Improving SSD Read Latency via Coding

Hyegyeong Park, *Member, IEEE* and Jaekyun Moon, *Fellow, IEEE*

Abstract—We study the potential enhancement of the read access speed in high-performance solid-state drives (SSDs) by coding, given speed variations across the multiple flash interfaces and assuming occasional local memory failures. Our analysis is based on a queuing model that incorporates both read request failures and NAND element failures. The NAND element failure in the present context reflects various limitations on the memory element level such as bad blocks, dies or chips that cannot be corrected by error control coding (ECC) typically employed to protect pages read off the NAND cells. Our analysis provides a clear picture of the storage-overhead and read-latency trade-offs given read failures and NAND element failures. We investigate two different ways to mitigate the effect of NAND element failures using the notion of multi-class jobs with different priorities. A strong motivation for this work is to understand the reliability requirement of NAND chip components given an additional layer of failure protection, under the latency/storage-overhead constraints.

Index Terms—NAND flash memory, solid-state drives, NAND component failures, distributed storage, latency analysis, queuing theory, error-control coding

1 Introduction

As the demand continues for higher storage density in solid-state drives (SSDs), the NAND process size inevitably shrinks, resulting in considerable difficulties in maintaining yield in the NAND manufacturing process. Strong error control coding (ECC) [2], [3], [4], [5], [6] and well-tailored signal processing [7], [8], [9] have been increasingly used to help relieve this burden on NAND manufacturing. For example, advanced low-density parity-check (LDPC) coding [10] is now widely deployed to protect pages read off the NAND cells, allowing raw bit error rates (RBERs) in accessing the NAND cells to drop to extremely low levels [2], [3], [4]. An unfortunate price paid is the considerable increase in the read latency as generating soft read values of the cells needed for the LDPC decoder requires multiple sensing/read operations that are highly time-consuming.

In this work, we analyze the problem from a different perspective. We investigate the power of ECC while bringing the *read access time* into the picture. The read access time is the read response time, which is defined as the time taken since the read request until the controller receives the data. Specifically, we explore trade-offs between coding-overhead and read-access-time first time use of RBER and memory element failure probability. To do this, we exploit the idea and analytical tools from the studies on coding for *distributed storage*. A distributed storage system consists of multiple storage servers which contain information in a distributed manner. Simple replication or erasure coding is often applied across the distributed servers to improve data reliability throughout the system.

While prior work exists on the use of erasure coding to improve download time of distributed storage [11], [12], [13], [14], [15], [16], here we are concerned with SSD-specific read access time and the failure event models that have not been previously used in the SSD system performance analysis. In particular, we introduce a new model for the read access time distribution reflecting a step increase in read latency that occurs every time read access fails. We also extend the (n, k) fork-join model introduced in [14] to include the following two types of failures: (a) the read failure in accessing individual NAND cells in spite of using ECCs, and (b) the NAND element failure event. The NAND element here can be the chip/die/block/page, which is similar to the silicon element in the redundant array of independent silicon elements (RAISE) [17]. To handle such NAND element failure events, multiple priority classes are included in our analysis. While the multiple classes with different priorities are also considered in [15], [16], there the priority is introduced to model the heterogeneity of the cloud storage data and there is no consideration of the NAND-specific system modeling. Thus, the approaches cannot be directly applied to the problem at hand.

In [2], progressive memory-sensing/LDPC-code-decoding is utilized to reduce system latency in the face of read failure events. In contrast to the work of [2], we also consider the NAND element failures. We handle the NAND element failures under the two different proposed policies, which are called *instantaneous repair* (IR) and *postponed repair* (PR) policies. The IR policy gives the NAND element failure the higher priority for service (i.e., reconstruction) under the preemptive-resume priority. On the contrary, the PR policy assigns the lower priority to the NAND element failure under the non-preemptive priority.

The main contributions of this paper are as follows:

1) We demonstrate how coding improves the read access time in SSDs.
2) We present upper bounds for the mean read access time, while also considering the SSD-specific read

• *The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, South Korea. E-mail: parkh@kaist.ac.kr, jmoon@kaist.edu.*

Manuscript received 9 Apr. 2019; revised 21 Nov. 2019; accepted 22 Feb. 2020. Date of publication 6 Mar. 2020; date of current version 6 Nov. 2020. (Corresponding author: Hyegyeong Park.) Recommended for acceptance by H. Jiang. Digital Object Identifier no. 10.1109/TC.2020.2978823

access time distribution and failure events of the read request and NAND element.

3) We propose two types of policies against NAND element failures so that the reduction of read access time due to the proposed coding is maintained even in the presence of the NAND element failures.
4) Through the analysis and numerical simulation result, we provide necessary insights into new trade-offs related to the tolerable level of physical memory failure rates.

Our contributions listed lay down a path toward more efficient utilization of storage overheads under the consideration of the target yields in the NAND manufacturing process.

Compared to the conference version [1] of this work, the present paper adds significantly in the following ways: first of all, this paper suggests an improved way of mitigating the effect of NAND element failure based on a new PR policy. If the NAND element failure arises frequently the IR policy introduced in [1] can be inefficient, since all the read operations are suspended to repair the failed element as soon as a failure event occurs. Adopting the PR policy, the repair process can wait until all read requests in the queues have been processed thanks to additional coding. The present paper provides a new latency analysis including the effect of the proposed PR policy against NAND element failure. Through both analysis and numerical simulations, it is shown that the PR policy can have a significantly better latency performance for a certain range of RBERs compared to the IR policy. Also, NAND element failure modeling and corresponding repair job modeling are new in the present paper. The underlying assumptions associated with modeling are clarified with more detailed descriptions and clearly justified based on the results of previous literature.

The rest of this paper is organized as follows. Section 2 contains a review of the relevant literature. Section 3 presents the description of the target problem and system model. In Section 4, we discuss a layered coding structure for SSDs based on the internal parallelism. In Section 5, analysis of read access time without the consideration of NAND element failures is given. In Sections 6 and 7, we provide the analysis of read access time in the existence of NAND element failures. Two different types of approaches to protect the system against NAND element failures are introduced as well. In Section 8, numerical simulation results show that the proposed layered coding gives a reduction in latency relative to the system without layered coding. A comparison between the latency performance under the two proposed policies against NAND element failures is also provided. Finally, the paper draws conclusions in Section 9.

2 Related Work

2.1 Distributed Storage Coding

Coding for distributed storage is an active area of research. For providing fault tolerance to distributed storage, 3-replication has been widely used in the Google File System (GFS) [18] and Hadoop Distributed File System (HDFS) [19]. This method simply stores three exact copies of the original data across the distinct servers so that a failed copy of the data can be recovered by accessing only one other intact copy of the data. That is, replication is optimal in repair bandwidth.[1] However, replication has a high cost in terms of storage overhead. Erasure codes such as the well-known Reed-Solomon (RS) codes [20] have been deployed as well, which have the minimum storage overhead while requiring an expensive cost in repair bandwidth.

After the work of Dimakis *et al.* [21], which introduced the regenerating codes that could achieve the optimal trade-off between the repair bandwidth and storage, there has been the following line of work on efficient codes in terms of various resource parameters: access I/O [22], [23], locality [24], [25], [26], download time [11], [12], [13], [14], [15], [16] and energy consumption [15]. Moreover, trade-offs among key resource parameters have been explored: among repair bandwidth, storage and reliability in [27] and among I/O, storage and network bandwidth in [28].

2.2 High-Level Coding

The chip level memory failure is a growing concern and a variety of techniques related to the redundant-array-of-independent disks (RAID) already exist to address this issue [29], [30]. For example, *Chipkill* from IBM [31], *Advanced ECC* from HP [32] and *redundant array of independent NAND* (RAIN) from Micron [33] all provide fault tolerance on the chip level. RAISE from SandForce [17] further considers data protection on the die/block/page level. The superblock/superpage from Micron [34] is another approach to improving speed by grouping multiple blocks/pages across chips and planes [29], [34], [35], [36]. This improves the throughput since the multiple pages with a large volume involved in the same superpage/superblock can be read/written/erased simultaneously.

