Analysis of Reliability Dynamics of SSD RAID

Yongkun Li, *Member, IEEE* Patrick P. C. Lee, *Member, IEEE* John C. S. Lui, *Fellow, IEEE*

Abstract—Solid-state drives (SSDs) have been widely deployed in desktops and data centers. However, SSDs suffer from bit errors, and the bit error rate is *time dependent* since it increases as an SSD wears down. Traditional storage systems mainly use parity-based RAID to provide reliability guarantees by striping redundancy across multiple devices, but the effectiveness of traditional RAID schemes in SSDs remains debatable. In particular, an open problem is how different parity distributions over multiple devices influence the reliability of an SSD RAID array. That is, should we evenly distribute parties as suggested by conventional wisdom, or unevenly distribute parties as recently proposed for SSD RAID? To address this fundamental problem, we propose the first analytical model to quantify the reliability dynamics of an SSD RAID array as it ages. Specifically, we develop a "non-homogeneous" continuous time Markov chain model, and derive the transient reliability solution. We validate our model via trace-driven simulation and conduct numerical analysis to analyze the reliability dynamics of SSD RAID arrays subject to different parity distributions, error rates, and SSD array configurations. Our model enables system practitioners to decide the appropriate parity distribution based on their reliability requirements.

Index Terms—Solid-state Drives, RAID, Reliability, CTMC, Transient Analysis

1 Introduction

SSDs emerge to be the next-generation storage medium. Today's SSDs mostly build on NAND flash memories, and provide several enhancements over hard disks including better I/O performance, lower energy consumption, and higher shock resistance. As SSD prices continue to drop nowadays, they have been widely deployed in desktops and large-scale data centers [10, 14].

Even though enterprise SSDs provide multiple enhancements over hard disks, they are susceptible to wear-outs and bit errors. First, SSDs can only write data to clean pages, and so they necessitate erase operations to reset flash blocks back to the clean state (see §2.1 for details). However, each flash block can only tolerate a limited number of erasures before wearing out. Second, bit errors are very common in SSDs due to read disturbs, program disturbs, and retention errors (see §2.1 for details). Moreover, bit error rates of SSDs are time-varying, and in particular, increase as SSDs issue more erase operations [5, 12, 27, 35, 38]. Last, bit errors of SSDs become more severe when the density of flash cells increases and the feature size decreases [13]. Thus, SSD reliability remains a legitimate concern

RAID (redundant array of independent disks) [32] provides an option to improve reliability of SSDs. Using parity-based RAID (e.g., RAID-4, RAID-5), the original data is encoded into *parities*, and the data and parities are striped across multiple SSDs to provide storage redundancy against device failures. RAID has been widely used in tolerating hard disk failures. However, traditional RAID introduces a different reliability problem to SSDs. Conventional wisdom suggests that parities should be *evenly* distributed across multiple drives to achieve better load balancing, such as in RAID-5. Balakrishnan *et al.* [2] find that RAID-5 introduces the problem of correlated device failures, in which all SSDs wear at the same rate and fail simultaneously, thereby causing data loss. To address this problem, they propose *Diff-RAID* as a RAID-5 variant that enhances the SSD RAID reliability by keeping *uneven* parity distributions.

One open problem is how different parity distributions generally influence the reliability of an SSD RAID array subject to different error rates and array configurations. *In other words, should we distribute parities evenly or unevenly across multiple SSDs with respect to the SSD RAID reliability?* This motivates us to characterize the SSD RAID reliability using analytical modeling, which enables us to readily tune different input parameters and determine their impact on reliability. However, analyzing the SSD RAID reliability is challenging. Unlike hard disk drives in which error arrivals are commonly modeled as a constant-rate Poisson process (e.g., see [29, 34]), SSDs have an increasing error arrival rate as they wear down with more erase operations.

In this paper, we formulate a continuous time Markov chain (CTMC) model to analyze the effects of different parity placement strategies, such as traditional RAID-5 and Diff-RAID [2], on the reliability dynamics of an SSD RAID array. To capture the time-varying bit error rates in SSDs, we formulate a *non-homogeneous* CTMC model, and conduct transient analysis to derive the reliability metric (which we formally define in §2) at any specific time instant. Thus, the model characterizes how the reliability of an SSD RAID array changes over time

- *Yongkun Li is with the School of Computer Science and Technology, University of Science and Technology of China. (Email: yongkunlee@gmail.com)*
- *Patrick P. C. Lee and John C. S. Lui are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. (Emails: {pclee,cslui}@cse.cuhk.edu.hk)*
- *An earlier conference version of the paper appeared in IEEE SRDS 2013 [23]. In this journal version, we extend the prior work for general error rates. We also propose a new optimization technique to speed up the model computation. Finally, we include additional evaluation results.*

throughout its whole lifespan. To the best of our knowledge, this is the *first* analytical study that quantifies the reliability dynamics of an SSD RAID array.

This paper makes two key contributions:

- We formulate a non-homogeneous CTMC model to characterize the reliability dynamics of an SSD RAID array. We use the uniformization technique [8, 17, 33] to derive the transient reliability of the array. Since the state space of our model increases with the SSD size, we develop optimization techniques to reduce the computational cost of transient analysis. We also quantify the corresponding error bounds of the uniformization and optimization techniques. Using the SSD simulator [1], we validate our model via trace-driven simulation.
- We conduct extensive numerical analysis to compare the reliability of an SSD RAID array under RAID-5 and Diff-RAID [2]. We observe that Diff-RAID, which places parities unevenly across SSDs, only improves the reliability over RAID-5 when the error rate is not too large, while RAID-5 is reliable enough if the error rate is sufficiently small. On the other hand, when the error rate is very large, neither RAID-5 nor Diff-RAID can provide high reliability, so increasing fault tolerance (e.g., RAID-6 or a stronger ECC) becomes necessary.

The rest of this paper proceeds as follows. In §2, we formulate our model that characterizes the reliability dynamics of an SSD RAID, and formally define the reliability metric. In §3, we derive the transient system state using uniformization and optimization techniques. In §4, we validate our model via trace-driven simulation. In §5, we present numerical results on how different parity placement strategies influence the RAID reliability. In §6, we review related work, and finally in §7, we conclude the paper.

2 System Model

In this section, we formulate a non-homogeneous continuous-time Markov chain (CTMC) model to characterize the reliability dynamics of an SSD RAID array for different parity distributions across multiple SSDs. We focus on two parity-based RAID schemes: RAID-5 [32], which distributes parties evenly across all drives and achieves load balancing, and Diff-RAID [2], a RAID-5 variant which operates by (i) distributing parties unevenly across all drives and (ii) redistributing parities each time when a worn-out SSD is replaced so that the oldest SSD always has the most parities and wears out first. Note that we can adapt our analysis of Diff-RAID for *any* uneven parity distribution (see §2.2). For example, we can apply our analysis to RAID-4, which places all parities in a single SSD, or WeLe-RAID [39], which chooses a parity distribution that balances the aging rates of all SSDs. Thus, our analysis mainly focuses on RAID-5 and Diff-RAID. Our primary goal is to provide a comprehensive study on the reliability dynamics of SSD RAID arrays under different parity distributions.

2.1 Background on Flash Memory and SSD

Flash memory is composed of flash cells, each of which is a CMOS transistor with a floating gate between the control gate and the channel. Flash cells store data by trapping charge on the floating gate, and each gate may store one or more bits. Flash memory supports three primary operations: program, erase, and read, and data can only be programmed to erased flash cells.

Flash memory can be classified as two types according to the organization of flash cells: NAND flash and NOR flash. Today's SSDs mainly build on NAND flash memory. An SSD is composed of multiple *chips*, each of which is organized in multiple *blocks*. Each block typically has 64 or 128 fixed-size *pages* of size 4KB or 8KB each. Both read and write operations are performed in units of pages. Data can only be written (done by flash-level program operations) to *clean* pages. SSDs perform an erase operation in units of blocks to reset all pages of a block into clean pages. To improve write performance, SSDs use *out-of-place* writes, i.e., to update a page, the new data is written to a clean page while the original page is marked as invalid. We refer readers to [1] for a more detailed description about the SSD organization.

Flash memory is not error-free, and may fail in various ways. In particular, bits stored in flash cells may become corrupted due to program disturbs and read disturbs. Almost all errors appear due to cells with unintended voltage. For example, programing one page may make all the pages in the block experience weak programming voltage, which causes data errors. Similarly, reading data from a page may also have a weak programming effect on other pages in the same block, which again leads to data corruption. In addition to program disturbs and read disturbs, bit errors may also happen due to data retention, mainly due to charge loss in flash cells. Finally, other factors like quantum-level noise effects and erratic tunneling may also lead to bit errors in flash memory. Bit errors in flash memory have been well studied by the literature (e.g., [5, 12, 27, 35, 38]). Therefore, both bit-level protection like ECC and device-level protection like RAID are necessary for SSDs.

2.2 SSD RAID Basics

We now describe the organization of an SSD RAID. Table 1 lists the main notation used in this paper.

Figure 1 shows an SSD RAID organization. We consider the device-level RAID organization where the array is composed of $N+1$ SSDs numbered from 0 to N. In this paper, we focus on *single-fault tolerance*, in which the array can tolerate against a single SSD failure, as assumed in RAID-5 and Diff-RAID [2] Each SSD is divided into multiple non-overlapping *chunks*, each of which can be mapped to one or multiple physical pages. The array is further divided into *stripes*, each of which is a collection of $N+1$ chunks from the $N+1$ SSDs. Within a stripe, there are N data chunks, and one parity chunk encoded from the N data chunks. We call a chunk an

Specific Notations of SSD		
M	:	Erasure limit of each block (e.g., 10K)
B	:	Total number of blocks in each SSD
$\lambda_i(t)$	:	Error rate of a chunk in SSD i at time t
Specific Notations of RAID Array		
N	:	Number of data drives (i.e., an array has $N+1$ SSDs)
S	:	Total number of stripes in an SSD RAID array
p_i	:	Fraction of parity chunks in SSD i, and $\sum_{i=0}^{N} p_i = 1$
k	:	Total number of erasures performed on SSD RAID array (i.e., system age of the array)
k_i	:	Number of erasures performed on each block of SSD i (i.e., age of SSD i)
T	:	Average inter-arrival time of two consecutive erase operations on SSD RAID array
$\pi_j(t)$	:	Probability that the array has j stripes that contain exactly one erroneous chunk each, $(0 \le j \le S)$
$\pi_{S+1}(t)$	:	Probability that at least one stripe of the array contains more than one erroneous chunk, so $\sum_{j=0}^{S+1} \pi_j(t) = 1$
$R(t)$	:	Reliability at time t, i.e., probability that no data loss happens until time t, $R(t) = \sum_{j=0}^{S} \pi_j(t)$

TABLE 1: Main notation used in this paper.

