A flexible simulation tool for estimating data loss risks in storage arrays

Conference Paper · May 2013

DOI: 10.1109/MSST.2013.6558435

CITATIONS
9

READS
82

4 authors, including:

Jehan-Francois Paris
University of Houston
165 PUBLICATIONS **2,570** CITATIONS
SEE PROFILE

Thomas J. E. Schwarz
Marquette University
152 PUBLICATIONS **1,912** CITATIONS
SEE PROFILE

Darrell D. E. Long
University of California, Santa Cruz
316 PUBLICATIONS **9,286** CITATIONS
SEE PROFILE

A Flexible Simulation Tool for Estimating Data Loss Risks in Storage Arrays

Hsu-Wan Kao, Jehan-François Pâris
Computer Science Dept.
University of Houston
Houston, TX 77204-3010
shewankao@gmail.com, jfparis@uh.edu

Thomas Schwarz, S. J.
Depto. de Informática y
Ciencias de la Computación
U. Católica del Uruguay
11600 Montevideo, Uruguay
tschwarz@ucu.edu.uy

Darrell D. E. Long[1]
Computer Science Dept.
University of California
Santa Cruz, CA 95064
darrell@cs.ucsc.edu

***Abstract*—Proteus is an open-source simulation program that can predict the risk of data loss in many disk array configurations, among which, mirrored disks, all levels of RAID arrays and various two-dimensional RAID arrays. It characterizes each array by five numbers, namely, the size n of the array, the number n_f of simultaneous disk failures the array will always tolerate without data loss, and the respective fractions f_1, f_2 and f_3 of simultaneous failures of $n_f + 1$, $n_f + 2$ and $n_f + 3$ disks that will not result in a data loss. As any simulation tool, Proteus imposes no restriction on the distributions of failure and repair events. Our measurements have shown a surprisingly good agreement with the results obtained through analytical techniques and no measurable difference between values obtained assuming deterministic repair times and those assuming exponential repair times.**

I. INTRODUCTION

Providing trustworthy estimates of the reliability of fault-tolerant disk arrays is a difficult task because analytical techniques are based on assumptions that are never realized in practice and simulation techniques require writing a new simulation program for each array organization we want to investigate.

We wrote the Proteus simulation program to address these issues. First, Proteus is flexible and can be parameterized to model most fault-tolerant disk array organizations. Second, Proteus is designed to run fast, which is important because obtaining tight confidence intervals for the reliability of highly fault-tolerant disk arrays often requires millions of simulation runs. Finally, Proteus is written in Python 3, a freely available language that has been ported to many programming environments.

We used Proteus to evaluate the five-year reliability of various fault-tolerant disk array organizations, including RAID levels 4, 5, and 6 and two-dimensional RAID arrays. Our results show excellent agreement with the five-year reliability figures obtained through analytical techniques. In addition, they present no difference between the values obtained assuming deterministic repair times and those assuming exponential repair times.

The remainder of our paper is organized as follows. Section 2 reviews relevant fault-tolerant disk array organizations. Section 3 introduces our simulator and Section 4 presents experimental results and compares them with those obtained through analytical methods. Finally, Section 5 has our conclusions.

[1] Supported in part by Grant CCF-1219163, by the Department of Energy under Award Number DE-FC02-10ER26017/DE-SC0005417 and by the industrial members of the Storage Systems Research Center

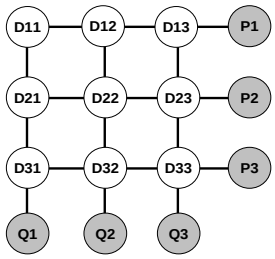


Fig. 1. A two-dimensional RAID array with 9 data and 6 parity disks.

II. FAULT-TOLERANT DISK ARRAYS

RAID arrays were the first disk array organizations to utilize erasure coding in order to protect data against disk failures [PGK88, SB92, CL+94]. While RAID levels 3, 4 and 5 only tolerate single disk failures, RAID level 6 organizations use $(n-2)$-out-of-n codes to protect data against double disk failures [BM93]. EvenOdd, Row-Diagonal Parity and the Liberation Codes are three implementations of RAID level 6 that use only XOR operations to construct their parity information [BB+95, CE+04, GX+08, P08]. Corbet et al. then Huang and Xu proposed similar coding schemes for correcting triple failures [C+03, HX05]..

