Reliability Modelling of Disk Subsystems with Probabilistic Model Checking

Technical Report UCSC-SSRC-09-05
May 2009

K. Gopinath
gopi@csa.iisc.ernet.in

Jon Elerath
jelerath@comcast.net

Darrell Long
darrell@cs.ucsc.edu

Storage Systems Research Center
Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
`http://www.ssrc.ucsc.edu/`

Reliability Modelling of Disk Subsystems with Probabilistic Model Checking

K. Gopinath[*]
Indian Institute of Science,
Bangalore 560 012, INDIA

Jon Elerath
NetApp Inc.,
Sunnyvale, CA

Darrell Long
Univ. California, Santa Cruz,
Santa Cruz, 95064, CA

Abstract

We discuss how probabilistic model checking, a variant of model checking, can be useful in modelling multiple failures in disk subsystems, as it allows specification of probabilities or rates of transitions also. Probabilistic model checking can be used to not only check the correctness of formulae that are augmented with a probability operator, but also to compute, as a side effect, probabilities and Markov rewards in response to queries expressed in an expressive logic. With exponential distributions for failures, it is possible to accurately compute the reliability of many non-trivial disk systems. Exploiting the technique of symmetry reduction, it is possible to handle even larger fully symmetric systems. For non-exponential distributions, such as Weibull models of disk reliability, we can analyze reasonable sized systems (such as RAID4/5/6) if good approximations based on exponomials are possible. We report our experiences and compare our results with those from simulation.

Keywords: Probabilistic model checking, disk storage systems, continuous time Markov chains, RAID, Weibull models.

1 Introduction

Storage systems are complex due to a large number of components: disks, controllers, interconnects. Each of these components may have complex failure modes; a disk can suffer, for example, a whole disk failure, a sector failure or a partially written sector ("torn") write [1]. In addition, the reliability model of a disk is complex, given its realization as an electro-mechanical device of considerable complexity. This leads to an enormous state space if we choose to model them in any detail.

Simulation is a widely used and powerful technique for evaluating such systems. Simulation is the most flexible since it allows us to use arbitrary distributions (such as Weibull common in reliability studies) and even traces. However, simulations take very long to run as the events that we are trying to model (data loss) are rare. Further, it requires sophistication to design statistically valid simulation runs and to interpret simulation results. We propose and investigate *Probabilistic Model Checking* [21] as a technique for analyzing the reliability of disk-based storage systems.

Model checking is a formal method for evaluating the correctness of systems. The correctness of microprocessor designs in the industry depends critically on model checking portions of its subsystems as the expensive FDIV Pentium bug has made the use of model checking techniques mandatory. Model checking has been used recently [22] to study correctness of RAID5 protocols with latent sector errors (when a disk sector cannot be read), torn writes and so on. Model checking is different from a proof-theoretic approach such as natural deduction; it is fully automated once the model and the formula to be checked are specified and is therefore suitable for regular use in industry.

Probabilistic model checking (PMC) [21] is a formal verification technique for the modelling and analysis of systems that exhibit stochastic behaviour. While propositions with probabilistic operators can be checked for validity, it is also possible to compute certain probabilities in the model as a side effect. This is similar to the use of standard model checking to derive solutions from counterexamples. For example, the cannibals problem can be model checked with the assertion that there is no solution; a model checker that gives all the counterexamples provides us the desired solution to the problem. Since the algorithms for PMC proceed by computing the actual probabilities necessary to evaluate a given formula with a probability operator, no additional work is needed to compute the probability that a certain path formula has.

PMC is an exact technique (modulo numerical analysis errors which are common to all numerical methods) and hence any inaccuracies in the results arise from the modelling itself or approximations used for distributions. Since only negative exponential distributions can be specified in current PMC tools, the usefulness of this technique

[*]gopi@csa.iisc.ernet.in (contact author, ph.831-459-2177), jon.elerath@netapp.com, darrell@cs.ucsc.edu

for reliability models based on non-exponential distributions (such as Weibull) depends on how well approximations can be carried out. We discuss this in the last part of the paper.

The automation possible in PMC is useful in solving reasonably large and non-trivial Markov models with exponential distributions; in comparison, analytical techniques require considerable effort, even for small systems, in solving coupled differential equations using Laplace transforms [19] while simulation requires code development and careful statistical validation of the results. While individual simulation runs can be fast, simulation for rare events in reliability studies requires many runs to reduce the variance of the results (proportional to $1/\sqrt{p}$, p being the rare event probability) and techniques such as importance sampling have to be used. However, many of its techniques are not easy to use and are still a research topic [3].

Though PMC encounters roadblocks in terms of state explosion in larger or very detailed models, we believe it is a good midway solution between simulation and analytical models: the models are high level and there is considerable automation in handling a rich query language based on temporal logic used for posing reliability queries.

While current tools do not exploit multicore clusters well, as the algorithms have not been parallelized, it will be more attractive once they are effectively parallelized on large clusters. In addition, techniques for identifying and eliminating redundant states due to symmetry are an important part of the computer-aided verification (CAV) discipline and hence any advance in this area can be used immediately. Fully symmetric systems do arise naturally; we discuss the reliability of RAID5 systems from 5 to 10 disks as well as RAID-DP[4] systems with 8 disks under realistic error models.

The outline of the paper is as follows. We first give some background on the technique of PMC. We then consider a few disk subsystems considered in literature and analyze them under realistic error models using PRISM [13], a tool for PMC. For Weibull models, we present our modelling methodology and the approximations used. We discuss the results of these models and how they compare with simulation. Due to the difficulty of computing good approximations for Weibull distributions, the agreement is still not very satisfactory. Finally, we will conclude with some problems with the technique and future work.

2 Previous Work

Elerath et al. [11] have recently used simulations to study enhanced reliability modelling of RAID4 storage systems. There have been many other studies that have used simulation or analytical modelling for the study of multi-disk storage systems but we cite just two more: Qin et al. [7], Paris et al. [2]. Hafner and his colleagues have developed analytical models of reliability for networked storage using RAID, and for non-MDS codes [17, 16]. These methods, however, do not have significant automation as discussed earlier. Krioukov et al. [22] have used model checking to estimate probability of error in a 5-disk RAID; it is not clear if their technique can handle large state spaces or uses a general query logic with probabilistic operators.

The System Availability Estimator (SAVE) tool [5] has a reliability-based language that solves a continuous time Markov chain for user specified measures such as steady state availability, interval availability, reliability and the mean time to failure. Importance sampling [5] is also supported for fast simulation using "failure biasing".