Unlike in the RAID-like systems or systems with the superblock/superpage-level parity, we also aim at reducing the read access time using codes. In the conference version [1] of this paper, we have introduced a *layered coding* structure boosted by outer MDS coding across NAND elements,[2] in addition to the inner soft-decision ECC such as the LDPC code. Typical SSD architecture is based on an array of NAND flash memory packages. Such packages and the flash controller are interconnected with multiple channels. Data accesses can be parallelized and conducted independently over channels. The importance of exploiting internal parallelism in high-performance SSDs is thoroughly investigated in [37], [38]. The highly parallelized structure of SSDs opens up the possibility of introducing queuing theoretic analysis. In the case under consideration in this work, reducing read access time is possible via the high-level erasure coding across parallel channels, since during the read the original data can be reconstructed by accessing any k fastest available channels out of n parallel ones.

1. The repair bandwidth is defined as the amount of data to be downloaded to repair one data unit failure.

2. We refer to a symbol in the erasure code as a node. Since we deploy the outer MDS code across NAND elements, any NAND elements such as page/block/die/chip can be regarded as nodes. Note also that in layered coding, the outer ECC and high-level (erasure) coding are terms that can be used interchangeably.

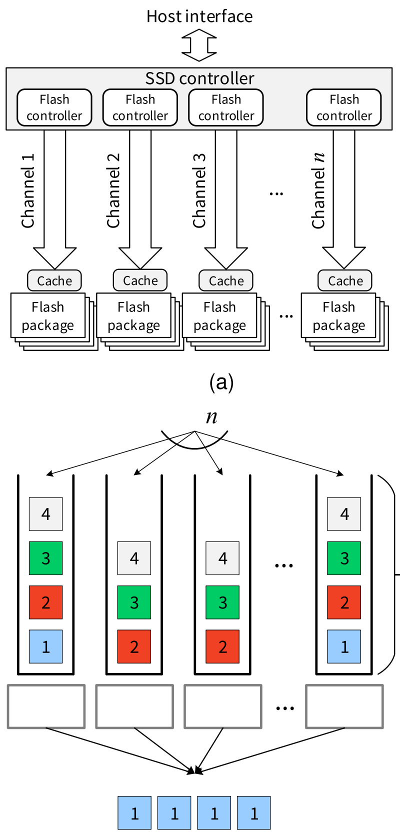


Fig. 1. NAND flash interface as a queuing model. (a) NAND interface with n channels combating latency variations. (b) Fork-join queuing model with n queues.

3 Target Problem and System Description

3.1 Target Problem

A failure of NAND elements such as the pages, blocks, dies or chips is catastrophic in that it cannot be recovered by the ECC applied over a page. In this paper, we consider the problem of mitigating failures and improving data access time by introducing high-level erasure coding. More specifically, we are interested in doing a latency analysis of the SSD system equipped with high-level erasure coding, and exploring the trade-off relationships between the coding-overhead and read access time. We further aim to investigate how the trade-off changes in the presence of failure events. We thus define the types of failures that we are interested in as follows.

Definition 1.

- *Read request failures: LDPC code decoding failures that invoke additional voltage sensing*
- *NAND element failures: occasional failures of NAND elements that require data reconstruction*

In the progressive memory sensing and decoding scheme, once the LDPC code decoding run fails, the controller increases memory sensing quantization level by one. The sensing level increment continues until the decoding succeeds or the number of retries reaches the highest sensing precision. The decoding failure events, which invoke re-sensing of the memory cells, are considered as read failure events in the present work.

3.2 NAND Interface Modeling With Queues

In our modeling of the SSD system, there are multiple channels[3] each of which is connected to the NAND flash package which consists of multiple NAND flash chips. Each channel is assumed to have its queue due to the presence of processing speed variations across different memory chip interfaces (see Fig. 1a). The speed variations arise from the various types of random noise that degrades the accuracy of the LDPC decoder input, which in turn affects the LDPC code decoding failure probability and the number of multiple sensing/read operations to generate soft read values of the cells for the LDPC decoder. Such speed variations tend to be large especially in high-performance SSDs due to the large number of NAND chips deployed. The pages read off each channel are typically protected by ECC. Our assumption here, however, is that there are occasional hard errors that cannot be corrected at this level. This might be due to bad blocks, dies or chips. In this sense, NAND elements deployed across distinct NAND flash packages that are connected with different flash channels can be interpreted as data nodes in distributed storage with their own queue. This view is consistent with some of the existing high-performance enterprise SSD architectures [39], where separate NAND controllers or caches/buffers are employed that help smooth out the access speed variations across the parallel NAND channel interfaces [40]. Fig. 1b shows the fork-join queuing model [41] where incoming job requests are split into n parallelized queues. The fork-join queue has been widely adopted for describing the parallel and distributed processing systems. In this paper, the NAND flash interface in Fig. 1a is modeled as a queuing model as described in Fig. 1b.

For the reader's convenience, key parameters extensively used in this paper are briefly defined in Table 1.

4 Layered Coding Structure

In order to provide additional data protection to correct full page/block/die failures, we introduce the high-level erasure code as the outer ECC across NAND elements. It is a similar concept to *striping* in the RAID viewpoint. Such a combination of an inner ECC and an outer ECC across the NAND elements forms a *layered coding* structure in Fig. 2. Soft-decision ECCs commonly used in practical SSDs have significantly stronger error correction capability than hard-decision ECCs such as the Bose-Chaudhuri-Hocquenghem (BCH) code [2]. However, an increased latency caused by multiple retries of read-voltage sensing is inevitable to obtain stronger error correction performance. This structure invokes the outer ECC each time the inner ECC (e.g., LDPC codes) decoding step fails (see Fig. 3). Latency caused by consecutive read retries of the LDPC code can be reduced by the help of the outer high-level erasure code. We wish to improve the read access time and NAND element failure tolerance by leveraging the outer code.

Incorporating the outer ECC in the memory system, we propose a framework wherein the NAND memory interfaces

3. Channels refer to the paths that connect the SSD controller to the NAND flash memory to carry traffic. Multiple channels enable parallel read/write operations of data.

TABLE 1
Glossary of Important Notation

Notation	Description
λ	rate of the read request arrivals
N	number of possible outcomes per a sensing level
$N_{\max}$	maximum number of the possible outcomes in the read access time distribution
N_s	maximum number of sensing level
$P_{\text{fail},i}$	probability of read request failure at the ith sensing level
$P_{\text{fail}}^{(1)}$	probability that hard decision decoding fails
$\tau_{\text{sen-ref}}$	latency of sensing reference hard-decision voltages
τ_{sen}	latency of sensing a set of additional voltage levels to yield one soft-decision quantization level
τ_{xfer}	latency of transferring the additional data read by increasing a sensing level from the flash to the controller
τ_{dec}	decoding delay of an LDPC code
τ_{prog}	programming latency
N_p	number of pages to be reconstructed

in an SSD act as *distributed data storage*. Layered coding and distributed storage coding clearly have parallels in using ECCs across the distinct nodes to provide node-level data reliability.

We consider the following storage system which is modeled using the (n, k) fork-join system introduced in [14].

Definition 2. *An* (n, k) *MDS-outer-coded* layered coding system *(or simply* (n, k) *layered coding) consists of* n *NAND elements which are deployed across the distinct NAND flash packages.*

- *Data is split into* k *chunks and then stored over* n *nodes after applying an* (n, k) *MDS code.*
- *By the property of* (n, k) *MDS code, accessing any* k *out of* n *nodes enables reconstruction of the desired data.*
- *Each incoming request of a read job is forked into* n *first-come-first-served (FCFS) queues and any* k *finished requests out of* n *complete the corresponding job, after which the remaining* $n-k$ *unfinished requests can be discarded from the queues.*

Here we assume a negligible request cancellation overhead, which means that the queue can serve the next waiting request immediately after a request is canceled. Moreover, the encoding/decoding of the outer code requires no significant additional cost, as it takes simple exclusive OR operations as in RAID SSDs [39]. It is also noteworthy that the (n, k) layered coding structure assumes that a data chunk larger than n pages is accessed at once, which is valid in usual NAND flash applications. Then, most of the data can benefit from layered coding by forming n-page groups, and there exists only a small or even a negligible portion of data that cannot.