Two popular combinations of RAID organizations are RAID 10 and RAID 01. RAID 10 organizations group their constituting disks groups into pairs of mirrored disks and combines these pairs of disks into a RAID level 0 organization. Conversely, RAID 01 organizations consist of two RAID level 0 arrays that mirror each other.

Two-dimensional RAID arrays, or 2D-Parity arrays, such as the one displayed in Figure 1, were investigated by Schwarz [S94] and Hellerstein et al. [HG94] who noted that these arrays tolerate all double disk failures but did not investigate how they reacted to triple or quadruple disk failures. More recently, Lee patented a two-dimensional disk array organization including prompt parity updates in one dimension and delayed parity updates in the second dimension [L04]. Pâris et al. [PSL07] investigated two-dimensional RAID arrays that reorganized themselves after a disk failure and noted that two-dimensional RAID arrays also tolerate most triple failures.

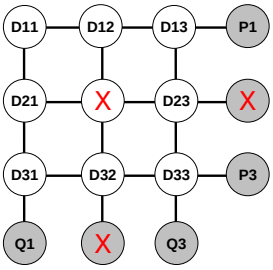


Fig. 2. A triple failure resulting in a data loss.

III. The Proteus Simulator

Our main motivation for using a simulation approach was its higher accuracy. While Markov models require disk failure and repair processes to obey a Poisson law, simulation allows us to use arbitrary distributions, among which failure distributions describing variable failure rates and repair time distributions with smaller coefficients of variation than the exponential distribution.

At the same time, we wanted to avoid the two main disadvantages of the simulation approach, namely the need to write a different simulation program for each array configuration being investigated and the long simulation runs. This latter consideration was especially important in our case because data losses are rare events and we might have to perform hundreds of thousands or even millions of runs of our model in order to observe a single data loss.

In other words, we wanted to develop a simulation program that was both very flexible and very fast. These two objectives were met by using a very simple and very flexible disk array model, which we will describe next.

A. *The disk array model*

A key feature of Proteus is its disk array model: it describes the topology of any disk array using only five parameters, namely:

1. The total number of disks n in the array;
2. The number n_f of simultaneous disk failures the array will always tolerate without data loss;
3. The fraction f_1 of simultaneous failures of $n_f + 1$ disks that will not result in a data loss;
4. The fraction f_2 of simultaneous failures of $n_f + 2$ disks that will not result in a data loss.
5. The fraction f_3 of simultaneous failures of $n_f + 3$ disks that will not result in a data loss.

For instance, all RAID level 0 arrays can be characterized by their size n and the four parameters

$$n_f = f_1 = f_2 = f_3 = 0,$$

since RAID level 0 tolerate no disk failures.

In the same way, we can characterize all RAID level 1 to 5 by their size and the four parameters

$$n_f = 1, f_1 = f_2 = f_3 = 0,$$

because these organizations tolerate all single disk failures and no double disk failures. The case of RAID level 6 arrays is fairly similar: they can be characterized by their sizes and the four parameters

$$n_f = 2, f_1 = f_2 = f_3 = 0.$$

For more complex disk array configurations, we cannot assume that the three parameters f_1, f_2, and f_3 are equal to zero because doing so would underestimate the array reliability. Consider for instance the two-dimensional disk array depicted in Figure 1. As we can see in Figure 2, the failure of an arbitrary data disk (D_{22} in our example) and its two parity disks (P_2 and Q_2) will always result in a data loss. We observe that these triple failures represent only a small fraction of all potential triple disk failures: for a two-dimensional disk array with n^2 data disks and $2n$ parity disks, only n^2 out of the $\binom{n^2+2n}{3}$ possible triple failures are fatal. As the size of the array grows, the ratio

$$\alpha = \frac{n^2}{\binom{n^2+2n}{3}}$$

quickly decreases: it becomes less than 1 percent for $n \geq 4$ and less than 0.1 percent for $n \geq 8$. Conversely, the fraction $f_1 = 1 - \alpha$ of triple failures the array will tolerate without data loss goes closer to unity as the size of the array increases.

