Hierarchical RAID: Organization, Operation, Reliability and Performance

Alexander Thomasian and Yujie Tang
Shenzhen Institutes of Advanced Technology - SIAT
Chinese Academy of Sciences
University Town, Xili Nanshan, Shenzhen 518055, China
alexthomasian@gmail.com and yj.tang@siat.ac.cn

Abstract

We consider two level Hierarchical RAID (HRAID) arrays with erasure coding at both levels. The main advantage of HRAID is tolerating disk array controller failures in addition to disk failures, so that it is suitable for storage clouds based on bricks. We consider an HRAID with N nodes consisting of independent RAID controllers and M disks per node. Each node tolerates the failure of ℓ disks and the controllers use a software protocol to coordinate their activities to implement an inter-node RAID level tolerating k node failures. We postulate MDS coding which incurs the minimal level of redundancy at both levels. HRAIDk/ℓ can fail with a minimum of $(k+1)(\ell+1)$ disk failures, regardless of N and M, but the probability of this event is very small. The maximum number of disk failures with no controller failures is $N \times \ell + (M - \ell)k$. We specify examples of HRAID organization, describe distributed transactions to ensure the integrity of updates, and assess the small write penalty in HRAID and compare it with single level RAID. We report Monte-Carlo simulation results to obtain the Mean Time to Data Loss (MTTDL) assuming repeated successful restripings via parity sparing until such space is exhausted. For fixed $k \times \ell$, an approximate reliability analysis, which does not take into account controller failures, shows that $k < \ell$ yields the higher reliability, but simulation results show that the opposite is true for less reliable controllers.

1. Introduction

The RAID paradigm based on mirroring and erasure coding was developed to deal with disk storage unreliability. The five level RAID classification was later extended in [4] to seven levels, most notably RAID level 6 to tolerate two disk failures. We are interested in erasure coding rather than mirroring, since it introduces a lower level of storage redundancy and reduces power consumption mainly attributable to disk rotations.

RAID5 utilizes one parity disk per $N > 1$ disks to cope with single disk failures or unreadable disk blocks due to *Latent Sector Errors (LSEs)* [17]. These are masked by accessing surviving disks to reconstruct missing data. The read load at the surviving disks is doubled when a disk fails, but this load increase can be reduced by adopting the clustered RAID paradigm [11].

The RAID5 rebuild process systematically reconstructs the contents of a failed disk on a spare disk to return the system to normal mode operation, provided a spare disk or spare areas are available for this purpose. Rebuild in RAID5 may be unsuccessful due to a second disk failure, before it is completed. LSEs are a more likely reason for unsuccessful rebuilds. This has led to *2-Disk-Failure-Tolerant (2DFT)* arrays, such as StorageTek's Iceberg with the *Reed-Solomon (RS)* coding, which served as the basis of the RAID6 classification. EVENODD [3], *Row-Diagonal Parity (RDP)* [5], and X-code [25] are RAID6 arrays extending RAID5 by using two parities instead of one, thus reducing the high computational cost associated with RS coding. All these codes are *Maximum Distance Separable (MDS)*, i.e., incur the minimum level of redundancy of ℓ disks to tolerate ℓ disk failures [22]. 3DFT arrays referred to as RAID7, are possible with RS coding, and have been proposed as extensions of EVENODD and RDP [22].

Disk scrubbing and intra-disk redundancy are two additional methods to deal with LSEs, which can be used in conjunction with all RAID levels [8, 17]. We postulate that rebuild is always successful, but we remain silent on the issue of silent errors. Instead of spare disks or spare areas, we utilize parity sparing [12], to hold the contents of failed disks and nodes. Distributed sparing evaluated in [19] is preferable to parity sparing when a higher reliability is not required, because it does not incur the overhead associated

with the small write penalty. A RAID7 with P, Q, R check blocks can reconfigure onto a RAID6 with P and Q check blocks, RAID6 reconfigures onto RAID5 with parity P, and RAID5 onto RAID0.

There has been interest in the overall reliability of storage systems including disk array controllers in the context of storage bricks. This issue was addressed a decade ago in [2] in the context of *Hierarchical RAID (HRAID)*, with lower level RAID controllers for disks connected to higher level RAID controllers. We postulate self-sufficient RAID nodes, where controllers in addition to controlling disks implement a higher level RAID functionality by coordinating their activities.

HRAIDk/ℓ with MDS coding tolerates $0 \leq k \leq 3$ failures at the node level and $0 \leq \ell \leq 3$ failures at the disk level. There is no data loss as long as there are not more than k node failures. We postulate in Section 6 that more than ℓ disk failures at a node also amount to a node failure. Strictly speaking, in the case of HRAID1/1 data recovery is possible with two failed disks at two nodes, as long as the coordinates of the four disks do not constitute a rectangle.

It was shown in [2] that the highest storage efficiency for an HRAID1/1 array is attained for $N = M$ [2]: $u = (N-1)^2/N^2$ or 81% for $N = M = 10$. More generally, the storage efficiency for HRAIDk/ℓ with MDS coding is: $u = (N-k)(M-\ell)/(NM) = 1 - k/N - \ell/M + (k\ell)/(NM)$, which is lower than that of RAID($k + \ell$): $u = 1 - (k+\ell)/(NM)$. This is simply due to the fact that inter-node codes are repeated at all nodes, so that with MDS coding at both levels HRAID is not MDS overall. HRAID1/1 tolerates all three, but in practice many more disk failures, while RAID6 with the same number of check blocks tolerates at most two disk failures. As specified in Section 2, in addition to the data block, which may have to be read unless its cached to compute the difference block, three check blocks need to be updated in HRAID1/1, but this also provides protection against node failures.