Trivedi and his collaborators have developed SHARPE and GSHARPE tools for reliability and performance modelling [9, 10]. For example, Malhotra and Trivedi[8] have analyzed RAID systems using SHARPE. While these tools have been used for research and in industry, it is not clear if any special techniques have been used to handle large state spaces.

PMC has been used in many performance studies but we are not aware of its use for studying storage reliability.

3 Primer on Prob. Model Checking

Model Checking [12] checks if a set of formulae have a model, i.e. a set of valuations (an interpretation) that satisfies the formulae. The formulae can be propositional, temporal, first order logic and so on but the propositional temporal logic is most often used. Linear temporal logic (LTL) deals with specifying (propositional) properties, relative to the current state in a path, in the future (≥ 1 states), past (≤ 1 states) and next state whereas branching time logic or computation tree logic (CTL) deals with possible futures, including a notation for describing a property in all states in all futures ("globally"), in some state in some future ("exists"), in the next state from the current state in all futures, etc. *Probabilistic* Model Checking [21], in addition, can specify properties on paths using a probability operator. The probabilistic version of CTL most widely studied is PCTL (probabilistic CTL) and CSL (continuous stochastic logic).

Model checking requires two inputs:

- a description of the system, usually given in some high level modelling formalism such as process algebra or Petri nets.

- a specification of one or more desired properties of the system, often in CTL, LTL, PCTL or CSL.

Using these inputs, a model checker constructs a model of the system, typically a labelled state-transition system in which each state represents a possible configuration and each transition represents an evolution of the system from one configuration to another over time. This is typically done by exhaustive exploration (the state space is explicitly generated) or by symbolic methods (the state space is represented implicitly using a formula in propositional logic, often encoded in space efficient data structures such as binary decision diagrams (BDDs), or multi-terminal BDDs (MTBDDs)). It is then possible to automatically verify whether or not each property is satisfied, based on a systematic and exhaustive exploration of the constructed state-transition system.

In probabilistic model checking, the models are provided with additional quantitative information regarding transitions (either probabilities or rates) and the times at which they do so. In practice, these models are typically Markov chains or Markov decision processes. If probabilities are used in the Markov chains and time is modelled in discrete steps, we have the discrete-time Markov chains (DTMC). To model reliability or performance, however, the most suitable model is continuous-time Markov chains (CTMCs), in which transitions between states are assigned positive, real-valued rates of negative exponential distributions. CTMCs have a continuous (real-valued) model of time and probabilistic choice but no non-determinism; they specify rates of making a transition instead of probabilities. The logic used for CTMCs is Continuous Stochastic Logic (CSL) that is based on PCTL; for example, it has a steady state operator that computes steady state probabilities.

Formally, if AP is the fixed, finite set of atomic propositions used to label states with properties of interest and $\mathbb{R}$ the reals, then a CTMC is a tuple (S, R, L):

- S is a set of finite states
- $R : S \times S \to \mathbb{R}_{\geq 0}$, the transition rate matrix.
- $L : S \to 2^{AP}$ is a labelling function which associates each state with a set of atomic propositions.

If there is more than one transition available at a state, there is a "race condition" and the first transition to be triggered determines the next state. The time spent in state s before any such transition occurs is exponentially distributed with the rate $E(s) = \Sigma_{s' \in S} R(s, s')$, the exit rate; the transition is triggered in time t with probability $1 - e^{-E(s) \cdot t}$. The probability of moving to state s' is given by $R(s, s')/E(s)$, and the probabilities between the various states then defines a probability matrix for an embedded DTMC. A CTMC can be augmented with rewards, attached to states and/or transitions of the model. Rewards can be used, for example, to count how many time units are spent in a state.

The underlying computation in probabilistic model checking involves a combination of graph-theoretical algorithms (for temporal logic model checking and qualitative probabilistic model checking) and numerical computation (for quantitative probabilistic model checking, i.e. calculation of probabilities and reward values).

There are many research tools that have been developed for probabilistic model checking. We will primarily use the PRISM model checker developed at Birmingham/Oxford [13]; it has support for all of the above with PCTL/CSL logic and generally considered to have good support for exploring large state spaces [14]. It also uses techniques such as "uniformization" for controlling some numerical errors. In reliability models with widely differing rates for failures and repairs (4-5 orders of magnitude), the convergence parameter depends on the stiffness of the problem. With many of the problems considered here, a (relative) convergence parameter of 1E-6 is good enough. Another well regarded tool is MRMC (Markov Reward Model Checker) [15]. A discussion of the merits of some of the tools available is given in [14].

3.1 Logic

The logic used in many probabilistic model checkers such as PRISM is PCTL which extends CTL by adding a probability operator $\mathcal{P}$. Example: under any scheduling of processes, probability that event A occurs is at least p (at most p): $\mathcal{P}_{\geq p}$ ($\mathcal{P}_{\leq p}$). The syntax of PCTL is as follows [21]; here AP is any atomic proposition, relop the relational operator, and X is the next state operator.

- state formulae:

  ϕ :: true | AP | $\phi1$ & $\phi2$ | not ϕ | $\mathcal{P}_{\text{relop } c}(\alpha)$
  relop :: $\geq$ | $\leq$ | $>$ | $<$

- path formulae:

  α :: X ϕ | $\phi1$ until$^k \phi2$ | $\phi1$ until $\phi2$

"X ϕ" is true if ϕ is true in all the states reachable in one step from the current one. The "$\phi1$ until $\phi2$" path formula is true if $\phi2$ becomes true in some future state and $\phi1$ is true starting from the 1st state in the path till that state. If the `until` is bounded (until^k), this transition has to happen in exactly k steps. Note that a PCTL formula is always a state formula with the path formulas occurring only inside the $\mathcal{P}$ operator. Since we will use CTMC for our studies, we discuss its logic CSL next.

3.2 CSL over CTMC

Continuous stochastic logic (CSL) is based on PCTL. Instead of the bounded `until` operator, it has a `until` interval operator; a path formula "$\phi 1$ until$^{t_1,t_2}\phi 2$" is true if $\phi 2$ is true some time instant in the interval $[t_1, t_2]$ and $\phi 1$ holds at all preceding time instants. It has, in addition, a steady state operator that computes the probability of staying in a state satisfying a formula in the long run *r*elop p. Example: if a wireless channel is free, the probability that it will be busy within t time units is less than 10%: "$\mathcal{P}_{<0.1}$[free until <= t busy]".