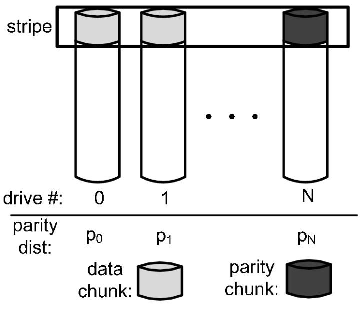


Fig. 1: Organization of an SSD RAID array.

erroneous chunk when uncorrectable bit errors appear in that chunk; or a *correct chunk* otherwise. Since we focus on single-fault tolerance, we require that each stripe contains at most one erroneous chunk without data loss so that it can be recovered from other surviving chunks in the same stripe.

Suppose that each SSD contains B blocks, and the array contains S stripes (i.e., S chunks per SSD). For simplicity, we assume that all S stripes are used for data storage, although in practice an SSD is usually over-provisioned [1]. To generalize our analysis, we organize parity chunks in the array according to some probability distribution. Let SSD i contain a fraction p_i of parity chunks (where $0 \le i \le N$ and $0 \le p_i \le 1$). In the special case of RAID-5, parity chunks are *evenly placed* across all devices, so $p_i = \frac{1}{N+1}$ for all i if the array consists of $N+1$ drives. For Diff-RAID, p_i's can be arbitrarily defined subject to the condition $\sum_{i=0}^{N} p_i = 1$.

Each block in an SSD can only sustain a limited number of erasures, and it is worn out after the limit. We denote the erasure limit by M. To enhance the durability of an SSD, efficient wear-leveling techniques are often deployed in the flash translation layer so as to balance the number of erasures across all blocks within the SSD. To simplify our analysis, we assume in this paper that each SSD achieves perfect wear-leveling such that every block within an SSD has exactly the same number of erasures. Let k_i ($\le M$) be the number of erasures that have been performed on each block in SSD i, where $0 \le i \le N$. We denote k_i as the *age* of each block in SSD i, or equivalently, the age of SSD i as perfect wear-leveling is assumed. When an SSD reaches its erasure limit, we assume that it is replaced by a new SSD. For simplicity, we treat k_i as a continuous value. Let k be the total number of erase operations that the whole array has processed, and we call k the *system age* of the array.

2.3 Age Characterization

In this subsection, we characterize the age of each SSD for a given RAID scheme. In particular, we derive k_i, denoting the age of SSD i, when the whole array has already performed a total of k erase operations. This characterization enables us to model the error rate in each SSD accurately (see §2.4).

We first quantify the *aging rate* of each SSD in an array. Let r_i be the aging rate of SSD i. Note that updating a data chunk also has the parity chunk updated. We first suppose that each data chunk has the same probability of being accessed. On average, the *ratio* of the aging rate of SSD i to that of SSD j can be expressed as [2]:

$$\frac{r_i}{r_j} = \frac{p_i N + (1-p_i)}{p_j N + (1-p_j)}. \tag{1}$$

Equation (1) states that the parity chunk ages N times faster than each data chunk. Given r_i's, we can quantify the probability of SSD i being the target drive for each erase operation, which we denote by q_i. We model q_i by making it proportional to the aging rate of SSD i, i.e.,

$$q_i = \frac{r_i}{\sum_{i=0}^{N} r_i} = \frac{p_i N + (1-p_i)}{\sum_{i=0}^{N}(p_i N + (1-p_i))}. \tag{2}$$

We first characterize the age of Diff-RAID, which places parities unevenly and redistributes parity chunks after the worn-out SSD is replaced. Diff-RAID aims to maintain the age ratios and always wear out the oldest SSD first. To mathematically characterize the system age of Diff-RAID, define A_i as the remaining fraction of erasures that SSD i can sustain right after an SSD replacement. Clearly, $A_i = 1$ for a brand-new drive and $A_i = 0$ for a worn-out drive. Without loss of generality, we assume that the drives are sorted by A_i in descending order, i.e., $A_0 \ge A_1 \ge \cdots \ge A_N$, and we have $A_0 = 1$ as it is the newly replaced drive. Diff-RAID performs parity redistribution after each drive replacement to guarantee that the aging ratio in Equation (1) remains unchanged. Therefore, the remaining fraction of erasures for each drive will *converge*, and the values of A_i's in the steady state are given by [2]:

$$A_i = \frac{\sum_{j=i}^{N} r_j}{\sum_{j=0}^{N} r_j} = \frac{\sum_{j=i}^{N}(p_j N + (1-p_j))}{\sum_{j=0}^{N}(p_j N + (1-p_j))}, \quad 0 \le i \le N. \tag{3}$$

In this paper, we study Diff-RAID at the steady state, i.e., after the age distribution right after each drive replacement converges. Thus, the remaining fractions of erasures of SSDs in Diff-RAID always follow the distribution of A_i's in Equation (3).

We now characterize k_i for Diff-RAID. Recall that each SSD has B blocks. Due to perfect wear-leveling, every block of SSD i has the same probability q_i/B of being the target block for an erase operation. Thus, if the array has processed k erase operations, the age of SSD i is:

$$\textbf{Diff-RAID:}\quad k_i = \Big(\frac{kq_i}{B} \bmod \frac{q_i}{q_N}(M - k_{N0})\Big) + k_{i0}, \quad (4)$$

where $k_{i0} = M(1 - A_i)$ is the initial number of times that each block of SSD i has been erased right after a drive replacement, and the operator mod denotes the modulo operation. The rationale of Equation (4) is as follows. Since we sort the SSDs by A_i in descending order, SSD N always has the highest aging rate and will be replaced first. Thus, after each block of SSD N has performed $(M - k_{N0})$ erasures, SSD N will be replaced, and each block of SSD i has just been erased $\frac{q_i}{q_N}(M - k_{N0})$ times. Therefore, for SSD i, a drive replacement happens when each block has been erased *every* $\frac{q_i}{q_N}(M - k_{N0})$ times. Moreover, the initial number of erasures on each block of SSD i right after a drive replacement is k_{i0}. Thus, the age of SSD i is derived as in Equation (4). Since $k_{i0} = M(1 - A_i)$ and $A_N = q_N$, k_i can be rewritten as:

$$\textbf{Diff-RAID:}\quad k_i = ((kq_i/B) \bmod Mq_i) + M(1 - A_i). \quad (5)$$

We now characterize the age of RAID-5. RAID-5 keeps parity chunks intact, and will not redistribute them during a drive replacement. So after the array has performed k erasures, each block of SSD i has just performed kq_i/B erasures, and an SSD will be replaced every time when each block performed M erasures. Thus,

$$\textbf{RAID-5:}\quad k_i = (kq_i/B) \bmod M. \quad (6)$$

Note that Equations (5) and (6) are general enough to characterize the age of SSD i for any given aging ratio $\frac{r_i}{r_j}$'s. Thus, the age characterizations for both Diff-RAID and RAID-5 (with Equations (5) and (6)) are also applicable to non-uniform workload as long as the aging ratio is given, which can be measured by replaying workload with a simulator. In particular, the aging ratio of Diff-RAID is mainly governed by the parity distribution, while that of RAID-5 depends on the workload. In other words, SSDs in a RAID-5 array may also have different aging rates under non-uniform workload. We will further study the impact of non-uniform workload on RAID reliability in §5.4 via numerical analysis.

2.4 Reliability Characterization

We model the error rate of an SSD as a function of the SSD age. We assume that the error arrival processes of different chunks in an SSD are independent. Since different chunks in an SSD have the same age, they must have the same error rate. We let $\lambda_i(t)$ represent the error rate of each chunk in SSD i at time t, and model it as a function of the number of erasures on SSD i at time t, which is denoted by $k_i(t)$ (the notation t may be dropped if the context is clear). Furthermore, to reflect that bit errors increase with the number of erasures, we model the error rate based on a Weibull distribution [36], which has been widely used in reliability engineering. Formally,

$$\lambda_i(t) = c\alpha(k_i(t))^{\alpha-1}, \quad \alpha > 1, \quad (7)$$

where α is called the *shape parameter* and c is a constant.

We assume that the error rate of an SSD varies with the number of erasures on the SSD only, and the error rate between two adjacent erasures is constant. Specifically, let t_k be the time point of the k^{th} erasure on the array. Then during the time period (t_k, t_{k+1}) (i.e., between the k^{th} and $(k+1)^{th}$ erasures), the error rate remains unchanged and is determined by the value of k. Thus, we can express $k_i(t) = k_i(k)$ if $t \in (t_k, t_{k+1})$, and the function $k_i(k)$ is determined by Equation (5) and (6). We further assume that the error arrivals are modeled as a Poisson process with the error rate $\lambda_i(t)$.

We now formulate a continuous-time Markov chain (CTMC) model to characterize the reliability dynamics of an SSD RAID array. Recall that the array provides single-fault tolerance for each stripe. We say that the CTMC is at state i if and only if the array has i stripes that contain exactly one erroneous chunk each, where $0 \le i \le S$. Data loss happens if any one stripe contains more than one erroneous chunk, and we denote this state by $S+1$. Let $X(t)$ be the system state at time t. Formally, we have $X(t) \in \{0, 1, ..., S+1\}, \forall t \ge 0$. To derive the system state, we let $\pi_j(t)$ be the probability that the CTMC is at state j at time t $(0 \le j \le S+1)$, so the system state can be characterized by the vector $\boldsymbol{\pi}(t) = (\pi_0(t), \pi_1(t), ..., \pi_{S+1}(t))$.

Let us consider the state transition of the CTMC, as depicted in Figure 2. For each stripe, if it contains one erroneous chunk, then the erroneous chunk can be reconstructed from the surviving chunks in the same stripe. We assume that only one stripe can be reconstructed at a time, and that the reconstruction time follows an exponential distribution with rate μ. To elaborate, suppose that the RAID array is currently at state j, if an erroneous chunk appears in one of the $(S-j)$ stripes that originally have no erroneous chunk, then it will move to state $j+1$ with rate $(S-j)\sum_{i=0}^{N}\lambda_i(t)$; if an erroneous chunk appears in one of the j stripes that already have another erroneous chunk, then the system will move to state $S+1$ (the state of data loss) with rate $j\sum_{i=0}^{N}\lambda_i(t)$.