Expressions for the *Mean Time To Data Loss (MTTDL)* for RAID5/6/7 disk arrays are derived in [2] with exponential failure and service rates. This model has been extended to take into account sector errors in [15], which also develops a hierarchical reliability model for an HRAID-like system. In addition to exponential failure rates for disks, it is postulated that the distribution of time to data loss at the nodes is also exponentially distributed (see Section 4.2 in [22]). No validation results are provided for the approximations involved.

The failure of a controller is tantamount to a node failure, including all of the disks at the node, since they become inaccessible. Providing multiple paths to disks as in [13] does not comply with the assumptions of self-contained nodes, but internal multipathing and duplexing of controller is a feasible solution. We assume that the higher level RAID functionality remains intact as long as the number of failed controller does not exceed the limit k set by the higher level RAID paradigm.

Relative frequency of hardware component replacements for the most frequently replaced components is given in Table III in [16]. For 3-out-of-10 systems considered in that paper, hard drive replacements are 30.6%, 34.8%, and 49.1% for HPC1, COM1, and COM2, respectively. Power supply and memory failures dominate in COM1 at 34.8%, and 20.1%, respectively. Table II in [16] summarizes the causes for failure as: CPU 44%, memory 29%, hard drive 16%, PCI motherboard 9%, and power supply 2%, i.e., the combined effect of non-disk failures exceeds disk failures 6-fold. A major advantage of HRAID is that it can tolerate controller failures, which are more catastrophic than disk failures, since each controller failure results in the loss of data at all the disks attached to it.

We use Monte Carlo simulation to assess HRAIDk/ℓ reliability taking into account both controller and disk failures. Simulation results show that $k < \ell$ is preferable, when disk failure rates are lower than controller failure rates, but the reverse is true otherwise.

We postulate one or more routers external to HRAID, which send data access requests to the nodes. Data updates, even those of individual blocks, need to be processed as *transactions (txns)*, to ensure correctness of updates and data integrity. *Concurrency Control (CC)* in a highly concurrent storage system was also considered in [1]. We utilize distributed txns with *Two-Phase Commit (2PC)* for the same purpose [14]. Only locking at the level of disk blocks is considered for the sake of brevity, while locking ranges of sectors would be generally required [14]. CC methods for distributed storage systems are discussed in [9].

Tandem's approach of using computer pairs cross-connected to mirrored disks is generalized in [13], which uses duplexed controllers and a crosshatch interconnect structure for RAID1. MTTDL is the metric used for comparing different configurations in this study. There have been numerous studies of interconnection network reliability, which are beyond the scope of this discussion. HRAID is pertinent to the storage bricks paradigm, where each brick consists of a storage controller, a cache, and a number of disks [24]. A study in the context of IBM's Icecube project estimates the probability of storage bricks becoming inaccessible as failed bricks are not repaired [6]. In this study, we only consider the failure of disks and RAID controllers, which lead to the failure of the entire node.

Simulation results for HRAID1/1 in [2], which can tolerate all three disk failures, shows that the probability of data loss due to four simultaneous disk failures is lower for $N > M$. The probability that the number of failed disks per node exceeds two is decreased when disk failures are distributed over a larger number of nodes. This is confirmed

in our simulation study of three HRAID1/2 configurations with 9×16, 12×12, and 16×9 disks. We report the distribution of number of failed disks leading to data loss for varying node failure rates.

The approximate reliability method developed in [20] is used to provide a general approximate expression for RAIDℓ. We use an approximate reliability analysis method to show that HRAID1/2 is more reliable than HRAID2/1.

This paper is organized as follows. In Section 2, we specify the organization of HRAID and describe erasure coding at intra-disk and inter-node level. A brief discussion of HRAID performance appears in Section 3. Distributed transactions for CC are discussed in Section 4. In Section 5 we obtain approximate expressions for RAID and HRAID reliability. A Monte-Carlo simulation to obtain the MTTDL is described and simulation results are given in Section 6. In Section **??** we summarize the conclusions of this study. In the Appendix we provide the reliabilities of several HRAID configurations using the approximate method.

2. Coding Across the Nodes

An additional level of data protection can be introduced by replication or erasure coding across storage nodes, which adhere to RAID. We adopted erasure coding to save disk space as noted earlier. Redundancy at the higher level is necessary for tolerating node, as well as disk failures, since the failure rate for non-disk hardware at the nodes may be higher than disk failures rates.

Across the N nodes the RAID paradigm applied is k-*Failure-Tolerant (kFT)* while the RAID level for the M disks at each node is ℓDFT. We restrict our discussion to HRAIDs with $M = N$, since it simplifies the check strip layout. Disk load imbalance for $N \neq M$ can be overcome by an appropriate permutation of stripes, so that all nodes are equally populated with check strips. We adopt two horizontal codes in our discussion, while a combination of vertical and horizontal codes are allowed in Grid files [10]. The vertical X-code with horizontally placed check strips is restricted to a prime number of disks and hence of little interest [22]. There is also the issue of load imbalance when disk failures occur in X-code [23].

We postulate MDS codes which protect against k node and ℓ disk failures (per node). When a controller fails, all the disks attached to it become inaccessible, unless there are multiple connections to the disks as in [13], but this option is not relevant to this study.