The nodes in layered coding are deployed in the different NAND flash packages. This allows us to assume that the failures occur independently. Although high-level coding can have a page/block/chip as a node, we confine our interest for the time being to the high-level coding with *pages* as its nodes for ease of presentation. Even so, the models and analyses can be easily extended to the case where high-level coding regards other NAND elements as nodes.

For illustrative purposes, consider the fork-join model in Fig. 4 corresponding to $(n, k) = (10, 4)$. This $(10, 4)$ layered coding has a page as its node. We denote each page accessed by job X by "Page X-Y". Y denotes the index of a flash package where Page X-Y stored. This means that pages with the same X index compose a $(10, 4)$ high-level code. Read job 1 has 10 requests to access Page 1-1 to Page 1-10. When a request approaches the head of a queue, the corresponding page is accessed. In Fig. 4, since four (Pages 1-2, 1-3, 1-5, 1-8) out of ten pages for job 1 have already been

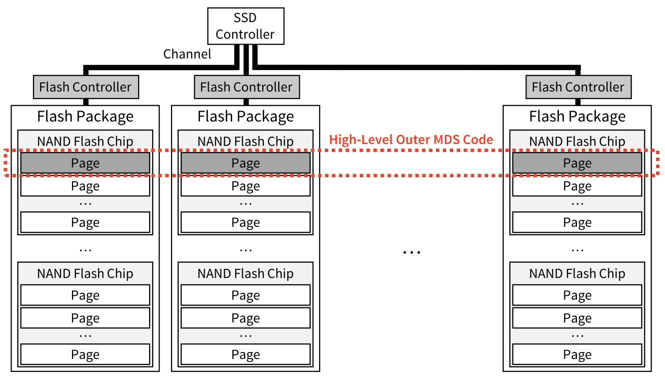
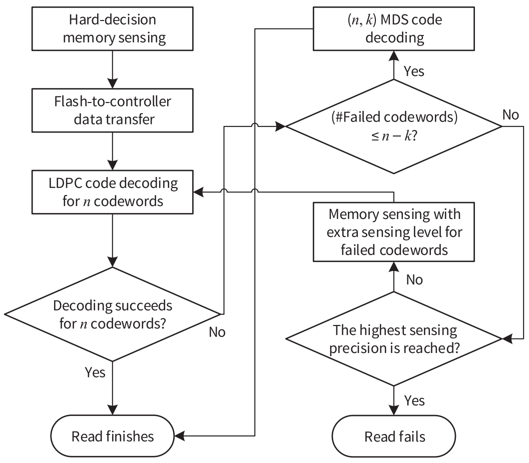


Fig. 2. Example of the layered coding structure. In this example, the pages deployed across the distinct flash packages are the nodes of the outer MDS code. Each page is protected by an inner LDPC code. The shaded pages are the nodes of high-level coding represented by a box in dashed red line.


Hard-decision memory sensing
Flash-to-controller data transfer
LDPC code decoding for n codewords
Decoding succeeds for n codewords?
Yes
No
Read finishes
(n, k) MDS code decoding
Yes
(#Failed codewords) ≤ n − k?
No
Memory sensing with extra sensing level for failed codewords
No
The highest sensing precision is reached?
Yes
Read fails
Fig. 3. Flow of the (n, k) MDS-coded layered coding structure.

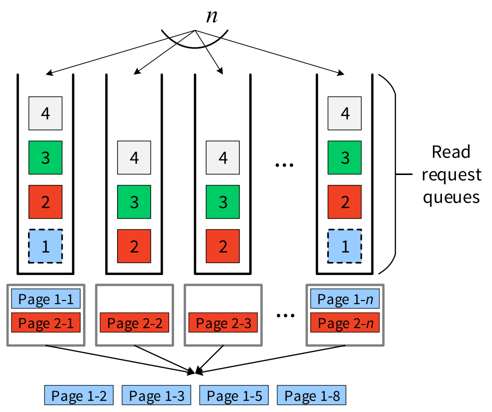


Fig. 4. Fork-join queuing model with n queues. For an $(n,k)=(10,4)$ MDS code under consideration, accessing any four out of ten pages complete the corresponding job.

served, the remaining six tasks for job 1 abandon (presented by dashed squares in the queues) their queues and job 1 exits the system.

5 Analysis of Read Access Time

For latency analysis, we assume that the read requests occur randomly according to the Poisson process with rate λ as in [14], [15]. The read access time at each node is governed by the behavior of memory sensing and decoding as we shall see in Section 5.1. Note that it is possible to assume that the read access time at each node is independent since nodes are deployed across the different flash packages. The reason is that the LDPC decoder failure rate largely depends on read noise (system noise) and write noise (due to media as well as written input). It is obvious that the read system noise (e.g., electronic circuit noise) across different channels are independent.[4]

5.1 Read Access Latency in SSDs

Soft-decision ECCs (e.g., LDPC codes) perform significantly better than hard-decision ECCs such as the BCH codes [2]. Error correction capability of LDPC codes highly depends on the quality of input information. Such input information is computed by using the digitally quantized threshold voltage of the memory cells in a page from successive multiple read decisions. Fig. 5 illustrates progressive sensing and decoding [2] for the multi-level-cell (MLC) NAND flash channel. Once decoding fails following hard-decision sensing, the controller invokes soft-decision LDPC code decoding. To do this, the sensing level between the adjacent storage states increases by one. The procedure is repeated until LDPC code decoding succeeds or the highest sensing precision is reached. The use of soft-decision thus causes severe latency overhead due to the multiple read operations with different reference voltages.

Let N_s denote the maximum sensing level. Based on the progressive sensing and decoding technique, the memory

4. Even though the physical media age together and thus the average decoding errors may increase together between two different NAND chips, the media noise processes as well as the written local data are independent across different chips.

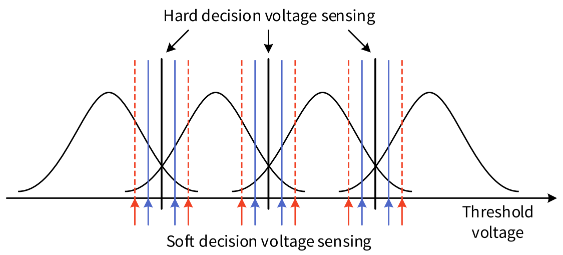


Fig. 5. Hard- and soft-decision voltage sensing for the multi-level-cell (MLC) NAND flash.

sensing and decoding latency to reach the ith sensing level is defined as follows.

Definition 3. *We define the* memory sensing and decoding latency *where decoding succeeds at the ith sensing level, for a given N_s and an integer i such that $1 \le i \le N_s$ by*

$$\tau_i = \tau_{\text{sen-ref}} + \tau_{\text{xfer}} + \tau_{\text{dec}} + (i-1)(\tau_{\text{sen}} + \tau_{\text{xfer}} + \tau_{\text{dec}}), \tag{1}$$

where $\tau_{\text{sen-ref}}$ denotes the latency of sensing reference hard-decision voltages, τ_{sen} is the latency of sensing a set of additional voltage levels to yield one soft-decision quantization level (denoted by same colored lines in Fig. 5), τ_{xfer} represents the latency of transferring the data read by increasing a sensing level from the flash to the controller, and τ_{dec} indicates the decoding delay of an LDPC code.

Note that τ_i is undefined for $i > N_s$ and we declare that the read from the (n,k) layered coding system fails as described in Fig. 3.