We now define the *reliability* of an SSD RAID array at time t, and denote it by $R(t)$. Formally, we define it as the probability that the array has not encountered data loss until time t, and we have

$$R(t) = \sum_{j=0}^{S}\pi_j(t). \quad (8)$$

Note that our model captures the time-varying nature

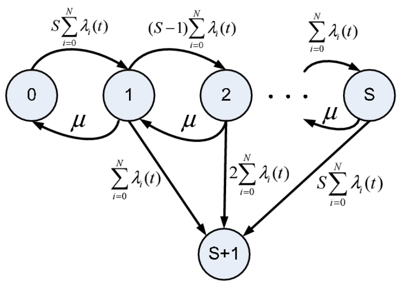

$S\sum_{i=0}^{N}\lambda_i(t)$
$(S-1)\sum_{i=0}^{N}\lambda_i(t)$
$\sum_{i=0}^{N}\lambda_i(t)$
0
1
2
S
μ
μ
μ
$\sum_{i=0}^{N}\lambda_i(t)$
$2\sum_{i=0}^{N}\lambda_i(t)$
$S\sum_{i=0}^{N}\lambda_i(t)$
S+1

Fig. 2: State transition of the non-homogeneous CTMC.

of reliability over the lifespan of the SSD array. The CTMC is *non-homogeneous*, because the error arrival rate $\lambda_i(t)$ is time-varying depending on the system age. Next, we show how to analyze this non-homogeneous CTMC.

3 Transient Analysis of CTMC

In this section, we describe how to derive $\boldsymbol{\pi}(t)$, the system state of an SSD RAID array at any time t, and then compute the instantaneous reliability $R(t)$ based on Equation (8). There are three major challenges in deriving $\boldsymbol{\pi}(t)$. First, it involves transient analysis, which is different from conventional steady state analysis. Second, the underlying CTMC $\{X(t), t \geq 0\}$ is *non-homogeneous* as the error arrival rate $\lambda_i(t)$ is time-varying. Third, the CTMC can have a very large state space, thereby making the derivation computationally expensive.

In the following, we present the mathematical framework for analyzing the non-homogeneous CTMC so as to compute the transient system state. We construct an algorithm based on the mathematical analysis. We also develop optimization techniques to speed up our analysis when the CTMC has a large state space.

3.1 Mathematical Analysis Framework

Note that error rates within a period (t_k, t_{k+1}) $(k = 0, 1, 2, ...)$ are assumed to be constant. If we only focus on a particular time period of the CTMC, i.e., $\{X(t), t_k < t \leq t_{k+1}\}$, then it becomes time-homogeneous. Therefore, an intuitive way to derive the transient solution of the CTMC $\{X(t), t \geq 0\}$ is to divide it into multiple time-homogeneous CTMCs $\{X(t), t_k < t \leq t_{k+1}\}$ $(k = 0, 1, 2...)$. Then we leverage the *uniformization* technique [8, 17, 33] to analyze these time-homogeneous CTMCs one by one in time ascending order. Uniformization is a well-established technique to solve the transient solution of a CTMC. Therefore, to derive $\boldsymbol{\pi}(t_{k+1})$, we first derive $\boldsymbol{\pi}(t_1)$ from the initial state $\boldsymbol{\pi}(0)$. We then take $\boldsymbol{\pi}(t_1)$ as the initial state and derive $\boldsymbol{\pi}(t_2)$ from $\boldsymbol{\pi}(t_1)$, and so on.

However, this intuitive approach may take a prohibitively long time to derive $\boldsymbol{\pi}(t_{k+1})$ when k is very large. Recall that k denotes the number of erasures performed on an SSD RAID array. It can grow up to $(N+1)BM$, where both B (the number of blocks in an SSD) and M (the erasure limit) can be huge, say, B =100K and M=10K (see §5). Therefore, applying the uniformization technique directly is computationally infeasible to derive the reliability, especially when the array performs a lot of erasures.

Thus, we propose an optimization technique for our analysis by aggregating multiple time periods and fixing the duration of the aggregated periods. Our observation is that the difference of the generator matrices at two consecutive periods is generally very small. Thus, we combine s consecutive periods together into an *epoch*, where s is called the *step size*. Furthermore, we fix the epoch length as sT, where T denotes the average inter-arrival time of two consecutive erase operations. To analyze the non-homogeneous CTMC over s periods $\{X(t), lsT < t \leq (l+1)sT\}$ (where $l = 0, 1, ...$), we approximate it with another time-homogeneous CTMC denoted by $\{\tilde{X}(t), lsT < t \leq (l+1)sT\}$. The derivation of $\boldsymbol{\pi}((l+1)sT)$ given $\boldsymbol{\pi}(lsT)$ proceeds as follows.

Step 1: Constructing a time-homogeneous CTMC $\{\tilde{X}(t), lsT < t \leq (l+1)sT\}$ with generator matrix $\tilde{\boldsymbol{Q}}_l$. Note that there are s periods in the epoch $(lsT, (l+1)sT)$. We denote the generator matrices of the original Markov chain $\{X(t)\}$ during each of the s periods by $\boldsymbol{Q}_{ls}$, $\boldsymbol{Q}_{ls+1}$, ... , $\boldsymbol{Q}_{(l+1)s-1}$. To construct $\{\tilde{X}(t), lsT < t \leq (l+1)sT\}$, we define $\tilde{\boldsymbol{Q}}_l$ as a function of the s generator matrices.

$$\tilde{\boldsymbol{Q}}_l = f(\boldsymbol{Q}_{ls}, \boldsymbol{Q}_{ls+1}, ..., \boldsymbol{Q}_{(l+1)s-1}), \quad l = 0, 1, ... \quad (9)$$

We will discuss how to construct the generator matrix $\tilde{\boldsymbol{Q}}_l$ for different types of error rates in later discussion.

Step 2: Deriving the system state $\tilde{\boldsymbol{\pi}}((l+1)sT)$ in the time-homogeneous CTMC $\{\tilde{X}(t)\}$. To derive the system state at time $(l+1)sT$, which we denote as $\tilde{\boldsymbol{\pi}}((l+1)sT)$, we solve the Kolmogorov's forward equation and we have

$$\tilde{\boldsymbol{\pi}}((l+1)sT) = \tilde{\boldsymbol{\pi}}(lsT)\sum_{n=0}^{\infty}\frac{(\tilde{\boldsymbol{Q}}_l sT)^n}{n!}, \; l = 0, 1, ... \quad (10)$$

where the initial state is $\tilde{\boldsymbol{\pi}}(0) = \boldsymbol{\pi}(0)$.

Step 3: Applying uniformization to solve Equation (10). We let $\tilde{\boldsymbol{P}}_l = \boldsymbol{I} + \frac{\tilde{\boldsymbol{Q}}_l}{\tilde{\Lambda}_l}$ where $\tilde{\Lambda}_l \geq \max_{ls \leq k \leq (l+1)s-1} \max_{0 \leq i \leq S+1} |q_{i,i}(k)|$. Based on the uniformization technique [8], the system state at time $(l+1)sT$ can be derived as follows.

$$\tilde{\boldsymbol{\pi}}((l+1)sT) = \sum_{n=0}^{\infty} e^{-\tilde{\Lambda}_l sT}\frac{(\tilde{\Lambda}_l sT)^n}{n!}\tilde{\boldsymbol{v}}_l(n), \; l = 0, 1, ... \quad (11)$$

where $\tilde{\boldsymbol{v}}_l(n) = \tilde{\boldsymbol{v}}_l(n-1)\tilde{\boldsymbol{P}}_l$ and $\tilde{\boldsymbol{v}}_l(0) = \tilde{\boldsymbol{\pi}}(lsT)$. The system initial state is $\tilde{\boldsymbol{\pi}}(0) = \boldsymbol{\pi}(0)$.

Step 4: Truncating the infinite summation in Equation (11) with a quantifiable error bound. Since Equation (11) requires an infinite summation, to make the computation feasible, we drop the negligible terms after an appropriate point. In particular, we denote the truncation point for the epoch $(lsT, (l+1)sT)$ by U_l and denote the system state at time $(l+1)sT$ after truncation by $\hat{\boldsymbol{\pi}}((l+1)sT)$. Based on Equation (11) and the truncation point U_l, $\hat{\boldsymbol{\pi}}((l+1)sT)$ can be easily expressed as follows.

$$\hat{\boldsymbol{\pi}}((l+1)sT) = \sum_{n=0}^{U_l} e^{-\tilde{\Lambda}_l sT}\frac{(\tilde{\Lambda}_l sT)^n}{n!}\hat{\boldsymbol{v}}_l(n), l = 0, 1, ... \quad (12)$$

where $\hat{\boldsymbol{v}}_l(n) = \hat{\boldsymbol{v}}_l(n-1)\tilde{\boldsymbol{P}}_l$ and $\hat{\boldsymbol{v}}_l(0) = \hat{\boldsymbol{\pi}}(lsT)$. Note that the initial state in the current epoch $\hat{\boldsymbol{v}}_l(0)$ is the solution obtained from the previous epoch, which also applies truncation, and hence is different from $\tilde{\boldsymbol{v}}_l(0)$ in Equation (11). However, for simplicity, we still assume that system starts from the same state, so $\hat{\boldsymbol{\pi}}(0) = \boldsymbol{\pi}(0)$.

To determine the truncation point in each epoch, we quantify the error caused by truncating the infinite series. Note that truncation is performed and an error is introduced in each epoch. Let ϵ_l be the (total) truncation error until time $(l+1)sT$.

$$\epsilon_l = ||\hat{\boldsymbol{\pi}}((l+1)sT) - \tilde{\boldsymbol{\pi}}((l+1)sT)||_1, \tag{13}$$

where $\tilde{\boldsymbol{\pi}}((l+1)sT)$ and $\hat{\boldsymbol{\pi}}((l+1)sT)$ are computed from Equations (11) and (12), respectively. Based on our definition, we can bound ϵ_l in Theorem 1.

Theorem *1:* Given the step size s and the truncation point U_l for each epoch $(lsT, (l+1)sT)$, the truncation error until time $(l+1)sT$ is

$$\epsilon_l \le \epsilon_{l-1} + \Big(1 - \sum\nolimits_{n=0}^{U_l} e^{-\tilde{\Lambda}_l sT}(\tilde{\Lambda}_l sT)^n/n!\Big),\ l = 0, 1, \ldots \tag{14}$$

where $\epsilon_0 = ||\hat{\boldsymbol{\pi}}(0) - \boldsymbol{\pi}(0)||_1 = 0$.

Proof: Please refer to §1 of the supplementary file.

Note that as shown in Equation (14), ϵ_{l-1} represents the total truncation error until time lsT, and the second part is exactly the bound of the error introduced only in the epoch $(lsT, (l+1)sT)$.