Dedicating a disk to parity in RAID4 has the following disadvantages [4]: (i) The check disk is not accessed at all by read requests. (ii) In a write intensive environment with updates of small blocks, the check nodes may become a bottleneck. These problems are resolved in RAID5 by adopting a left-symmetric organization, i.e., the parity strips are placed in repeating right-to-left diagonals [4]. Similarly to RAID4, instead of designating one node out of N nodes as a check node, the associated inefficiency is alleviated by distributing the check strips at that node across all nodes. The same concept is applicable to more than one check strip for inter-node protection.

Data may be striped at the level of individual nodes or across all disks crossing node boundaries. The latter was shown to be detrimental to performance according for variable bit rate continuous media file servers in [18], but is postulated in this study, since it is expected to balance load across nodes. Our allocation is not reflected in the numbering of strips in the following examples. We distribute P parities across all disks and Q parities across nodes. Q parities are protected by P parities, but not vice-versa. There is a similarity to RDP, where one set of check blocks is protected by the other set [5]. Q parities apply at the inter-node level to data only.

Example 1: An HRAID1/1 with four nodes and disks: $N = M = 4$ is shown in Fig. 1. $D^n_{i,j}$ is used for data strips and $P^n_{i,j}$ and $Q^n_{i,j}$ denote local and global check strips, respectively. The superscript $1 \leq n \leq N$ denotes the node number, i is the row number, and $1 \leq j \leq M$ represents the disk number. The P and Q are in fact parities, which are rotated from disk to disk and from node to node to balance the load for check block updates. Only the first four rows are shown because the pattern repeats itself.

To update $d^2_{4,1}$, a disk block in strip $D^2_{4,1}$, with $d^{2new}_{4,1}$ we have the following steps (we use $R(.)$ and $W(.)$ to denote the writing of a block).
(1) Read old block $R(d^{2old}_{4,1})$, compute difference block, and write new block:
$d^{2diff}_{4,1} = d^{2new}_{4,1} \oplus d^{2old}_{4,1}$, $W(d^{2new}_{4,1})$;
(2) Computing new check blocks and write them:
$p^{2new}_{4,3} = p^{2old}_{4,3} \oplus d^{2diff}_{4,1}$, $W(p^{2new}_{4,3})$;
$q^{1new}_{4,1} = q^{1old}_{4,1} \oplus d^{2diff}_{4,1}$, $W(q^{1new}_{4,1})$;
$p^{1new}_{4,4} = p^{1old}_{4,4} \oplus q^{1diff}_{4,1}$, $W(p^{1new}_{4,4})$.

Disk failures can be dealt with firstly inside a node, but otherwise using the coding scheme across nodes, which is slower and more expensive, but the only choice when a node fails. In the case of a node failure, data strips are reconstructed by accessing corresponding data and Q strips at other nodes. The P parities at a node are reconstructed last using Q parities and data strips. As an example, we assume Node 1 has failed and reconstruct its first stripe. $D^1_{1,1} = D^2_{1,1} \oplus Q^4_{1,1}$, $D^1_{1,2} = Q^3_{1,2} \oplus D^4_{1,2}$, $Q^1_{1,4} = D^2_{1,4} \oplus D^3_{1,4}$, $P^1_{1,3} = D^1_{1,1} \oplus D^1_{1,2} \oplus Q^1_{1,4}$.

Just in case Disk 1 at Node 2 has failed, it has to be reconstructed first: $D^2_{1,1} = P^2_{1,2} \oplus Q^2_{1,3} \oplus D^2_{1,4}$. This computation is invoked so as to make it possible to compute $D^1_{1,1}$. In fact, it is best to interleave rebuild at the disk level with rebuild at the node level, because of the caching benefit.

We postulate that a node fails with the failure of the

Node 1				Node 2				Node 3				Node 4			
$D^1_{1,1}$	$D^1_{1,2}$	$P^1_{1,3}$	$Q^1_{1,4}$	$D^2_{1,1}$	$P^2_{1,2}$	$Q^2_{1,3}$	$D^2_{1,4}$	$P^3_{1,1}$	$Q^3_{1,2}$	$D^3_{1,3}$	$D^3_{1,4}$	$Q^4_{1,1}$	$D^4_{1,2}$	$D^4_{1,3}$	$P^4_{1,4}$
$D^1_{2,1}$	$P^1_{2,2}$	$Q^1_{2,3}$	$D^1_{2,4}$	$P^2_{2,1}$	$Q^2_{2,2}$	$D^2_{2,3}$	$D^2_{2,4}$	$Q^3_{2,1}$	$D^3_{2,2}$	$D^3_{2,3}$	$P^3_{2,4}$	$D^4_{2,1}$	$D^4_{2,2}$	$P^4_{2,3}$	$Q^4_{2,4}$
$P^1_{3,1}$	$Q^1_{3,2}$	$D^1_{3,3}$	$D^1_{3,4}$	$Q^2_{3,1}$	$D^2_{3,2}$	$D^2_{3,3}$	$P^2_{3,4}$	$D^3_{3,1}$	$D^3_{3,2}$	$P^3_{3,3}$	$Q^3_{3,4}$	$D^4_{3,1}$	$P^4_{3,2}$	$Q^4_{3,3}$	$D^4_{3,4}$
$Q^1_{4,1}$	$D^1_{4,2}$	$D^1_{4,3}$	$P^1_{4,4}$	$D^2_{4,1}$	$D^2_{4,2}$	$P^2_{4,3}$	$Q^2_{4,4}$	$D^3_{4,1}$	$P^3_{4,2}$	$Q^3_{4,3}$	$D^3_{4,4}$	$P^4_{4,1}$	$Q^4_{4,2}$	$D^4_{4,3}$	$D^4_{4,4}$

Figure 1. A 4 by 4 HRAID1/1 with $N = 4$ nodes and $M = 4$ disks per node.

