Modeling SSD RAID Reliability under General Settings

Zhiyong Wu [1], Yongkun Li [1], Patrick P. C. Lee [2], Yinlong Xu [1]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong
wzylucky@mail.ustc.edu.cn, {ykli, ylxu}@ustc.edu.cn, pclee@cse.cuhk.edu.hk

ABSTRACT

Solid-state drives (SSDs) are susceptible to the limited number of program/erase (P/E) cycles and uncorrectable flash errors, and hence achieving high reliability of SSD storage systems is a critical issue. RAID provides a viable option for enhancing system reliability by distributing redundancy across a number of SSDs. However, the flash error rate of an SSD increases with the number of P/E cycles, and this time-varying nature complicates the reliability analysis of SSD RAID. In addition, there remains very limited formal analysis that quantifies the reliability dynamics of an SSD RAID array under general settings. To this end, we propose a new continuous time Markov chain (CTMC) model to characterize the reliability dynamics of SSD RAID over time under two general settings: (1) fault tolerance against a general number of device failures and (2) non-uniform workload. We validate the correctness of our CTMC model via trace-driven simulations. Based on our model, we further analyze the impact of different RAID parameters on the reliability dynamics of an SSD RAID array.

CCS CONCEPTS

• **Computer systems organization** → **Reliability**; • **Information systems** → *Storage management*; • **Theory of computation** → *Probabilistic computation*;

KEYWORDS

SSD RAID, Reliability, CTMC, Transient Analysis

1 INTRODUCTION

1.1 Background and Motivation

NAND-flash-based solid-state drives (SSDs) have revolutionized traditional storage architectures since they provide higher I/O performance, lower power consumption, and higher reliability than hard disk drives (HDDs). SSDs have inherently distinct internal architecture and I/O characteristics from traditional HDDs. Specifically, they are composed of NAND flash memory that is organized as *blocks*, each of which further contains a fixed number (e.g., 64 or 128) of *pages* of size several kilobytes (e.g., 4KB or 8KB) each. SSDs perform data read and write based on three basic operations provided by flash memory: *read*, *write* (or *program*), and *erase*, which operate in units of pages, pages, and blocks, respectively. We refer readers to [1] for a more detailed description of the SSD architecture. As the price of SSDs has been dropping in recent years, we have witnessed an increasing adoption of commercial SSDs in both desktops and even large-scale data centers [30].

Although SSDs are increasingly adopted, there remain reliability concerns that prohibit their wide deployment. First, each flash block in an SSD can only tolerate a limited number of program/erase (P/E) cycles. The typical limit is 100K for single-level cell (SLC) SSDs, and it drops to 10K for multi-level cell (MLC) SSDs [5] and even several thousand for triple-level cell (TLC) SSDs [11]. In addition, even though SSDs often employ error correction codes (ECC) for data protection, flash errors are quite common and *uncorrectable* in SSDs due to read disturbs, write disturbs, and data retention [4, 10, 11, 27]. Even worse, the flash error rate of an SSD increases as flash blocks undergo more P/E cycles [4, 10, 27, 36, 37]; the reason is that the physical materials of the underlying flash cells, which trap electric charges to store bit information, deteriorate after multiple erase operations [4]. Furthermore, to increase the SSD capacity, manufacturers deploy high-density flash cells for SSDs, but make the trade-off of further reducing the P/E cycle limit of flash blocks and degrading flash reliability [11].

RAID (Redundant Array of Independent Disks) [32] provides a viable option for enhancing the reliability of SSD storage systems by striping data redundancy across multiple SSDs. SSD RAID has been well explored by the literature (see Section 7). However, deploying SSD RAID remains challenging and is subject to various pitfalls [16, 28], mainly because of distinct I/O characteristics of SSDs.

In this work, we argue that quantitative analysis of the reliability dynamics of general configurations of SSD RAID is of major significance, since a quantitative model can be used to guide system designers to decide the appropriate RAID configuration that satisfies the reliability and storage capacity constraints. In addition, a quantitative model can be used to assist system administrators to manage SSD RAID arrays, so as to satisfy the desired reliability requirements during system operation. For example, system administrators can use the model to estimate the run-time reliability of SSD RAID arrays, and then decide whether reliability enhancement techniques(e.g., replacing aged SSDs early, scheduling full recovery operations) are necessary for increasing the RAID reliability.

However, there remain limited mathematical models in the literature that accurately characterize the reliability of SSD RAID. Unfortunately, characterizing the reliability dynamics of SSD RAID accurately is a non-trivial task. First, the flash error rate of an SSD increases as flash blocks undergo more P/E cycles. The time-varying nature of the flash error rate needs to be taken into account in the reliability modeling of SSDs. Second, SSDs typically contain a large number of flash blocks. For example, a 256GB SSD contains around one million blocks of size 256KB each. Thus, keeping track of the dynamics of all these blocks implies a significant computational cost. Finally, the reliability dynamics of SSD RAID often depends on a wide variety of factors, including the RAID configurations, data loss patterns, and workload patterns. This requires a general analytical model that can be adaptive for various settings. The literature [23] makes the first attempt of characterizing the reliability dynamics of SSD RAID, but it only considers single-fault tolerance and does not take into account workload patterns.

1.2 Our Contributions

In this paper, we propose a *general* mathematical model that quantifies the reliability dynamics of SSD RAID. By general, we aim for two general settings: (1) an SSD RAID provides fault tolerance against a general number of SSD failures and (2) the model is applicable for non-uniform workloads. Our model takes into account the time-varying nature of flash errors of SSDs. To this end, we make the following contributions:

- We formulate a novel non-homogeneous continuous time Markov Chain (CTMC) model that addresses more general settings in the deployment of an SSD RAID array. The CTMC model take into account a general number of SSD failures.
- We extend the CTMC model to analyze the reliability dynamics of SSD RAID arrays subject to non-uniform workloads, in which SSDs may wear out at different rates due to the non-uniform access patterns.
- We conduct trace-driven simulations using the DiskSim simulator with SSD extensions [1] to validate the accuracy of our analysis. We also conduct extensive numerical analysis to study the impact of different factors, including workloads, error dynamics, error recovery capabilities, and array configurations. Our study provides insights into the selection of the appropriate RAID configuration for SSDs.

The remainder of the paper proceeds as follows. In Section 2, we provide the necessary background on SSD RAID organizations, and also characterize the error dynamics of SSDs. In Section 3, we present the CTMC model that characterizes the system dynamics of SSD RAID under uniform workload, and then derive the RAID reliability. In Section 4, we extend our modeling framework to address non-uniform workload. In Section 5, we validate the accuracy of our model via trace-driven simulations. In Section 6, we conduct numerical analysis using our model and study the impact of general settings on SSD RAID reliability. In Section 7, we present related work, and finally in Section 8, we conclude the paper.

2 PROBLEM FORMULATION

In this section, we first present the background details of the organization of an SSD RAID array. We then present mathematical models that characterize the system dynamics of an SSD RAID array and the time-varying flash error rate of an SSD. We also provide justifications for our modeling.

2.1 Basics of SSD RAID

We consider a device-level RAID array composed of n SSDs that can tolerate up to m failures (where $m < n$). Figure 1 shows the detailed organization, in which we number the n SSDs from 0 to $n-1$. Each SSD contains B physical blocks, each of which can sustain up to M P/E cycles. To implement RAID, we assume that each SSD is further divided into S non-overlapping *chunks*, and each chunk can be mapped to one or multiple physical pages. Correspondingly, the whole RAID array is divided into S *stripes*, each of which comprises exactly n chunks from the n SSDs. Among the n chunks in each stripe, $n-m$ of them are *data chunks* that contain the uncoded information, and the remaining m of them are *parity chunks* that are encoded from the $n-m$ data chunks of the same stripe. Stripes are encoded independently. We assume that the placements of data and parity chunks are rotated across stripes for load balancing [33], so that parity chunks are evenly distributed across SSDs. The value m determines the fault tolerance level. In particular, $m = 1$ means a RAID-5 system, while $m = 2$ means a RAID-6 system.

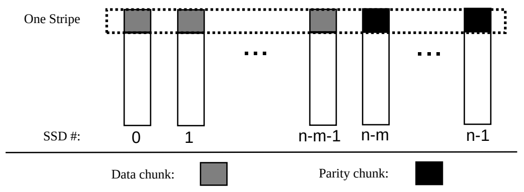


Figure 1: Organization of an SSD RAID array consisting of n SSDs and tolerating up to m failures.

Bit errors have been demonstrated to be prevalent in SSDs. In this work, we assume that data loss is due to bit errors only, although other failure types such as device crashes are also possible. We say that a chunk is an *erroneous chunk* if it contains uncorrectable bit errors; otherwise, we call it a *correct chunk*. In SSD RAID, a stripe can contain at most m erroneous chunks without data loss.