Let us now consider quadruple failures. As Figure 3 shows, the sole quadruple disk failures resulting in a data loss are:

1. The failure of a data disk, its two parity disk and any other disk; and
2. The failure of four disks placed at the summits of a rectangle.

Out of the $\binom{n^2+2n}{4}$ possible quadruple failures the array can experience, we can enumerate:

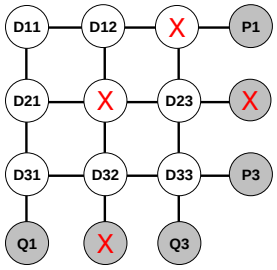
1. $n^2(n^2+2n-3)$ failures of a data disk, its two parity disk and any of the n^2+2n-3 remaining disks;
2. $\binom{n}{2}^2 + 2n\binom{n}{2}$ failures of four disks placed at the summits of a rectangle;

for a total of $n^2\left(n^2+2n-3\right)+\binom{n}{2}^2+2n\binom{n}{2}$ fatal quadruple failures.

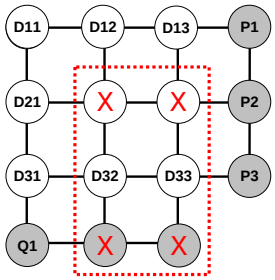
As we observed for triple failures, the fraction

$$\beta = \frac{n^2\left(n^2+2n-3\right)+\binom{n}{2}^2+2n\binom{n}{2}}{\binom{n^2+2n}{4}}$$

of fatal quadruple failures decreases with the size of the array: it becomes less than 4 percent for $n \geq 4$ and less than 0.4 percent for $n \geq 8$. In the same way, the fraction



(a) The failures of a data disk, its two parity disks and any other disk.



(b) The failures of four disks at the summits of a rectangle.

Fig. 4. Examples of quadruple failures resulting in a data loss.

$f_2 = 1 - \beta$ of quadruple failures the array will tolerate without data loss goes closer to unity as the size of the array increases.

We could, if we wanted, extend this approach to quintuple failures or even sextuple failures. We should however keep in mind that disk repair rates typically are more than one thousand times faster than their failure rates. As a result, each individual disk is likely to remain operational most of the time and the probability that any reasonably-sized disk array will experience five or six simultaneous disk failures is very low. Hence, we can safely assume that $f_3 = 0$. We nevertheless decided to keep this parameter as it might be useful for simulating very large disk arrays.

The other parameters of our model are the disk failure and repair time distributions.

In addition to its flexibility, our disk array model results in a very simple simulation model: it represents the disk array being investigated as a single entity whose state is described by its number of failed disks. That number will be decreased by one each time a failure event occurs and increased by one each time a disk gets repaired.

At the same time our model has four important limitations. First, it assumes that all disks have the same failure and repair distributions. The main reason for this restriction was keeping the simulator simple by limiting the number of its input parameters. Nothing actually prevents anyone from adding this feature as the change would be fairly simple to implement.

Second, our model assumes that disk failures and repair are independent events. The main reasons for our choice were keeping the simulator simple and the lack of an agreement on how to model correlated failures.

Third, our model imposes some restrictions on the disk repair process. It postulates that there will be no delay between the time a disk failure occurs and the start of the repair process. In addition, it also assumes that disk repairs can proceed in parallel without interfering with each other.

Fourth, our model assumes that we can safely neglect the probability that the disk array will experience more than $n_f + 3$ simultaneous disk failures and still undergo no data loss. This assumption is strictly true for RAID organizations 0 to 6 as they always experience a data loss after the simultaneous failure of more than n_f disks. It is fairly accurate for more complex disk array organizations as long they do not exceed, say, a few hundred disks.

B. *Selecting a reliability index*

Most extant studies of disk array reliability use array *mean time to data loss* (MTTDL) as a reliability index because it can be easily computed through analytical methods. Unfortunately, MTTDL is not the best predictor of the reliability of actual disk array. It captures instead the behavior of ideal disk arrays that would operate for tens or hundreds of years without ever being replaced unless they fail. This is not true for actual disks, which are replaced every five to seven years, often without having ever failed. As result, these estimates tend to underestimate the reliability of actual arrays. This is especially true for highly redundant disk arrays that are designed to operate without being repaired during their useful lifetime [PS+08].