$\ell + 1^{st}$ disk st the node, even though the controller has not failed.

Example 2: There are $N = 5$ nodes and $M = 5$ disks per node. We have the RAID6 paradigm across the nodes with $k = 2$ and each node is a RAID5 with $\ell = 1$. We use P to denote the intra-node RAID5 parity and Q and R for inter-node RS code. Only the first row is shown for brevity.
$(D^1_{1,1}, D^1_{1,2}, P^1_{1,3}, Q^1_{1,4}, R^1_{1,5})$;
$(D^2_{1,1}, P^2_{1,2}, Q^2_{1,3}, R^2_{1,4}, D^2_{1,5})$;
$(P^3_{1,1}, Q^3_{1,2}, R^3_{1,3}, D^3_{1,4}, D^3_{1,5})$;
$(Q^4_{1,1}, R^4_{1,2}, D^4_{1,3}, D^4_{1,4}, P^4_{1,5})$,
$(R^5_{1,1}, D^5_{1,2}, D^5_{1,3}, P^5_{1,4}, Q^5_{1,5})$.

If node 5 fails reconstruct using R strips.
$(D^1_{1,1}, D^1_{1,2}, P^1_{1,3}, Q^1_{1,4}, D^5_{1,2})$,
$(D^2_{1,1}, P^2_{1,2}, Q^2_{1,3}, D^5_{1,3}, D^2_{1,5})$,
$(P^3_{1,1}, Q^3_{1,2}, -, D^3_{1,4}, D^3_{1,5})$,
$(Q^4_{1,1}, -, D^4_{1,3}, D^4_{1,4}, P^5_{1,5})$.
If nodes 4 and 5 fail use both Q and R strips.
$(D^1_{1,1}, D^1_{1,2}, P^1_{1,3}, D^4_{1,3}, D^4_{1,4})$,
$(D^2_{1,1}, P^2_{1,2}, -, D^5_{1,2}, D^2_{1,5})$,
$(P^3_{1,1}, -, D^5_{1,3}, D^3_{1,4}, D^3_{1,5})$.

As strips are relocated the system has a means of referring to them by their original coordinates. There is the issue of using inter-node check blocks for parity sparing to deal with disk failures. Let's assume that two disks had failed at one of the nodes, before any node failures occurred. After applying parity sparing to the P parity we could have restriped using the Q or S strips, but then we would have been unable to deal with internode failures.

The volume of data required for rebuild at the node level requires the transmission of the contents of the surviving $N - k$ nodes to a spare node for reconstruction. The high bandwidth interconnect which is adequate for normal mode processing, may be inadequate during inter-node rebuild processing. The volume of data to be moved for rebuild can be reduced by using the pyramid code [7], which uses two instead of one parity strip, each covering half of the data strips. Depending on the interconnection network, an appropriate method can be developed to minimize the volume of data to be transmitted. For example, the contents of pairs of nodes can be XORed and combined with other pairs. One stripe at a time can be reconstructed to minimize intermediate space requirements. Since the system is not quiesced when rebuild is in progress, attention should be paid to updates targeting data on the failed node. The read redirection method may be applied so that data that has already been materialized is read directly [11].

3. Performance Analysis

We consider the cost of disk storage accesses in processing updates to small data blocks for *Online Transaction Processing (OLTP)* applications, since disk bandwidth constitutes the bottleneck in this case. The updating of each data block requires the updating of corresponding local and global check blocks. The number of inter-node messages for updating global check blocks is determined by k. The updating of each global check block requires the updating of its local check blocks.

Given that disk access bandwidth constitutes a bottleneck, we compute the impact of the HRAID coding scheme assuming that data and check blocks are uniformly distributed. We denote the mean service time for Single Read (SR) and Single Write (SW) accesses by $\overline{x}_{SR}$ and $\overline{x}_{SW}$, respectively. f_R denotes the fraction of reads and $f_W = 1 - f_R$ denotes the fraction of writes. Alternatively, we are given R:W with $f_R = R/(R+W)$ and $f_W = W/(R+W)$.

We have the following maximum I/O rates per disk for RAID0, RAIDℓ, and HRAIDk/ℓ, respectively:
$\lambda_0 = [f_R\overline{x}_{SR} + f_W\overline{x}_{SW}]^{-1}$.
$\lambda_\ell = \{f_R\overline{x}_{SR} + f_W[(\ell+1)(\overline{x}_{SR} + \overline{x}_{SW})]\}^{-1}$.
$\lambda_{k/\ell} = \{f_R\overline{x}_{SR} + f_W[(k+1)(\ell+1)(\overline{x}_{SR} + \overline{x}_{SW})]\}^{-1}$.

For HRAID k/ℓ the updating of a data block requires $(\ell+1)$ SR accesses to read data and local check blocks and $(\ell+1)$ SWs to write them. Also $k(\ell+1)$ SRs are required to read the corresponding inter-node check blocks and $k(\ell+1)$ SWs to write them.