While the above query language is powerful (one is not limited to a "canned" set of queries one can pose), CSL is still not powerful enough to handle some natural queries such as: find the time at which the probability becomes a certain value. This is needed, for example, in computing the five 9's reliability [2]. Currently, this has to be solved by the "bisection" method by guessing the time and then increasing or decreasing the time. This simple procedure is guaranteed to be effective as the reliability function over time is a decreasing monotone function.

3.3 Example of Model Checking a formula

We will take a very simple formula "$X\phi$" for illustration. For simplicity, let us assume that the full state graph has already been generated from the model. We can therefore find which of the states satisfies ϕ; let us represent all the states whether they satisfy this formula by a state-indexed column vector of 0 or 1s, B. If the formula that is being model checked is $X\phi$, the matrix vector product of the embedded probability matrix of the induced DTMC and B gives the required probabilities for $X\phi$. For other path operators (such as `until`), more involved computations (such as fixpoints) are needed but they all boil down to matrix vector multiplications or solving linear equations [21].

We will not discuss the computational complexity of model checking here due to lack of space; the most expensive complexity is cubic in the number of states.

3.4 PRISM modelling language

PRISM has a simple, state-based language for DTMCs/CTMCs/MDPs with process algebra-based primitives. Modules can be defined for each system component and they can be composed in parallel. Variables are finite-valued and either local or global to the module. It has guarded commands that are labelled with probabilities (in case of DTMC/MDP) or rates (CTMC). Though we do not use it in this paper, synchronization between modules is possible through action labellings; they can also be used for process algebra-style operations. For illustration, we consider a simple RAID5[1] model in PRISM with a very simple on/off model of the disk but that handles latent sector errors (LSE) and rebuilds the failed disk onto a spare disk [16] (note that we use the exact same "optimized" Markov model they used in their studies). h >1 means that the rebuild cannot be successful.

```
ctmc  // Continuous Time Markov Chain
//Next 4 decls common to all code examples below
const int MTTFd; // MTTF of disk
const int MTTRd; // MTTR of disk
const double lambda=1/MTTFd; //failure rate
const double mu=1/MTTRd; //repair rate

const int d; // num of disks
const double HER; // hard error rate/GB
const int dcap; // disk capacity in GB
const double h=(d-1)*dcap*HER;//rebuild perr

module raid5
 s: [0..2] init 0; //state of disk
     //s=0: working; 1: repair; 2: fail
 [] (s=0) -> d*(1-h)*lambda:(s'=1) +
             d*h*lambda:(s'=2);
 [] (s=1) -> mu: (s'=0) + (d-1)*lambda: (s'=2);
endmodule

rewards
 !(s=2): 1; // unit reward if disk OK
endrewards
```

The RAID5 model has only a single module; if there are more than one, the composite state space of the model is obtained, by default, as the cross-product of the states of all the modules by interleaving each guarded statement. The action labellings (inside `[]`) are empty in the two statements of the module as we do not use process algebra type synchronization between the modules. The two guard conditions are `(s=0)` and `(s=1)`. The second statement means: if state `s` is 1 (repair state), there are two possibilities (separated by the process-algebraic notation for composing processes, namely +): either a repair process at rate `mu` that takes the system to state 0 `(s'=0)` or another failure process with rate `(d-1)*lambda` that takes it to the fail state `(s'=2)`. A variable that is redefined has a prime (') as a suffix for the new version. Note that the arithmetic + can be used by overloading but not at the outermost level. The `&` is similarly overloaded: in the guards on the LHS, it means the logical operator (just as negation `!` and or `|`) whereas on the RHS it means conjunction of assignments.

The query language is based on CSL but augmented with Markov reward models: thus we can compute the mean time to data loss (MTTDL) by the formula `R=? [F (s=2)]`. Here, `R=?` is the query for computing the cumulative rewards until (`F`, signifying "finally") the state 2 is reached (`(s=2)`) i.e. until a disk fails or data gets corrupted during reorganization; this is realized by assigning a unit rate of accumulation of reward when in states that

[1]In RAID5, a parity disk block is computed as the xor of the n-1 data disk blocks. If a disk fails, the lost disk block (data or parity) can be rebuilt using the xor of the other n-1 disk blocks.

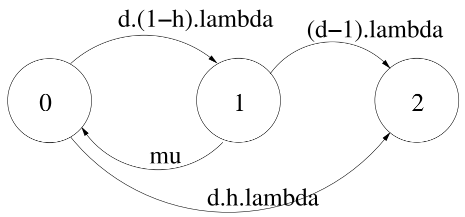


Figure 1: RAID5 Model from Rao et al. [16]

are not `(s=2)`: namely, `!(s=2):1`. Absorbing states are detected (state 2 is one) and handled without any difficulty.

Figure 1 gives the Markov Model. There is a direct and natural 1-1 mapping of the transitions in the Markov model and the disjunctive expressions in the PRISM model. In addition to computing MTTDL, we can compute the reliability at time t by model checking, `P=? [true U<=t (s=2)]`, which queries the probability (`P=?`) of reaching the fail state `(s=2)` until time t (`true U<=t`). Note that, in general, an analytical solution involves solving a potentially complex system of linear differential equations . This simple model can be solved analytically with ease [16] and PRISM, which uses numerical techniques corresponding to the analytical methods, gives identical results.

4 Case Studies

We model the following to study the suitability of probabilistic model checking for analyzing disk subsystems:

- A small disk array with/without LSEs and scrubbing
- Larger disk arrays with 3-state HMM ("Hidden Markov Model") models for disks that model burn-in failures; smaller disk arrays but with LSEs
- RAID5/6 systems with Weibull models for disk failures, LSEs, scrubbing and rebuild times.

4.1 Simple Reliability Models for Small Disk Arrays

4.1.1 SSPiRAL

Amer et al. [19] have considered an array structure that is more robust than mirrored schemes but is close to them in simplicity; it also requires fewer number of disks to be read fully for a rebuild compared to RAID5 schemes. A SSPiRAL layout is defined by three parameters: the degree of the system (the number of devices the data will be distributed across), the total number of devices (data and parity), and the x-order (the number of devices used

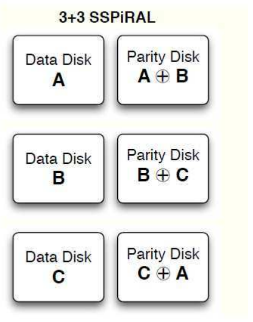


Figure 2: SSPiRAL 3+3 disk array from Amer et al.[19].

for the parity calculation; it is also the maximum number devices to be read in case of a rebuild).