We decided in favor of a more realistic index of disk array reliability, namely, their *five-year reliability*. It represents the probability that a given array will not experience a data loss over a useful lifetime of five years. In addition to being a better predictor of actual disk array reliability, five-year reliabilities can be directly measured through simulation by repeatedly simulating the behavior of a disk array over a period of five years and counting the number of times a data loss occurred. Unlike other approaches, this method does not require any assumptions about array failure distributions, such as assuming an exponential failure rate for the whole array.

C. *Simulation support*

Discrete simulation techniques are well suited to disk array reliability studies because these studies only focus on actions that take place when a disk fails or just after it has been replaced.

There are two approaches to the simulation of discrete systems. The *event-oriented approach* maintains an event list that contains all known future events sorted by the times at which they will occur. At each simulation step, the simulation program extracts the next event from the event list, updates the states of the system entities and possibly predicts the occurrence of the next events.

Another approach is possible. Various specialized simulation languages, among which Simscript [S13] and CSIM [S01], allow programmers to describe the behavior

of the system by defining special functions or processes that model the behavior of each system entity. This *process-oriented approach* allows the programmer to focus on the dynamic behavior of each system entities rather than on the mechanics of the simulation process. Its main disadvantage is its limited portability resulting from the proprietary nature of the best process-oriented languages.

While the process-oriented approach is essential for writing complex simulation models, the same is not true for simpler models like the one that we developed. We decided to use instead a freely available programming language and selected Python for three reasons. First, it is freely available on many computing platforms, among which Windows, Linux/Unix, Mac OS X, as well as the Java and .NET virtual machines. Second, its extensive libraries include good random number generators for the exponential and Weibull distribution as well as a fast built-in implementation of heaps. Finally, the conciseness of the language would result in a more readable program.

D. Proteus organization

Proteus consists of an input phase, a simulation phase and a very short data analysis phase.

The input phase prompts users for the parameters of the system in a self-explanatory fashion. The current implementation of Proteus offers two choices for the failure time distribution, namely exponential and Weibull as well as two possible repair time distributions, namely, exponential and deterministic. While the exponential distribution remains the "safe" traditional choice for modeling disk failures, the Weibull distribution allows users to study the behavior of disk arrays subject to either infant disk mortality during their early months or increased failure rates as the array ages.

In the same way, deterministic repair times are a good proxy for all repair time distributions with a smaller coefficient of variation than the exponential distribution.

The simulation phase consists of repeated runs of the simulation model. The main issue here is that we are simulating disk arrays that are not likely to fail during their useful lifetime. In some cases, we may thus have to simulate hundreds of thousands of runs in order to observe one data loss.

Figure 4 displays the pseudocode for an individual run of the model. The program starts by reinitializing the model and scheduling the first failure events for all the disks then goes through a fairly conventional event-oriented simulation. The actual code includes two domain-specific optimizations. Because disk mean times to fail are much larger than five years, Proteus does not schedule failure—and repair—events past the array lifetime. Second, it skips simulation runs whenever fewer than n_f disk failures are scheduled as these runs cannot result in a data loss even if the failed disks were never repaired.

The sole Proteus routine worth discussing here is the function that handles disk failures. As Figure 6 shows, it consists in a sequence of cascading ifs testing whether the failure results in a data loss.

```
reset simulation clock to zero
reset number of failed disks to zero
reset dataloss to False
clear event list (implemented as a heap)
schedule 'STOP' event at end of array lifetime
for all disks in array :
        schedule first 'FAILURE' event
while dataloss == False :
        extract first event from event list
        clock = time
        if event_type == 'STOP' :
                break
        elif event_type == 'REPAIR' :
                repair()
        elif event_type == 'FAILURE' :
                failure()
```

Fig. 4. Pseudo-code for an individual run of the model.

```
nfailed += 1
reset dataloss to False
if nfailed <= nf :
        dataloss = False    # fast exit
elif nfailed == nf + 1 :
        if random() > f1 :
                dataloss = True
elif nfailed == nf + 2 :
        if random() > f2
                dataloss = True
elif nfailed == nf + 3 :
        if random() > f3 :
                dataloss = True
else :
        dataloss = True
 if dataloss:
        update global dataloss count
else :
        schedule disk repair event
```

Fig. 6. Pseudo-code for the disk failure routine.