To simplify the discussion we assume that disk head settling time is small so that $\overline{x}_{SW} \approx \overline{x}_{SR}$. The relative throughputs for RAID0, RAIDℓ and HRAIDk/ℓ are:
$1 : [1 + f_W(1+2\ell)]^{-1} : [1 + f_W + 2f_W(\ell k + \ell + k)]^{-1}$.

Earlier studies of disk I/O traces indicated that read requests predominate in OLTP workloads, such that $f_R = 0.8$ and $f_W = 0.2$ or R:W=4:1, but with the advent of very large read caches $f_R = f_W = 0.5$ or R:W=1:1 is more plausible, The maximum throughput drops to 0.42 for RAID6 and to 0.31 for HRAID1/2 for $f_W = 0.2$. The drop is more drastic for $f_W = 0.5$: 0.40 for RAID6 and 0.15 for HRAID1/2.

The above discussion has ignored the cost of storage transactions for acquiring locks and 2PC, which are dis-

cussed in Section 4. Lock requests and commit related messages are logged to NVS, so that no disk accesses are required. The analysis can be extended to degraded mode operation due to disk and node failures and restriped systems after the completion of the rebuild process using paring sparing.

4. Storage Transactions

Race conditions arise when the updating of two data blocks, such as $d^1_{1,1}$ and $d^1_{1,2}$ in Example 1 affect the same parity block $p^1_{1,3}$. Unless *Concurrency Control (CC)* is applied the outcome of two simultaneous updates may be: $p^{1new}_{1,3} = d^1_{1,1} \oplus p^{1old}_{1,3}$ or $p^{1new}_{1,3} = d^1_{1,2} \oplus p^{1old}_{1,3}$, but not $p^{1new}_{1,3} = d^1_{1,1} \oplus d^1_{1,2} \oplus p^{1old}_{1,3}$. CC in this case can be handled easily by the local controller.

A more complex situation arises in updating nonlocal Q check blocks, which are affected by updates at other nodes. A distributed *transaction (txn)* is required to ensure the atomicity of updates [14]. These txns are initiated by frontends which route requests to the nodes, which also acts as a commit coordinator. The identity of all the blocks required by a txn to update data blocks is known a priori, so that all required locks are predeclared. The front-end handling the updating of a data block issue lock requests as part of I/O requests to all involved nodes. Low cost versions of 2PC may be used, because we postulate a highly reliable interconnection network. A large variety of network RAID protocols are otherwise available [9]. It is worthwhile to mention that higher level CC protocols are still required for OLTP [14].

Locks are held in NVS and lock handling does not involve disk accesses, the same applies for logging data written for 2PC, so that this overhead was not taken considered in the simplified performance analysis. There is an implicit shared (S) lock requests associated with read requests and an exclusive (X) lock associated with write requests. In what follows we use $R(.)$ and $W(.)$ to denote the reading and writing of a block. The txn to update $d^{1new}_{1,1}$ is as follows:
(1)- $R(d^{1old}_{1,1})$, $R(p^{1old}_{1,2})$, $R(q^{4old}_{1,1})$, $R(p^{4old}_{1,4})$;
(2)- $d^{1diff}_{1,1} = d^{1old}_{1,1} \oplus d^{1new}_{1,1}$;
(3)- $W(d^{1new}_{1,1})$,
(4)- $p^{1new}_{1,3} = d^{1diff}_{1,1} \oplus p^{1old}_{1,3}$, $q^{4new}_{1,1} = q^{4old}_{1,1} \oplus d^{1diff}_{1,1}$, $p^{4new}_{1,4} = p^{4old}_{1,4} \oplus q^{4diff}_{1,1}$;
(5)-$W(p^{1neq}_{1,3})$,$W(q^{4new}_{1,1})$,$W(p^{4new}_{1,4})$.

If $d^{1old}_{1,1}$ was not cached and Disk 1 has failed then the txn is canceled and a new txn is issued, which involves reads to reconstruct the missing block, i.e., $d^{1old}_{1,1} = d^1_{1,2} \oplus p^1_{1,3} \oplus q^1_{1,4}$ In the case of Node 1 failure $d^{1old}_{1,1}$ can be reconstructed by using Q check blocks as $d^{1old}_{1,1} = d^2_{1,1} \oplus q^4_{1,1}$.

5. APPROXIMATE RAID and HRAID RELIABILITY

Let $r = 1 - \epsilon$ denote the reliability of each disk, where $\epsilon \ll 1$ is the disk unreliability. We use the method in [21] to obtain approximate reliability expressions for RAID($\ell+4$), $1 \le \ell \le 3$ given N disks: For RAID5, which is a 1DFT, we have:

$$R_1 = r^N + N(1-r)r^{N-1} = (1-\epsilon)^N + N\epsilon(1-\epsilon)^{N-1}$$
$$\approx 1 - \binom{N}{2}\epsilon^2 + 2\binom{N}{3}\epsilon^3 - \dots$$

It is shown in [21] that the smallest power n of ϵ^n with a nonzero coefficient determines the maximum number of disk failures. The formulas for ℓDFTs with $\ell \ge 1$ are as follows:

$$R_\ell \approx 1 - \binom{N}{\ell+1}\epsilon^{\ell+1} + (\ell+1)\binom{N}{\ell+2}\epsilon^{\ell+2} - \dots$$

Note that we are subtracting the probability of $\ell+1$ disk failures which would lead to data loss.