Basic Model For the 3-degree, 2-order design with 3 data disks + 3 parity disks design with x=2 (Fig. 2) and not considering latent sector errors, we can specify the reliability model in PRISM as follows (we assume the 1st 4 declarations in the previous model):

```
const int MAX=4; // max failures
// formula is a ‘‘macro’’
formula cond = ((s1=1) & (s2=1)  & (s3=1)
               |(s1=1) & (s12=1) & (s31=1)
               |(s2=1) & (s12=1) & (s23=1)
               |(s3=1) & (s31=1) & (s23=1));
module disks
 s1: [0..1] init 0;
 s2: [0..1] init 0;
 s3: [0..1] init 0;
 s12: [0..1] init 0;
 s23: [0..1] init 0;
 s31: [0..1] init 0;
 fail: [0..MAX] init 0;
 dataloss:[0..1] init 0;

 [] (fail<MAX-1) & (s1=0) ->
         failrate: (s1’=1) & (fail’=fail+1);
 [] (fail<MAX-1) & (s2=0) ->
         failrate: (s2’=1) & (fail’=fail+1);
 [] (fail<MAX-1) & (s3=0) ->
         failrate: (s3’=1) & (fail’=fail+1);
 [] (fail<MAX-1) & (s12=0) ->
         failrate: (s12’=1) & (fail’=fail+1);
 [] (fail<MAX-1) & (s23=0) ->
         failrate: (s23’=1) & (fail’=fail+1);
 [] (fail<MAX-1) & (s31=0) ->
         failrate: (s31’=1) & (fail’=fail+1);
 [] (fail>0) & (s1=1)    ->
         repairrate: (s1’=0) & (fail’=fail-1);
 [] (fail>0) & (s2=1)    ->
         repairrate: (s2’=0) & (fail’=fail-1);
 [] (fail>0) & (s3=1)    ->
         repairrate: (s3’=0) & (fail’=fail-1);
 [] (fail>0) & (s12=1)    ->
         repairrate: (s12’=0) & (fail’=fail-1);
 [] (fail>0) & (s23=1)    ->
         repairrate: (s23’=0) & (fail’=fail-1);
 [] (fail>0) & (s31=1)    ->
         repairrate: (s31’=0) & (fail’=fail-1);
endmodule
```

```
rewards
    !((fail=MAX-1) & cond | (fail=MAX)): 1;
endrewards
```

The 3+3 system enters a fail state if any 4 disks fail or 4 of the specified configurations with 3 disks fail. We can compute the probability of failure at time t by the formula

```
P=? [true U<=t ((fail=MAX) | (fail=MAX-1) &
     ((s1=1) & (s2=1) & (s3=1)
     |(s1=1) & (s12=1) & (s31=1)
     |(s2=1) & (s12=1) & (s23=1)
     |(s3=1) & (s31=1) & (s23=1)))]
```

The graph plotting the probability of failure against time from 1 year to 100 years turns out to be linear and is given in Figure 3(a). On an Opteron single core[2], it takes about 2.5 min elapsed time for the whole run. The previously reported results on SSPIRAL [19] are approximate as the analytical approximation used aggregates the states of data and parity disks, when appropriate, for tractability. It has only 5 states and 8 transitions compared to the 114 states and 700 transitions of the PRISM model obtained by an optimized cross-product operation. It may be possible to remove still some of the redundant states and transitions arising from symmetry in the system but current state of art in symmetry reduction [23] requires "fully symmetric systems" which SSPIRAL is not (as the parity and data disks are not interchangeable). We present the results reported by PRISM and the analytical results, along with running PRISM on the approximate analytical model SSPIRAL (fig. 3 of [19]) for a disk with MTTF of 10^5 hours and MTTR of 30 hours in Table 1 (being one set of parameters used in [19]). It cannot be done analytically without approximations, and simulation requires considerable number of runs to be sure of statistically valid results. For a confidence interval of 95% and maximum relative half-width (or deviation) of 10%, the number of simulation runs necessary is $384 \cdot (1-p)/p$ where p is the probability of the rare event [3]. Since p is quite small (of the order of 1E-6) for this example, large number of runs are necessary even for a deviation of just 10% unless special methods are used in the simulation (such as importance sampling). The model checking technique provides a much quicker and accurate answer and with much less user effort, with a relative error of less than 1E-6.

However, just as with the simple RAID5 model, this model is not realistic. It does not take into account latent sector errors (LSE) that are detected when a sector cannot be read nor does it take into account the non-exponentially distributed (total) failure rates of the disk that are seen in practice. We will take into account LSE first and the latter once we develop better disk models (see Sec. 4.3).

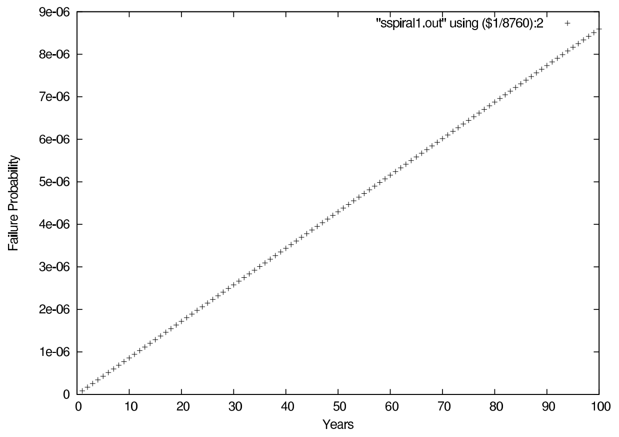
[2]For this and other results reported in this paper, we use one core of a 2GHz Opteron. Also, we have selected the SOR technique for solving the Markov chains with a convergence parameter of 1e-06. We have used the default 'hybrid" method for representing data structures using MTBDDs; using sparse matrices is faster but needs more memory.

years	PRISM	Analyt[19]	Analyt(PRISM)
4	3.77E-7	3.78E-07	3.78E-7
5	4.72E-7	4.73E-07	4.72E-7
20	1.89E-6	1.89E-06	1.89E-6
100	9.44E-6	9.46E-06	9.45E-6

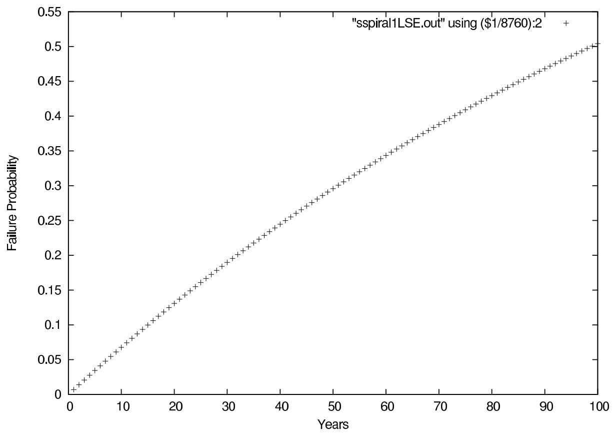
Table 1: Failure probability vs time for SSPIRAL 3+3 array: PRISM vs approx. analytical [19] ("Analyt"), and approx. analytical model in PRISM ("Analyt(PRISM)").

