RAID 5 or RAID 6) and a redundancy arrangement with a fault tolerance of 1 between nodes.

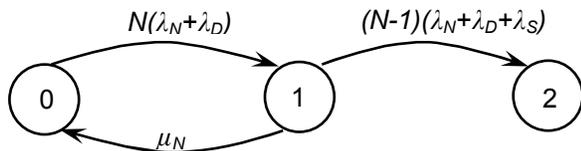


Figure 5. Markov Model for Fault Tolerance 1; Nodes with Internal RAID

Here N is the number of nodes in the node set, λ_N is the node failure rate and μ_N is the node rebuild rate. The array failure rate, λ_D , and the rate of sector error during a re-stripe, λ_S , are a function of the internal RAID level in

the nodes.

State 0 is when the storage system is fully operational. The system transitions to state 1 when either a node fails or a node experiences an array failure. In this state, the data of this node is rebuilt on the remaining nodes in the node set. (This is described in Section 5). State 2 represents the data loss state caused by a second node or array failure or a sector error during an internal RAID re-stripe while the node rebuild is in progress.

The *MTTDL* for this scheme (internal RAID, node fault tolerance 1) is given by:

$$MTTDL_{IR,NFT1} = \frac{\mu_N + (2N-1)(\lambda_N + \lambda_D) + (N-1)\lambda_S}{N(N-1)(\lambda_N + \lambda_D)(\lambda_N + \lambda_D + \lambda_S)}$$

$$\approx \frac{\mu_N}{N(N-1)(\lambda_N + \lambda_D)(\lambda_N + \lambda_D + \lambda_S)}$$

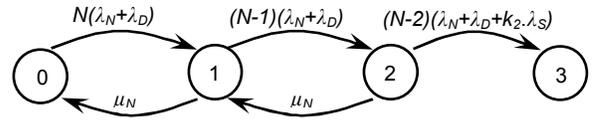


Figure 6: Markov Model for Fault Tolerance 2; Nodes with Internal RAID

Figure 6 shows the Markov model for nodes with internal RAID and an erasure code with a fault tolerance of 2 between nodes. As can be seen above, this scheme tolerates two failures; a third failure during the node rebuild operation results in a data loss event, State 3. We will explain the factor k_2 (and corresponding k_3 below) in section 5.2.1.

The *MTTDL* for this scheme (internal RAID, node fault tolerance 2) is:

$$MTTDL_{IR,NFT2} \approx \frac{\mu_N^2}{N(N-1)(N-2)(\lambda_N + \lambda_D)^2(\lambda_N + \lambda_D + k_2 \cdot \lambda_S)}$$

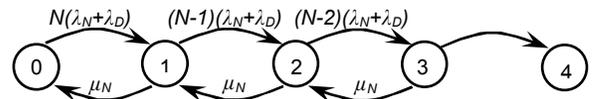


Figure 7: Markov Model for Fault Tolerance 3; Nodes with Internal RAID

Figure 7 shows the Markov model for nodes with internal RAID and an erasure code with a fault tolerance of 3 between nodes. This scheme tolerates three failures; a fourth failure during the node rebuild operation results in a data loss event, State 4.

The *MTTDL* for this scheme (internal RAID, node fault tolerance 3) is:

$$MTTDL_{IR,NFT3} \approx \frac{\mu_N^3}{N(N-1)(N-2)(N-3)(\lambda_N + \lambda_D)^3(\lambda_N + \lambda_D + k_3 \cdot \lambda_S)}$$

In general, for nodes with internal RAID, with node fault tolerance t , the *MTTDL* is:

$$MTTDL_{IR,NFTt} \approx \frac{\mu_N^t}{N(N-1)\dots(N-t)(\lambda_N + \lambda_D)^t(\lambda_N + \lambda_D + k_t \cdot \lambda_S)}$$

4.3. Nodes without Internal RAID

In these types of nodes, there is no RAID scheme deployed between the drives of the node and each of the drives is visible externally. In configurations for nodes without internal RAID, individual drives within each node are used to realize the erasure code between nodes. We assume that no more than one drive per node is used in each redundancy set, that is, each block of a data stripe is on a different node; thus, each node failure causes only a single erasure on each redundancy set.