We next obtain the reliability of HRAID1/2 and HRAID2/1 with N nodes at the higher level and M disks at the lower level, so that the total number of disks is $N \times M$ in both cases:

$$R_{1/2} = R_2^N + N(1-R_2)R_2^{N-1}$$
$$\approx 1 - \frac{N(N-1)M^2(M-1)^2(M-2)^2}{72}\epsilon^6 + \dots$$

It follows that HRAID1/2 tolerates all five disk failures, but may encounter a data loss with the sixth disk failure, although the possibility of this event is very small. Consider the configuration with one failed RAID6 due to three failed disks and another RAID6 with two failed disks, so that another disk failure at that node will lead to data loss. The probability of this event is $p_{1/2} = (M-2)/[(N-2)M + M - 2]$. In the case of HRAID2/1 all five disk failures can be tolerated as well. Consider two RAID5s with two failed disks, so that their data can be reconstructed by the higher level RAID paradigm. Data loss occurs when a third RAID5 array, which has a failed disk encounters a second disk failure. The probability of this event is $p_{2/1} = (M-1)/[(N-3)M + M - 1] > p_{1,2}$, This is shown by the analysis below.

$$R_{2/1} = R_1^N + N(1-R_1)R_1^{N-1} + \binom{N}{2}(1-R_1)^2R_1^{N-2}$$
$$\approx 1 - \frac{N(N-1)(N-2)M^3(M-1)^3}{24}\epsilon^6 + \dots$$

It is observed that from the disk reliability viewpoint $R_{1/2} > R_{2/1}$, since: $N > 2 + 2(M-2)^2/(3M(M-1))$.

Regardless of N and M with perfectly reliable controllers HRAIDk/ℓ can tolerate all possible $(k+1) \times (\ell + 1) - 1$ disk failures. In the case of HRAID1/1 four disk failures cannot be tolerated if their coordinates constitute a rectangle, but of course the probability of this event is very small.

HRAIDk/ℓ can tolerate at most $D_{max} = N \times \ell + (M - \ell)k$ disk failures without data loss. Each one of N nodes can tolerate ℓ disk failures by using its intra-node check code, but additionally the remaining $M - \ell$ nodes can tolerate k disk failures via inter-node redundancy. This is provided we allow internode data recovery at the disks, but this option is not considered in this study. For $M = N$ $D_{max} = N(k + \ell) - k\ell$ and given that $D_{red} = N(k+\ell)$ disks are redundant, we have $D_{max}/D_{red} = 1 - k\ell/(N(k + \ell)$, which is close to one, but then D_{max} is rarely attained.

6. Simulation Results

We simulate the configurations HRAIDk/ℓ for $0 \leq k, \ell \leq 3$. The parity sparing approach for rebuild is followed so that check strips are used to hold data blocks after a failure, so that there is a progression from $k/\ell \rightarrow k/(\ell-1)$ for $\ell \geq 1$ or $k/\ell \rightarrow (k-1)/\ell$ for $k \geq 1$, depending on whether there is a disk or node failure. Rebuild has the advantage HRAID performance is not degraded and in fact the overhead associated with small write penalty decreases with extra disk failures, since fewer check blocks need to be updated. Rebuild is no longer possible when the check strips are exhausted. As noted earlier if the number of failed disks exceeds ℓ in HRAIDk/ℓ, then the associated node is designated as failed, even though its controller has not failed and its nonfailed disks are accessible. This pessimistic assumption is justifiable in that not all the disks at the node can be used for reconstruction. A node may fail directly due to a controller failure. In both cases all nonfailed disks at the node are designated as failed.

Disk failure rates are exponentially distributed, which was shown to be a good approximation in an earlier study, although it has been shown that the Wieball distribution better represents this distribution [16]. We also assume that controllers also fail according to the negative exponential distribution. We are interested in relative rather than absolute reliabilities. The assumption that failures rates are exponentially distributed reduces the complexity of the simulator and the effort to generate the time to the next event. We have one event at a time, rather than a large number of pending events. We generate one failure at a time using the sum of failures rates by taking advantage of the aggregation property of the exponential distribution, but then determine the identity of the failed component probabilistically.

We use the batch means method to obtain MTTDLs with a confidence interval within 5% of the mean with a 95% confidence level. The MTTDL is obtained for $I_1 = 1000$ iterations, repeated $I_2 = 10$ times. We also obtain the percentage of cases data loss due to disk or controller failures and the mean number of failed disks and controllers over the 10,000 runs.

Procedure: Time to Data Loss

Inputs:

k: Fault tolerance at the level of nodes.
ℓ: Fault tolerance RAID(ℓ+4).
N: The number of controllers (or nodes).
M: The number of disks per node.
D: Total number of disks $D = N \times M$.
δ: Disk failure rate (one million hours)
$\gamma = \delta \times (0, 1/2, 1, 2)$ with exponential distribution.

Outputs (with initializations)

$Clock = 0$ simulation clock.
$F_{cd} = 0/1$ data loss due to controller/disk failure.
N_c: Number of failed controllers ($N_c = 0$).
N_d: Number of failed disks, including those made unavailable due to controller failures or node failures when the number of failed disks at a node exceeds ℓ ($N_d = 0$).
N_{df}: Number of failed disks due to disk failure ($N_{df} = 0$).