Model with LSE In the context of the configurations of the previous 3-disk failure dataloss cases, we have dataloss even if only 2 disks fail and the other 3rd disk has a LSE. If only one disk fails and 2 other necessary disks have LSEs, we assume that the system can recover as two disks having the LSE at the same sector position is improbably small. If the state of a disk i is x_i, with 0 being UP, 1 being LSE and 2 being DOWN, then all these dataloss cases can be conveniently coded as satisfying $\sum_{i=1}^{3} x_i > 4$. Adding regular scrubbing of disks, the modified PRISM code is given below (we again assume the 1st 4 declarations from the first code example):

```
const int MAX=4; // max failures
const double lse; //latent sector err rate
const double lscr; // scrubbing rate
const double HER; //hard error rate/GB
const int dcap; // disk capacity in GB
const double h=(3-1)*dcap*HER; //perrebuild
const int UP =0;
const int LSE =1;
const int DOWN =2;

global dataloss: bool init false;
global fail: [0..MAX] init 0;

formula fdataloss= (x1+x2 +x3 >4) |
                   (x1+x12+x31>4) |
                   (x2+x12+x23>4) |
                   (x3+x31+x23>4);
//data loss if 3 disks fail, or 2 disks
//fail and one other disk has an LSE in
//the configurations involving 3 failures

module disk1
x1: [0..2] init 0; //state of disk
 [](fail<MAX) & (x1=UP) ->
      lambda:(x1'=DOWN) & (fail'=fail+1)+
                             lse: (x1'=LSE);
 [](fail<MAX) & (x1=LSE) -> lscr: (x1'=UP)+
      lambda:(x1'=DOWN) & (fail'=fail+1);
 [](x1=DOWN) & !fdataloss ->
      (1-h)*mu: (x1'=UP) & (fail'=fail-1)+
          h*mu: dataloss'=true;
endmodule
// instantiate 5 more disks by renaming vars
module disk2 =disk1[x1=x2, x2=x1]  endmodule
module disk3 =disk1[x1=x3, x3=x1]  endmodule
module disk12=disk1[x1=x12,x12=x1] endmodule
module disk23=disk1[x1=x23,x23=x1] endmodule
module disk31=disk1[x1=x31,x31=x1] endmodule

rewards
     true: 1;
endrewards
```



(a) Basic Model



(b) With LSEs

Figure 3: (a) Failure probability of the SSPiRAL 3+3 array from 1 to 100 years. (b) The same with LSEs. Note that due to the 6 orders of magnitude difference, we cannot plot them together on the same graph. Each has 100 data points.

The failure in t hours is given by the query:

```
P=? [true U<=t (dataloss | (((x1=2)|(x2=2)|
      (x3=2)|(x12=2)|(x23=2)|(x31=2)) &
     ((x1+x2+x3>4) | (x1+x12+x31>4) |
      (x2+x12+x23>4) | (x3+x31+x23>4))))]
```

Taking the rate of LSE at 1 in 10^4 hours (close to the value of 9259 hours in [11]) and, scrub disk rate of once every 50 hours (rebuild times will be incorporated in the more detailed models in Sec. 4.3), the probability of failure is given in Figure 3(b); the total run takes 16041.6 secs (approx. 4.5 hours). Note the dramatic change of the failure probability by six orders of magnitude and the no longer strictly linear graph.

4.2 Simple Reliability Models for Large Disk Farms

IDEMA [20] has modelled disk reliability by specifying variable rates of failures by specifying the rates for the first 3 months, then for the next 6 months, etc. Qin et al. [7] have used a 3-state HMM for approximating this model (Figure 4). A disk is modelled as being in the burn-in phase (state 0), burnt-in phase (state 1) or failed state (state FS). Using the parameters in [7], we can investigate, say, a 100-node disk systems using the 3-HMM model (Figure 5); we can also investigate much larger disk systems from 200 to 2000 disks or more (we use a simple C++ program to generate the PRISM programs given the required number of disks). Given that the failure model in [7] assumes data loss on the second disk failure (*dataloss*), the number of states and transitions in the Markov model increase only linearly with number of disks; it is therefore possible to analyze even much larger disk systems in PRISM but we give results only up to 2000 disks (Table 2). Such results are not easy with other techniques. When the first failure occurs, an exponential repair process is started. In addition, an exponential process replaces the disk "before" end of life (EOL) of the disk. Note that with a 2000 disk system, the system fails within about 15 days due to only one type of disk failure (whole disk failure during burn-in and after), but this is optimistic as the model does not take into account latent sector failures.

#disks	states	edges	exectime(secs)	MTBF
50	102	399	0.066	161500
100	202	799	0.079	42148
500	1002	3999	0.462	2729
600	1202	4799	0.748	2071
700	1402	5599	1.270	1644
1000	2002	7999	4.793	965
2000	4002	15999	24.167	347

Table 2: MTBF (in hrs) computed by the query `R=? [F (s=dataloss)]` with parameters from [7], with s the state.

In addition to computing the survival time of the system for various nodes, we can compute the failure of the system at time t by model checking the following formula

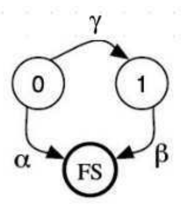

$$\begin{tikzcd} 0 \arrow[r, bend left, "\gamma"] \arrow[dr, "\alpha"'] & & 1 \arrow[dl, "\beta"] \\ & FS & \end{tikzcd}$$

Figure 4: 3-state HMM disk reliability model from [7]. γ is the burn-in rate; α and β are the pre- and post- burn-in failure rates.

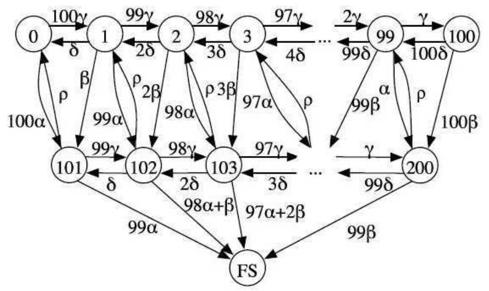


Figure 5: A 100-disk Markov reliability model assuming the 3-state HMM disk model. From [7].