Figure 8 shows the Markov model for nodes without internal RAID and an erasure code of fault tolerance 1 between nodes. Although there are only a few new parameters used in this model, we list all the parameters:

- $N =$ node set size
- $d =$ drives per node
- $\lambda_N =$ node failure rate
- $\lambda_d =$ drive failure rate
- $\mu_N =$ node rebuild rate
- $\mu_d =$ drive rebuild rate
- $h_N =$ probability of an uncorrectable error during node rebuild = $d \cdot (R-1) \cdot C \cdot HER = d \cdot h$
- $h_d =$ probability of an uncorrectable error during drive rebuild = $(R-1) \cdot C \cdot HER = h$
- $R =$ redundancy set size
- $C =$ drive capacity
- $HER =$ disk hard error rate expressed in hard errors per number of bytes read

State 0 is when the system is fully operational. State 1 corresponds to a node failure that will not experience an uncorrectable error during node rebuild. State 2 corresponds to a drive failure that will not experience an uncorrectable error during drive rebuild. State 3 represents a data loss state – either due to a second node or drive failure or due to an uncorrectable error during rebuild.

The *MTTDL* for this scheme (no internal RAID, node fault tolerance 1) is:

$$MTTDL_{NIR,NFT1} \approx \frac{\mu_d \mu_N}{N(N-1)(\lambda_N + d\lambda_d)(\mu_d \lambda_N + d\mu_N \lambda_d) + Nd h \mu_d \mu_N (\lambda_d + \lambda_N)}$$

Figure 9 shows the Markov model for nodes without internal RAID and an erasure code of fault tolerance 2 between nodes. The h_{xy} parameters are probabilities of an uncorrectable error during a second node or drive rebuild ($y = N$ or d respectively), after an initial node or drive failure ($x = N$ or d respectively). We will show how these parameters are determined in Section 5.2.2.

Comparing Figures 8 and 9, we observe that the number of states are doubled and that the state transitions in Figure 8 are represented in two subsets in Figure 9 – states 1, 2, 3 and 7; and states 4, 5, 6 and 7. We observe a similar representation when we look at the Markov model for nodes without internal RAID and an erasure code of fault tolerance 3 between nodes – Figure 10 in [11]. With the doubling of states with each increase in fault tolerance, conventional techniques to obtain a

parameterized closed form solution for these higher levels of fault tolerance are not practical. However, a recursive method can be developed to solve these and more general Markov models with arbitrary fault tolerance between nodes. Such a method is described in detail in the appendix in [11].

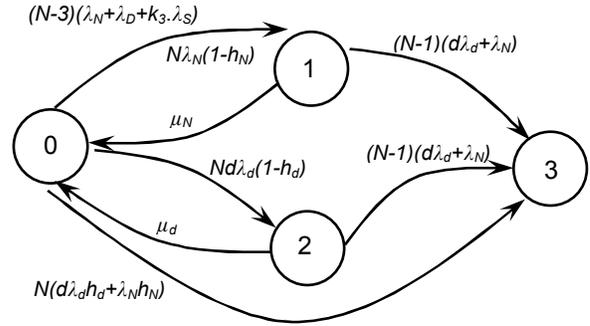


Figure 8. Markov Model for Fault Tolerance 1; Nodes without Internal RAID

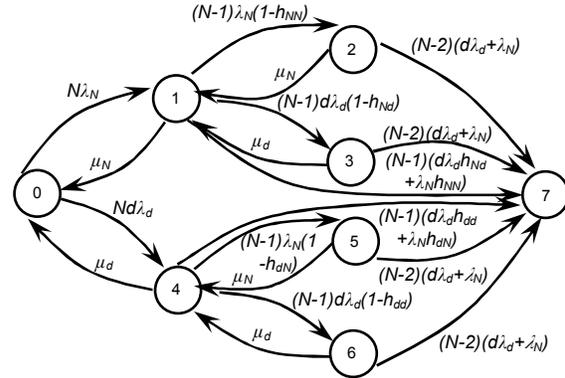


Figure 9. Markov Model for Fault Tolerance 2; Nodes without Internal RAID

The *MTTDL* for the last two schemes will be shown in Section 5.2 following the explanation of the h parameters.

5. Implications of Distributed Data

5.1 Node Rebuild Time

We mentioned earlier that the fail-in-place service model implies that the set of nodes is over-provisioned with spare capacity to deal with subsequent failures that will result in a loss of usable capacity. This model, coupled with the even distribution of data, implies that spare capacity is also evenly distributed among the nodes. Thus, when a node fails, the data on the failed node is rebuilt by all the remaining nodes evenly, utilizing their spare capacity. Similarly, in configurations without internal RAID, when a drive fails, the data on the failed drive is rebuilt on all the remaining drives in the node set. This is not the case for nodes with internal RAID: a drive failure results in a re-striping operation, removing the failed drive from the array and restoring redundancy.