3.2 Upper and Lower Bounds of RAID Reliability

We thus far present a general process to compute the transient system state of a non-homogeneous CTMC. However, an open problem is how to construct the homogeneous CTMC and characterize the corresponding error. In this subsection, we discuss how to choose the generator matrix $\tilde{\boldsymbol{Q}}_l$ for different types of error rates (see Equation (9)). We also derive the upper and lower bounds of the system reliability.

Without loss of generality, we focus on a particular epoch $(lsT, (l+1)sT)$. Recall that each epoch contains s time periods, and the CTMC $\{X(t)\}$ in each period is time-homogeneous. We first write down the generator matrix of $\{X(t)\}$ in each period. Precisely, each element of the matrix $\boldsymbol{Q}_k$ $(ls \le k \le ls+s-1)$ is

$$q_{i,j}(k) = \begin{cases} -S\sum\nolimits_{i=0}^{N}\lambda_i(t), & i=j=0, \\ -\mu - S\sum\nolimits_{i=0}^{N}\lambda_i(t), & 0<i\le S,\ j=i, \\ (S-i)\sum\nolimits_{i=0}^{N}\lambda_i(t), & 0\le i<S, j=i+1, \\ i\sum\nolimits_{i=0}^{N}\lambda_i(t), & 0<i\le S, j=S+1, \\ \mu, & 0<i\le S, j=i-1, \\ 0, & \text{otherwise}\ , \end{cases} \tag{15}$$

where $\sum_{i=0}^{N}\lambda_i(t)$ can be computed based on Equations (5)-(7). Note that $\sum_{i=0}^{N}\lambda_i(t)$ represents the error rate of one stripe, and it changes when SSD replacement happens. Without loss of generality, we assume that the SSD replacement only occurs in the epoch boundaries, which can be achieved by adjusting the step size s. Thus, the error rate $\sum_{i=0}^{N}\lambda_i(t)$ is a monotonically increasing function of system age k within the epoch $(lsT, (l+1)sT)$.

Now we can construct the generator matrix $\tilde{\boldsymbol{Q}}_l$. We identify the corresponding upper and lower bounds of the system reliability, as stated in Theorem 2.

Theorem *2:* Let $R(t)$ be the reliability under the original non-homogeneous CTMC $\{X(t)\}$, and also let $R_1(t)$ and $R_2(t)$ be the reliabilities under the homogeneous CTMC $\{\tilde{X}(t)\}$ with generator matrices $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls+s-1}$ and $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls}$ for the epoch $(lsT, (l+1)sT)$, respectively. We have

$$R_1(t) \le R(t) \le R_2(t).$$

Proof: Please refer to §2 of the supplementary file.

Remarks: The gap between the upper and lower bounds heavily depends on the step size s. In particular, if $s = 1$, then there is no approximation error and the gap is just zero, while the computational cost for deriving the reliability becomes very high. Therefore, there is a trade-off between accuracy and computational overhead.

Based on Theorem 2, we have the following corollary.

Corollary *1:* For any generator $\tilde{\boldsymbol{Q}}_l$ which is a linear combination of $\boldsymbol{Q}_k$'s $(ls \le k \le ls+s-1)$, i.e., $\tilde{\boldsymbol{Q}}_l = \sum_{k=ls}^{ls+s-1} c_k \boldsymbol{Q}_k$ where $\sum_{k=ls}^{ls+s-1} c_k = 1$ and $0 \le c_k \le 1$, the approximation error of reliability is no bigger than $R_2(t) - R_1(t)$.

Proof: Please refer to §3 of the supplementary file.

Remarks: We can have better choices for constructing $\tilde{\boldsymbol{Q}}_l$ according to Corollary 1. In particular, if the error rate $\lambda_i(t)$ in Equation (7) is a linear function, i.e., $\alpha = 2$, then we can take $\tilde{\boldsymbol{Q}}_l$ as an average over the s generator matrices of the original CTMC. Mathematically,

$$\tilde{\boldsymbol{Q}}_l = \frac{1}{s}\sum_{k=ls}^{ls+s-1} \boldsymbol{Q}_k, \quad l = 0, 1, \cdots,\ \text{if}\ \alpha = 2. \tag{16}$$

Note that Equation (16) can be transformed to $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls+\frac{s-1}{2}}$ as the error rate is linear when $\alpha = 2$, so the computational cost of constructing $\tilde{\boldsymbol{Q}}_l$ in Equation (16) is $O(S)$. This is because each generator matrix only contains $O(S)$ non-zero elements according to Equation (15).

On the other hand, if the error rate is non-linear, then the computational cost of constructing $\tilde{\boldsymbol{Q}}_l$ using Equation (16) becomes $O(Ss)$ as each non-zero item costs $O(s)$ and there are $O(S)$ non-zero items in each generator matrix. To reduce the computation time, for the case of general α, one may choose $\tilde{\boldsymbol{Q}}_l$ as in Equation (17) so that the computational cost of constructing $\tilde{\boldsymbol{Q}}_l$ is still $O(S)$.

$$\tilde{\boldsymbol{Q}}_l = \frac{\boldsymbol{Q}_{ls} + \boldsymbol{Q}_{ls+s-1}}{2},\ l = 0, 1, \cdots (\text{ for } \forall\ \alpha > 1). \tag{17}$$

Note that when we compute the lower bound $R_1(t)$ and upper bound $R_2(t)$ with generator matrices $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls+s-1}$ and $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls}$, respectively, the computational cost of constructing $\tilde{\boldsymbol{Q}}_l$ is obviously $O(S)$ as each generator matrix only contains $O(S)$ non-zero elements.

3.3 Algorithm for Reliability Computation

We now present a formal algorithm that computes the reliability of an SSD RAID array. In particular, we focus on the reliability dynamics from time zero to the time when the K^{th} erase operation has been performed on the RAID array. Without loss of generality, we assume that K is a multiple of the step size s. Moreover, we denote the maximum acceptable truncation error by ϵ.

Algorithm 1 Algorithm for Reliability Computation

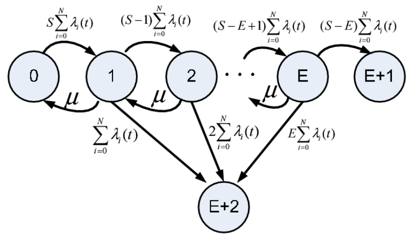
```
1: Initialize: $\hat{\boldsymbol{\pi}}(0) = \boldsymbol{\pi}(0), R(0) = 1$
2: for $l = 0 \to \frac{K}{s} - 1$ do
3:     Construct generator matrix $\tilde{\boldsymbol{Q}}_l$;
4:     Choose $\tilde{\Lambda}_l \geq \max_{ls \leq k < (l+1)s} \max_{0 \leq i \leq S+1} |q_{i,i}(k)|$;
5:     Let $\tilde{\boldsymbol{P}}_l = \boldsymbol{I} + \frac{\tilde{\boldsymbol{Q}}_l}{\tilde{\Lambda}_l}$;
6:     Let $\epsilon_l \leftarrow 0$; $n \leftarrow 0$; $\hat{\boldsymbol{\pi}}((l+1)sT) \leftarrow \boldsymbol{0}$; $\hat{\boldsymbol{v}}_l(0) \leftarrow \hat{\boldsymbol{\pi}}(lsT)$;
7:     while $1 - \epsilon_l > \frac{s\epsilon}{K}$ do
8:         $\epsilon_l \leftarrow \epsilon_l + e^{-\tilde{\Lambda}_l sT} \frac{(\tilde{\Lambda}_l sT)^n}{n!}$;
9:         $\hat{\boldsymbol{\pi}}((l+1)sT) \leftarrow \hat{\boldsymbol{\pi}}((l+1)sT) + e^{-\tilde{\Lambda}_l sT} \frac{(\tilde{\Lambda}_l sT)^n}{n!} \hat{\boldsymbol{v}}_l(n)$;
10:        $n \leftarrow n + 1$;
11:        $\hat{\boldsymbol{v}}_l(n) \leftarrow \hat{\boldsymbol{v}}_l(n-1)\tilde{\boldsymbol{P}}_l$;
12:    end while
13:    $R((l+1)sT) = \sum_{i=0}^{S} \hat{\pi}_i((l+1)sT)$
14: end for
```

Algorithm 1 presents the pseudo-code of the algorithm. Lines 3 to 12 derive the system state in one epoch with s time periods based on the flow described in §3.1. Lines 4 to 6 initialize the necessary parameters. Lines 7 to 12 implement Equation (12), while the truncation point is determined based on Equation (14) and the given maximum error tolerance. Note that the condition in Line 7 indicates that the maximum allowable error in one epoch is $\frac{s\epsilon}{K}$ as there are $\frac{K}{s}$ epochs and the aggregate maximum allowable error is ϵ. After computing the system state at time lsT, Line 13 computes the RAID reliability based on the definition in Equation (8).

3.4 Speeding up the Computation of Algorithm 1

Note that the dimension of the matrix $\tilde{\boldsymbol{P}}_l$ is $(S+2) \times (S+2)$ (S is the number of stripes), which can be very large for large-scale SSDs. To speed up the computation of Algorithm 1, in addition to aggregating multiple time periods as described in §3.1, we propose another optimization technique by truncating the states with large state numbers from the CTMC, so as to reduce the dimension of $\tilde{\boldsymbol{P}}_l$ and hence the computational cost. Intuitively, if an array contains many stripes with exactly one erroneous chunk, it is more likely that a new erroneous chunk appears in one of such stripes (and hence data loss occurs) rather than in a stripe without any erroneous chunk. We can thus remove such states with large state numbers without losing accuracy.

Specifically, as the state number i increases, the transition rate $q_{i,i+1}(k)$ decreases, while the transition rate $q_{i,S+1}(k)$ increases. That is, if the current state number is higher, it is more likely to transit to the state of data loss (i.e., state $S+1$). In other words, it is unlikely for a system to contain too many stripes with exactly one erroneous chunk, because either some erroneous chunks will be recovered soon, or another new error may appear in the same stripe so that data loss happens. Therefore, to reduce the computational cost, we can truncate the states with large state numbers so as to reduce the state space of the Markov chain. We truncate the states with state number greater than E, and let $E+1$ represent the case where more than E stripes contain exactly one erroneous chunk. We take state $E+1$ as an absorbing state, and denote the state of data loss by $E+2$. Figure 3 depicts the truncated state transition diagram. According to Equation (8), we define the reliability as



$S\sum_{i=0}^{N}\lambda_i(t)$
$(S-1)\sum_{i=0}^{N}\lambda_i(t)$
$(S-E+1)\sum_{i=0}^{N}\lambda_i(t)$
$(S-E)\sum_{i=0}^{N}\lambda_i(t)$
0
1
2
E
E+1
μ
μ
μ
$\sum_{i=0}^{N}\lambda_i(t)$
$2\sum_{i=0}^{N}\lambda_i(t)$
$E\sum_{i=0}^{N}\lambda_i(t)$
E+2

Fig. 3: State transition after truncation.

$$R(t) = \sum_{j=0}^{E} \pi_j(t). \tag{18}$$

The intuition of taking state $E+1$ as an absorbing state is that as long as the array contains more than E stripes with an erroneous chunk, we assume that data loss happens. That is, the computed reliability is a lower bound. As we increase E, the computed reliability is more accurate (i.e., the approximation error drops) at the expense of incurring a higher computational cost. Clearly, if E is set as S, then the transition diagram in Figure 3 is exactly the same as that in Figure 2, and Equation (18) gives the exact solution.