Before we discuss the distribution of the read access time, we present our assumptions as follows. Let $N_{\max}$ denote the maximum number of the possible outcomes in the read access time distribution. Then, assume that (a) the possible outcomes of the read access time are scattered about τ_i and are represented by $N\ (=N_{\max}/N_s)$[5] equally spaced values with a maximum dispersion of $\alpha\tau_i$ (on one side) to reflect the data's spread from the mean in reality, where α is a dispersion coefficient,[6] and that (b) each ith set of the N possible outcomes representing the ith sensing level satisfies the discrete uniform distribution on $[\tau_i(1-\alpha), \tau_i(1+\alpha)]$, which, as N grows, tends to a continuous distribution. The dispersion comes from the assumption that nodes of (n,k) layered coding are deployed across the multiple distinct NAND flash packages. We model the local distribution associated with each sensing quantization level to be a bounded uniform distribution from the following reasons: For any given quantization level, there is clearly a minimum possible read time. The read delay is also upper-bounded because a maximum limit is always set in practice for the LDPC decoder iteration for a given quantization level. Assuming uniformity within the upper/lower bounds simply reflects the maximum uncertainty on the actual read latency inside these bounds.

5. We assume that $N_{\max}$ is divisible by N_s for simplicity.
6. In this assumption, the amount of spread increases as τ_i increases, since α is fixed. As the number of sensing increases, the uncertainties associated with individual read retries add up and it is natural to assume a wider local variation with a larger number of read retries.

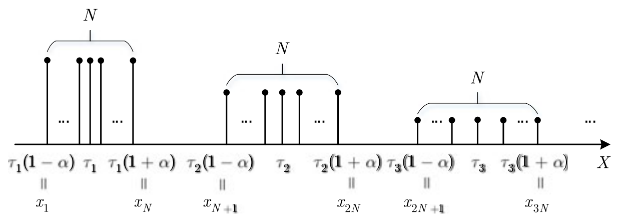

N
N
N
$\tau_1(1-\alpha)$ τ_1 $\tau_1(1+\alpha)$ $\tau_2(1-\alpha)$ τ_2 $\tau_2(1+\alpha)$ $\tau_3(1-\alpha)$ τ_3 $\tau_3(1+\alpha)$ X
x_1 x_N x_{N+1} x_{2N} x_{2N+1} x_{3N}

Fig. 6. Illustration of the read access time distribution at a page.

In this work, we present the analysis of read access time considering two types of failure in Definition 1. The read request failure is included in the analysis by assuming the progressive sensing and decoding and using the definition of latency that grows with an increasing sensing level as in (1). In this section, we first focus on the latency analysis for the system having read failures only, and the analysis considering both the read and NAND element failures is dealt with in Sections 6 and 7.

5.2 An Upper Bound on the Read Access Time

Exact latency analysis of the (n,k) fork-join system is difficult. Only bounds are known even for the (n,k) fork-join system with no failure events [14]. We also focus on the bounds on the mean read access time. We confine our interest to the upper bounds as they can give enough insights into the read latency by providing the maximum average cost in terms of latency.

For the latency analysis, let $P_{\text{fail},i}$ denote the probability of read request failure at the ith sensing level. We further define: $P_{\text{fail}}^{(i-1)} \triangleq \prod_{i'=1}^{i-1} P_{\text{fail},i'}$ for integer i such that $1 \le i \le N_s$ and $P_{\text{fail}}^{(0)} \triangleq 1$. Here, $P_{\text{fail}}^{(1)}$ denotes the probability that hard decision decoding fails, and $P_{\text{fail}}^{(0)} \triangleq 1$ is introduced for ease of exposition and has no physical meaning. We then present the distribution of the read access time in the following proposition.

Proposition 4. *The probability mass function of the read access time distribution X is given by*

$$\begin{aligned} p_j &\triangleq \Pr[X = x_j] \text{ for } 1 \le j \le N_{\max} \\ &= \frac{1}{N} P_{\text{fail}}^{(i-1)}(1 - P_{\text{fail},i}), \end{aligned} \tag{2}$$

where

$$\begin{aligned} x_j &= \tau_i\left(1 - \alpha + \frac{2\alpha}{N-1}(j - N(i-1) - 1)\right), \\ i &= \lceil j/N \rceil . \end{aligned} \tag{3}$$

Proof. The probability that LDPC code decoding succeeds at the ith sensing level is represented by

$$\Pr[T = \tau_i] = P_{\text{fail}}^{(i-1)}(1 - P_{\text{fail},i}) \text{ for } 1 \le i \le N_s. \tag{4}$$

From Definition 3 and the fact that the ith set of N read access time samples lies in the range of $[\tau_i(1-\alpha), \tau_i(1+\alpha)]$ as depicted in Fig. 6, we represent a sample of read access time x_j as follows:

$$x_j = \tau_i\left(1 - \alpha + \frac{2\alpha}{N-1}(j - N(i-1) - 1)\right).$$

The probability mass function of X is then given by

$$p_j \triangleq \Pr[X = x_j] = \frac{1}{N}\Pr[T = \tau_i], \tag{5}$$

where the last equality in (5) follows since each set of the possible outcomes of the read access time includes N points. Plugging (4) into (5), we have the proposition. □

Fig. 6 shows the read access time distribution in Proposition 4. The probability $p_j \triangleq \Pr[X = x_j]$ shall be provided in Proposition 4. Note that the probability $p_j \triangleq \Pr[X = x_j]$ highly depends on the LDPC decoding performance as described in Eqs. (4) and (5). Therefore, the relative magnitude of the probability at each sensing level may vary depending on RBER.

Using the distribution given in Proposition 4, we state an upper bound on the mean read access time as follows.

Theorem 5. *Assuming no NAND element failure occurs, the mean read access time of the (n,k) MDS-outer-coded layered coding system is upper bounded by*

$$S = \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} \binom{n}{l} x_j (R_{j,l} - R_{j-1,l}) \tag{6}$$

$$+ \frac{\lambda \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} \binom{n}{l} x_j^2 (R_{j,l} - R_{j-1,l})}{2\left(1 - \lambda \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} \binom{n}{l} x_j (R_{j,l} - R_{j-1,l})\right)}, \tag{7}$$

provided

$$\lambda \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} \binom{n}{l} x_j (R_{j,l} - R_{j-1,l}) < 1, \tag{8}$$

where $R_{j,l} \triangleq r_j^l (1-r_j)^{n-l}$ and $r_j = 1 - P_{\text{fail}}^{(i-1)}(1 - (\frac{j}{N} - (i-1))(1 - P_{\text{fail},i}))$.

Proof. In order to obtain an upper bound, we resort to the split-merge system as in [42], [43], which is a degraded version of the fork-join system where all n nodes are kept from continuing with the next job until k out of n requests are served. The mean access time of the split-merge system is modeled as an $M/G/1$ system[7] with the service time governed by kth order statistic. The kth order statistic is defined as the kth smallest sample of n random variables [45].

Let $X_1, X_2, \ldots, X_n$ be random samples from a discrete distribution with the probability mass function $f_X(x_j) = p_j$, where $x_1 < x_2 < \ldots$ are the possible samples of X. For a sample of size n, let $X_{1:n}, X_{2:n}, \ldots, X_{n:n}$ denote the order statistics from the sample. Then the kth order statistic from the sample is given by [46]

$$\Pr[X_{k:n} = x_j] = \sum_{l=k}^{n} \binom{n}{l} \left(r_j^l (1-r_j)^{n-l} - r_{j-1}^l (1-r_{j-1})^{n-l} \right),$$

where $r_0 = 0, r_1 = p_1, r_2 = p_1 + p_2, \ldots, r_j = p_1 + p_2 + \ldots + p_j, \ldots$

7. An $M/G/1$ queue is a single-server queuing system where arrivals are Markovian and the service times have a general distribution [44].