Rebuild time, and hence the rebuild rate, is a key component in the expressions for MTTDL. We will describe a model to determine rebuild time accurately. Our model of rebuild time is based on the amount of data that is transferred during a rebuild.

We assume that in a rebuild, a destination node receives all the required redundancy data and performs the necessary exclusive-OR (or equivalent) operations to generate the data it will write on its drive(s). This is not the only rebuild algorithm, but it is sufficient for our analyses; alternate algorithms do not change the results significantly.

For a node set size of N , a redundancy set size of R and a fault tolerance of t across nodes, we express the amounts of data below in units of a node's worth of data. Note that this means that R nodes are involved in the rebuild of one lost data object.

- Amount of data rebuilt by each node $= \frac{1}{N-1}$
- Amount of data received by each node from other nodes to rebuild the above $= \frac{R-t}{N-1}$
- Total data received by all the $N-1$ nodes $= (N-1) \frac{R-t}{N-1} =$ total data sourced by all $N-1$ nodes
- Total data sourced by each node $= \frac{1}{N-1} (N-1) \frac{R-t}{N-1} = \frac{R-t}{N-1}$

The effective rebuild time will be the maximum time required to move data in and out of nodes, to and from disks, and through the interconnecting network, depending on where the bottleneck lies.

- Hence, total data in and out of a node $= 2 * \frac{R-t}{N-1}$
- Total data to and from the disks in a node $= \frac{R-t}{N-1} + \frac{1}{N-1}$
- The total data flowing in the interconnecting network $= R-t$

5.2 Scope of Sector Error

We stated earlier that we assume that an uncorrectable read error causes a data loss event only when the redundancy set is in a critical state. The even distribution of data across all the nodes implies that, for fault tolerance 2 or higher, when a redundancy set is critical, only a portion of a node's data (or drive's data in the case of no internal RAID) is critical.

This is illustrated in Figure 10. Let us assume that we have an erasure code of fault tolerance 2 between nodes and that the nodes have internal RAID. The X's indicated failed nodes. Each failed node is a part of two redundancy sets, one shared with the other failed node and one otherwise independent. However, only the shared set is

critical; the other has lost one node but can tolerate a second loss.

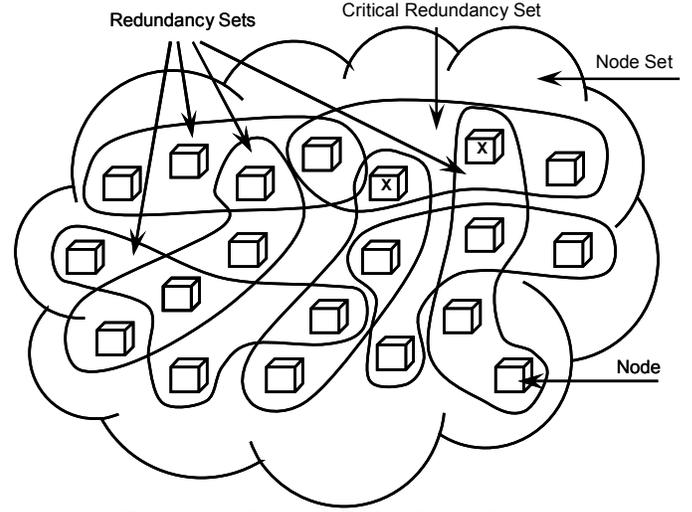


Figure 10. Critical Redundancy Sets

5.2.1 Nodes with Internal RAID

The fraction of redundancy sets that are critical and hence, can contribute to a sector loss are represented in the k_2 and k_3 terms in the MTTDL expressions for internal RAID, fault tolerance 2 and 3 respectively. Each node is a

part of $\binom{N-1}{R-1}$ redundancy sets. Thus,

$$k_2 = \frac{\binom{N-2}{R-2}}{\binom{N-1}{R-1}} = \frac{R-1}{N-1}, \text{ and}$$

$$k_3 = \frac{\binom{N-3}{R-3}}{\binom{N-1}{R-1}} = \frac{(R-1)(R-2)}{(N-1)(N-2)}$$

5.2.2 Nodes without Internal RAID

For nodes without internal RAID, we used h -with-subscript terms to represent probabilities of encountering uncorrectable sector errors during critical rebuilds. These probabilities depend on the amount of critical data that must be read for a rebuild operation, which in turn is derived from critical redundancy sets. Unlike nodes with internal RAID, redundancy sets may be critical because of combinations of node and drive failures.