2.2 System Dynamics

We now characterize the system dynamics of an SSD RAID array, in particular, the wearing process in the presence of I/O requests. We assume that SSDs can achieve perfect wear-leveling, since efficient wear-leveling techniques are often deployed in SSDs to balance the number of P/E cycles on blocks. Note that if all blocks have the same number of P/E cycles, then all pages in an SSD also have the same number of P/E cycles. This further implies that all chunks in an SSD have the same number of P/E cycles since each chunk consists of one or multiple pages. Thus, we can focus on the number of P/E cycles at the chunk level for each SSD. Let $k_i(t)$, where $0 \le k_i(t) \le M$, denote the number of P/E cycles performed on each chunk on the i^{th} SSD at time t, where $i = 0, 1, \cdots, n-1$, and we call $k_i(t)$ the *age* of SSD i. We treat $k_i(t)$ as a continuous value in range $[0, M]$, so that the time-varying bit error rate in SSDs can be modeled by a continuous function (see Section 2.3). In addition, we define the age of an SSD RAID based on the number of P/E cycles performed on the array. Let $k_{RAID}(t)$ denote the age of a RAID array, which represents the total number of P/E cycles until time t. Since each block can only sustain M P/E cycles, an SSD must be replaced if its age reaches M. Here, we assume that the replacement is an immediate operation.

We proceed to characterize the relationship between the age of an SSD RAID array and the age of an SSD. Given the age of array $k_{RAID}(t)$, our goal is to derive the age of SSD i, i.e., $k_i(t)$ where $i = 0, 1, \cdots, n-1$. To model this relationship, we define another parameter r_{ij}, which denotes the ratio of the aging rate of SSD i to that of SSD j, where $i, j = 0, 1, \cdots, n-1$. We call r_{ij} as the *aging ratio*, whose physical meaning is that if the number of P/E cycles performed on SSD j is one, then the number on SSD i is r_{ij}. Clearly, $r_{ii} = 1$ where $i = 0, 1, \cdots, n-1$. Now $k_i(t)$ can be expressed in terms of $k_{RAID}(t)$ as follows.

$$k_i(t) = \left[\frac{k_{RAID}(t)}{(\sum_{l=0}^{n-1} r_{li})B} \right] \bmod M, \quad i = 0, 1, \cdots, n-1. \qquad (1)$$

In Equation (1), $1/(\sum_{l=0}^{n-1} r_{li})$ denotes the average probability of an P/E cycles being performed on SSD i, and mod represents the modulo operation. We use the modulo operation because each block can only sustain M P/E cycles, and an SSD will be replaced with a brand new one immediately when it reaches its P/E cycle limit.

We have assumed that parity chunks are evenly distributed among n SSDs (see Section 2.1). Thus, the aging ratio depends on the access patterns of a given workload pattern. For example, for uniform workload, data chunks have the same probability of being accessed by each request, so the aging ratio r_{ij} equals to one for any i, j. On the other hand, for non-uniform workload, some "hot" data chunks may be frequently accessed, so the number of writes to each SSDs varies significantly and the aging ratio may be different across SSDs. We assume that the aging ratio can be estimated by replaying the I/O trace via simulation.

2.3 Modeling the Time-varying Error Rate

2.3.1 Error Model. We now develop a mathematical model to characterize the time-varying bit error rate in SSDs. We assume that the error arrival processes of different chunks are independent, and let $\lambda_i(t)$ denote the bit error rate of each chunk in SSD i at time t. We model $\lambda_i(t)$ as a function of the age $k_i(t)$ of SSD i. More precisely, let $\lambda_i(t) = f(k_i(t))$, where $f(\cdot)$ is a monotone increasing function of $k_i(t)$. In this work, we consider a special case of $\lambda_i(t)$ by modeling the inter-arrival time of bit errors to be Weibull distribution [41], which is one of the most widely used lifetime distrubitons in reliability engineering and is also used to carve the relationship between the bit error rate and the number of erasures in an SSD in literature [23]. Mathematically, $\lambda_i(t)$ can be formally modeled as follows:

$$\lambda_i(t) = c\alpha \left[k_i(t)\right]^{\alpha-1}, \quad \alpha > 1, \tag{2}$$

where c is a constant and α is called the *shape parameter*. In this paper, we set $\alpha > 1$ to model the behavior that the bit error rate increases with the system age of an SSD. By tuning the shape parameter α, we can study various kinds of error behaviors. Specifically, if α is set as $1 < \alpha < 2$, then the error rate function $\lambda_i(t)$ is a concave function with respect to the system age; if $\alpha = 2$, then $\lambda_i(t)$ is a linear function; if $\alpha > 2$, then $\lambda_i(t)$ is a convex function. We call these three kinds of error rates the *concave* error rate, the *linear* error rate, and the *convex* error rate, respectively. We believe that the error rate model in Equation (2) is general enough to capture various kinds of error behaviors. We will study the impact of different error rates on SSD RAID reliability in Section 6.4.

2.3.2 Justifications. Our modeling of the bit error rate of an SSD in Equation (2) assumes a monotone increasing function. We provide justifications based on current measurement studies.

Many studies on SSDs show that the flash error rate of an SSD shows an increasing trend as flash blocks undergo more P/E cycles[4, 10, 27, 36, 37]. This increasing trend is obvious especially for MLC NAND flash [10]. While the increasing trend is not monotone, we model the error rate as a monotone increasing function as an approximation for tractability of our analysis.

We note that a recent large-scale field study by Facebook [26] argues that the SSD failure rate does not monotonically increase with the number of P/E cycles. However, we find that the increasing trend with the number of P/E cycles actually accounts for the majority of an entire SSD lifetime. Specifically, the SSD lifecycle failure pattern can be divided into three periods. The first two periods (namely the early detection and early failure periods) are very short and they both finish very quickly, while the last period (namely the wearout period) dominates and the error rate increases with the number of P/E cycles. For example, for the 720GB SSDs (in Platforms A and B), after writing around 15TB data, the early failure period ends. We can deduce that the early failure period ends at only around $15TB/720GB \approx 21$ P/E cycles for each block, assuming that perfect wear-leveling is used. Note that the number of P/E cycles that a block in an SSD can tolerate is typically on the order of thousands. Thus, if we assume that the P/E cycle limit is 1,000 cycles, then in over 98% of the SSD lifetime, an SSD is in the wearout period and its failure rate *does* increase with the number of P/E cycles. Similar statistics are also given in [26].

Another recent large-scale field study by Google [36] shows that both the raw bit error rate and the probability that an SSD sees an uncorrectable error increase with the number of P/E cycles, although the increasing rate is slower than commonly assumed.

We emphasize that our analysis framework on SSD RAID reliability does not depend on a particular (e.g., concave, linear, convex) error distribution. Our analysis remains applicable even defining $\lambda_i(t)$ as a different monotone increasing function of $k_i(t)$.

3 ANALYSIS OF RELIABILITY DYNAMICS

We propose a general mathematical model that analyzes the reliability dynamics of an SSD RAID array versus the age of the array. Specifically, we first develop a continuous time Markov chain (CTMC) model to characterize the error dynamics. And then solve the CTMC model via uniformization technique [6] and the optimization techniques [23]. While authors in [23] also propose a CTMC model for the reliability dynamics of SSD RAID, it is only limited for single-fault tolerance. Here, we extend the CTMC model to address two *general* settings: (1) we consider an SSD RAID array that tolerates a general number m of SSD failures, and (2) we consider both uniform and non-uniform workloads that can characterize general data access patterns (see Section 4). We emphasize that our contribution mainly lies in the formulation of a novel CTMC model that addresses more general settings in the deployment of an SSD RAID array (see Sections 3.1 and 4), while the techniques of solving the model (see Section 3.2) are adapted from the standard mathematical approaches in stochastic analysis.

We first consider the case of uniform workload where the aging ratio r_{ij}'s are equal to one, then extend the case of general non-uniform workload in Section 4. For uniform workload, the aging rates of different SSDs are equal (see Section 2.2), and we have

$$k_i(t) = \frac{k_{RAID}(t)}{nB} \bmod M, \quad \forall i.$$

Therefore, the error arrival rates of different chunks in the same stripe are also the same according to the definition in Equation (2). For simplicity, we represent the rate as $\lambda(t)$, and hence we have

$$\lambda(t) = c\alpha \left[\left(\frac{k_{RAID}(t)}{nB} \right) \bmod M \right]^{\alpha-1}, \quad \alpha > 1. \tag{3}$$

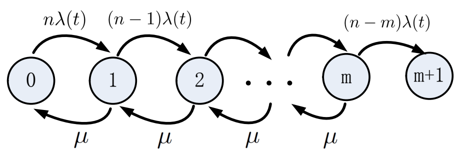


Figure 2: State transition diagram of the CTMC model under uniform workload.