To compute the system state after state truncation, let $\{\bar{X}(t), t \geq 0\}$ be the new CTMC with generator matrix $\bar{\boldsymbol{Q}}_k$ during period $(kT, (k+1)T)$ and $\bar{\boldsymbol{\pi}}((k+1)T)$ be the system state at time $(k+1)T$. Similar to Equation (10), given the initial state $\bar{\boldsymbol{\pi}}(kT)$, the system state at time $(k+1)T$ for $\{\bar{X}(t), t \geq 0\}$ can be derived as follows.

$$\bar{\boldsymbol{\pi}}((k+1)T) = \bar{\boldsymbol{\pi}}(kT) \sum_{n=0}^{\infty} \frac{(\bar{\boldsymbol{Q}}_k T)^n}{n!}. \tag{19}$$

If we denote the error caused by truncating the states at time kT by $\bar{\epsilon}_k$, then $\bar{\epsilon}_k$ can be formally defined as follows.

$$\bar{\epsilon}_k = \sum_{i=0}^{E} |\bar{\pi}_i(kT) - \pi_i(kT)|,$$

where $\bar{\pi}_i(kT)$ and $\pi_i(kT)$ represent the probabilities that the system is at state i at time kT for the truncated CTMC and the original CTMC, respectively. Clearly, $\bar{\epsilon}_0 = 0$ as both CTMCs start from the same initial state. The bound of the state truncation error is

$$\bar{\epsilon}_k \leq \bar{\pi}_{E+1}(kT). \tag{20}$$

Again, we can follow the steps in §3.1 to compute the reliability under the truncated CTMC $\{\bar{X}(t), t \geq 0\}$.

3.5 Model Implementation

Our implementation of Algorithm 1 uses the following inputs. We fix $s = BM/20$, meaning that for each SSD, we consider at least 20 time points before it reaches its lifetime of BM erasures. We choose this step size because it also achieves high accuracy and only takes small computation time, which will be justified in §4.2. The error bound is fixed at $\epsilon = 10^{-3}$. We also set $\boldsymbol{\pi}(0) = (1, 0 \cdots 0)$ to indicate that the array has no error initially. Finally, to configure parameter E, we note that if E is smaller, we have a lower computation cost, while the approximation error due to state truncation is larger. We set $E = 500$ by default, as we find that the error is very close to zero when E reaches 500.

4 Model Validation

In this section, we validate via trace-driven simulation the accuracy of our CTMC model on quantifying the RAID reliability $R(t)$ (see §4.1). We also study the trade-off between model accuracy and computation time under different step sizes (see §4.2).

4.1 Comparisons between Model and Simulation

We simulate the I/O behavior of an SSD using Microsoft's SSD simulator [1], which is extended from DiskSim [3]. Since each SSD contains multiple chips that can be configured to be independent of each other and handle I/O requests in parallel, without loss of accuracy, we consider RAID at the chip level (as opposed to device level). Specifically, we configure each chip to have its own data bus and control bus and treat it as one drive, and also treat the SSD controller as the RAID controller where parity-based RAID is built.

Our validation measures the reliability of RAID-5 and Diff-RAID. We consider $N + 1$ chips where N may be configured to have different values. For traditional RAID-5, parity chunks are evenly placed across the $N+1$ chips; for Diff-RAID, we place 10% of parity chunks in each of the N chips and the remaining parity chunks in the last flash chip. Since we set $N \leq 7$ in our evaluation, the choice of our parity distribution for Diff-RAID is feasible.

We generate synthetic uniform workload in which write requests access the addresses of the entire address space with equal probability. To simulate the process of error arrivals and recovery, we generate both error events and recovery events and feed them into the SSD simulator as special types of I/O requests. Specifically, we generate an initial error event and an initial recovery event before our simulation starts, and add them into the event queue of the simulator. Note that the events in the event queue, including those related to I/O requests and the error/recovery events, are sorted according to their arrival times, and each time the simulator will process the event at the head of the queue. When an error event or a recovery event is processed, we identify the error position and change the corresponding metadata. We then generate another error event or recovery event and add it into the event queue. Error events are generated based on Poisson arrivals with rate determined by the current system age k of the array, while recovery events are generated based on an exponential recovery time distribution with a fixed rate $\mu = 1$. As the array ages, i.e., each time when an erase operation happens, we update the error arrival rate accordingly by varying the variable $k_i(t)$ based on Equation (7), and a new error event is generated with the new error arrival rate.

To obtain simulation results for RAID reliability, the workload that we generate lasts until data loss happens or all drives in the array are worn out and replaced at least once. Then we record the array age (i.e., the number of erasures performed on the array) when the simulation stops. Thus, we obtain only one number in each simulation run, and this number represents the least number of erasures that the array has undergone when data loss happens. To obtain the reliability metric defined in Equation (8), we run the simulation 1000 times for each parameter setting, and then calculate the probability of no data loss at any array age based on the 1000 numbers we obtain. Note that the reliability defined in this paper is a continuous function of age, so the more number of simulations we run for each parameter setting, the smoother reliability curve is. For time consideration, we only run 1000 times for each parameter setting, which we find suffices to obtain a smooth reliability curve.

To speed up our simulation, we consider a small-scale array, in which each chip contains 80 blocks with 64 pages each, and the spare factor is set as 0.2. The chunk size is set to be the same as the page size. We set a low erasure limit of 50 for each block to avoid waiting for too long to wear out an SSD, i.e., we let $M = 50$.

To validate the accuracy of our model, we consider different configurations by varying the parameters c, N, α and M. In the interest of space, we here only present the results in the case of varying the parameter c. Other results are shown in §4 of the supplementary file. The value of c determines the overall frequency of the error arrivals (see Equation 7), we fix $N = 3$ and $\alpha = 2$. We consider three cases where $c = 1.0 \times 10^{-6}$, $c = 0.5 \times 10^{-6}$ and $c = 0.2 \times 10^{-6}$, which represent three cases in which the error rate is larger than, comparable to, and smaller than the recovery rate, respectively. We call them the *error dominant case*, *comparable case*, and *recovery dominant case*, respectively.

Figures 4 shows the reliability results obtained from the model and simulation for different values of c. In each figure, the horizontal axis represents the array age, which denotes the number of erasures performed on the array, and the vertical axis shows the reliability, which denotes the probability of no data loss until the array age. We show the reliability dynamics of a RAID array until all drives wear out. In particular, we plot the upper bound and the lower bound obtained from our model, which are derived by setting $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls}$ and $\tilde{\boldsymbol{Q}}_l = \boldsymbol{Q}_{ls+s-1}$, respectively. We also plot the results with the reliability obtained from the simulator. From the figures we observe that the gap between the upper bound and the lower bound is generally very small, and that the model and simulation results are very close in all cases. We will further study the implication of parity distributions and the impact of different RAID parameters by using our model in §5.

4.2 Trade-off between Model Accuracy and Computation Time

Finally, we validate how our model can trade between model accuracy and computation time by tuning the step size s. Since the computation time is too large if the step size is set as too small, we vary the step size s from B to BM (note that Diff-RAID will have one drive worn out and replaced after every BM erasures). In this evaluation, we consider a specific set of parameters, whose practical justifications are provided in §5.1. We fix $S = B = 131,072$, $N = 9$, $\mu = 10^{-3}$, $T = 10^{-2}$, and $M = 10^4$. We also consider different error arrival patterns with $\alpha = 2, 3$, and 1.5, which correspond to the linear, convex, and concave error rates, respectively. Without loss of generality, for each α, we only show one particular error rate where the corresponding parameter c is set as 0.4×10^{-13}, 0.267×10^{-17} and 0.533×10^{-11} for $\alpha = 2, 3$, and 1.5, respectively.

To characterize the accuracy of the model, we show the gap between the upper bound and the lower bound, i.e., $R_2(t) - R_1(t)$. Note that since the error is larger when the system age is higher, i.e., the gap becomes larger as t increases, we focus on the gap at the time when all drives have just been replaced once, i.e., at time $(N+1)BMT$. On the other hand, the computation time is estimated by implementing Algorithm 1 with Java. The algorithm runs on a PC with a dual core CPU running at 3.0 GHz and 4 GB memory. We show the results for both traditional RAID-5 and Diff-RAID with parity distribution computed by Equation (21) with $\sigma = 1$.

Figure 5 shows the results under different types of error rates. The horizontal axis indicates the step size, the left vertical axis shows the gap between the upper bound and the lower bound, and the right vertical axis shows the computation time. We see that the computation time decreases as the step size increases, while the gap between the upper bound and the lower bound increases. The gap may even reach up to 0.5 if the step size is set as large as $BM \approx 1.3 \times 10^9$. Thus, there is a trade-off between model accuracy and computation time when we choose the step size. We point out that in the case of RAID-5 with concave error rate (see Figure 5(c)), the gap is very close to zero even for a large step size $s = BM$. The reason is that the marginal increase of the error rate decreases as SSD ages due to the concavity of the error rate function, and the RAID reliability remains very high when system age is small as RAID-5 is deployed with brand-new SSDs. Thus, the variance of RAID reliability in one computation interval is small and the gap between the upper bound and the lower bound is very close to zero. On the other hand, this is not the case for Diff-RAID, as SSDs for constructing Diff-RAID are initially consumed by a certain fraction of erasures based on the distribution of A_i's (see Equation (3)).

To choose an appropriate step size, note that the gap between the upper bound and the lower bound is very small for all cases when the step size s is smaller than $\frac{BM}{20} \approx 6.5 \times 10^7$, and the computation time is only less than one minute if s is greater than $\frac{BM}{20}$. Thus, setting $s = \frac{BM}{20}$ achieves a good trade-off, and we fix $s = \frac{BM}{20}$ in Algorithm 1. Last but not the least, according to the trend shown in the figures, we infer that it takes a very long time to derive the reliability if we set the step size as one. This further shows the importance of our optimization techniques in speeding up the computation.