The combinations and corresponding fractions of critical redundancy sets for fault tolerance 2 are:

- two nodes: $\frac{\binom{N-2}{R-2}}{\binom{N-1}{R-1}} = \frac{R-1}{N-1}$ of a node;

- two drives: $\frac{d^{R-2} \binom{N-2}{R-2}}{d^{R-1} \binom{N-1}{R-1}} = \frac{1}{d} \frac{R-1}{N-1}$ of a drive;

- and a drive and a node: $\frac{d^{R-1} \binom{N-2}{R-2}}{d^{R-1} \binom{N-1}{R-1}} = \frac{R-1}{N-1}$ of a

drive.

The probability of encountering a hard error while rebuilding a drive if the entire drive is critical is $(R-2) \cdot C \cdot HER$.

Now, if $h = \frac{(R-1)(R-2)}{N-1} \cdot C \cdot HER$, then

$$h_{NN} = dh, \quad h_{Nd} = h_{dN} = h \quad \text{and} \quad h_{dd} = \frac{h}{d}.$$

Similarly, the combinations and corresponding fractions of critical redundancy sets for fault tolerance 3 are:

- three nodes:

$$\frac{\binom{N-3}{R-3}}{\binom{N-1}{R-1}} = \frac{(R-1)(R-2)}{(N-1)(N-2)} \text{ of a node;}$$

- two nodes and a drive:

$$\frac{d^{R-1} \binom{N-3}{R-3}}{d^{R-1} \binom{N-1}{R-1}} = \frac{(R-1)(R-2)}{(N-1)(N-2)} \text{ of a drive;}$$

- two drives and a node:

$$\frac{d^{R-2} \binom{N-3}{R-3}}{d^{R-1} \binom{N-1}{R-1}} = \frac{1}{d} \frac{(R-1)(R-2)}{(N-1)(N-2)} \text{ of a drive;}$$

- and three drives: $\frac{d^{R-3} \binom{N-3}{R-3}}{d^{R-1} \binom{N-1}{R-1}} = \frac{1}{d^2} \frac{(R-1)(R-2)}{(N-1)(N-2)}$ of

a drive.

The probability of encountering a hard error while rebuilding a drive if the entire drive is critical is $(R-3) \cdot C \cdot HER$.

Now, if $h = \frac{(R-1)(R-2)(R-3)}{(N-1)(N-2)} \cdot C \cdot HER$, then

$$h_{NNN} = dh, \quad h_{NNd} = h_{NdN} = h_{dNN} = h,$$

$$h_{Ndd} = h_{dNd} = h_{ddN} = \frac{h}{d} \quad \text{and} \quad h_{ddd} = \frac{h}{d^2}.$$

We use these parameters to solve the Markov models and obtain the corresponding MTDLs, shown in Figure 11. A general solution for arbitrary fault tolerance is described in the appendix in [11].

6. Baseline Reliability

We use the closed form solutions for the MTDL for the various configurations and determine baseline reliability using parameters defined below. We assume that desktop/ATA drives are used in the nodes. The parameters are chosen based on drive specifications, observed reliability and ‘typical’ configurations.

$MTTF_N$ = node MTTF = 400,000 hours

$MTTF_d$ = drive MTTF = 300,000 hours

HER = drive hard error rate = 1 sector in 10^{14} bits read

C = drive capacity = 300 GB

Maximum drive throughput = 150 I/O ops/sec.

Drive sustained transfer rate (avg) = 40 MB/sec.

N = node set size = 64

R = redundancy set size = 8

d = drives per node = 12

Re-stripe command size = 1 MB

Rebuild command size = 128 KB

Link speed = 10 Gbps (800 MB/sec. sustained)

Capacity utilization = 75%

Bandwidth utilization for rebuild, re-stripe = 10%

The link speed needs clarification. The rebuild performance depends on the total rate data can move in and out of the node over all links. We assume that nodes are physically sealed units shaped like cubes and are stacked together to build larger three-dimensional structures. Nodes communicate with adjacent nodes through links on each of their six surfaces. [3] has more information on effective bandwidth of such structures.