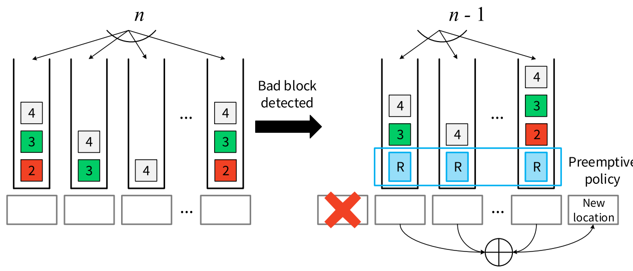


Fig. 7. The reconstruction process from a NAND block failure under the IR policy. When a NAND block fails, regular read jobs denoted by the square boxes are interrupted, and the virtual repair job requests appear (represented by the rectangular boxes marked by "R") and move to the head of queues (under preemptive priority). The $n-1$ intact nodes help the reconstruction of the failed node's contents to a new location. Taking the k fastest copying requests out of $n-1$ is sufficient for the reconstruction.

For the read time distribution of (2),

$$r_j = 1 - P_{\text{fail}}^{(i-1)}\left(1-\left(\frac{j}{N}-(i-1)\right)\left(1-P_{\text{fail},i}\right)\right), \tag{9}$$

where $i = \lceil j/N \rceil$.

The Pollaczek-Khinchin mean-value formula [47] gives the mean access time of an $M/G/1$ system S in terms of its first two moments.

$$S = \mathbb{E}[X_{k:n}] + \frac{\lambda\mathbb{E}[X_{k:n}^2]}{2(1-\lambda\mathbb{E}[X_{k:n}])}\ . \tag{10}$$

The mth moment of the read access time distribution is given by

$$\begin{aligned}\mathbb{E}[X_{k:n}^m] &= \sum_{j=1}^{N_{\max}} x_j^m \Pr[X_{k:n}=x_j] \\ &= \sum_{j=1}^{N_{\max}}\sum_{l=k}^{n}\binom{n}{l} x_j^m (R_{j,l}-R_{j-1,l}),\end{aligned} \tag{11}$$

where $R_{j,l} \triangleq r_j^l(1-r_j)^{n-l}$ and r_j is defined in (9).

The condition in (8) is from the stability requirement for the $M/G/1$ queue, i.e., $\lambda\mathbb{E}[X_{k:n}] < 1$. □

Note that (7) is composed of the two separate terms that are the mean actual read access time and mean waiting time in queues. The stability condition requirement in (8) guarantees the queue length not to increase without bound.

6 Analysis of Read Access Time With Instantaneous Repair From NAND Element Failures

6.1 Repair Under the IR Policy

In this section, we provide an analysis of the read access time including the presence of NAND element failures in addition to considering read request failures only as in Section 5. Queuing with breakdown or failure of queues can be interpreted as queuing with *preemptive resume priority* [48]. Under the preemptive resume priority policy, the service is interrupted when a higher priority request arrives (see Fig. 7). Assume that we have two classes of priorities ($c \in \{1,2\}$), where jobs of each class include $n_{(c)}$ nodes. We denominate the normal job to access the node as a *regular* read job. When a NAND element failure event occurs, we imagine that a *virtual* repair job enters the system at that moment. We give a higher priority to the repair job (class $c=1$) and a lower priority to the read job (class $c=2$). The read requests at the heads of the queues are interrupted immediately when a repair job (i.e., NAND element failure event) arrives. Read requests resume from the point of interruption once the repair requests have been served. In this work, we refer to such a repair policy as IR.

Under the IR policy, once a block failure is detected, the neighbors of the failed block embark on a reconstruction phase to maintain the desired redundancy level. Here, we are speaking of the block failure since it is the most typical form of NAND element failure in practical SSDs. Throughout the remainder of our analysis, we shall focus on block failures. We shall also consider the bad blocks detected by the failure of the program operation, as typically happens in real SSDs. Note that bad block detection is done independently of the current read job in the queue. When the program operation fails, the contents of the pages already written in the corresponding bad block need to be duplicated to another block. The repair process takes the k fastest copying requests out of the $n-1$ nodes. For copying the contents of the failed block in the repair process, we reserve a small memory buffer space which is in line with the typical memory buffer size in real SSDs [49], [50]. Each block typically contains hundreds to thousands of pages in SSDs and the failed block is reconstructed in a page-by-page fashion.[8] Both the repair job and the read job can be described as a collection of the k fastest requests and thus be represented by the kth order statistic $X_{(c),k:n_{(c)}}$. Note that $n_{(1)} = n-1$ and $n_{(2)} = n$ for our purposes. To reduce notational burden, we shall simply write $X_{k:n_{(c)}}$ instead of $X_{(c),k:n_{(c)}}$.

We focus on the repair from a single block failure since it is the dominant failure pattern. Multiple block failures can be handled by consecutive reconstructions from single block failures. An (n,k) MDS coded storage system, in fact, tolerates $n-k$ failures without repair. Thus, it would be possible to build a more elaborated model that continues working without entering the repair phase until t failures occur, where t $(\leq n-k)$ is a threshold of the failure tolerance. However, we leave this for future work.

6.2 Analysis of Read Access Time

Under the additional consideration of NAND element failures, we also obtain an upper bound for the mean read access time by taking the split-merge system, the degraded version of (n,k) fork-join system. We make certain assumptions on the node failures and read jobs. The failures and read jobs occur randomly (according to the Poisson process with rates $\lambda_{(1)}$ and $\lambda_{(2)}$, respectively). The read time in this section is identically defined to (1) even with the read failures considered (i.e., $\tau_{(2),i} = \tau_i$ and $x_{(2),j} = x_j$). The repair time (or, equivalently, service time of the repair jobs) is defined similarly to (1), but the repair job essentially

8. Since a block failure essentially means the failure of all pages in it, the two terms - node failure and NAND element failure - are used interchangeably in the remainder of this paper.

consists of reconstructing the data of the failed node using information from adjacent nodes and programming it to the clean location. Thus, the memory sensing, decoding and programming latency to reach the ith sensing level is given by the following definition.

Definition 6. *We define the* memory sensing and decoding latency of the repair job *where decoding of the read operation for repair succeeds at the ith sensing level, for a given N_s and an integer i such that $1 \le i \le N_s$ as follows:*

$$\begin{aligned}\tau_{\mathrm{rep},i} &= \tau_i + \tau_{\mathrm{prog}} \\ &= \tau_{\mathrm{sen-ref}} + \tau_{\mathrm{xfer}} + \tau_{\mathrm{dec}} \\ &\quad + (i-1)(\tau_{\mathrm{sen}} + \tau_{\mathrm{xfer}} + \tau_{\mathrm{dec}}) + \tau_{\mathrm{prog}},\end{aligned} \tag{12}$$

where τ_{prog} denotes the programming latency due to the reconstruction of a failed NAND element. We assume the block-level failure here since it represents the typical NAND element failure.

From this definition, note that the repair job also requires to read data to reconstruct the failed node. Using Definition 6, we establish the following definition for the memory sensing and decoding latency of the repair job and read job at each sensing level under the IR policy. We discuss the number of nodes involved in each class of job as well. The parameters are based on the jobs of two different priority classes.

Definition 7. *Under the IR policy, for an (n,k) MDS-outer-coded layered coding system, the number of nodes involved in the repair and read job (denoted by $n_{(1)}$ and $n_{(2)}$, respectively), and the memory sensing and decoding latency of the repair and read job where decoding of the read operation succeeds at the ith sensing level (denoted by $\tau_{(1),i}$ and $\tau_{(2),i}$, respectively) for integer i such that $1 \le i \le N_s$ are defined as follows:*

$$\begin{aligned} n_{(1)} &= n-1, \quad \tau_{(1),i} = N_p \tau_{\mathrm{rep},i}, \quad \text{(repair job requests)} \\ n_{(2)} &= n, \qquad \tau_{(2),i} = \tau_i\,. \qquad \text{(read job requests)},\end{aligned}$$

where N_p denotes the number of pages to be reconstructed.[9]

In the same sense as Section 5, but considering the NAND element failure events, we present the read time distribution reflecting the data's spread from the mean in the following proposition.