Finally, the data analysis phase computes a confidence interval for the five-year reliability of the array. As these values are fairly close to 1, we express then in "nines" using the formula $n_n = -\log_{10}(1 - R_d)$, where R_d is the five-year reliability of the array. Thus a reliability of 0.999 would be represented by 3 nines, a reliability of 0.9999 by 4 nines and so on.

Since array reliabilities were fairly close to 1, we had to use the Wilson confidence interval instead of the more common Gaussian interval.

IV. EXPERIMENTAL RESULTS

We present two series of experimental results comparing the reliability figures obtained by Proteus with those derived from analytical models. Since the outcomes of these analytical models were mean times to data loss (MTTDLs), these values were converted into five-year reliabilities using the formula

$$R_d = \exp(-\frac{d}{MTTDL})$$

where d is a five-year interval expressed in the same units as the MTTDL. Observe that the above formula implicitly assumes that long-term failure rate $1/MTTDL$ does not significantly differ from the average failure rate during the first five years of the array.

A. *Evaluating the impact of repair times distributions on the reliability of RAID arrays level 5 and 6*

In our first series of experiments, we measured the five year reliabilities of two distinct RAID arrays assuming both exponential and deterministic repair times and compared these values with these obtained through analytical techniques.

The first array we investigated was a RAID level 5 array consisting of five disks with the parity data thus occupying 20 percent of the disk space. The second disk array was a RAID level 6 array with 10 disks and thus the same space overhead as the first array.

We assumed a disk mean time to fail (MTTF) of one hundred thousand hours, which corresponds to slightly less than one failure every eleven years. This rate is at the high end of the failure rates observed by Pinheiro et al. [PWB07] as well as Schroeder and Gibson [SG07]. The three disk mean times to repair (MTTRs) we selected were one day, two days and five days.

Since RAID level 5 arrays tolerate at most one disk failure while RAID level 6 arrays tolerate at most two disk failures, the topology of our RAID level 5 array was modeled with the five parameters

$$n = 5,\ n_f = 1,\ f_1 = f_2 = f_3 = 0,$$

and that of our level 6 array with the parameters

$$n = 10,\ n_f = 2,\ f_1 = f_2 = f_3 = 0.$$

All simulations were repeated three times.

Tables I to IV summarize our results. As we can see, the five-year reliability values obtained through stochastic methods always fall inside our confidence intervals. In addition, we did not observe any significant difference between the reliability figures for deterministic repair times and those for exponential repair times. This confirms earlier observations made by Carroll and Long on the impact of repair time distributions on the availability of replicated data in the presence of site failures [CL89].

B. *Two-dimensional RAID arrays*

We present here results for a two-dimensional RAID array model with 64 data disks plus 16 parity disks and no superparity disk. Like the two RAID organizations we previously investigated the space overhead of this array is 20 percent. Space considerations prevent us from discussing here the case of two-dimensional arrays with a superparity disk. Interested readers are referred to Kao's thesis [K12].

Because the array tolerates all double disk failures without a data loss but not all triple failures, its n_f parameter is set to two. As we did in Section 2, we will assume that the array will tolerate most triple and quadruple failures but neglect to take into account the probability of no data loss after a quintuple failure. As a result, the values of its f_1, f_2 and f_3 parameters are

TABLE I. CONFIDENCE INTERVALS FOR THE FIVE-YEAR RELIABILITY OF A RAID LEVEL 5 ARRAY WITH FIVE DISKS ASSUMING DETERMINISTIC REPAIR TIMES.

Disk MTTF: 100,000 hours						
MTTR	One Day		Two Days		Five Days	
Array Reliability	Confidence Interval (nines)		Confidence Interval (nines)		Confidence Interval (nines)	
Run 1	2.67	2.68	2.37	2.38	1.98	1.99
Run 2	2.67	2.68	2.37	2.38	1.98	1.99
Run 3	2.67	2.68	2.37	2.38	1.98	1.99

TABLE II. CONFIDENCE INTERVALS FOR THE FIVE-YEAR RELIABILITY OF A RAID LEVEL 5 ARRAY WITH FIVE DISKS ASSUMING EXPONENTIAL REPAIR TIMES.