Simulation Variables:

Two states: 0 for failed and 1 for operational.
$C[0 : N-1]$ the state controllers ($C[n] = 1, \forall n$).
$D[0 : N-1, 0 : M-1]$ the state of $N \times M$ disks ($D[n, m] = 1, \forall n, \forall m$).
N_n : Number of failed nodes ($N_n = 0$).
$F[0 : N-1]$ the number of failed disks at node n ($F[n] = 0, \forall n$).
Instances of pseudo random-variables in $(0, 1)$ are denoted by u_i.

We report MTTDLs in thousands of hours, the mean number of failed controllers and disks at the time of data lose in different configurations with different controller failure rates (γ) with respect to the disk failure rates (δ), which is always set to one million hours.

The following conclusions can be drawn from the simulation results:

- Higher values of k and ℓ are required to achieve a dramatic increase in MTTDL. Comparing the MTTDL for $k = \ell = 0$ with $k = \ell = 3$ there is a 40-fold increase for $\gamma = 0$, 30-fold for $\gamma = 1$, and 25-fold for $\gamma = 2$. [1]

[1] Although controllers do not fail when $\gamma = 0$, a node is considered to be failed when the number of failed disks at the node exceeds ℓ in this study.

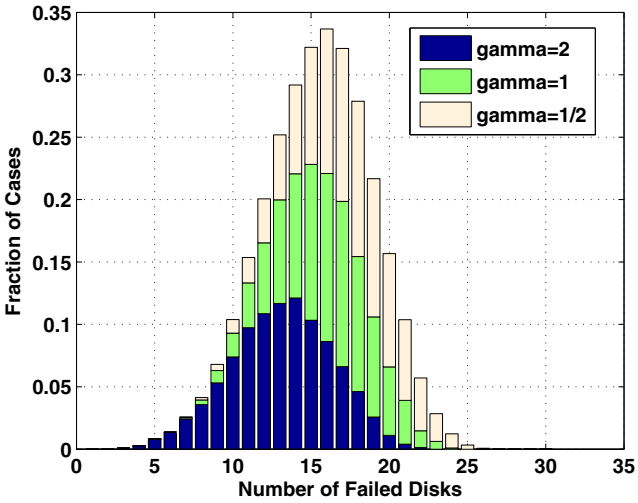

Figure 2. The distribution of the number of failed disks for varying values of γ for HRAID2/2.

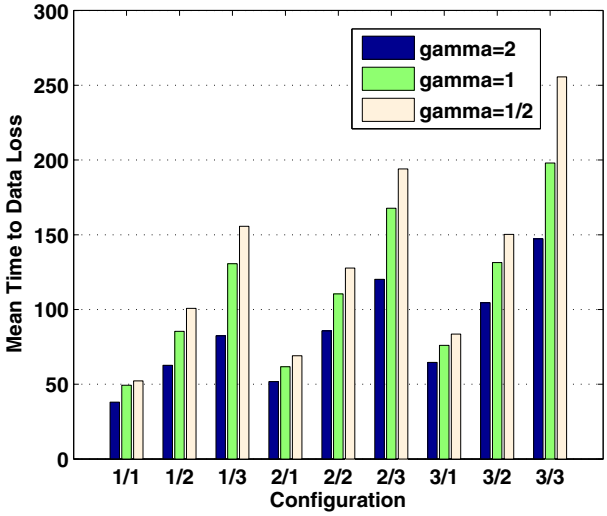

Figure 3. The MTTDL for different HRAID configurations in thousands of hours.

- For the most reliable configuration considered in this study ($k = \ell = 3$) and for $\gamma = 0, 1, 2$ the MTTDL decreases as 288, 198, and 147.2 in thousands of hours.
- When γ is larger the number of controller failures at the time of data loss is larger, but the number of failed disks is smaller.
- For $0 \leq \gamma \leq 2$ HRAIDk/ℓ is better for $\ell > k$, but there is a crossover point for $3 \leq \gamma \leq 4$ (not reported here).
- The fraction of cases that data loss occurs due to controller failures does not increase for higher values of γ for larger values of k and ℓ.
- The fraction of cases data loss is due to controller failure becomes larger with increasing ℓ when k is smaller.

Fig. 2 provides the distribution of the number of failed disks when data loss occurs. The number of failed disks increases when controllers are more reliable, since otherwise the system fails due to controller failures. The MTTDLs are shown in Fig. 3. We observe that the MTTDL increases as k and ℓ increase. In addition, the MTTDL becomes smaller with increasing values of γ. As shown in Fig. 4 that disk failures dominate in most cases, but this is less so for $\gamma = 2$ for HRAID1/3 for example.

The simulator described in this section can be simplified to model disk failures while ignoring time. Given that controller failures rate are negligibly small, the MTTDL can be estimated using the mean number of disk failures to data loss (say $\bar{n}$).

$$MTTDL = \sum_{i=1}^{\bar{n}} [(D-i)\delta]^{-1}$$

References

[1] K. Amiri, G.A. Gibson, and R.A. Golding, "Highly Concurrent Shared Storage", *Proc. 20th Int'l Conf. on Distributed Computing Systems (ICDCS'00)*, Taipei, Taiwan, April 2000, 298-307.

[2] S. H. Baek, W. Kim. E. J. Joung, and C. W. Park, "Reliability and Performance of Hierarchical RAID with Multiple Controllers", *Proc. 20th ACM Symp. on Principles of Distributed Computing (PODC'01)*, Newport, RI, Aug. 2001, pp. 246-254.