We specify the reliability target in terms of data loss events per PB-year. We view reliability from a manufacturer’s perspective and choose a target that tracks the field population of such storage systems. We use the manufacturer’s view rather than a single customer view because our customers have a high expectation of reliability from our systems and a data loss event at any of our customers is perceived as a very significant event. Such events can make other customers feel vulnerable causing them to lose confidence in our storage systems. We set a reliability target that a field population of 100 systems each with a petabyte of logical capacity will experience less than one data loss event in 5 years. This translates to less than 2×10^{-3} data loss events per PB-year.

Figure 12 shows a baseline comparison of the 9 configurations using the parameters defined above. We observe the following:

1. Configurations with node fault tolerance of 1 do not meet our reliability target.
2. At fault tolerance 3, the internal RAID configurations exceed the target by 5 orders of magnitude.

$$MTTDL_{NIR,NFT2}$$

$$\approx \frac{\mu_d^2 \mu_N^2}{N(N-1)(N-2)(\lambda_N + d\lambda_d)(\mu_d \lambda_N + d\mu_N \lambda_D)^2 + N(R-1)(R-2) \cdot C \cdot HER \cdot d\mu_d \mu_N (\lambda_d + \lambda_N)(\mu_d \lambda_N + \mu_N \lambda_d)}$$

$$MTTDL_{NIR,NFT3}$$

$$\approx \frac{\mu_d^3 \mu_N^3}{N(N-1)(N-2)(N-3)(\lambda_N + d\lambda_d)(\mu_d \lambda_N + d\mu_N \lambda_D)^3 + N(R-1)(R-2)(R-3) \cdot C \cdot HER \cdot d\mu_d \mu_N (\lambda_d + \lambda_N)(\mu_d \lambda_N + \mu_N \lambda_d)^2}$$

Figure 11. MTTDL for No Internal RAID, Node Fault Tolerance 2 and 3

- There is no significant difference between internal RAID 5 and internal RAID 6 especially for fault tolerance 2 or higher. We will discuss why this is the case in Section 8.

Based on the above observations, for the analysis that follows, we will not consider configurations with fault tolerance of 1 between nodes. Also, for configurations with internal RAID, we only use RAID 5 as RAID 6 does not provide any advantage. Further, we do not include the configuration at fault tolerance 3, internal RAID in the sensitivity analyses (item 3. above) as this is well beyond our reliability requirements. This leaves us with three configurations for sensitivity analyses:

- Fault Tolerance 2 without internal RAID,
- Fault Tolerance 2 with internal RAID 5, and
- Fault Tolerance 3 without internal RAID.

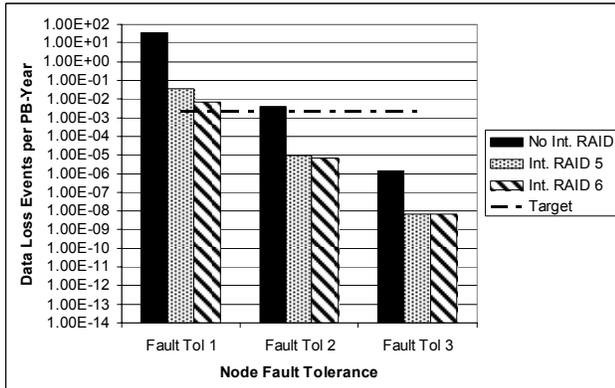


Figure 12. Baseline Comparison

7. Sensitivity Analyses

We will perform sensitivity analyses of the reliability to the following parameters: drive MTTF, node MTTF, rebuild block size, link speed, node set size, redundancy set size, and drives per node. As we vary these parameters one at a time, we will keep all the other parameters at their baseline level, except for drive and node MTTF. For the latter two, we will use two values, one at each end of a practical range, namely:

Drive MTTF (hours): low 100,000; high 750,000;

Node MTTF (hours): low 100,000; high 1,000,000.

Figure 13 shows the sensitivity to disk drive MTTF. We observe that the configuration at fault tolerance 2, no internal RAID does not meet the target at all for low node MTTF, and marginally meets it for high node MTTF. The other two configurations exceed the target – some more

comfortably than the others – over the entire range. FT 2, Internal RAID 5 appears to be relatively insensitive to drive MTTF, especially for low node MTTF – clearly, it is limited by node MTTF and provides one view why RAID 6, which protects from a further drive failure, does not offer any advantage (see Section 8).