9's	Time
5	4.5 hours
4	45 hours
3	415 hours
2	3487 hours
1	34354 hours

Table 3: Time interval for k 9's probability of availability.

`P=? [true U<=t (s=dataloss)]`. By using this formula for various values of t, we can also compute the time for k 9's reliability [2] which is given in Table 3.

4.2.1 Taking care of LSEs

To make the above model more realistic, let us combine the first two models: RAID5 with latent sector failures and the HMM models. It is not practical to use more than 8 to 10 devices as part of a parity group due to LSEs; higher number of devices makes the rebuild unlikely to complete without errors. Using the combined error model for a RAID5 across 10 disks, we need to have, for a 100 disk system, 10 instances of these; we can run these in parallel and the system fails as a whole if any of them fails. However, the state explosion for even this small system is tremendous: there are 36,439,964,604,647 states and 1,445,587,465,949,086 transitions (obtained by constructing the model in PRISM); note that each 10-disk HMM model has atleast 20 bits of state and a simple cross product results in atleast 20^{10} states. Instead of 10 instances, if we use 5 instances (for a composite 50 disk system), the resulting system is feasible for model checking as is (with 6,223,163 states and 122,712,702 transitions). With 500GB disks, and using the same previous parameters and hard error rate of 8E-06 per GB, the total time for model checking (both construction and execution) is 10799.4 secs (approx. 3 hours on an Opteron core).

In comparison with the basic (non-RAID5) 50-node HMM model (where any 2 errors anywhere causes the system to "lose" data), the 5 RAID5 50-node system is more robust even after taking into account the LSEs as

disksize GB	exectime secs	MTTDL hours
100	23701.58	251882
200	16655.61	148016
500	10604.99	66466
1000	8151.33	34939
1500	7263.93	23876
2000	6740.96	18241

Table 4: Time to data loss (MTTDL) for 5 copies of a RAID5 with 10-disk HMM-based model and handling LSEs during reconstruction.

k	etime	states	trans	MTTDL
5	8086	78430	1316612	66466
6	39033	361790	7033400	55525
7	154405	1480050	32429540	47712

Table 5: MTTDL of k copies of 10-disk RAID5 HMM model using symmetry reduction.

the 2 errors have to be in the same RAID5 instance for data loss in the latter case (Table 4).

To reduce the size of the models, we can exploit the technique of symmetry reduction that maps equivalent states into a single state. In SMP systems, for example, the caches are indistinguishable for certain properties: the system behaviour depends on whether it has cached a line or not and whether it is exclusive or shared, but it is not important for some properties which particular cache has the exclusive line. Such symmetric states can be factored out to get a much smaller ("quotient"ed) state space. When the symmetry reduction technique is applied to the disk system under consideration, we can now solve from 5 to 8 copies of the 10-disk RAID5 models (Table 5). The reduction in the number of states and transitions is, for the 5 copies of RAID5, by two orders of magnitude.

4.3 Weibull Modelling for Disk Systems

To make the disk model more realistic, we can use the Weibull reliability models which have given good results for modelling disks [18]. The 2-parameter (β, η) Weibull probability density function is given by:

$$f(t) = (\beta/\eta)(t/\eta)^{(\beta-1)}e^{-(t/\eta)^{\beta}}$$

where β is the shape parameter, η is the scale parameter (here, "life"). Note that Weibull models either model increasing failure rates (IFR; $\beta > 1$) or decreasing failure rates (DFR; $\beta < 1$); $\beta = 1$ makes it an exponential distribution. Since PRISM does not support anything other than exponential distributions, we need to model Weibull using a sum of exponential distributions (*expono-*

mials). Probabilistic model checking can, therefore, investigate many of the disk-based designs if Weibull models can be approximated well using exponential distributions. Malhotra and Reibman [10] have described methods for approximating many distributions such as deterministic, Weibull, or lognormal using exponentials.

4.3.1 Modelling of a Single Disk

Elerath et al. [11] have modelled errors resulting from whole failures of disks, and LSEs arising from the intensity of read traffic to disks using Weibull models. They are also used for the time taken for disk repairs, the rebuild time after a failure of a disk and scrubbing time:

- Time to operational failure (TTOp) with a 2-parameter Weibull (shape=1.12, scale=461386 hrs)
- Time to restore (TTR) with a 3-parameter Weibull (shape=2, scale=12 hours and offset 6 hours)
- Time to scrub (TTScr) with a 3-parameter Weibull (shape=3, scale=168 hours and offset 6 hours)
- Time to latent defect (TTLd) with shape=1 (an exponential distribution) and scale=9259 hours

We do not use the 3-parameter Weibull distribution in this paper due to its significant additional complexity in approximating distributions and the attendant increase in the state space. The intent of using the 3-parameter model is to signify that the repair or scrub time is at least a minimum amount and avoid having to use an unrealistic exponential time distribution; however, a deterministic repair/scrub time can as well be used as a good first approximation. Since we will be using M-stage Erlang models (M-Erlang) – a convolution of M exponentials – our modelling is simpler and it also captures some variation of the repair/scrub time.

4.3.2 Approximating the Weibull Distributions

The realization of approximations to the Weibull models is however tricky. The simple technique of matching moments such as the mean (1st) and variance (2nd) is useful in some cases. Consider the modelling of a Weibull IFR by a M-stage Erlang distribution:

$$E(t, M, \lambda) = 1 - \sum_{k=0}^{M-1} ((\lambda t)^k / k!) e^{-\lambda t}$$

The first two moments of the M-Erlang are given by

$$m_1 = M/\lambda \text{ and } m_2 = M(M+1)/\lambda^2$$

(hence, $\lambda = m_1/(m_2 - m_1^2)$ and $M = m_1^2/(m_2 - m_1^2)$) whereas those for Weibull IFR distributions are given by

$$m_1 = \eta\Gamma((\beta+1)/\beta) \text{ and } m_2 = \eta^2\Gamma((\beta+2)/\beta)$$

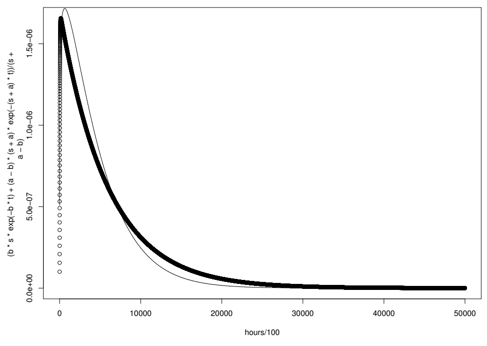


Figure 6: PDF of Weibull IFR (thin line) and for our exponomial approx. (thick line) for disk reliability.