[3] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures", *IEEE Trans. on Computers 44*(2): 192-202 (Feb. 1995)

[4] P. M. Chen, E. L. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-Performance, Reliable Secondary Storage", *ACM Computing Surveys 26*(2): 145-185 (June 1994).

[5] P. F. Corbett, R. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction", *Proc. USENIX Conf. on File*

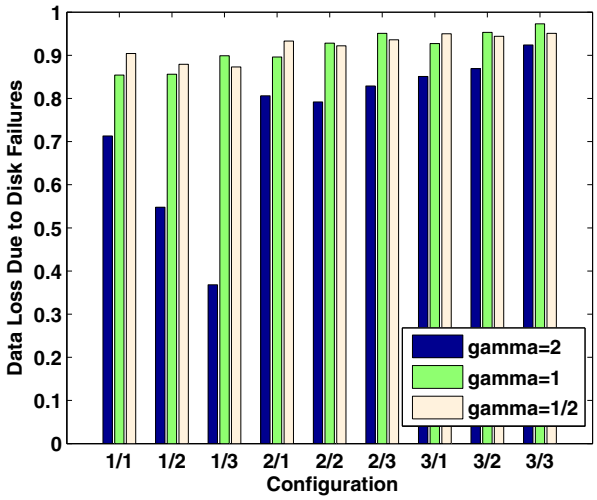

Figure 4. The percentage of cases data loss due to disk failures.

and Storage Technologies (FAST'04), San Francisco, CA, March-April 2004, 1-14

[6] C. Fleiner et al. "Reliability of Modular Mesh-Connected Intelligent Storage Brick Systems", *IBM J. Research and Development 50*(2-3): 199-208 (2006).

[7] C. Huang, M. Chen, and J. Li, "Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems", *Proc. 6th IEEE Int'l Symp. on Network Computing and Applications (NCA 2007)*, Cambridge, MA, July 2007, 79-86.

[8] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk Scrubbing versus Intra-Disk Redundancy for High-Reliability Raid Storage Systems", *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Annapolis, MD, June 2008, pp. 241-252.

[9] D. R. Kenchammana-Hosekote, R.A. Golding, C. Fleiner, and O. Zaki, "The Design and Evaluation of Network RAID Protocols", *IBM Research Report RJ 10316*, Almaden Research Center, CA, March 2004.

[10] M. Li, J. Shu, and W. Zheng, "GRID Codes: Strip-Based Erasure Codes with High Fault Tolerance for Storage Systems", *ACM Trans. on Storage (TOS) 4*(4): Article 15 (2009).

[11] R. R. Muntz and J. C.-S. Lui, "Performance Analysis of Disk Arrays under Failure", *Proc. 16th Int'l Conf. on Very Large Data Bases (VLDB'90)*, Brisbane, Queensland, Australia, Aug. 1990, pp. 162-173.

[12] A. L. Narasimha Reddy, J. A. Chandy, and P. Banerjee, "Design and Evaluation of Gracefully Degradable Disk Arrays", *J. Parallel and Distributed Computing (JPDC) 17*(1-2): 28-40 (1993).

[13] S.W. Ng, "Crosshatch Disk Array for Improved Reliability and Performance," *Proc. 21st Ann'l Int'l Symp. on Computer Architecture (ISCA'94)*, Chicago, IL, April 1994, pp. 255-264.

[14] R. Ramakrishnan and J. Gehrke, *Database Management Systems, 3rd ed.*, McGraw-Hill 2003.

[15] K.K Rao, J.L. Hafner, and R.A. Golding, "Reliability for Networked Storage Nodes", *Proc. Int'l Conf. on Dependable Systems and Networks (DSN 2006)*, Philadelphia, PA, June 2006, 237-248.

[16] B. Schroeder and G.A. Gibson, "Understanding Disk Failure Rates: What Does an MTTF of 1,000,000 Hours Mean to You?" *ACM Trans. on Storage 3*(3): Article 8 (2007)

[17] B. Schroeder, S. Damouras, and P. Gill. "Understanding Latent Sector Errors and How to Protect Against Them", *ACM Trans. on Storage (TOS) 6*(3): Article 8 (2010).

[18] P.J. Shenoy and H.M. Vin, "Efficient Striping Techniques for Variable Bit Rate Continuous Media File Servers", *Performance Evaluation 38*(3-4): 175-199 (1999).

[19] A. Thomasian and J. Menon, "RAID5 Performance with Distributed Sparing", *IEEE Trans. Parallel Distributed Systems 8*(6): 640-657 (1997).

[20] A. Thomasian, "Multi-level RAID for Very Large Disk Arrays"; *ACM SIGMETRICS Performance Evaluation Review 33*(4): 17-22 (2006).

[21] A. Thomasian, "Shortcut Method for Reliability Comparisons in RAID," *Journal of Systems and Software (JSS) 79*(11): 1599-1605 (2006).

[22] A. Thomasian and M. Blaum, "Higher Reliability Redundant Disk Arrays: Organization, Operation, and Coding", *ACM Trans. on Storage (TOS) 5*(3): Article 7 (2009).

[23] A. Thomasian and J. Xu, "X-code Double Parity Array Operation with Two Disk Failures", *Information Processing Letters 111* 568-574 (2011).

[24] W.A. Wilcke et al. "IBM Intelligent Bricks Project - Petabytes and Beyond", *IBM J. Research and Development 50*(2/3): 181-198 (2006).

[25] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding", *IEEE Trans. Information Theory 45*(1): 272-276 (1999).