5 Numerical Analysis

In this section, we use our proposed model to conduct numerical analysis on the reliability dynamics of a large-scale SSD RAID system with respect to different parity placement strategies. We consider an SSD RAID array with a large number of blocks in each SSD and consider realistic erasure limit by setting a large value for M, and hence it is computationally infeasible to derive the reliability from trace-driven simulation. To this end, we summarize the lessons learned from our analysis.

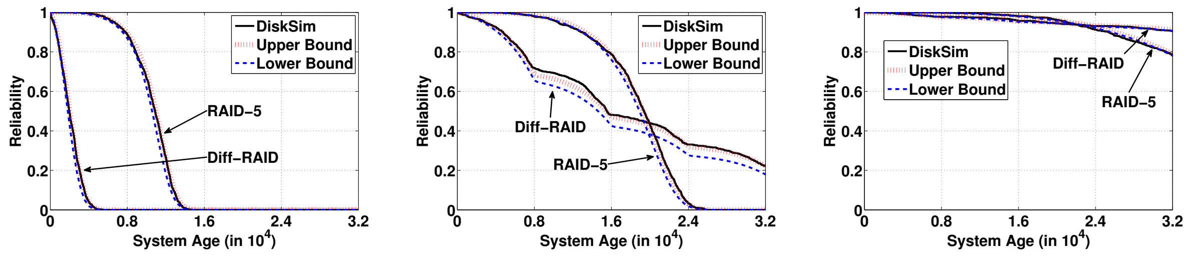
5.1 Choices of Default Model Parameters

We first describe the default model parameters used in our analysis, and provide justifications for our choices.

We consider an SSD RAID array composed of $N + 1$ SSDs, each being modeled by the same set of parameters. By default, we set $N = 9$. Each block of an SSD has 64 pages of size 4KB each. We consider 32GB SSDs with $B = 131,072$ blocks. We configure the chunk size equal to the block size, i.e., there are $S = B = 131,072$ chunks[1]. We also have each block sustain M =10K erase cycles.

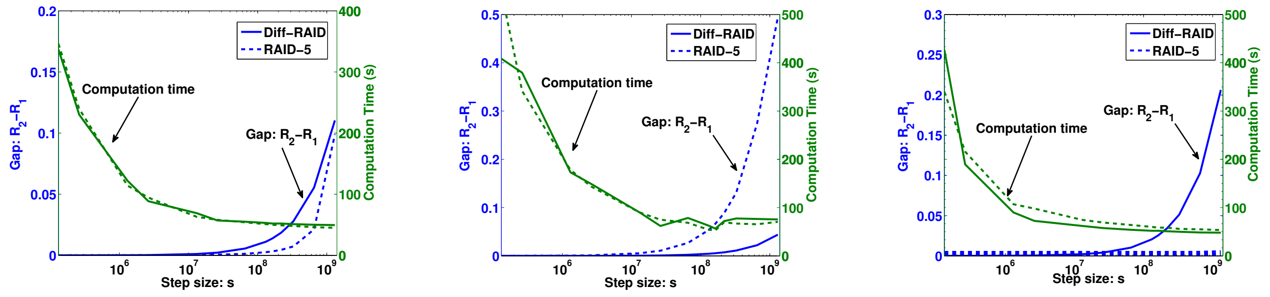
We now describe how we configure the error arrival rate $\lambda_i = c\alpha k_i^{\alpha-1}$ and the error recovery rate μ. We employ 4-bit ECC protection per 512 bytes of data, the industry standard for today's MLC flash. Based on the uncorrectable bit error rates (UBERs) calculated in [2], we choose the UBER when an SSD reaches its rated lifetime (i.e., the erasure limit M is reached) from the range $[10^{-16}, 10^{-18}]$. Now the probability that a chunk contains at least one bit error is roughly in the range of $[2 \times 10^{-10}, 2 \times 10^{-12}]$. Based on the analysis on real enterprise workload traces [30], a RAID array can have several hundred gigabytes of data being accessed per day. If the write request rate is set as 1TB per day (i.e., 50

1. In practice, SSDs are over-provisioned [1], so the actual number of blocks (or chunks) that can be used for storage (i.e., S) should be smaller. However, the key observations of our results here still hold.



(a) $c = 1.0 \times 10^{-6}$ (b) $c = 0.5 \times 10^{-6}$ (c) $c = 0.2 \times 10^{-6}$

Fig. 4: Model validation with respect to different values of c ($N = 3$ and $\alpha = 2$).



(a) Linear error rate ($\alpha = 2$) (b) Convex error rate ($\alpha = 3$) (c) Concave error rate ($\alpha = 1.5$)

Fig. 5: Trade-off between accuracy and computation time.

chunks per second), then the error arrival rate per chunk at its rated lifetime (i.e., $\lambda_i = c\alpha M^{\alpha-1}$) is approximately in the range $[10^{-8}, 10^{-10}]$.

To set the parameters c and μ, we consider three cases: (1) error arrivals are dominant, (2) error arrivals are comparable to error recoveries, and (3) error recoveries are dominant, which correspond to the *error dominant*, *comparable*, and *recovery dominant* cases, respectively. Note that the aggregate error arrival rate when all $N + 1$ drives are going to die out is $c\alpha M^{\alpha-1}S(N+1)$, which is roughly in the range $[10^{-2}, 10^{-4}]$ if $N = 9$, so we first fix $\mu = 10^{-3}$, and then consider three cases of error patterns. In particular, in the case where $\alpha = 2$, we set $c = 1.1 \times 10^{-13}$, $c = 0.4 \times 10^{-13}$, and $c = 0.1 \times 10^{-13}$ so to represent the above three cases, respectively. Note that when $\alpha = 2$ and $c = 0.4 \times 10^{-13}$, the aggregate error arrival rate of the array when all SSDs reach their rated lifetime is around $2cMS(N+1) \approx 10^{-3} = \mu$ (where $N = 9$, M =10K, and $S = 131,072$).

As in §4, we also consider the cases when $\alpha = 3$ and $\alpha = 1.5$, and set the corresponding coefficient c according to the same maximum error rate $c\alpha M^{\alpha-1}$ as in the case where $\alpha = 2$. Specifically, we set c as 0.73×10^{-17}, 0.267×10^{-17}, and 0.667×10^{-18} when $\alpha = 3$, and set it as 0.147×10^{-10}, 0.533×10^{-11}, and 0.133×10^{-11} when $\alpha = 1.5$.

We now configure T, the time interval between two neighboring erase operations. Suppose that there are 1TB of writes per day as described above. The inter-arrival time of write requests is around 3×10^{-4} seconds for 4KB page size. Thus, the average time between two erase operations is 1.9×10^{-2} seconds as an erase is triggered after writing 64 pages. In practice, each erase causes additional writes (i.e., write amplification [15]) as it moves data across blocks, so T should be smaller. Here, we fix $T = 10^{-2}$ seconds.

We compare the reliability dynamics of RAID-5 and Diff-RAID. For RAID-5, each drive holds a fraction $\frac{1}{N+1}$ of parity chunks; for Diff-RAID, we choose the parity distribution (i.e., p_i's for $0 \le i \le N$) based on a truncated normal distribution. Specifically, we consider a normal distribution $\mathcal{N}(N+1, \sigma^2)$ with mean $N+1$, and standard deviation σ, and let f be the corresponding probability density function. We then choose p_i's as follows:

$$p_i = \frac{\int_i^{i+1} f(x)dx}{\int_0^{N+1} f(x)dx}, \quad 0 \le i \le N. \tag{21}$$

We can choose different distributions of p_i by tuning the parameter σ. Intuitively, the larger σ is, the more evenly p_i's are distributed. We consider three cases: $\sigma = 1$, $\sigma = 2$, and $\sigma = 5$. Suppose that $N = 9$. Then for $\sigma = 1$, SSD N and SSD $N - 1$ hold 68% and 27% of parity chunks, respectively; for $\sigma = 2$, SSD N, SSD $N-1$, and SSD $N-2$ hold 38%, 30%, and 18% of parity chunks, respectively; for $\sigma = 5$, the proportions of parity chunks range from 2.8% (in SSD 0) to 16.6% (in SSD N). Giving p_i's, the age of each block of SSD i (i.e., k_i) can be computed.

5.2 Impact of Different Error Dynamics

We now show the numerical results of RAID reliability. We evaluate the reliability using Equation (16) for the linear error rate and using Equation (17) for the non-linear error rate. We assume that the drive replacement can be completed immediately after the oldest SSD reaches its rated lifetime. When the oldest drive is replaced, all its chunks (including any erroneous chunks) are copied to the new drive. Thus, the reliability remains the same. We use the parameters described in §5.1.

Since the reliability dynamics in the case of non-linear error rate is similar as that in the case of linear error rate, in the interest of space, here we focus on the linear error rate and show the results in the case of non-linear error rate in § 5 of the supplementary file. Figure 6 shows the reliability for three cases: error dominant, comparable, and recovery dominant. We elaborate the results below.

Case 1: Error dominant case. Figure 6(a) first shows the numerical results for the error dominant case. Initially, RAID-5 achieves very good reliability. As SSDs wear down, the bit error rate increases, and this makes the RAID reliability decrease very quickly. In particular, the reliability drops to zero (i.e., data loss always happen) when the array performs around 5×10^9 erasures. For Diff-RAID, the more evenly parity chunks are distributed, the lower RAID reliability is. In the error dominant case, since error arrival rate is much bigger than the recovery rate, the RAID reliability drops to zero very quickly no matter what parity placement strategy is used. We note that Diff-RAID is less reliable than traditional RAID-5 in the error dominant case. The reason is that for Diff-RAID, the initial ages of SSDs when constructing RAID are non-zero, but instead follow the convergent age distribution (i.e., based on A_i's in Equation (3)). When error rate is very large, the array suffers from low reliability even if the array only performs a small number of erasures. However, for RAID-5, since it is always constructed with brand-new SSDs, it starts with a very high reliability.

Case 2: Comparable case. Figure 6(b) shows the results for the comparable case. RAID-5 achieves very good reliability initially, but decreases dramatically as the SSDs wear down. Also, all drives wear down at the same rate, the reliability of the array is about zero when all drives reach their erasure limits, i.e., when the system age is around 1.3×10^{10} erasures. Diff-RAID shows different reliability dynamics. Initially, Diff-RAID has less reliability than RAID-5, but the drop rate of the reliability is much slower than that of RAID-5 as SSDs wear down. The reason is that Diff-RAID has uneven parity placement, SSDs are worn out at different times and will be replaced one by one. When the worn-out SSD is replaced, other SSDs perform fewer erase operations and have small error rates. This prevents the whole array suffering from a very large error rate as in RAID-5. Also, the reliability is higher when the parity distribution is more skewed (i.e., smaller σ), as also observed in [2].