Proposition 8. *For a priority class $c \in \{1,2\}$, the probability mass function of the read time distribution $X_{(c)}$ in the existence of NAND element failure events is given by*

$$\begin{aligned} p_{(c),j} &\triangleq \Pr[X_{(c)} = x_{(c),j}] \text{ for } 1 \le j \le N_{\max} \\ &= \frac{1}{N} P_{\mathrm{fail}}^{(i-1)}(1 - P_{\mathrm{fail},i}),\end{aligned} \tag{13}$$

where

$$x_{(c),j} = \tau_{(c),i}\left[1 - \alpha + \frac{2\alpha}{N-1}[j - N(i-1) - 1]\right]. \tag{14}$$

9. We assume that the failure of program operation, if occurs, takes place when programming to the blocks in which about half of the pages are already written. $2Np$ is thus the number of pages in a block.

Proof. Since this proposition expands Proposition 4 for two priority classes, we have this proposition by simply replacing X, x_j and τ_i with $X_{(c)}$, $x_{(c),j}$ and $\tau_{(c),i}$, respectively. $P_{\mathrm{fail},i}$ which means the probability of read request failure at the ith sensing level is commonly used for both priority classes to reflect the effect of the read failures; the repair job also includes sensing and decoding of the data to reconstruct the failed node. □

We also focus on obtaining an upper bound on the mean read access time as we did in Section 5.

Theorem 9. *Under the IR policy, the mean read access time of the repair and read job in the (n,k) MDS-outer-coded layered coding system is upper bounded as follows:*

- $c = 1$ *(for repair job requests)*

$$\begin{aligned} S_{\mathrm{IR}}^{(1)} &= \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)} \\ &\quad + \frac{\lambda_{(1)}}{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j}^2 D_{j,l}^{(1)} \\ &\quad \cdot \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)}\right)^{-1}.\end{aligned} \tag{15}$$

- $c = 2$ *(for read job requests)*

$$\begin{aligned} S_{\mathrm{IR}}^{(2)} &= \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(2)}} \binom{n_{(2)}}{l} x_{(2),j} D_{j,l}^{(2)} \\ &\quad \cdot \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)}\right)^{-1} \\ &\quad + \frac{1}{2} \sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j}^2 D_{j,l}^{(v)} \\ &\quad \cdot \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)}\right)^{-1} \\ &\quad \cdot \left(1 - \sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j} D_{j,l}^{(v)}\right)^{-1}.\end{aligned} \tag{16}$$

where $D_{j,l}^{(c)} \triangleq R_{j,l}^{(c)} - R_{j-1,l}^{(c)}$, $R_{j,l}^{(c)} \triangleq r_{(c),j}^{l}(1 - r_{(c),j})^{n_{(c)}-l}$ and $r_{(c),j} = 1 - P_{\mathrm{fail}}^{(i-1)}\big(1 - (\frac{j}{N} - (i-1))(1 - P_{\mathrm{fail},i})\big)$.

The stability condition is

$$\sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j} D_{j,l}^{(v)} < 1\,. \tag{17}$$

Proof. Let $X_{k:n_{(c)}}$ ($c \in \{1,2\}$) be the read access time distribution for the priority class c, which is based on the kth order statistic from the $n_{(c)}$ samples. Under the IR policy, the repair job and read job are given the priority classes $c = 1$ and $c = 2$, respectively. As can be seen in Definition 7, $\tau_{(1),i} = \tau_{\mathrm{rep},i}$ and $\tau_{(2),i} = \tau_i$. Substituting $\tau_{(c),i}$ into (14) provides $x_{(c),j}$ for the both classes. Then, we have

$$\Pr[X_{k:n_{(c)}} = x_{(c),j}]$$
$$= \sum_{l=k}^{n_{(c)}} \binom{n_{(c)}}{l} \left(r^l_{(c),j}(1 - r_{(c),j})^{n_{(c)}-l} - r^l_{(c),j-1}(1 - r_{(c),j-1})^{n_{(c)}-l} \right), \tag{18}$$

where $r_{(c),0} = 0, r_{(c),1} = p_{(c),1}, r_{(c),2} = p_{(c),1} + p_{(c),2}, \ldots, r_{(c),j} = p_{(c),1} + p_{(c),2} + \ldots + p_{(c),j}, \ldots$

For the read time distribution of (13),

$$r_{(c),j} = 1 - P^{(i-1)}_{\text{fail}} \left(1 - \left(\frac{j}{N} - (i-1) \right) (1 - P_{\text{fail},i}) \right).$$

Using (18) and $x_{(c),j}$ obtained above, the mth moment of $X_{k:n_{(c)}}$ under the read time distribution is calculated from

$$\begin{aligned}\mathbb{E}[X^m_{k:n_{(c)}}] &= \sum_{j=1}^{N_{\max}} x^m_{(c),j} \Pr[X_{k:n_{(c)}} = x_{(c),j}] \\ &= \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(c)}} \binom{n_{(c)}}{l} x^m_{(c),j} \left(R^{(c)}_{j,l} - R^{(c)}_{j-1,l} \right),\end{aligned} \tag{19}$$

where $R^{(c)}_{j,l} \triangleq r^l_{(c),j}(1 - r_{(c),j})^{n_{(c)}-l}$.

Since we modeled the degraded version of the proposed system under the IR policy as an $M/G/1$ queue with preemptive resume priority, the average delay for class c, denoted by $S^{(c)}_{\text{IR}}$, is written as [44]

$$\begin{aligned}S^{(c)}_{\text{IR}} &= \frac{\mathbb{E}[X_{k:n_{(c)}}]}{1 - \sum_{v=1}^{c-1} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}]} \\ &+ \frac{\sum_{v=1}^{c} \lambda_{(v)} \mathbb{E}[X^2_{k:n_{(v)}}]}{2(1 - \sum_{v=1}^{c-1} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])(1 - \sum_{v=1}^{c} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])}.\end{aligned} \tag{20}$$

Substituting (19) into (20) and defining $D^{(c)}_{j,l} \triangleq R^{(c)}_{j,l} - R^{(c)}_{j-1,l}$ yield the upper bounds of the mean read access time $S_{(c)}$ for class c ($c \in 1, 2$). The condition in (17) is from the stability requirement for the $M/G/1$ queue, namely: $\lambda_{(1)}\mathbb{E}[X_{k:n_{(1)}}] + \lambda_{(2)}\mathbb{E}[X_{k:n_{(2)}}] < 1$. $\square$

Note that (20) reduces to a form similar to (10) for $c = 1$, since a job of class 1 cannot see any other jobs of higher priority than itself while it stays in the queue.

The upper bound on the mean read access time of the (n, k) MDS-coded memory system under read request failure and NAND element failure, S_{IR}, is the expected time between a job arrival and the point of service completion where any k out of n requests have been served. For the system under consideration, it is given by the weighted sum of the mean read access time of two classes as follows:

$$S_{\text{IR}} = \frac{\lambda_{(1)} S^{(1)}_{\text{IR}} + \lambda_{(2)} S^{(2)}_{\text{IR}}}{\lambda_{(1)} + \lambda_{(2)}}.$$

If the stability condition in (17) is not satisfied, the mean latency of the corresponding $M/G/1$ queuing system will grow without bound. The violation of the stability condition causes the mean latency of the read job (or both the read and repair jobs) to be infinite. Namely, the violation of the stability condition means that there exists a certain priority class c so that jobs with a priority class lower than c have the mean latency growing without bound.