Equating the two, we take M as the floor of the value and compute λ using the new value of M [10]. For TTR, we get M=3 and $\lambda = M/m_1 = 3/10.63472 = 0.2821$. For TTScr, we get M=8 and $\lambda = M/m_1 = 8/150.0206 = 0.05333$. Since an 8-stage approximation is too expensive, we will use a simpler 3-stage one, with $\lambda = 0.01922$.

For TTOp, we get M=1 (a single stage!), hence it is not useful; we need to use other methods. As mentioned earlier (Sec. 4.2), Qin et al. [7] have used a HMM model that achieves something close to a Weibull model by using 3-state and 4-state HMM models (these are also Cox models). We follow this line of work as there is intuitive justification for the model. A disk is modelled as being in the burn-in phase ($x0$), burnt-in phase ($x1$) or failed state ($x2$). Assuming the transitions are modelled exponentially with rates s ($x0$ to $x1$), a ($x0$ to $x2$), b ($x1$ to $x2$), we can derive the pdf of the fail state as follows

$$(b \cdot s \cdot e^{-bt} + (a-b) \cdot (s+a) \cdot e^{-(s+a)t})/(s+a-b).$$

Discovering values of s, a and b that match those by a given Weibull distribution has been done through a trial and error method as the publicly available tools such as EMpht [6] (expectation-maximization fitting for phase type distributions) did not give good results on parameters of interest such as η =461386 and β=1.12 (the algorithm can get stuck in an insignificant local minima or even a saddlepoint [6]). A reasonably good approximation for TTOp is a=1e-07, b=1.7e-06 and s=0.00035 (though the skew ("asymmetry" of distribution) and kurtosis ("peakedness") of these curves do not match that closely). The PDF of the failure density function for Weibull IFR and our approximation are given in Fig. 6.

4.3.3 Weibull Disk Approx. using only Exponentials

Using Elerath et al.'s work as the starting point but using deterministic distributions for repair and for scrub, we now discuss how we construct our composite disk model

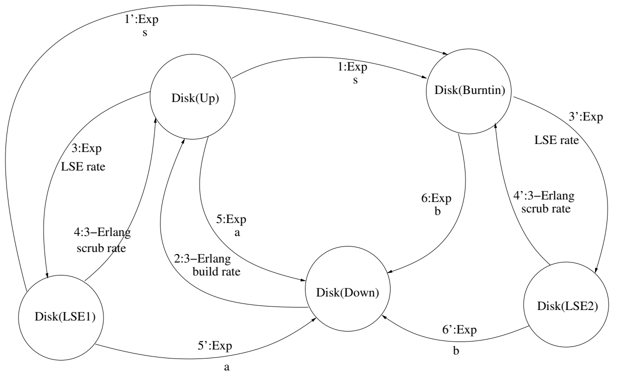


Figure 7: Our “Weibull” disk model approx. using only exponentials. Transitions are labelled with an id and the distribution (exponential, or M-Erlang). All transitions with the same id have the same rate (below). The transition from Disk(Down) to system error state is not given as it is relevant only with multi-disk RAID modelling when a LSE occurs elsewhere during rebuild.

(Fig. 7). When there are “competing” transitions out of a state (such as the Weibull transition for modelling whole disk failure and the exponential for modelling the LSE), we need to cross-product the substates resulting from approximating each of these “macro-transitions”. To avoid serious state explosion with multiple disks, our cross-product model of the disk adds states and transitions as few as possible. First, we add two Disk(LSE) states to take care of LSEs before and after the disk is completely burnt-in. Similarly, we add new transitions (such as between the Disk(LSE) state to the burnt-in state and from these two LSE states to the Disk(DOWN) state) but we have not added some transitions such as those modelling the substates corresponding to the scrubbing macrostate transition as their impact is likely to be negligible.

4.3.4 Modelling multiple disks in RAID5

By using the above model for a disk and running multiple instances of disks for a RAID5 configuration, we can compute time to data loss as well as the probability of failure at time t. Without symmetry reduction, the state explosion problem increases dramatically with the accuracy (number of bits needed to represent the state diagram) used to represent the approximation to a distribution using exponentials. Our results are given in Table 6. Note that we cannot run RAID5 models with 8 and more disks without symmetry reduction due to the serious state explosion; for a 7 disk RAID5 model, there are 125 million states and 1.34 billion transitions; the time to compute the MTTDL takes about 36 hours on a single core of an Opteron. With symmetry reduction, however, we can run 8 to 10-disk RAID5 models in significantly less time.

years	DDFs	sDDFs	years	DDFs	sDDFs
1	15.7	10.1	6	88.7	75.9
2	30.9	22.9	7	102.6	89.8
3	45.8	35.1	8	116.2	105.1
4	60.3	48.5	9	129.7	120.4
5	74.6	61.3	10	142.9	136.0

Table 7: Double-disk failures (DDF) per 1000 RAID5s with Weibull models in PRISM with symm. reduction, and with simulation (sDDF).

years	DDFs	sDDFs	years	DDFs	sDDFs
1	2.27	1.92	6	13.47	14.82
2	4.52	3.84	7	15.70	18.24
3	6.77	6.46	8	17.94	21.52
4	9.00	9.32	9	20.17	24.56
5	11.24	12.16	10	22.40	28.16

Table 8: Double-disk failures (DDF) per 1,000,000 RAID6/RAID-DPs with Weibull models in PRISM with symm. reduction, and with simulation (sDDF).

Note that Elerath et al. have used 8-disk RAID5 in their simulation studies. For β=1.12 and scrub rate of 168 hours, the double-disk failure (DDF) rate for 1000 RAID5 instances is given in Table 7; our results using the above models in PRISM are similar. The set of approximations used are reasonable but, at lower values of time, we do have a significant difference (such as a DDF of 10.1 at the end of the first year vs 15.7 in our model). Further, it remains mostly linear with time whereas it is mildly convex in the simulation results [11].

RAID6/RAID-DP Model Using the previous model of the disks for the RAID6/RAID-DP[4] configuration where failure results when there are either two whole disk failures and an LSE, or more than two whole disk failures, we get the following results (Table 8); the system failures are now much more rare than in the previous RAID5 case.