Case 3: Recovery dominant case. Figure 6(c) shows the results for the recovery dominant case. Note that the vertical axis in Figure 6(c) starts from 0.9 but not 0. RAID-5 shows high reliability in general. Between two replacements (which happens every 1.3×10^{10} erasures), its data loss probability drops by within 3%. Its reliability drops slowly right after each replacement, and its drop rate increases as it is close to be worn out. Diff-RAID shows higher reliability than RAID-5 in general, but the difference is small (e.g., less than 6% between Diff-RAID for $\sigma = 1$ and RAID-5). Therefore, in the recovery dominant scenario, we may deploy RAID-5 instead of Diff-RAID, as the latter introduces higher costs in parity redistribution in each replacement and has smaller I/O throughput due to load imbalance of parities.

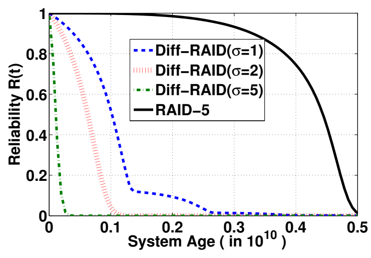
5.3 Impact of Different Array Configurations

We further study via our model how different array configurations affect the RAID reliability. We focus on Diff-RAID and generate the parity distribution p_i's with $\sigma = 1$. Without loss of generality, in the following, we only focus on the case when $\alpha = 2$. Our goal is to validate the robustness of our model on characterizing the reliability for different array configurations.

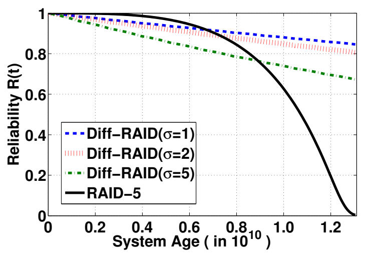
Impact of N. Figure 7(a) shows the impact of the RAID size N. We fix other parameters as the same in the comparable case shown in Figure 6(b), i.e., $\mu = 10^{-3}$, $c = 0.4 \times 10^{-13}$, and $M = 10^4$. The larger the system size, the lower the RAID reliability. Intuitively, the probability of having one more erroneous chunk in a stripe increases with the stripe width (i.e., $N+1$). Note that the reliability drop is significant when N increases. For example, at 2.6×10^{10} erasures, the reliability drops from 0.7 to 0.2 when N increases from 9 to 19.

Impact of ECC. Figure 7(b) shows the impact of different ECC lengths. We fix $\mu = 10^{-3}$, $M = 10^4$, and $N = 9$. We also fix the raw bit error rate (RBER) as 1.3×10^{-6} [2], and compute the uncorrectable bit error rate using the formulas in [27]. Then as described in §5.1, we derive c for different ECCs that can correct 3, 4, 5 bits per 512 byte sector, and the corresponding values are 4.4×10^{-11}, 4.7×10^{-14}, and 4.2×10^{-17}, respectively. We observe that the RAID reliability drops to zero very quickly for 3-bit ECC at around 10^5 erasures, while the reliability for 5-bit ECC starts to decrease until the array performs 10^{11} erasures. This shows that the RAID reliability heavily depends on the reliability of each single SSD, or the ECC length employed in each SSD.

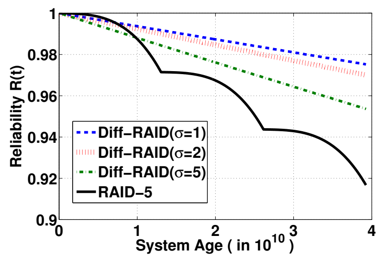
Impact of M. Figure 7(c) shows the impact of the erasure limit M, or the endurance of a single SSD, on the RAID reliability. We fix other parameters with $\mu = 10^{-3}$, $N = 9$ and $c = 0.4 \times 10^{-13}$. We observe that when M decreases, the RAID reliability increases. For example, at 1.3×10^{10} erasures, reliability increases from 0.85 to 0.99 when M decreases from 10K to 1K. This is mainly because the parameter c is fixed, so the maximum error rate of SSDs (i.e., $c\alpha M^{\alpha-1}$) is bigger when M is larger. Recall that error rates increase with the number of erasures in SSDs. We now have the increase of bit error rates capped by the small erasure limit. The trade-off is that the SSDs are worn out and replaced more frequently with smaller M.



(a) Error dominant case ($c = 1.1 \times 10^{-13}$)

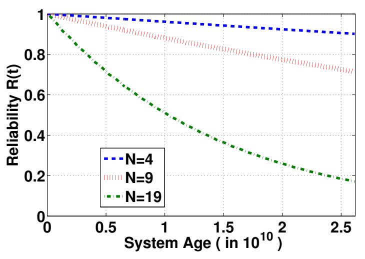


(b) Comparable case ($c = 0.4 \times 10^{-13}$)

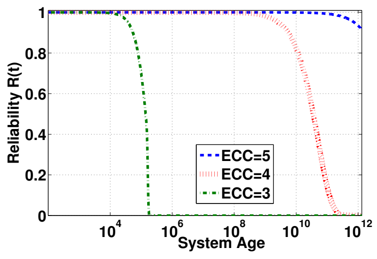


(c) Recovery dominant case ($c=10^{-14}$)

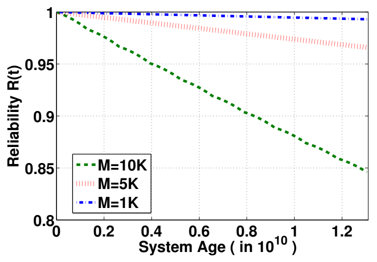
Fig. 6: Reliability dynamics of SSD arrays (Linear error rate with $\alpha = 2$).



(a) Impact of N



(b) Impact of ECC length



(c) Impact of M

Fig. 7: Impact of different RAID configurations on the reliability.

5.4 Impact of Non-uniform Workload

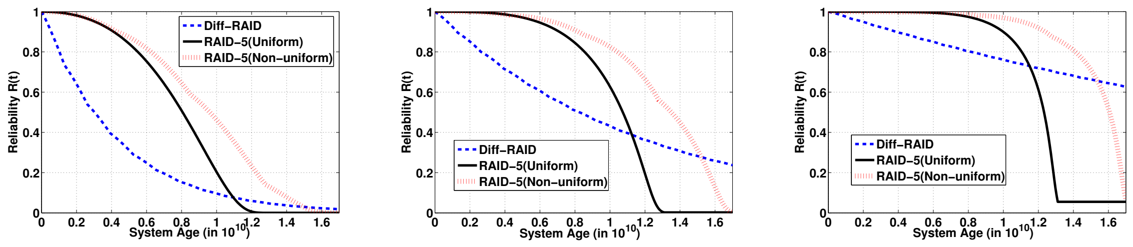
As we stated in §2.3, our model can also be applied to analyze non-uniform workload as long as the aging ratio is given. To study the impact of non-uniform workload, we let $N = 9$ and focus on the comparable case. Since the aging ratio for RAID-5 depends on workload, while it mainly depends on the parity distribution for Diff-RAID, we fix the aging ratio for Diff-RAID and only vary it for RAID-5. In particular, we set $(r_0 : r_1 : \cdots : r_N) = (4 : 1 : \cdots : 1)$ for Diff-RAID. For RAID-5, we consider two cases: (1) $(r_0 : r_1 : \cdots : r_N) = (1 : 1 : \cdots : 1)$, which corresponds to the case of uniform workload where all SSDs receive the same number of requests and age at the same rate, and (2) $(r_0 : r_1 : \cdots : r_N) = (4 : 1 : \cdots : 1)$, which corresponds to the case of non-uniform workload where the first drive receives more requests and ages four times faster than other drives.

Figure 8 shows the impact of workload on RAID reliability with different error rates. Note that the aging ratio $(4 : 1 : \cdots : 1)$ implies that the first SSD ages four times faster than others. That is, the first SSD will be replaced four times when other drives are just replaced once for RAID-5. Based on parameters described before, all drives in the array will be replaced at least once after every $13BM \approx 1.7 \times 10^{10}$ erasures, so we show the RAID reliability until the system age reaches 1.7×10^{10}. We see that the reliability of RAID-5 under non-uniform workload is always higher than that under uniform workload for all types of error rates. However, even if SSDs in a RAID-5 array age at different rates under non-uniform workload, correlated device failures may still happen as it is still possible that all drives wear out simultaneously, which makes the array suffer from a low reliability. For example, in the case where the aging ratio is $(4 : 1 \cdots 1)$, when the first SSD has been replaced four times, all other SSDs are also worn out. Simultaneous wearing can only be avoided by calling parity redistribution as in Diff-RAID. In particular, the relationship between Diff-RAID and RAID-5 we derived from the previous analysis on uniform workload still holds for non-uniform workload.

5.5 Discussion

Our model effectively analyzes the RAID reliability with regard to different RAID configurations. The numerical results presented in previous subsections address the fundamental question of whether we should distribute evenly or unevenly parities across multiple SSDs to achieve high SSD RAID reliability (see §1). In particular, our results provide several insights as follows.

- The error dominant case may correspond to the low-end SSDs with high bit error rates, especially when these types of SSDs have low I/O bandwidth for RAID reconstruction. Both traditional RAID-5 and Diff-RAID show low reliability. A higher degree of fault tolerance becomes necessary in this case, and both methods of using stronger ECC that can correct more than 4-bit errors (note that we assume 4-bit ECC in the numerical study) and deploying a RAID that can tolerate against two or three failures may be adequate. However, the comparison of the two methods in improving SSD RAID reliability is out of the scope of this paper.
- When the error arrival and recovery rates are similar, Diff-RAID, with uneven parity distribution, achieves higher reliability than RAID-5, especially



(a) Concave error rate ($\alpha = 1.5$) (b) Linear error rate ($\alpha = 2$) (c) Convex error rate ($\alpha = 3$)

Fig. 8: Impact of workload on RAID reliability. For Diff-RAID, the aging ratio is fixed as $(4 : 1 : \cdots : 1)$ which can be achieved under both uniform workload and non-uniform workload by adjusting the parity distribution. For RAID-5, the aging ratios are $(1 : 1 : \cdots : 1)$ and $(4 : 1 : \cdots : 1)$, which correspond to the cases of uniform workload and non-uniform workload, respectively.

when RAID-5 reaches zero reliability when all SSDs are worn out simultaneously. This conforms to the findings in [2].