Disk MTTF: 100,000 hours						
MTTR	One Day		Two Days		Five Days	
Array Reliability	Confidence Interval (nines)		Confidence Interval (nines)		Confidence Interval (nines)	
Run 1	2.67	2.68	2.37	2.38	1.98	1.99
Run 2	2.68	2.69	2.38	2.38	1.98	1.99
Run 3	2.67	2.69	2.37	2.38	1.99	1.99
Analytic	2.679		2.379		1.985	

TABLE III. CONFIDENCE INTERVALS FOR THE FIVE-YEAR RELIABILITY OF A RAID LEVEL 6 ARRAY WITH TEN DISKS ASSUMING DETERMINISTIC REPAIR TIMES.

Disk MTTF: 100,000 hours						
MTTR	One Day		Two Days		Five Days	
Array Reliability	Confidence Interval (nines)		Confidence Interval (nines)		Confidence Interval (nines)	
Run 1	4.94	5.12	4.42	4.52	3.65	3.69
Run 2	4.94	5.12	4.44	4.53	3.63	3.67
Run 3	4.97	5.16	4.39	4.47	3.63	3.66

TABLE IV. CONFIDENCE INTERVALS FOR THE FIVE-YEAR RELIABILITY OF A RAID LEVEL 6 ARRAY WITH TEN DISKS ASSUMING EXPONENTIAL REPAIR TIMES.

Disk MTTF: 100,000 hours						
MTTR	One Day		Two Days		Five Days	
Array Reliability	Confidence Interval (nines)		Confidence Interval (nines)		Confidence Interval (nines)	
Run 1	4.94	5.12	4.42	4.52	3.65	3.69
Run 2	4.94	5.12	4.44	4.53	3.63	3.67
Run 3	4.97	5.16	4.39	4.47	3.63	3.66
Analytic	5.043		4.443		3.651	

$$f_1 = 1 - \alpha = 0.999221,\ f_2 = 1 - \beta = 0.996105,\ f_3 = 0$$

As before, the disk MTTF was set to 100,000 hours. We assumed exponential repair times and investigated this time a range of disk repair times extending from half a day to ten days. The results of our simulations were compared to the results obtained through Markov analysis and presented previously [PS+12].

TABLE V. CONFIDENCE INTERVALS FOR THE FIVE YEAR RELIABILITY OF A TWO-DIMENSIONAL RAID ARRAY WITH 64 DATA DISKS AND 16 PARITY DISKS.

MTTR (days)	Five-Year Reliability		
	L-bound (nines)	U-bound (nines)	Analytic (nines)
0.5	5.739	6.863	5.911
1	5.079	5.440	5.295
1.5	4.783	5.023	4.923
2	4.505	4.673	4.649
2.5	4.365	4.505	4.426
3	4.206	4.321	4.236
3.5	4.009	4.100	4.068
4	3.911	3.991	3.917
4.5	3.740	3.806	3.779
5	3.628	3.686	3.651

As before, the disk MTTF was set to 100,000 hours. We assumed exponential repair times and investigated this time a range of disk repair times extending from half a day to ten days. The results of our simulations were compared to the results obtained through Markov analysis and presented previously [PS+12].

As Table 5 shows, the five-year reliability values obtained through stochastic methods always fall inside our confidence intervals. In addition, we observe that our two-dimensional RAID organization offers better five-year reliability figures than the RAID level 6 organization we investigated before. This is a superb result when we consider that our two-dimensional array has the same space overhead as the RAID level 5 organization and holds eight times as much data.

V. CONCLUSION

Previously, practitioners who desired to obtain good estimates of the reliability of a disk array were forced to choose between accepting the limits of analytical methods and having to write a different simulation program for each disk array configuration.

We have presented Proteus a flexible portable simulation tool for evaluating the risk of data loss in fault-tolerant disk arrays. Its main advantage is its ability to simulate a wide range of disk array configurations without any reprogramming. This was made possible through the key observation that disk array reliability can be characterized by the probability that the array will survive a given number of disk failures.