The sensitivity to node MTTF is shown in Figure 14. FT 2, Internal RAID 5 shows the most sensitivity to node MTTF and all three configurations show increased sensitivity with high drive MTTF. FT 2, No Internal RAID again does not meet the target for the most part.

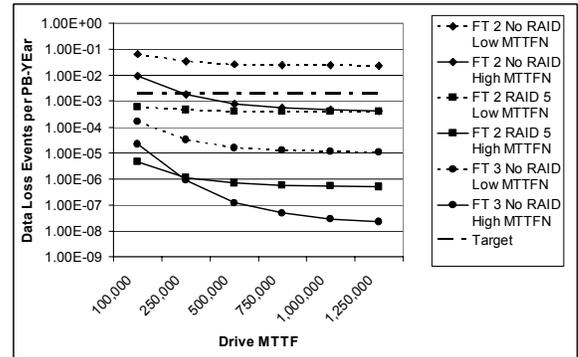


Figure 13. Sensitivity to Drive MTTF

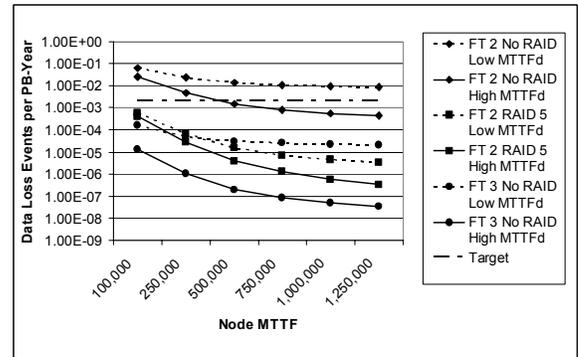


Figure 14. Sensitivity to Node MTTF

The rebuild block size affects the node and the drive rebuild rate, μ_N and μ_d respectively. As we saw in Sections 4 and 5, these are key parameters for the MTTDL. From Figure 15, it can be seen that the rebuild block size affects the reliability significantly. FT2, No Internal RAID does not meet the target for low MTTF. The other two configurations meet the target if the rebuild block size is 64 KB or larger.

The rebuild rate is determined by the slower of the data transfers – across the network between nodes or within a node to and from the disk drives. With the parameters as defined (12 drives per node, 150 I/O operations/second, and so on), the rebuild rate is constrained by the link speed up to around 3 Gb/s beyond which it is constrained by the disk drives.

This can be seen in Figure 16 which shows sensitivity to link speed at 3 points – 1, 5 and 10 Gb/s. There is no difference in reliability between the last two points.

We now look at sensitivity to the configurable parameters – node set size, redundancy set size and drives per node. Figure 17 shows the sensitivity to node set size. As can be seen, FT 2, No Internal RAID shows some sensitivity to the node set size, but the other two configurations are relatively insensitive to it.

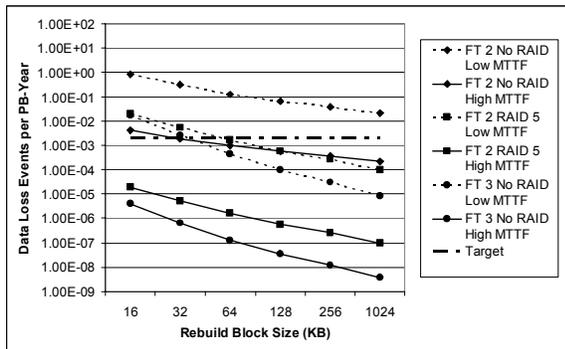


Figure 15. Sensitivity to Rebuild Block Size

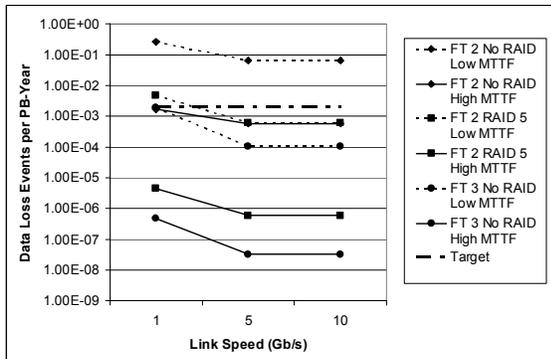


Figure 16. Sensitivity to Link Speed

The sensitivity to redundancy set size is shown in Figure 18. It can be seen that all configurations appear to become less reliable as the redundancy set size increases, with about an order of magnitude difference between the extremes.