3.1 Markov Model

Since the stripes of an SSD RAID array are encoded independently (see Section 2.1), and the error arrival processes in different chunks are assumed to be independent, we only need to develop a Markov model to characterize the dynamics of one particular stripe.

We formulate a CTMC model to characterize the dynamics of a stripe. Since the error arrival rates of different chunks in a stripe are the same, we define the state of a stripe as the number of erroneous chunks within the stripes. Formally, we say that a stripe is at state i if it contains i erroneous chunks, where $i = 0, 1, 2, \cdots, m$. Data loss happens if a stripe contains more than m erroneous chunks. We call this state as the "data loss" state, and denote it as state $m + 1$ for the ease of presentation. Let $X(t)$ denote the state of a stripe at time t. Then we have $X(t) \in \{0, 1, \cdots, m + 1\}$, $\forall t \geq 0$, and the CTMC model can be represented as $\{X(t), t \geq 0\}$.

We now specify the state transitions of the CTMC model. Note that if a stripe contains no more than m erroneous chunks, then they can always be reconstructed from the surviving chunks in the same stripe. We assume that erroneous chunks are reconstructed one by one, and the reconstruction time of one erroneous chunk follows an exponential distribution with rate μ. On the other hand, the error arrival processes of different chunks in a stripe are independent, and the arrival rate of each chunk is $\lambda(t)$. Figure 2 depicts the state transitions of the CTMC model with respect to the error arrival and recovery processes. Specifically, suppose that the stripe is currently at state j. Then it will move to state $j + 1$ if one more erroneous chunk appears in the stripe. The corresponding rate is $(n - j)\lambda(t)$ as the stripe contains $n - j$ correct chunks at this state, and each of them may independently experience errors with rate $\lambda(t)$. On the other hand, if one erroneous chunk is recovered from the surviving chunks, then the stripe will move to state $j - 1$, and the corresponding rate is μ.

To represent the state transitions of CTMC model , we let $\mathbf{Q}(t) = [q_{ij}(t)]_{0\leq i,j\leq m+1}$ be the rate matrix. We have $q_{ii}(t) = -\sum_{j\neq i} q_{ij}(t)$, and $q_{ij}(t)$'s when $j \neq i$ are expressed as:

$$q_{ij}(t) = \begin{cases} (n-i)c\alpha\left[\frac{k_{RAID}(t)}{nB} \bmod M\right]^{\alpha-1}, & 0\leq i\leq m, j=i+1, \\ \mu, & 1\leq i\leq m, j=i-1, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $q_{ij}(t)$ denotes the transition rate from i to j at time t.

To derive $X(t)$, let $\pi_j(t)$ be the probability of the CTMC model being at state j at time t, i.e., $\pi_j(t) = \text{Prob}\{X(t) = j\}$. Therefore, the system state of a stripe at time t can be characterized by the probability vector $\boldsymbol{\pi}(t) = (\pi_0(t), \pi_1(t), \cdots, \pi_{m+1}(t))$.

We now define a metric that characterizes the reliability of an SSD RAID array. Note that the state $m + 1$ denotes the data loss event, and it is an absorbing state. Thus, $\pi_{m+1}(t)$ represents the probability that data loss happens in a stripe before time t; in other words, $1 - \pi_{m+1}(t) = \sum_{j=0}^{m} \pi_j(t)$ denotes the probability that there is no data loss until time t. Since the error arrival and error recovery processes in different stripes are independent, we can define the *transient reliability* of an SSD RAID array as follows:

$$R(t) = \left(\sum_{j=0}^{m} \pi_j(t)\right)^S, \tag{5}$$

where S denotes the number of stripes in the array, and $R(t)$ is the probability that no stripe has encountered data loss until time t. We can now use the definition in Equation (5) to study the reliability dynamics of an SSD RAID array at different times. In particular, once the system state at any time t, i.e., $\boldsymbol{\pi}(t)$ is obtained, we can derive the transient reliability using Equation (5).

3.2 Transient Analysis

We now perform transient analysis to derive the transient state probability of the non-homogeneous CTMC model. Our analysis is mostly based on uniformization [6] and the optimization techniques [23], all of which follow the standard procedures in solving a CTMC model. To make the paper self-contained, we summarize the main idea that is sufficient for the analysis, while we refer readers to [6, 23] for formal analysis and proofs.

Although the error rate of each chunk $\lambda(t)$ is time-varying, it only varies with the array age, i.e., the number of P/E cycles (see Equation (3)). Thus, for the time period between two consecutive P/E cycles, the age of the array $k_{RAID}(t)$ remains unchanged, so the error rate $\lambda(t)$ is a constant. In other words, the CTMC model is homogeneous during the period of two consecutive P/E cycles. Let T denote the average length of the time period of two consecutive P/E cycles performed on the SSD RAID array, and we can rewrite the age of the array as $k_{RAID}(t) = k$ if $t = kT$. The original non-homogeneous CTMC model $\{X(t), t \geq 0\}$ can now be decomposed into multiple time-homogeneous CTMC models $\{X(t), t \in [kT, (k+1)T)\}$, where $k = 0, 1, 2\cdots$. For each time-homogeneous CTMC model, we can use the uniformization technique [6] for analysis. Specifically, given the system state at time kT, which is the initial state of the CTMC $\{X(t), t \in [kT, (k + 1)T)\}$, we can approximate the system state at time $(k + 1)T$, denoted by $\boldsymbol{\pi}((k + 1)T)$, with a given error bound ϵ_k. Mathematically, let $\tilde{\boldsymbol{\pi}}((k + 1)T)$ denote the approximated system state at time $(k + 1)T$. We can derive $\tilde{\boldsymbol{\pi}}((k + 1)T)$ by using the uniformization technique [6], and the results are stated as follows.

Baseline computations of the CTMC model: Given the initial state $\boldsymbol{\pi}(kT)$ of the CTMC model $\{X(t), t \in [kT, (k + 1)T)\}$ and the upper bound of the approximation error ϵ_k, the system state at time $(k + 1)T$ can be approximated as follows.

$$\tilde{\boldsymbol{\pi}}((k+1)T) = \sum_{n=0}^{U_k} e^{-\Lambda_k T}\frac{(\Lambda_k T)^n}{n!}\mathbf{v}_k(n), \tag{6}$$

where $\mathbf{v}_k(n) = \mathbf{v}_k(n-1)\mathbf{P}_k$, $\mathbf{v}_k(0) = \boldsymbol{\pi}(kT)$, $\mathbf{P}_k$ is defined as $\mathbf{P}_k = \mathbf{I} + \frac{\mathbf{Q}_k}{\Lambda_k}$ with $\mathbf{I}$ being an identity matrix, $\mathbf{Q}_k$ being the rate matrix computed via Equation (4) by substituting $k_{RAID}(t)$ with k, and Λ_k satisfying $\Lambda_k \geq \max_{0\leq i\leq m+1} |-q_{ii}(kT)|$, and U_k is an integer that satisfies the following inequality.

$$\sum_{n=0}^{U_k} e^{-\Lambda_k T}\frac{(\Lambda_k T)^n}{n!} \geq 1 - \epsilon_k. \tag{7}$$

We point out that the above solution is a direct application of the uniformization technique, and the approximation error in Equation (6) is introduced due to the truncation of the infinite series. The truncation point U_k is determined by ϵ_k. In our study, we set $\epsilon = 10^{-9}$, and deduce U_k accordingly.

Note that computing the system state at time kT requires to first derive $\boldsymbol{\pi}(T)$ from the initial state $\boldsymbol{\pi}(0)$, followed by deriving $\boldsymbol{\pi}(2T)$ from $\boldsymbol{\pi}(T)$, and so on. Thus, the computation time will be very huge for a very large k. For example, a 256GB SSD may contain one million blocks of size 256KB each, and each block may sustain 10K P/E cycles. It is possible for an array containing multiple SSDs to have at least 10^{10} P/E cycles.

To speed up computations, we apply the optimization techniques in [23]. Specifically, instead of solving time-homogeneous CTMC models for each time period, we merge s time periods into a large time interval, and perform transient analysis on the combined time-homogeneous CTMC models with s successive periods. Therefore, the CTMC models $\{X(t), t \in [kT, (k+1)T)\}$ (where $k = 0, 1, 2, \cdots$) can be rewritten as $\{X(t), t \in [lsT, lsT + sT)\}$ (where $l = 0, 1, \cdots$). We can then derive $\boldsymbol{\pi}((l+1)sT)$ from $\boldsymbol{\pi}(lsT)$ by using single transient analysis, thereby greatly reducing the computation time.