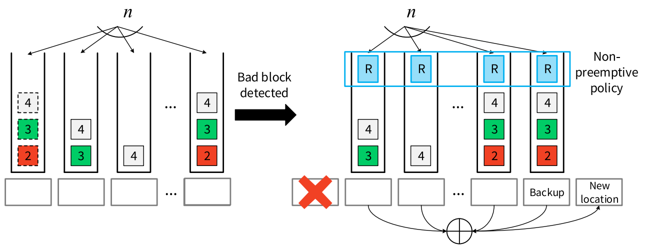


Fig. 8. The reconstruction process from a NAND block failure under the PR policy. Even if a NAND block fails, regular read jobs denoted by the square boxes continue to be served, and the virtual repair job requests having a lower priority appear (represented by the rectangular boxes marked by "R") and move to the tail of queues. When the request queue is idle, the reconstruction process starts. The n intact nodes including the backup block help the reconstruction of the failed block's contents to a new location. Taking the k fastest copying requests out of n is sufficient for the reconstruction.

7 Analysis of Read Access Time With Postponed Repair From NAND Element Failures

7.1 Repair Under the PR Policy

In Section 6, we have assumed that the read job requests are interrupted and the system enters the reconstruction phase every time the NAND element failure events occur. However, this might be impractical under certain circumstances since all the read job requests would have to be suspended and wait for the failed node to recover. To get around this issue, we introduce in this section the PR policy wherein the system suspends the repair process until the read request queue is idle. The repair process is depicted in Fig. 8.

The desired upper bound on the mean read access time of this system can be represented as an $M/G/1$ queue with *non-preemptive priority* [48]. Under the non-preemptive priority, the service continues without interruption even if a higher priority job arrives during the service time of an undergoing job. Here we assume two classes of priority as in Section 6. For the system under consideration here, we assign a higher priority (class $c = 1$) to the read jobs and a lower priority (class $c = 2$) to the repair jobs (or failure events). Since the read job is given the higher priority for service, even if a node failure occurs while a read job is being performed, the intact nodes can continue the read job. The repair job is executed when there is no read request in the queue because it has a lower priority. From the assumption that the non-preemptive priority is given in this case, the repair job is not preempted by the read jobs when the repair job is being served. This modeling is justified from the observation that the arrival rate of the failure is generally lower compared to the arrival rate of the read jobs.

In an attempt to simplify the problem, we assume that the number of intact blocks that can be accessed is maintained even when there occurs a block failure. To do so, suppose that we have one backup block that supports the coded data read from the time that a block failure occurs until the reconstruction is done. Then, we have $n_{(1)} = n_{(2)} = n$ under the PR policy as described in Definition 10. Under this scenario, the actual coding overhead of the MDS code in the proposed system is

$k/(n+1)$ instead of k/n. However, the backup block is activated only when there exists a failed block, which means that it is used for a limited time period. We thus stick to the notation of (n,k) MDS-outer-coded layered coding for this system. To guarantee a seamless transition to the use of the backup block instead of the failed block, it is further assumed that job requests in the queue of the failed block can be easily moved to the queue of the new location.

7.2 Analysis of Read Access Time

Since we modeled the (n,k) MDS-outer-coded layered coding system under the PR policy as an $M/G/1$ queue with non-preemptive priority, first we present the latency distribution for jobs of each class. We use the following definition of the memory sensing and decoding latency.

Definition 10. *Under the PR policy, for an (n,k) MDS-outer-coded layered coding system, the number of nodes involved in the read and repair job (denoted by $n_{(1)}$ and $n_{(2)}$, respectively), and the memory sensing and decoding latency of the read and repair job where decoding of the read operation succeeds at the ith sensing level (denoted by $\tau_{(1),i}$ and $\tau_{(2),i}$, respectively) for integer i such that $1 \le i \le N$ are defined as follows.*

$$\begin{aligned} n_{(1)} &= n\,, \quad \tau_{(1),i} = \tau_i\,, && \text{(readjobrequests)}\\ n_{(2)} &= n\,, \quad \tau_{(2),i} = N_p \tau_{\text{rep},i}\,. && \text{(repairjobrequests)}. \end{aligned}$$

Using Definition 10, we now present the mean read access time of the proposed system under the PR policy.

Theorem 11. *Under the PR policy, the mean read access time of the repair and read job in the (n,k) MDS-outer-coded layered coding system is upper bounded as follows:*

- $c = 1$ *(for read job requests)*

$$\begin{aligned} S_{\text{PR}}^{(1)} &= \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)} \\ &+ \frac{1}{2} \sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j}^2 \\ &\cdot D_{j,l}^{(v)} \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)} \right)^{-1}. \end{aligned} \tag{21}$$

- $c = 2$ *(for repair job requests)*

$$\begin{aligned} S_{\text{PR}}^{(2)} &= \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(2)}} \binom{n_{(2)}}{l} x_{(2),j} D_{j,l}^{(2)} \\ &+ \frac{1}{2} \sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j}^2 D_{j,l}^{(v)} \\ &\cdot \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)} \right)^{-1} \\ &\cdot \left(1 - \sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j} D_{j,l}^{(v)} \right)^{-1}. \end{aligned} \tag{22}$$

The stability condition is

$$\sum_{v=1}^{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(v)}} \binom{n_{(v)}}{l} \lambda_{(v)} x_{(v),j} D_{j,l}^{(v)} < 1\,. \tag{23}$$

Proof. The overall proof process is similar to the proof of Theorem 9. As in Section 6, consider $X_{k:n_{(c)}}$ $(c \in \{1,2\})$, the read access time distribution for priority class c. Under the PR policy, the read and repair job has the priority classes $c = 1$ and $c = 2$, respectively. Substituting $\tau_{(1),i} = \tau_i$ and $\tau_{(2),i} = \tau_{\text{rep},i}$ in Definition 10 into (18), $\Pr[X_{k:n_{(c)}} = x_{(c),j}]$ can be similarly obtained by using the approach of the proof for Theorem 9. We then easily calculate $\mathbb{E}[X_{k:n_{(c)}}^m]$ associated with the PR policy by using (19). The average delay for class c under non-preemptive priority policy, denoted by $S_{\text{PR}}^{(c)}$, is written as [44]

$$\begin{aligned} S_{\text{PR}}^{(c)} &= \mathbb{E}[X_{k:n_{(c)}}] \\ &+ \frac{\sum_{v=1}^{2} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}^2]}{2(1 - \sum_{v=1}^{c-1} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])(1 - \sum_{v=1}^{c} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])}\,. \end{aligned} \tag{24}$$

Substituting $\mathbb{E}[X_{k:n_{(c)}}^2]$ and $\mathbb{E}[X_{k:n_{(c)}}^1]$ which are calculated above into (24) yields the upper bounds of the mean read access time.

The condition in (23) is also from the stability requirement for the $M/G/1$ queue, namely: $\lambda_{(1)}\mathbb{E}[X_{k:n_{(1)}}] + \lambda_{(2)}\mathbb{E}[X_{k:n_{(2)}}] < 1$. The stability condition is the same as given in (17). □

Note that (21) automatically leads to the additional latency burden due to the effect of NAND element failures as stated below.