From Figure 19, it can be seen that there is very little sensitivity to the number of drives per node. It should be noted that we are measuring normalized reliability – data loss events per PB-Year. As a result, with some parameters such as drives per node, there is a cancellation effect. Increasing the number of drives in a node can result in decreased reliability per node – however, fewer such nodes will be required to yield a petabyte.

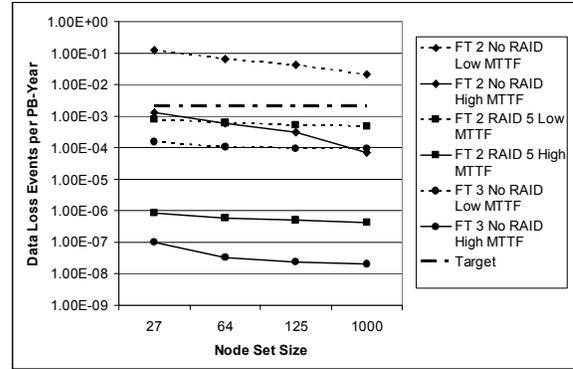


Figure 17. Sensitivity to Node Set Size

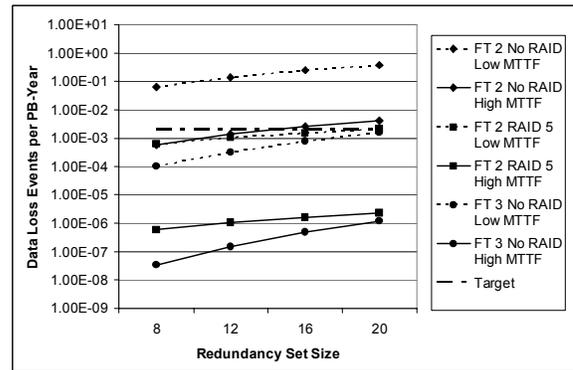


Figure 18. Sensitivity to Redundancy Set Size

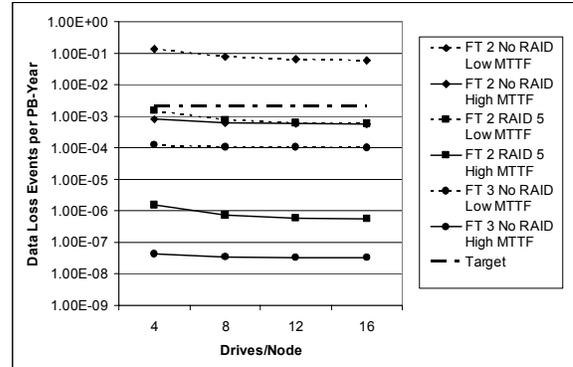


Figure 19: Sensitivity to Drives per Node

8. Discussion

The baseline reliability analysis in Section 6 showed that RAID 6 does not offer any advantage over RAID 5 when used internal to networked storage nodes. This is because the reliability of a networked storage system as a whole is affected by both drive and node failures. When RAID 5 is used internally, the effect of drive failures is considerably minimized such that the susceptibility to node failures becomes a dominant factor. Providing further tolerance to drive failures by using RAID 6 does not alleviate the susceptibility to node failures. It is interesting to note that we need to obtain a balance of protection against both drive and node failures – increasing the protection for one without correspondingly

increasing it for the other does not result in an overall increase in reliability.

The sensitivity analyses in section 7 reveal interesting results. Firstly, we see that there is very little sensitivity to the configuration parameters – node set size and drives per node – and a little more pronounced sensitivity to redundancy set size. We alluded to the reason for the insensitivity to drives per node earlier. Similar arguments apply to the node set size. In the latter case, there is an additional factor. Even though increasing the node set size increases the size of the failure domain, the fraction of critical redundancy sets decreases.

We also see that the reliability is constrained by disk drive bandwidth rather than network bandwidth if the link speed is 3 Gb/s or higher, resulting in no change in reliability at higher link speeds. By using drive bandwidth more efficiently through the use of larger rebuild block sizes, we see significant improvements in reliability. In fact, the rebuild block size is a design parameter with the most significant impact on reliability.