To construct a homogeneous CTMC model to approximate the non-homogeneous one $\{X(t), t \in [lsT, lsT + sT)\}$, we carefully configure the error rate to make the approximation accurate. Formally, if we denote the constructed homogeneous CTMC as $\{\bar{X}(t), t \in [lsT, lsT + sT)\}$, and denote its rate matrix as $\bar{\mathbf{Q}}_l = [\bar{q}_{ij}(l)]$, then we have $\bar{\mathbf{Q}}_l = \mathbf{Q}_{ls+\frac{s}{2}}$. We use $\bar{\boldsymbol{\pi}}((l+1)sT)$ to denote the system state at time $(l+1)sT$ for the CTMC model $\{\bar{X}(t), t \in [lsT, lsT + sT)\}$.

Again, we can approximate the system state $\bar{\boldsymbol{\pi}}((l+1)sT)$ by using the uniformization technique [6]. Mathematically, we define $\bar{\Lambda}_l \geq \max_{0 \leq i \leq m+1} | - \bar{q}_{ii}(l)|$ and let matrix $\bar{\mathbf{P}}_l = \mathbf{I} + \frac{\bar{\mathbf{Q}}_l}{\bar{\Lambda}_l}$. We use $\tilde{\bar{\boldsymbol{\pi}}}((l+1)sT)$ to denote the approximated system state at time $(l+1)sT$ with a given upper bound $\bar{\epsilon}_l$. The approximation results can be derived as follows.

Accelerated computations of the CTMC model: To analyze the CTMC model $\{\bar{X}(t), t \in [lsT, lsT + sT)\}$, the approximation of the system state $\bar{\boldsymbol{\pi}}((l+1)sT)$, which is denoted as $\tilde{\bar{\boldsymbol{\pi}}}((l+1)sT)$, can be derived as:

$$\tilde{\bar{\boldsymbol{\pi}}}((l+1)sT) = \sum_{n=0}^{\bar{U}_l} e^{-\bar{\Lambda}_l sT} \frac{(\bar{\Lambda}_l sT)^n}{n!} \mathbf{v}_k(n), \tag{8}$$

where $\mathbf{v}_k(n) = \mathbf{v}_k(n-1)\bar{\mathbf{P}}_l$, $\mathbf{v}_k(0) = \tilde{\bar{\boldsymbol{\pi}}}(lsT)$, and $\bar{U}_l$ is an integer that satisfies the following inequality.

$$\sum_{n=0}^{\bar{U}_l} e^{-\bar{\Lambda}_l sT} \frac{(\bar{\Lambda}_l sT)^n}{n!} \geq 1 - \bar{\epsilon}_l. \tag{9}$$

Note that the solution of the CTMC model $\{\bar{X}(t), t \in [lsT, lsT + sT)\}$ is a direct application of the results in Equation (6) and Equation (7), with the only difference that the length of a time interval is sT instead of T. The value of s trades between performance and accuracy (see Section 6.2). In our study, we choose $s = \frac{BM}{20}$, which achieves reasonable performance while the approximation error remains negligible (see Section 5 and Section 6.2).

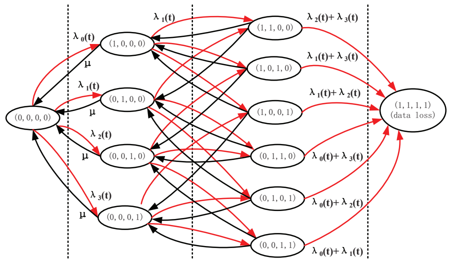


Figure 3: State transition diagram of the CTMC model with $(n, m) = (4, 2)$ under non-uniform workload.

4 NON-UNIFORM WORKLOAD

We now consider the case of non-uniform workload, in which the aging ratio r_{ij}'s are not equal to one. In this case, the error rates of chunks in different SSDs are different as they are subject to different write intensities and their ages are no longer the same (see Equations (1) and (2)). Thus, we cannot simply use the number of erroneous chunks to characterize the state of a stripe as in Section 3, but instead, we need to remember the positions of the erroneous chunks, or equivalently, the identifier of the SSD (i.e., $0, 1, \cdots, n-1$) in which the erroneous chunks reside.

We formulate a CTMC model to characterize the dynamics of a stripe. To recognize the position of each erroneous chunk, we use an n-dimensional 0-1 vector to characterize the errors within a stripe. Formally, let $\mathbf{b} = (b_0, b_1, \cdots, b_{n-1})$ be a vector where $b_i = 1$ indicates that the chunk from SSD i is an erroneous chunk, and 0 otherwise. Note that a stripe can have at most m erroneous chunks before data loss, so any vector $\mathbf{b}$ with $\sum_{i=0}^{n-1} b_i > m$ can be used to represent the state of data loss. For ease of presentation, we use the vector $\mathbf{e} = \{1, 1, \cdots, 1\}$ to denote the state of data loss. Let $X(t)$ denote the state of a stripe, and $X(t) \in \{(b_0, b_1, \cdots, b_{n-1}) | b_i = 0, 1 \text{ and } \sum_{i=0}^{n-1} b_i \leq m \text{ or } \sum_{i=0}^{n-1} b_i = n\}$. In particular, the state $X(t) = (0, 0, \cdots, 0)$, which we denote as $\mathbf{0}$, implies that no error appears in the stripe.

Take an SSD RAID array with $(n, m) = (4, 2)$ as an example, we illustrate the state transitions of the CTMC model. Figure 3 depicts the state transitions. Specifically, if the current state is $(0, 0, 0, 0)$, it means that the stripe contains no erroneous chunk. Also, it may transit to one of the states $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, $(0, 0, 1, 0)$ and $(0, 0, 0, 1)$, and the corresponding rates are $\lambda_0(t)$, $\lambda_1(t)$, $\lambda_2(t)$ and $\lambda_3(t)$, respectively. The physical meaning of transitions is that one erroneous chunk appears in SSD i, where $i = 0, 1, 2$, or 3. On the other hand, if the stripe already contains one erroneous chunk in the current state, e.g., $(1, 0, 0, 0)$, then it may transit to state $(0, 0, 0, 0)$ when the erroneous chunk is recovered. The stripe may also transit to a state containing two erroneous chunks if one more error appears. For example, it may transit to state $(1, 1, 0, 0)$, $(1, 0, 1, 0)$ and $(1, 0, 0, 1)$ if one more erroneous chunk appears at SSD 1, SSD 2 and SSD 3, respectively. The corresponding rates are $\lambda_1(t)$, $\lambda_2(t)$, and $\lambda_3(t)$. Note that for an array which contains four SSDs and can tolerate up to two failures, the CTMC has 12 states, and we use the state $(1, 1, 1, 1)$ to denote the state of data loss.

Figure 4 depicts the state transitions of the CTMC model $\{X(t), t \geq 0\}$ under general array configurations. We define $\mathbf{e}_j$ as the vector in which only the j^{th} element is one, i.e., $\sum_{j=0}^{n-1} \mathbf{e}_j = \mathbf{e}$. If the current

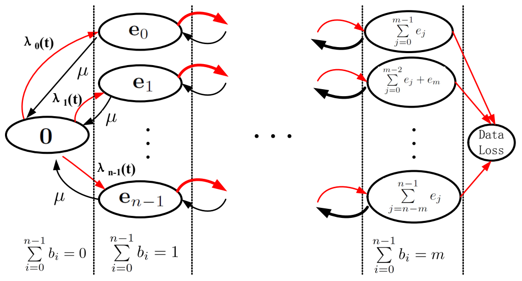


Figure 4: State transition diagram of the CTMC model with general configurations under non-uniform workload.

state is $\mathbf{0}$, it means that no erroneous chunk exists, so the stripe may transit to a state $\mathbf{b}$ containing exactly one erroneous chunk, i.e., the state where $\sum_{i=0}^{n-1} b_i = 1$. Moreover, there are n types of such transitions, which correspond to the cases where the erroneous chunk is at SSD i ($i = 0, 1, \cdots, n-1$), and the corresponding rates are $\lambda_0(t), \lambda_1(t), \cdots, \lambda_{n-1}(t)$, respectively. Thus, a stripe may transit from state $\mathbf{0}$ to state $\mathbf{e}_j$ with rate $\lambda_j(t)$. In general, if a stripe is currently at state $\mathbf{b}$ where $\sum_{i=0}^{n-1} b_i = j$, which means that this stripe contains j erroneous chunks, then it may transit to state $\mathbf{b}^+$ where $\sum_{i=0}^{n-1} b_i^+ = j+1$ if one more erroneous chunk appears, and the number of such transitions is $n-j$. On the other hand, it may also transit to state $\mathbf{b}^-$ in which $\sum_{i=0}^{n-1} b_i^- = j-1$ if one erroneous chunk is recovered, and the number of such transitions is j. We still assume that the time to recover an erroneous chunk in a stripe is exponentially distributed with rate μ as in Section 3, and random tie-breaking is used to select a chunk for recovery if multiple erroneous chunks are contained in a stripe.