However, both the RAID5 and RAID-DP results show that PMC gives results that are not in very good agreement with the simulation results as the DDFs of the former consistently show a mild concavity with time whereas the latter show the opposite mostly (as they are much more accurate with respect to sampling from a Weibull distribution). The main reason is that the our approximation has a hazard function that increases rapidly in the beginning and becomes flat thereafter; this is different for the actual Weibull IFR which increases as $(t/\eta)^{(\beta-1)}$. Using a 4-state Cox model (found manually) did not improve the results[3]. Our attempt to use a 10-state Cox model (us-

[3]If the CDF for the exponomial is $1 - \Sigma c_i e^{-a_i t}$, $a_i \leq a_{i+1}$, the

disks	etime*	states	trans	symmstates	symmtrans	MTTDL
5	601.8	742586	5838767	12376	76749	1510785
6	9640.2	9653618	89408965	37128	257355	1029710
7	129663.6	125497034	1337588683	100776	763359	748776
8	2244.9*	1631461442	19662987617	251940	2053698	570302
9	8804.7*	21208998746	285135363111	587860	5099460	460601
10	25090.2*	275716983698	4089945286077	1293292	11832444	380293

Table 6: MTTDL (hours) in a Weibull-based RAID5 model with 5-10 disks. The number of states and transitions are given for models with and without symmetry reduction. *The execution time (etime) (in seconds) for 8-10 disks are given for the run with symmetry reduction; without it, the state space is so large that it is currently impossible.

ing EMpht [6] to compute the parameters) was not successful as the approximation generated is not as close to the actual Weibull distribution as the previous 4-state Cox model. The simplified state diagram has an effect also but this is small as we have experimentally confirmed. Another minor contributor for the differences could be that the simulation results are based on averages of 10 experiments; on one set of results, the coefficient of variation is around 0.06 while it is around 1E-6 for PMC.

5 Conclusions

Probabilistic model checking is useful for analyzing practical disk subsystems, and gives results with high accuracy if distributions are exponential. For non-exponential ones, it can give good results if good approximations are possible using a few states. Our experience indicates that getting good approximations is the current bottleneck. Disk systems with up to 10 disks are easy to model check in PRISM if we use symmetry reduction; larger systems are difficult, from a computational viewpoint, due to state explosion; however, large parity groups are not useful as the rebuild in RAID-like systems then fails with high probability. If parallelism in large clusters can be exploited for the model checking problem, analysis of larger disk systems is possible.

We have used symmetry reduction [23] as a technique for reducing the state space of the system. However, the stringent requirement of "full symmetry" makes its applicability somewhat narrow; we plan to study this aspect as future work. Another problem is that the language for expressing a model is based on process algebra and hence it is not easy to code combinatorial reasoning often needed in reliability studies.

Acknowledgments Thanks are due to David Parker, T Schwarz, Jay Wylie, Hari Prasad, Kevin Greenan and A.Amer for their help. Funding from Kumar Malavalli and NSF is gratefully acknowledged.

hazard function is $\Sigma c_i a_i e^{-a_i t}/\Sigma c_i e^{-a_i t}$. If $a_1 \ll a_2 \ll a_3$, it is approx. $a_1 - a_1(c_2/c_1)e^{-a_2 t}$.

References

[1] L N. Bairavasundaram et al., "An Analysis of Latent Sector Errors in Disk Drives," SIGMETRICS'07, Jun'07.

[2] J F Paris, T J E Schwarz, "On the Possibility of Small, Service-Free Disk Based Storage Systems," 3rd Intl. Conf. on Availability, Reliability and Security, Mar'08.

[3] S. Juneja, P. Shahabuddin, "Rare-event Simulation Techniques: An Introduction and Recent Advances," Handbook on Simulation, Elsevier, 2006.

[4] Peter Corbett, et al., "Row-Diagonal Parity for Double Disk Failure Correction," FAST04, 2004.

[5] A M Blum, et al., "Modeling and Analysis of System Dependability Using the System Availability Estimator," Fault-Tolerant Computing, 1994. FTCS-24.

[6] S. Asmussen, et al., "Fitting phase-type distribution via the EM algorithm," Scand. J. Statist. 23, 419-441 (1996)

[7] Q Xin, T J E Schwarz, E L Miller, "Disk Infant Mortality in Large Storage Systems," MASCOTS 2005.

[8] Malhotra, M. and Trivedi, K. S. "Reliability analysis of redundant arrays of inexpensive disks," J. Parallel Distrib. Comput. 17, 146–151, 1993.

[9] Sahner R A, Trivedi K, Puliafi to A, "Performance & Reliability Analysis of Computer Systems: An Example-Based Approach Using SHARPE," Academic Publishers, 1997.

[10] Manish Malhotra, Andrew Reibman, "Selecting and Implementing Phase Approximations for Semi-Markov models," Commun. Statist. -Stochastic Models, 9(4), 1993.

[11] J. G. Elerath, M. Pecht, "Enhanced Reliability Modeling of RAID Storage Systems," DSN, 2007.

[12] E M Clarke, D Peled, O Grumberg, "Model Checking," MIT Press, 2000.

[13] www.prismmodelchecker.org

[14] H A Oldenkamp, "Probabilistic model checking. A comparison of tools," Master's Thesis, Univ. Twente, May'07.

[15] http://www.cs.utwente.nl/ zapreevis/mrmc/.

[16] K. K. Rao, James Lee Hafner, Richard A. Golding, "Reliability for Networked Storage Nodes," DSN 2006: 237-248.

[17] James Lee Hafner and KK Rao, “Notes on Reliability Models for Non-MDS Erasure Codes,” IBM Research Report RJ10391, 2006.

[18] Bianca Schroeder, Garth A. Gibson, “Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?,” FAST’07.

[19] Ahmed Amer et al., “Increased Reliability with SSPiRAL Data Layouts,” IEEE/ACM MASCOTS 2008. Baltimore.

[20] Intl. Disk Drive Equipment & Materials Assoc. (IDEMA), “R2-98: Specifi cation of hard disk drive reliability,” 1998.

[21] M. Kwiatkowska, “Model Checking for Probability and Time: From Theory to Practice,” In Proc. 18th IEEE Symp. on Logic in Computer Science (LICS’03), Jun’03.

[22] A. Krioukov et al., “Parity lost and parity regained,” FAST’08, 2008.

[23] M. Kwiatkowska, G. Norman and D. Parker. Symmetry Reduction for Probabilistic Model Checking. CAV’06, vol. 4144 LNCS, August 2006.