In contrast, drive and node MTTF are not easily controllable. Industry experience has indicated that drive MTTF can vary significantly between batches of drives and the same can be expected of nodes. The numbers we have used in the baseline analysis are conservatively realistic with the sensitivity analysis providing an insight into available headroom from a reliability perspective.

For the specific target we have chosen in this paper, it appears that either the [FT2, Internal RAID 5] or the [FT3, No Internal RAID] configurations meet the reliability requirement with the condition that the rebuild block size is at least 64 KB.

9. Conclusions

We have developed effective reliability models for networked storage nodes based on Markov chains. We deal with the complexity of solving large Markov models in two different ways – hierarchical models and recursive models. Using these methods, we are able to generate closed-form parametric solutions that have broad utility. We have chosen a specific reliability target in order to focus on a few redundancy configurations. However, the closed-form solutions we have presented may be used to determine redundancy configurations for a spectrum of reliability targets such as in systems that offer user-configurable goals.

We have also developed a model that utilizes basic parameters such as disk drive bandwidth and network link speed, to generate effective rebuild rates. System reliability, as we have seen, is impacted significantly by the rebuild rate; hence, obtaining a precise estimate using basic parameters ensures that the reliability results are accurate.

Acknowledgements

The authors thank Robert Garner, Deepak Kenchammana-Hosekote, Claudio Fleiner, Winfried Wilcke and Jody Glider for their input and contributions.

We also thank the anonymous reviewers for their helpful suggestions and comments.

References

- [1] M. Blaum, J. Brady, J. Bruck and J. Menon. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures. *IEEE Transaction on Computing*, February 1995, 44(2) 192-202.
- [2] P. Corbett, B. English, A. Goel, T. Gracanac, S. Kleiman, J. Leong and S. Sankar. Row Diagonal Parity for Double Disk Failure Correction. *FAST-2004: 3rd USENIX Conference on File and Storage Technologies*, March 2004.
- [3] C. Fleiner, R.B. Garner, J.L. Hafner, KK Rao, D.R. Kenchammana Hosekote, W. W. Wilcke and J.S. Glider. Reliability of modular mesh-connected intelligent storage brick systems. *IBM Journal of Res. and Dev.*, 50, No. 2/3, 2006, 199-208.
- [4] G.R. Ganger, J.D. Strunk, and A.J. Klosterman. Self-* Storage: Brick-based Storage with Automated Administration. Carnegie Mellon University Technical Report, CMU-CS-03-178, August 2003.
- [5] G.R. Goodson, J.J. Wylie, G.R. Ganger, and M.K. Reiter. Efficient Byzantine-tolerant erasure-coded storage. *Dependable Systems and Networks*, June 2004.
- [6] J.L. Hafner. WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems. *FAST-2005: 4th USENIX Conference on File and Storage Technologies*, December 2005
- [7] C. Huang and L. Xu. Star: An Efficient Coding Scheme for Correcting Triple Storage Node Failures. *FAST-2005: 4th USENIX Conference on File and Storage Technologies*, December 2005
- [8] E.K. Lee, C.A. Thekkath, C. Whitaker, and J. Hogg. A Comparison of Two Distributed Disk Systems. Research Report 155, Digital Systems Research Center, April 1998.
- [9] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale Storage Cluster – Delivering Scalable High Bandwidth Storage. *Proceedings of the 2004 ACM/IEEE Supercomputing Conference*, November 2004.
- [10] J.S. Plank and M.G. Thomason. A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide-Area Storage Applications. *Dependable Systems and Networks*, June 2004.
- [11] K. Rao, J.L. Hafner, and R. Golding. Reliability for Networked Storage Nodes. Technical Report RJ 10358, IBM Research, September 2005.
- [12] Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence. FAB: Building Distributed Enterprise Disk Arrays from Commodity Components. *ASPLOS 2004*, October 2004.
- [13] K.S. Trivedi. Probability and Statistics with Reliability, Queuing, and Computer Science Applications. *Prentice-Hall*, 1982
- [14] W.W. Wilcke et al. IBM Intelligent Bricks project – Petabytes and beyond. *IBM Journal of Res. and Dev.*, 50, No. 2/3, 2006, 181-197.
- [15] Q. Xin, E.L. Miller, T. Schwarz, D.D.E. Long, S.A. Brandt, and W. Litwin. Reliability Mechanisms for Very Large Storage Systems. *IEEE/ NASA Goddard Conference on Mass Storage Systems and Technologies*, April 2003.