We can now express the transition rate matrix of the CTMC $\{X(t), t \geq 0\}$, which we denote as $\mathbf{Q}(t) = [q_{\mathbf{bb}'}(t)]$ where $q_{\mathbf{bb}'}(t)$ denotes the transition rate from $\mathbf{b}$ to $\mathbf{b}'$ at time t. We have $q_{\mathbf{bb}}(t) = -\sum_{\mathbf{b}' \neq \mathbf{b}} q_{\mathbf{bb}'}(t)$, where $q_{\mathbf{bb}'}(t)$'s ($\mathbf{b}' \neq \mathbf{b}$) are stated as follows.

$$q_{\mathbf{bb}'}(t) = \begin{cases} \lambda_j(t), & 0 \leq \sum_{i=0}^{n-1} \mathbf{b}_i < m, \mathbf{b}' - \mathbf{b} = \mathbf{e}_j, j = 0, \cdots, n-1, \\ \sum_{j \in S} \lambda_j(t), & \sum_{i=0}^{n-1} \mathbf{b}_i = m, \mathbf{b}' = \mathbf{e}, S = \{j | (\mathbf{b}' - \mathbf{b}) \& \mathbf{e}_j = \mathbf{e}_j\}, \\ \frac{\mu}{(\sum_{i=0}^{n-1} \mathbf{b}_i)}, & 0 < \sum_{i=0}^{n-1} \mathbf{b}_i \leq m, \mathbf{b} - \mathbf{b}' = \mathbf{e}_j, j = 0, \cdots, n-1, \\ 0, & \text{otherwise,} \end{cases}$$

where $\lambda_j(t) = c\alpha \left[\frac{k_{RAID}(t)}{(\sum_{l=0}^{n-1} r_{lj})B} \bmod M \right]^{\alpha-1}$ and the notation & denotes the bitwise AND operation.

To derive $X(t)$, let $\boldsymbol{\pi}(t)$ denote the state probability of the CTMC model at time t. In particular, if the stripe is at state $\mathbf{b}$ at time t, then we have $\pi_{\mathbf{b}}(t) = \text{Prob}\{X(t) = \mathbf{b}\}$. Similar to the definition in Equation (5), we define the transient reliability of an SSD RAID array as follows.

$$R(t) = \left(\sum_{\mathbf{b}, \sum b_i \leq m} \pi_{\mathbf{b}}(t) \right)^S, \tag{10}$$

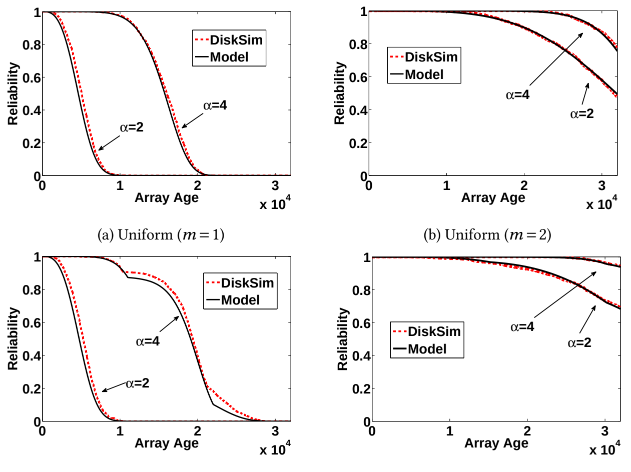
Again, $R(t)$ denotes the probability that no stripe has encountered data loss until time t.

To derive the reliability dynamics, we note that for a given rate matrix $\mathbf{Q}(t)$, we can directly apply the computation framework in Section 3.2. The only difference is that the dimension of the rate matrix in the non-uniform workload case is much larger. Nevertheless, the number of states in this CTMC model $\{X(t), t \geq 0\}$ is $1 + \sum_{i=0}^{m} \binom{n}{i}$. Even though it is large for general values of n and m, it has a small dimensionality for practical RAID settings. For example, Facebook uses the configuration $n = 14$ and $m = 4$ [35], and the number of states is smaller than 1,500. In addition, each state in the CTMC model cannot transit to an arbitrary state, but instead it only transits to at most n states in one step. Therefore, the computation process presented in Section 3.2 is computationally feasible, and it can be directly used to solve the extented CTMC model developed in this section.

5 MODEL VALIDATION

In this section, we validate the accuracy of our model via trace-driven evaluations using the DiskSim simulator [3] with SSD extension [1]. Since an SSD contains multiple flash chips, and these chips are configured to work independently and handle I/O requests in parallel, our validation considers RAID configurations at the chip level instead of the device level for simplicity. In particular, each chip is configured to have its own data bus, and RAID is implemented in the flash translation layer of the SSD controller. To speed up the simulation, we consider a small-scale RAID array. Specifically, we configure each chip to contain only 80 (i.e., $B = 80$) physical blocks, and set the P/E cycle limit of each block as 50 (i.e., $M = 50$) so that it wears out blocks quickly. We fix $n = 8$ chips. Thus, the SSD RAID array can sustain at most $n \times B \times M$ = 8×80×50=32K P/E cycles before all chips wear out in the equal aging rate case. Since an SSD usually deploys spare blocks, we set the spare factor as 0.2, which means the logical capacity of each chip is 64 blocks. We also configure each block with 64 pages of size 4KB each, and also set the chunk size as 4KB. Note that even for such a small-scale RAID array, it may still takes hours to estimate the reliability dynamics using trace-driven simulations on a commodity PC.

To simulate the error arrival and error recovery processes, we create two types of requests, which we call *error arrival events* and *error recovery events*, and inject them to the simulator. The arrival times of these events are determined according to the arrival rate $\lambda_i(t)$ and the recovery rate μ. In our validations, we fix $\mu = 1$, and set the maximum error rate $\lambda_i(M) = c\alpha M^{\alpha-1}$ as 10^{-3}. We consider two different error rates by setting the shape parameter α equal to 2 and 4, and the constant c can be derived accordingly based on the shape parameter α. We generate uniform workload to emulate the case of equal aging rates. Specifically, for each write request, we generate its address uniformly at random from the whole logical address space. This ensures that the writes are equally distributed across all chips. We point out that even if FTL employs address translation, it does not influence the distribution of writes to flash chips, since flash chips are configured to operate independently and data will not be moved across chips, i.e., update to a data page will still be directed to the chip to which the data originally belongs. We Note that this is also the default setting of the SSD simulator. After we evenly distribute writes across chips, all chips should have the same aging rate. In the case of unequal aging rates, we configure 80% of requests to uniformly access the data in the first SSD, and



(a) Uniform ($m = 1$)
(b) Uniform ($m = 2$)
(c) Non-uniform ($m = 1$)
(d) Non-uniform ($m = 2$)

Figure 5: Analysis validation for different workloads, array configurations, and error dynamics.

the remaining 20% of requests to uniformly access the data of other SSDs. Thus, the first SSD ages faster than the others. We measure the aging ratio when simulation stops, and use the measured aging ratio to derive the theoretical result of RAID reliability by using our model.

We obtain the reliability results as follows. For DiskSim simulations, we make 1,000 runs with different random seeds, and in each simulation run, we generate a workload and run until a data loss happens or all blocks wear out (i.e., after 32K P/E cycles have been performed). We record the array age when each simulation stops, and use the 1,000 values to derive the probability of no data loss. For modeling, we compute the reliability metrics based on Equations (5) and (10) for the cases of uniform and non-uniform workloads, respectively. We consider a RAID array with $n = 8$, and consider two values of m: $m = 1$ (RAID-5) and $m = 2$ (RAID-6).

Figure 5 shows both simulation and modeling results. Figures 5(a) and 5(b) show the results of RAID-5 and RAID-6 under uniform workload, respectively, and Figure 5(c) and Figure 5(d) show the results of RAID-5 and RAID-6 under non-uniform workload. Each figure shows the results of two error rates for $\alpha = 2$ and $\alpha = 4$. In each figure, the x-axis represents the array age, which denotes the number of P/E cycles performed on the array, while the y-axis shows reliability, which denotes the probability of no data loss until the array ages at the time indicated by the x-axis. Recall that T denotes the average time length of two consecutive P/E cycles (see Section 3.2), so the array age shown in the x-axis can be translated to the average total elapsed time of the array if we multiply the number of P/E cycles performed by T. Note that T decreases if the I/O arrival rate of a given workload increases. From the figures, we can see that our model accurately quantifies the reliability dynamics of an SSD RAID array for all cases. Moreover, the reliability of the array drops very fast after a certain age.