Proteus is an open-source program that will be made available from the web site of the Storage Systems Research Center of the University of California, Santa Cruz (www.ssrc.ucsc.edu**)** and the personal web site of one of the authors (www.cs.uh.edu/~paris).

REFERENCES

[BB+95] M. Blaum, J. Brady, J. Bruck, and J. Menon, EvenOdd: An efficient scheme for tolerating double disk failures in RAID architectures, *IEEE Trans. Computers* 44(2):192–202, 1995.

[BM93] W. A. Burkhard and J. Menon. Disk array storage system reliability. *Proc. 23rd International Symp. on Fault-Tolerant Computing*, pp. 432–441, 1993.

[CE+04] P. Corbett, B. 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 and Storage Technologies*, pp. 1–14, 2004.

[C+03] P. F. Corbett, R. M. English, and S. R. Kleiman, Triple parity technique for enabling efficient recovery from triple failures in a storage array, US Patent Application EP1324200 A1, 2003.

[CL+94] P. M. Chen, E. K. Lee, G. A. Gibson, R. Katz and D. A. Patterson. RAID, High-performance, reliable secondary storage, *ACM Computing Surveys* 26(2):145–185, 1994.

[CL89] J. L. Carroll and D. D. E. Long, The effect of failure and repair distributions on consistency protocols for replicated data objects, *Proc. 22nd Annual Simulation Symp.*, pp. 47–60, Mar. 1989.

[GX+08] W. Gang, L. Xiaoguang, L. Sheng, X. Guangjun, and L. Jing, Generalizing RDP codes using the combinatorial method, *Proc. 7th IEEE Int. Symp. on Network Computing and Applications*, pp. 93–100, 2008.

[HG94] L. Hellerstein, G. Gibson, R. M. Karp, R. H. Katz, and D.A. Patterson. Coding techniques for handling failures in large disk arrays. *Algorithmica*, 12(3-4):182-208, June 1994

[HX05] C. Huang and L. Xu, STAR: an efficient coding scheme for correcting triple storage node failures, *Proc. 4th USENIX Conf. on File and Storage Technologies*, pp. 197–210, Dec. 2005.

[K12] H.-W. Kao, *Proteus: A portable simulation program for estimating data loss risks in disk arrays*, MS Thesis, Dept. of Computer Science, U. of Houston, 2012.

[L04] W. S. Lee, *Two-dimensional storage array with prompt parity in one dimension and delayed parity in a second dimension*, US Patent #6675318 B1, 2004

[P08] J. S. Plank, The RAID-6 liberation codes, *Proc. 6th USENIX Conf. on File and Storage Technologies*, pp. 1–14, 2008.

[PGK88] D.A. Patterson, G. Gibson, and R.H. Katz, A case for redundant arrays of inexpensive disks (RAID). *Proc. SIGMOD Int. Conf*, pp. 109–116, 1988.

[PS+08] J.-F. Pâris, T. J. Schwarz, D. D. E. Long and A. Amer, When MTTDLs Are Not Good Enough: Providing Better Estimates of Disk Array Reliability, *Proc. 7th I2TS Symp.*, pp. 140–145, 2008

[PSL07] J.-F. Pâris, T. J. Schwarz and D. D. E. Long. Self-adaptive archival storage systems. *Proc. 26th Int. Performance of Computers and Communication Conf.*, pp. 246–253, 2007.

[PWB07] E. Pinheiro, W.-D. Weber and L. A. Barroso, Failure trends in a large disk drive population, *Proc. 5th USENIX Conf. on File and Storage Technologies*, pp. 17–28, 2007.

[S94] T. J. Schwarz, *Reliability and Performance of Disk Arrays*, Ph.D. Thesis, U. C. San Diego, 1994.

[S01] H. S. Schwetman, CSIM19: a powerful tool for building system models, *Proc. 2001 Winter Simulation Conference*, Vol. 1, pp. 250–255, 2001

[S13] www.simscript.com, retrieved February 1, 2013.

[SB92] T. J. E. Schwarz and W. A. Burkhard. RAID organization and performance. *Proc. 12th Int. Conf. on Distributed Computing Systems*, pp. 318–325, 1992.

[SG07] B. Schroeder and G. A. Gibson, Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you? *Proc. 5th USENIX Conf. on File and Storage Technologies*, pp. 1–16, 2007.