- In the recovery dominant case, which may correspond to the high-end SSDs that typically have very small bit error rates, even if Diff-DAID provides higher reliability than RAID-5, the improvement is very small and RAID-5 already achieves very high reliability. Thus, we may choose RAID-5 over Diff-RAID as it can achieve better load balancing because of the even distribution of parities. On the other hand, Diff-RAID requires the operation of parity redistribution during each SSD replacement. This may severely degrade the I/O performance of SSD RAID. Parity redistribution also significantly complicates the implementation of SSD replacement.

6 Related Work

There have been extensive studies on NAND flash-based SSDs. A detailed survey of the algorithms and data structures for flash memories is found in [11]. Recent papers empirically study the intrinsic characteristics of SSDs (e.g., [1, 6]), or develop analytical models for the write performance (e.g., [9, 15]) and garbage collection algorithms (e.g., [24]) of SSDs.

Bit error rates of SSDs are known to increase with the number of erasures [5, 12, 27, 35, 38]. To improve reliability, prior studies propose to adopt RAID for SSDs at the device level [2, 16, 21, 22, 26, 31], or at the chip level [19, 20]. These studies focus on developing new RAID schemes to improve the performance and endurance of SSDs. The performance and reliability implications of RAID on SSDs are experimentally studied in [18], and authors in [28] analyzed the impact of parity protection on the lifetime of SSD arrays. In contrast, our work focuses on quantifying reliability dynamics of SSD RAID under different parity distributions from a theoretical perspective. Authors of Diff-RAID [2] also attempt to quantify the reliability, but they only compute the reliability at the instants of SSD replacements, while our model captures the time-varying nature of error rates in SSDs and quantifies the instantaneous reliability during the whole lifespan of an SSD RAID array.

RAID was first introduced in [32] and has been widely used. Performance and reliability analysis on RAID in the context of hard disk drives has been extensively studied (e.g., see [4, 7, 25, 29, 37]). On the other hand, SSDs have a distinct property that their error rates increase as they wear down, so a new model is necessary to characterize the reliability of SSD RAID.

7 Conclusions

We developed the *first* analytical model that quantifies the reliability dynamics of SSD RAID. We built our model as a non-homogeneous continuous time Markov chain, and analyzed its transient state probability using the uniformization technique. We validated the correctness of our model via trace-driven simulation. With our model, one can characterize the reliability dynamics of SSD RAID arrays with different parity placement distributions. To demonstrate, we compared the reliability dynamics of traditional RAID-5 and the new Diff-RAID scheme under different error patterns and array configurations. Our model provides a useful tool to characterize the reliability of an SSD RAID array with regard to different scenarios.

Acknowledgments

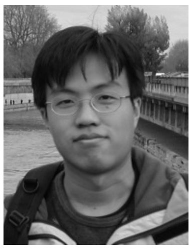
The work of Yongkun Li was supported in part by National Nature Science Foundation of China under Grant No. 61303048, and the Fundamental Research Funds for the Central Universities under Grant No. WK0110000040. The work of Patrick P. C. Lee was supported in part by seed grants from the CUHK MoE-Microsoft Key Laboratory of Human-centric Computing and Interface Technologies.

References

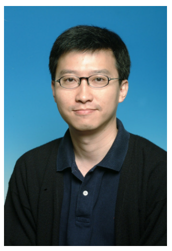
[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design Tradeoffs for SSD Performance. In *USENIX ATC*, Jun 2008.

[2] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. Differential RAID: Rethinking RAID for SSD Reliability. *ACM ToS*, 6(2):4, Jul 2010.
[3] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger. The DiskSim Simulation Environment Version 4.0 Reference Manual. Technical Report CMUPDL-08-101, May 2008.
[4] W. Burkhard and J. Menon. Disk Array Storage System Reliability. In *Proc. of IEEE FTCS*, Jun 1993.
[5] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis. In *DATE*, 2012.
[6] F. Chen, D. A. Koufaty, and X. Zhang. Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives. In *SIGMETRICS*, 2009.
[7] S. Chen and D. Towsley. A Performance Evaluation of RAID Architectures. *IEEE TC*, 45(10):1116–1130, 1996.
[8] E. de Souza e Silva and H. R. Gail. Transient Solutions for Markov Chains. *Computational Probability*, W. K. Grassmann (editor). Kluwer Academic Publishers:43–81, 2000.
[9] P. Desnoyers. Analytic Modeling of SSD Write Performance. In *SYSTOR*, Jun 2012.
[10] R. Enderle. Revolution in January: EMC Brings Flash Drives into the Data Center. http://www.itbusinessedge.com/blogs/rob/?p=184, Jan 2008.
[11] E. Gal and S. Toledo. Algorithms and Data Structures for Flash Memories. *ACM Comput. Surv.*, 37(2):138–163, 2005.
[12] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf. Characterizing Flash Memory: Anomalies, Observations, and Applications. In *MICRO*, Dec 2009.
[13] L. M. Grupp, J. D. Davis, and S. Swanson. The Bleak Future of NAND Flash Memory. In *FAST*, Feb 2012.
[14] K. Hess. 2011: Year of the SSD? http://www.datacenterknowledge.com/archives/2011/02/17/2011-year-of-the-ssd/, Feb 2011.
[15] X.-Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka. Write Amplification Analysis in Flash-based Solid State Drives. In *SYSTOR*, May 2009.
[16] S. Im and D. Shin. Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD. *IEEE TC*, 2011.
[17] A. Jensen. Markoff Chains As An Aid in The Study of Markoff Processes. *Scandinavian Actuarial Journal*, 3:87–91, 1953.
[18] N. Jeremic, G. Mühl, A. Busse, and J. Richling. The Pitfalls of Deploying Solid-state Drive RAIDs. In *SYSTOR*, 2011.
[19] J. Kim, J. Lee, J. Choi, D. Lee, and S. Noh. Improving SSD Reliability with RAID via Elastic Striping and Anywhere Parity. In *DSN*, pages 1–12, 2013.
[20] J. Kim, J. Lee, J. Choi, D. Lee, and S. H. Noh. Enhancing SSD Reliability Through Efficient RAID Support. In *APSys*, Jul 2012.
[21] S. Lee, B. Lee, K. Koh, and H. Bahn. A Lifespan-aware Reliability Scheme for RAID-based Flash Storage. In *Proc. of ACM Symp. on Applied Computing*, SAC '11, 2011.
[22] Y. Lee, S. Jung, and Y. H. Song. FRA: A Flash-aware Redundancy Array of Flash Storage Devices. In *ACM CODES+ISSS*, Oct 2009.
[23] Y. Li, P. P. C. Lee, and J. C. S. Lui. Stochastic Analysis on RAID Reliability for Solid-State Drives. In *IEEE SRDS*, 2013.
[24] Y. Li, P. P. C. Lee, and J. C. S. Lui. Stochastic Modeling of Large-Scale Solid-State Storage Systems: Analysis, Design Tradeoffs and Optimization. In *SIGMETRICS*, 2013.
[25] M. Malhotra and K. S. Trivedi. Reliability Analysis of Redundant Arrays of Inexpensive Disks. *J. Parallel Distrib. Comput.*, 17(1-2):146–151, Jan 1993.
[26] B. Mao, H. Jiang, S. Wu, L. Tian, D. Feng, J. Chen, and L. Zeng. HPDA: A Hybrid Parity-based Disk Array for Enhanced Performance and Reliability. *ACM ToS*, 8(1):4, Feb 2012.
[27] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill. Bit Error Rate in NAND Flash Memories. In *IEEE Int. Reliability Physics Symp.*, Apr 2008.
[28] S. Moon and A. L. N. Reddy. Don't Let RAID Raid the Lifetime of Your SSD Array. In *HotStorage*, pages 1–12, 2013.
[29] R. R. Muntz and J. C. S. Lui. Performance Analysis of Disk Arrays under Failure. In *VLDB*, Aug 1990.
[30] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In *EuroSys*, Mar 2009.
[31] K. Park, D.-H. Lee, Y. Woo, G. Lee, J.-H. Lee, and D.-H. Kim. Reliability and Performance Enhancement Technique for SSD Array Storage System Using RAID Mechanism. In *IEEE Int. Symp. on Comm. and Inform. Tech.*, 2009.
[32] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD*, Jun 1988.
[33] A. Reibman and K. S. Trivedi. Transient Analysis of Cumulative Measures of Markov Model Behavior. *Communications in Statistics-Stochastic Models*, 5:683–710, 1989.
[34] M. Schulze, G. Gibson, R. Katz, and D. Patterson. How Reliable Is A RAID? In *IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers*, 1989.
[35] H. Sun, P. Grayson, and B. Wood. Quantifying Reliability of Solid-State Storage from Multiple Aspects. In *SNAPI*, 2011.
[36] W. Weibull. A Statistical Distribution Function of Wide Applicability. *J. of Applied Mechanics*, 18:293–297, 1951.
[37] X. Wu, J. Li, and H. Kameda. Reliability Analysis of Disk Array Organizations by Considering Uncorrectable Bit Errors. In *SRDS*, Oct 1997.
[38] E. Yaakobi, L. Grupp, P. Siegel, S. Swanson, and J. Wolf. Characterization and Error-correcting Codes for TLC Flash Memories. In *ICNC*, 2012.
[39] D. Yimo, L. Fang, C. Zhiguang, and M. Xin. WeLe-RAID: A SSD-Based RAID for System Endurance and Performance. In *Network and Parallel Computing*, volume 6985 of *Lecture Notes in Computer Science*, pages 248–262. Springer Berlin Heidelberg, 2011.

Yongkun Li is currently an associate researcher in School of Computer Science and Technology, University of Science and Technology of China. He received the B.Eng. degree in Computer Science from University of Science and Technology of China in 2008, and the Ph.D. degree in Computer Science and Engineering from The Chinese University of Hong Kong in 2012. After that, he worked as a postdoctoral fellow in Institute of Network Coding at The Chinese University of Hong Kong. His research mainly focuses on performance evaluation of networking and storage systems.



Patrick P. C. Lee received the B.Eng. degree (first-class honors) in Information Engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now an assistant professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in various applied/systems topics including cloud storage, distributed systems and networks, operating systems, and security/resilience.



John C. S. Lui is currently a professor in the Department of Computer Science & Engineering at The Chinese University of Hong Kong. He received his Ph.D. in Computer Science from UCLA. He serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security . He is an elected member of the IFIP WG 7.3, Fellow of ACM, Fellow of IEEE and Croucher Senior Research Fellow. His current research interests are in communication networks, network/system security, network economics, network sciences, cloud computing, large scale distributed systems and performance evaluation theory.