6 RELIABILITY EVALUATION

In this section, we conduct numerical analysis on an SSD RAID array using our model. We consider more practical, larger-scale settings, and configure the parameters of our model based on realistic SSD and RAID configurations.

6.1 Model Parameters

We first describe the model parameters used for our numerical analysis. We consider SSDs with physical capacity of 256GB each, and configure each block in an SSD with 64 pages of page size 4KB each. Thus, there are 1M blocks in an SSD, i.e., $B = 2^{20}$. We set the P/E cycle limit of each block as 10K, i.e, $M = 10,000$. We set the spare factor to be 0.2, and configure the chunk size to be equal to the block size, meaning that the number of stripes in an array is $S = 0.8 \times B$.

To configure the error arrival and error recovery rates, we assume that 4-bit ECC is employed to protect 512 bytes of data. We choose the maximum uncorrectable bit error rate (UBER) from the range $[10^{-16}, 10^{-18}]$ [2], which is the UBER when flash blocks reach the P/E cycle limit. Since the chunk size is set as 256KB, the probability of a chunk containing at least one error can be computed, which is in the range of $[2 \times 10^{-10}, 2 \times 10^{-12}]$. To configure the error arrival rate per chunk, we note that in enterprise storage systems, a storage array may have several hundred of gigabytes of data being accessed each day [29]. If we set the amount of data being accessed each day as 1TB, which corresponds to 50 blocks per second, then the error arrival rate of each chunk at its rated lifetime, denoted by $\lambda(M) = c\alpha M^{\alpha-1}$, is in the range of $[10^{-8}, 10^{-10}]$. In our study, we fix $\lambda(M) = 10^{-9}$, so that the corresponding parameter c can be derived with a given α. With respect to the error recovery rate μ, we set the default value as 10^{-5}, and also study its impact on RAID reliability in Section 6.4. For the parameter T, which is the average duration between two consecutive erase operations, we configure it as follows. Since we assume there are 1TB of writes per day, the inter-arrival time of two consecutive page writes is around 3×10^{-4} seconds. Since we configure each block to have 64 pages, erase operation will be triggered after every 64 page writes. However, except for external user writes, SSDs may also introduce additional page writes due to garbage collection [13], so we fix $T = 10^{-2}$ seconds.

In the following, we present numerical results using our model with the above parameters. In particular, we first study the impact of parameter s on the tradeoff of performance and accuracy (see Section 6.2). We then study the impact of workloads by varying the parameter of aging ratio (see Section 6.3), the impact of error dynamics by varying α and μ (see Section 6.4), and the impact of different array configurations by varying n and m (see Section 6.5).

6.2 Impact of Stepsize s

In Section 3.2, to accelerate the transient analysis, we combine s consecutive time periods into one time interval, and construct a single CTMC model to approximate the system dynamics in combined time interval. However, this optimization introduces errors, and hence there is a tradeoff between performance and accuracy for different values of the parameter s. Here, we study this tradeoff. We use the parameters in Section 6.1 to simulate a practical and large-scale setting. In this evaluation, we fix the array configuration and error dynamics by setting $(n, m) = (8, 2)$ and $\alpha = 4$. We also fix the aging ratio by setting an equal aging rate.

Table 1 shows the tradeoff results. In particular, we show the numerical result of $R(nBM)$, which denotes the RAID reliability at the time when the system undergoes nBM P/E cycles, as well as the time required to compute $R(nBM)$ under different settings of s.

Table 1: Tradeoff between performance and accuracy.

s	time	$R(nBM)$
$4BM \approx 4.2 \times 10^{10}$	24 ms	0.9145
$2BM \approx 2.1 \times 10^{10}$	27 ms	0.8296
$BM/20 \approx 5.24 \times 10^{8}$	34 ms	0.7897
$BM/10^3 \approx 1.05 \times 10^{7}$	99 ms	0.7898
$BM/10^7 \approx 1.05 \times 10^{3}$	97132 ms	0.7888
$BM/10^8 \approx 105$	973600 ms	0.7957

We point out that the converged result of $R(nBM)$ can be used to denote the actual reliability as there is no approximation error if we set $s = 1$. Here, we do not run simulations to obtain the actual result because the running time will be extremely large for such a large-scale system considered in this section. The results show that the tradeoff indeed exists: while a large s can save a lot of computation time, it also introduces a big error. Fortunately, the reliability quickly converges. For example, when $s = BM/20$, the reliability is already very close to the actual value (note that this is also validated in Section 5), while the computation time under this setting is still very small. Thus, we set $s = BM/20$ by default.

6.3 Impact of Workloads

We first study the impact of workloads. Specifically, we compare RAID reliability under uniform workload (i.e., an equal aging rate) with that under non-uniform workload (unequal aging rates). For uniform workload, the aging ratio is set to $(1 : 1 \cdots 1 : 1)$. For non-uniform workloads, we consider two particular aging ratios for simplicity of illustration: $(1 : 1 \cdots 1 : 2)$ and $(1 : 1 \cdots 1 : 5)$. It means that the first $n - 1$ drives age at the same rate, and the last drive age two times or five times faster, respectively.

Figure 6 shows the numerical results under three different settings, with different array configurations (n, m) and the error shape parameter α. We observe that the reliability dynamics vary across workloads. Specifically, for RAID arrays with low reliability, the difference of reliability under different aging ratios is very small (see Figure 6(a)). On the other hand, the difference becomes significant for highly reliable RAID arrays (see Figure 6(b)), and unequal aging rates achieve higher reliability, mainly because SSDs wear out one by one in this case and this avoids correlated failures [2].

Figure 6(c) shows that the reliability curve under a non-uniform workload contains inflection points, and this happens when an SSD wears out and is replaced with a brand-new SSD. Also, the reliability in the case of unequal aging rates (e.g., for the aging ratio $(1 : 1 \cdots 1 : 5)$) is *smaller* than that in the case of an equal aging rate $(1 : 1 \cdots 1 : 1)$ before the inflection point, while it becomes larger afterwards. The main reason is that the flash error rate increases as an SSD undergoes more P/E cycles, and an SSD replacement slows down the decreasing rate of RAID reliability. Note that inflection points may exist at different array ages with respect to different aging ratios. They may appear too early such that the RAID reliability is still one, or too late so that the reliability has already dropped to zero. Thus, we may not observe any inflection point in some reliability curves.

Note that our model can also analyze any general aging ratio. To further show the reliability implications, we use two distributions to generate a general form of aging ratio. In this evaluation, we fix the array configuration and error dynamics by setting $(n, m) = (8, 2)$ and $\alpha = 2$, that is, we use the same parameters as in Figure 6(b). In particular, Figure 7(a) shows the reliability dynamics in the case where the aging rates of SSDs are generated from a truncated normal distribution, in which the aging rate of SSD i ($0 \le i \le n-1$) is $[\int_i^{i+1} f(x)]/[\int_0^n f(x)]$ where $f(x)$ is the probability density function of a normal distribution $N(n, \sigma^2)$. We show the reliability dynamics by setting $\sigma = 5$ and $\sigma = 3$, and the corresponding aging ratios are (0.058, 0.077, 0.098, 0.119, 0.14, 0.158, 0.171, 0.179) and (0.012, 0.026, 0.05, 0.088, 0.136, 0.189, 0.236, 0.263), respectively. Similarly, in Figure 7(b), we use a Zipf distribution to generate the aging ratio, and precisely, the aging rate of SSD i ($0 \le i \le n - 1$) is $[(i + 1)^{-\gamma}]/[\sum_{i=0}^{n-1}(i + 1)^{-\gamma}]$. We set the parameter $\gamma = 1$ and $\gamma = 2$, and the corresponding aging ratios are (0.367, 0.184, 0.123, 0.092, 0.074, 0.061, 0.053,0.046) and (0.655, 0.164, 0.073, 0.041, 0.026, 0.018, 0.013, 0.01), respectively. For comparison, we also show the reliability under an equal aging rate in both figures. We note that the reliability curves under different workloads share similar trends, i.e., they all decrease as the array ages. In the following, we only focus on uniform workload.

6.4 Impact of Error Dynamics

We now study the impact of error dynamics by varying α and μ. Figures 8(a) and 8(b) show the impact of α under $(n, m) = (8, 1)$ and $(n, m) = (8, 2)$, respectively. We consider three values of α: $\alpha = 2$ (linear error rate), $\alpha = 4$ (convex error rate), and $\alpha = 1.5$ (concave error rate). We observe that the reliability is the highest for a convex error rate, and it is the lowest for a concave error rate. The reason is that when the maximum error rate $\lambda_i(M)$ is fixed, a convex error rate gives the smallest bit error rate (see Equation (2)). Also, increasing the reliability of single SSDs (e.g., by decreasing the bit error rate) increases the RAID reliability, and the increase is more significant for a smaller fault tolerance degree m.

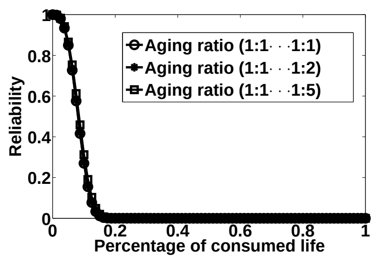
Figures 8(c) and 8(d) show the impact of μ under $(n, m) = (8, 1)$ and $(n, m) = (8, 2)$, respectively. We consider three values of μ: $\mu = 10^{-4}$ (fast recovery), $\mu = 10^{-5}$ (moderate recovery and the default value), and $\mu = 10^{-6}$ (slow recovery). We see that increasing the recovery rate greatly improves the RAID reliability, and the gain is more significant for a larger fault tolerance degree m.

6.5 Impact of Array Configurations

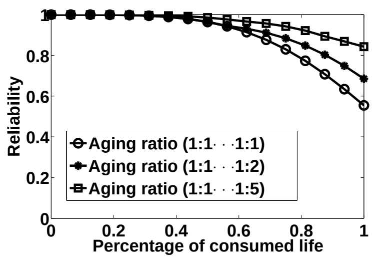
We now study the impact of different array configurations determined by the parameters n and m, they determine the storage overhead, which we define as $\frac{m}{n-m}$, and the reliability dynamics.

We first study the impact of array configurations with a fixed storage overhead. Specifically, we configure (n, m) to be $(4, 1)$, $(8, 2)$, and $(12, 3)$, all of which has the storage overhead fixed at $\frac{1}{3}$. Figure 9 shows the results for $\alpha = 2$ (linear error rate) and $\alpha = 4$ (non-linear error rate). We see that different array configurations have a significant impact on the RAID reliability, even though the storage overhead is fixed. In particular, a larger array (i.e., both n and m are large) always achieves a higher RAID reliability.

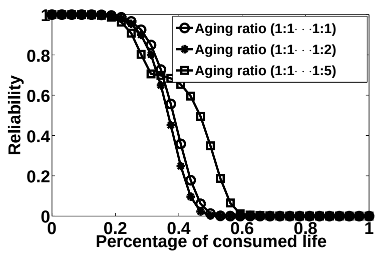
We further study the impact of array configurations by changing either n or m at a time (i.e., the storage overhead also changes). Specifically, we set $(n, m) = (8, 1)$ as default, and change the array configuration by either decreasing n from $n = 8$ to $n = 4$ or increasing m from $m = 1$ to $m = 2$. Figure 10 shows the results for different α's. The RAID reliability increases with the storage overhead. That



(a) $(n, m) = (8, 1)$, $\alpha = 2$

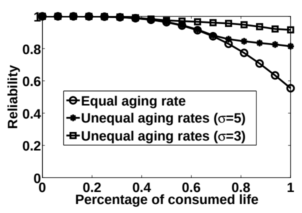


(b) $(n, m) = (8, 2)$, $\alpha = 2$

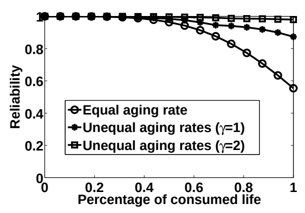


(c) $(n, m) = (8, 1)$, $\alpha = 4$

Figure 6: Impact of workloads on RAID reliability under different array configurations and error dynamics.

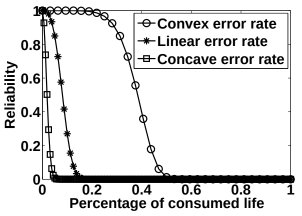


(a) Normal: $N(n, \sigma^2)$

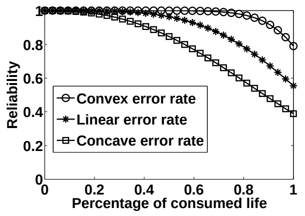


(b) Zipf: $f(k) \propto k^{-\gamma}$

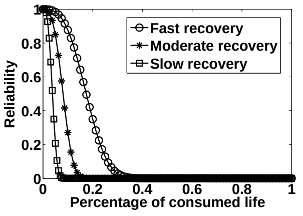
Figure 7: Reliability dynamics under general distributions of aging rates.



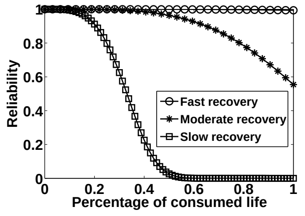
(a) Impact of error rate ($(n, m) = (8, 1)$)



(b) Impact of error rate ($(n, m) = (8, 2)$)

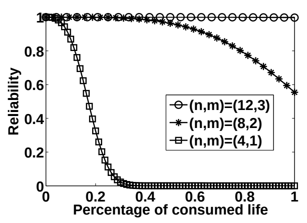


(c) Impact of recovery rate ($(n, m) = (8, 1)$)

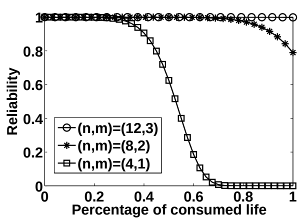


(d) Impact of recovery rate ($(n, m) = (8, 2)$)

Figure 8: Impact of different error arrival and recovery rates under different array configurations.

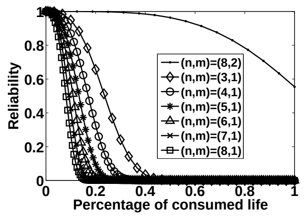


(a) $\alpha = 2$

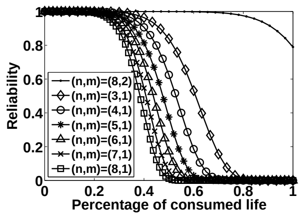


(b) $\alpha = 4$

Figure 9: Impact of (n, m) with fixed storage overhead.



(a) $\alpha = 2$



(b) $\alpha = 4$

Figure 10: Impact of (n, m) with variable storage overhead.

is, the storage-reliability trade-off exists if we only change the array configuration in one dimension (either n or m). However, the trade-off no longer holds if both dimensions are varied at the same time. For example, even if the configuration with $(n, m) = (8, 2)$ incurs smaller overhead than that of $(n, m) = (3, 1)$, the RAID reliability in the former case is larger. Also, incrementing m brings a higher reliability gain than decrementing n, although both cases increase the storage overhead.

7 RELATED WORK

NAND-flash-based SSDs have been studied in several aspects. Examples include (1) empirical study of the intrinsic characteristics of SSDs [1, 5, 17], (2) performance modeling on write performance and garbage collection algorithms [7, 12, 13, 22, 38, 39], and (3) reliability analysis via experiments and mathematical models [11, 23, 26, 36]. To improve SSD reliability, RAID has been adopted [2, 15, 18–21, 25, 31, 40]. These studies mainly focus on the architectural design of SSD RAID so as to improve the performance and durability of traditional RAID schemes. On the other hand, careless designs of SSD RAID can degrade reliability [16, 28].

On the theoretical side, RAID reliability was first formulated by Gibson and Patterson as mean-time-to-data loss (MTTDL) [9]. After that, various models were proposed to improve the reliability model of RAID systems, e.g., [8, 14, 34]. Also, Machida et al. proposed a Markov regenerative process model to address the non-exponential disk rebuild time for analyzing the performability of RAID storage systems [24]. For reliability analysis of SSDs, Li et al. [23] develop a mathematical model to analyze the reliability of SSD RAID arrays that provide single-fault tolerance. In particular, they focus on studying the impact of different parity distributions among devices, and compare the reliability of Diff-RAID [2], which places parities unevenly among devices, with that of traditional RAID-5. In contrast, we address this problem from a more general perspective.

Specifically, our model analyzes the reliability dynamics of SSD RAID that tolerates *any number* of failures subject to various array configurations and non-uniform workloads.

8 CONCLUSIONS

In this paper, we develop a general mathematical model to analyze the reliability dynamics of SSD RAID arrays with different configurations. Specifically, we build Markov chain models to characterize the error dynamics of SSD RAID, and use the uniformization and interval linearization technique to perform transient analysis. The accuracy of our model is validated via trace-driven simulations.

Based on our mathematical model, we conduct extensive numerical evaluations on the reliability of SSD RAID. We observe that RAID reliability under non-uniform workload may be higher than that under uniform workload, so making SSDs in a RAID age at different rates and wear out separately may be beneficial for RAID reliability, while this benefit requires a careful control on the aging rates of SSDs. We also find that both methods of reducing the error rate of single SSDs and increasing the recovery capability of RAID improve the RAID reliability, and the improvement is more pronounced for RAID tolerating fewer failures when using the former method, while the improvement is more pronounced for RAID tolerating more failures for the latter method. Finally, there is a trade-off between storage overhead and RAID reliability if we only change the array configuration in one dimension, i.e., varying either n or m. However, if we vary both of these parameters, we can have an SSD RAID with higher storage capacity and better reliability. This shows our model is indeed relevant in assisting developers to configure SSD RAID.

ACKNOWLEDGMENTS

This work was supported in part by National Nature Science Foundation of China (61772484 and 61379038), and Anhui Provincial Natural Science Foundation (1508085SQF214).

REFERENCES

[1] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. 2008. Design Tradeoffs for SSD Performance. In *Proc. of USENIX ATC.*

[2] Mahesh Balakrishnan, Asim Kadav, Vijayan Prabhakaran, and Dahlia Malkhi. 2010. Differential RAID: Rethinking RAID for SSD Reliability. *ACM Trans. on Storage* 6, 2 (Jul 2010), 4.

[3] John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. 2008. *The DiskSim Simulation Environment Version 4.0 Reference Manual.* Technical Report CMUPDL-08-101. Carnegie Mellon University.

[4] Yu Cai, E.F. Haratsch, O. Mutlu, and Ken Mai. 2012. Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis. In *Proc. of DATE.*

[5] Feng Chen, David A. Koufaty, and Xiaodong Zhang. 2009. Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives. In *Proc. of ACM SIGMETRICS.*

[6] E. de Souza e Silva and H. R. Gail. 2000. Transient Solutions for Markov Chains. *Computational Probability* W. K. Grassmann (editor). Kluwer Academic Publishers (2000), 43–81.

[7] Peter Desnoyers. 2012. Analytic Modeling of SSD Write Performance. In *Proc. of SYSTOR.*

[8] J. G. Elerath and M. Pecht. 2007. Enhanced Reliability Modeling of RAID Storage Systems. In *DSN.*

[9] Garth A. Gibson and David A. Patterson. 1993. Designing Disk Arrays for High Data Reliability. *J. Parallel Distrib. Comput.* 17, 1-2 (Jan. 1993), 4–27.

[10] Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. 2009. Characterizing Flash Memory: Anomalies, Observations, and Applications. In *Proc. of IEEE/ACM MICRO.*

[11] Laura M. Grupp, John D. Davis, and Steven Swanson. 2012. The Bleak Future of NAND Flash Memory. In *FAST.*

[12] Peter G. Harrison, Naresh M. Patel, and Soraya Zertal. 2010. Response Time Distribution of Flash Memory Accesses. *Performance Evaluation* 67, 4 (April 2010), 248 – 259.

[13] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. 2009. Write Amplification Analysis in Flash-based Solid State Drives. In *Proc. of SYSTOR.*

[14] Ilias Iliadis and Vinodh Venkatesan. 2015. Beyond MTTDL: A Closed-Form RAID-6 Reliability Equation. *Trans. Storage* 11, 2, Article 9 (March 2015), 10 pages.

[15] Soojun Im and Dongkun Shin. 2011. Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD. *IEEE Trans. on Computers* 60 (Jan 2011), 80–92.

[16] Nikolaus Jeremic, Gero Mühl, Anselm Busse, and Jan Richling. 2011. The Pitfalls of Deploying Solid-state Drive RAIDs. In *Proc. of SYSTOR.*

[17] Myoungsoo Jung and Mahmut Kandemir. 2013. Revisiting Widely Held SSD Expectations and Rethinking System-level Implications. In *SIGMETRICS.*

[18] Jaeho Kim, Jongmin Lee, Jongmoo Choi, Donghee Lee, and S.H. Noh. 2013. Improving SSD Reliability with RAID via Elastic Striping and Anywhere Parity. In *Proc. of IEEE/IFIP DSN.*

[19] Sehwan Lee, Bitna Lee, Kern Koh, and Hyokyung Bahn. 2011. A Lifespan-aware Reliability Scheme for RAID-based Flash Storage. In *Proc. of SAC.*

[20] Yangsup Lee, Sanghyuk Jung, and Yong Ho Song. 2009. FRA: A Flash-aware Redundancy Array of Flash Storage Devices. In *Proc. of ACM CODES+ISSS.*

[21] Yongkun Li, Helen H. W. Chan, Patrick P. C. Lee, and Yinlong Xu. 2016. Elastic Parity Logging for SSD RAID Arrays. In *Proc. of IEEE/IFIP DSN.*

[22] Yongkun Li, Patrick P. C. Lee, and John C. S. Lui. 2013. Stochastic Modeling of Large-Scale Solid-State Storage Systems: Analysis, Design Tradeoffs and Optimization. In *Proc. of SIGMETRICS.*

[23] Yongkun Li, Patrick P. C. Lee, and John C. S. Lui. 2016. Analysis of Reliability Dynamics of SSD RAID. *IEEE Trans. on Computers* 65, 4 (Apr 2016), 1131 – 1144.

[24] F. Machida, R. Xia, and K. Trivedi. 2015. Performability Modeling for RAID Storage Systems by Markov Regenerative Process. *IEEE Transactions on Dependable and Secure Computing* PP, 99 (2015), 1–1.

[25] Bo Mao, Hong Jiang, Suzhen Wu, Lei Tian, Dan Feng, Jianxi Chen, and Lingfang Zeng. 2012. HPDA: A Hybrid Parity-based Disk Array for Enhanced Performance and Reliability. *ACM Trans. on Storage* 8, 1 (Feb 2012), 4.

[26] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. 2015. A Large-Scale Study of Flash Memory Failures in the Field. In *Proceedings of ACM SIGMETRICS.*

[27] N. Mielke, T. Marquart, Ning Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L.R. Nevill. 2008. Bit Error Rate in NAND Flash Memories. In *IEEE Int. Reliability Physics Symp.*

[28] Sangwhan Moon and A. L. Narasimha Reddy. 2013. Don't Let RAID Raid the Lifetime of Your SSD Array. In *HotStorage.*

[29] Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. 2009. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In *Proc. of ACM EuroSys.*

[30] Jian Ouyang, Shiding Lin, Song Jiang, Zhenyu Hou, Yong Wang, and Yuanzheng Wang. 2014. SDF: Software-Defined Flash for Web-Scale Internet Storage Systems. In *Proc. of ACM ASPLOS.*

[31] Kwanghee Park, Dong-Hwan Lee, Youngjoo Woo, Geunhyung Lee, Ju-Hong Lee, and Deok-Hwan Kim. 2009. Reliability and Performance Enhancement Technique for SSD Array Storage System Using RAID Mechanism. In *IEEE ISCIT.*

[32] David A. Patterson, Garth Gibson, and Randy H. Katz. 1988. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. of ACM SIGMOD.*

[33] J.S. Plank, J. Luo, C.D. Schuman, L. Xu, and Z. Wilcox-O'Hearn. 2009. A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries for Storage. In *Proc. of USENIX FAST.*

[34] E. W. D. Rozier, W. Belluomini, V. Deenadhayalan, J. Hafner, K. Rao, and P. Zhou. 2009. Evaluating the Impact of Undetected Disk Errors in RAID systems. In *2009 DSN.* 83–92.

[35] Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros G. Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. 2013. XORing Elephants: Novel Erasure Codes for Big Data. *Proc. VLDB Endow.* 6, 5 (March 2013), 12.

[36] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. 2016. Flash Reliability in Production: The Expected and the Unexpected. In *Proc. of USENIX FAST.*

[37] Jonathan Thatcher, Tom Coughlin, Jim Handy, and Neal Ekker. 2009. NAND Flash Solid State Storage for the Enterprise: An In-depth Look at Reliability. In *SNIA report.*

[38] Benny Van Houdt. 2013. A Mean Field Model for a Class of Garbage Collection Algorithms in Flash-based Solid State Drives. In *Proc. of ACM SIGMETRICS.*

[39] Benny Van Houdt. 2013. Performance of Garbage Collection Algorithms for Flash-based Solid State Drives with Hot/cold Data. *Performance Evaluation* 70, 10 (Sep 2013), 692 – 703.

[40] Yu Wang, Wei Wang, Tao Xie, Wen Pan, Yanyan Gao, and Yiming Ouyang. 2014. CR5M: A Mirroring-powered Channel-RAID5 Architecture for An SSD. In *Proc. of IEEE MSST.*

[41] W. Weibull. 1951. A Statistical Distribution Function of Wide Applicability. *Journal of Applied Mechanics* 18 (1951), 293–297.