Corollary 12 (Impact of NAND element Failures). *Consider an (n,k) MDS-outer-coded layered coding system under the PR policy. The additional latency from NAND element failure is represented as follows:*

$$\begin{aligned} &\frac{1}{2} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(2)}} \binom{n_{(2)}}{l} \lambda_{(2)} x_{(2),j}^2 D_{j,l}^{(2)} \\ &\cdot \left(1 - \lambda_{(1)} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(1)}} \binom{n_{(1)}}{l} x_{(1),j} D_{j,l}^{(1)} \right)^{-1}. \end{aligned} \tag{25}$$

Proof. Substituting $c = 1$ into (24), we have the mean read access time of the read job

$$\begin{aligned} S_{\text{PR}}^{(1)} &= \mathbb{E}[X_{k:n_{(1)}}] \\ &+ \frac{\lambda_{(1)} \mathbb{E}[X_{k:n_{(1)}}^2]}{2\left(1 - \lambda_{(1)} \mathbb{E}[X_{k:n_{(1)}}]\right)} + \frac{\lambda_{(2)} \mathbb{E}[X_{k:n_{(2)}}^2]}{2\left(1 - \lambda_{(1)} \mathbb{E}[X_{k:n_{(1)}}]\right)}, \end{aligned} \tag{26}$$

which is represented by the summation of the mean service time of the read job and the mean waiting time of the read job in the queue. The summation of the first and second terms in (26) is equivalent to the mean read access

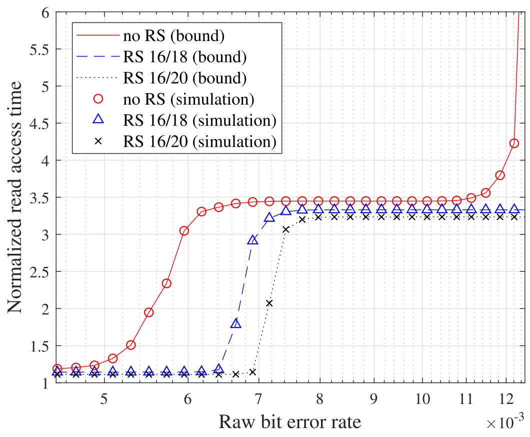


Fig. 9. Read access time for the layered RS codes with 1-KB LDPC codes without NAND element failures. RS codes of different rates k/n are used for comparison.

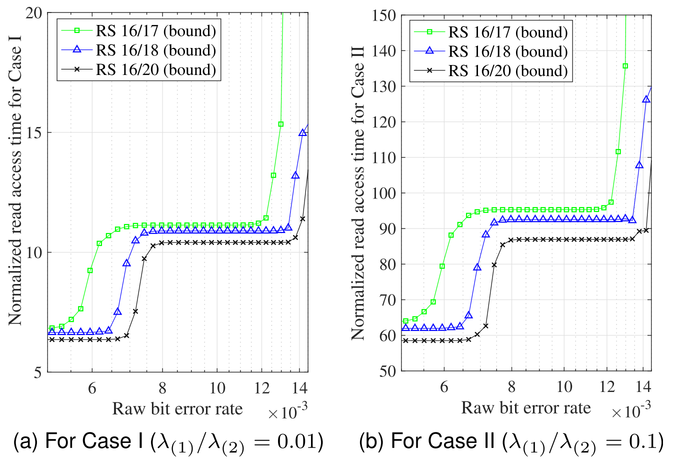


Fig. 10. Read access time of the combinations of RS codes and 1-KB LDPC codes with NAND element failures under the IR policy. RS codes of different rates k/n are used for comparison.

time without assuming NAND element failure events. Therefore, the last term in (26) can be regarded as an impact of NAND element failure events. Substituting $\mathbb{E}[X^2_{k:n_{(2)}}]$ and $\mathbb{E}[X_{k:n_{(1)}}]$ associated with the PR policy into

$$\frac{\lambda_{(2)}\mathbb{E}[X^2_{k:n_{(2)}}]}{2\left(1-\lambda_{(1)}\mathbb{E}[X_{k:n_{(1)}}]\right)},$$

yields (25). □

The upper bound on the mean read access time of the (n,k) MDS-outer-coded memory system under the PR policy is also given by the weighted sum of the read access time of two classes as follows:

$$S_{\text{PR}} = \frac{\lambda_{(1)}S^{(1)}_{\text{PR}} + \lambda_{(2)}S^{(2)}_{\text{PR}}}{\lambda_{(1)}+\lambda_{(2)}}.$$

As stated in Section 6, if the stability condition in (23) is not satisfied, the mean latency of the repair job (or both the repair and read jobs) will increase without bound.

In the latency analysis, we have not explicitly considered the background process that can interfere with the read requests, since this work focuses on the situation where the read operation is the major burden and how the high-level coding helps reduce read latency. In addition, as the background process occurs regardless of the use of layered coding, the present analysis will allow the assessment of the *improvement* in the overall latency with the use of layered coding. Moreover, although we do not consider the background process like garbage collection, we do include the effect of the failure handling process on read latency. Note that under the PR policy, the failure handling can be regarded as a background process interfering with the read process.

8 Quantitative Results and Discussion

For numerical simulations in this section, a 1-KB regular LDPC code of rate 0.8947 is employed as an inner ECC which is designed by using the progressive edge growth (PEG) algorithm [51]. As an outer ECC with the MDS property, an RS code is adopted. In our simulation, the number of data blocks k to be encoded is fixed to 16, in the range of the typical number for the NAND channels in commercial SSDs [52], while the number of nodes n is a variable. Here we only consider the storage overhead from 0 to 4. Higher storage overhead cases are excluded due to their impracticality.

The parameters related to sensing and decoding are set to $(\tau_{\text{sen-ref}}, \tau_{\text{sen}}, \tau_{\text{xfer}}, \tau_{\text{dec}}) = (96\ \text{s}, 96\ \text{s}, 5\ \text{s}, 8\ \text{s})$ based on the measurement results on 25 nm MLC NAND flash memory chips [2]. Each value is normalized by $\tau_{\text{sen-ref}} + \tau_{\text{xfer}} + \tau_{\text{dec}}$ in order to represent the read access time for a given RBER in multiples of the first hard-decision sensing and decoding delay. α is set to 0.2, which means the measured samples of the read access time are assumed to be dispersed up to 20 percent [53], [54] from τ_i's.

Fig. 9 shows the normalized mean read access time versus RBER. The bounds are fairly tight to the simulated result in all cases. We see that coding across NAND elements gives improved access time. There is a consistent access time reduction when we put one or more outer ECC parity nodes compared to the case without an outer RS code. For example, with an RS code having one or more parity nodes, we see up to about a 70 percent reduction in the mean read access time in the RBER region below 0.006. We see that as the code gets stronger (larger n and/or smaller k/n), the reduction in read access time becomes more pronounced at RBERs around 0.006 to 0.008. In the RBER region around 0.008 to 0.011, there is a consistent reduction in latency up to 10 percent for the systems with layered coding. At the right end of the plot, we see that the read access time of the case without the RS code tends to diverge. The read access time has a "knee" behavior because there exist ranges of tolerable RBERs for each level of sensing. The required number of reads increases in a step-like fashion due to the need to improve the sensing level as the RBER gets worse.

Fig. 10 presents the results as the read access time versus RBER with the NAND element failures under the IR policy. Here we assume $N_p = 32$ and consider the following two cases of NAND element failure: (Case I) $\lambda_{(1)}/\lambda_{(2)} = 0.01$ and (Case II) $\lambda_{(1)}/\lambda_{(2)} = 0.1$, i.e., the node failures occur 100 times and 10 times less frequently than the read request arrivals, respectively. The programming delay is set to $\tau_{\text{prog}} = 785.5$ s. This parameter is again normalized as in the

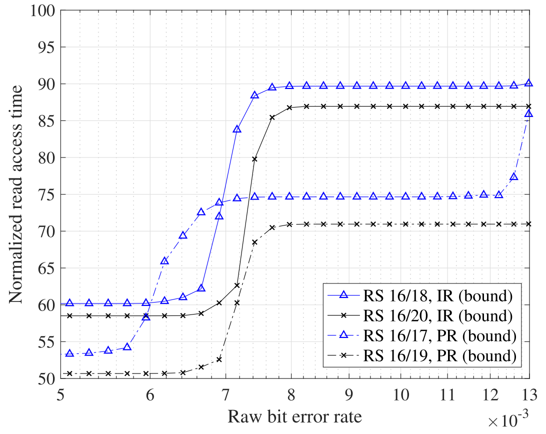


Fig. 11. Read access time of the combinations of RS codes and 1-KB LDPC codes with NAND element failures under the IR/PR policy. RS codes of different rates k/n are used for comparison.

case of the parameters related to sensing and decoding. Note that $n = k$ is impossible in our setting since the system cannot tolerate even one node failure in which case the access time diverges to infinity. Although not shown, we observed that the bounds were again tight to the simulated result as in the cases without the NAND element failure. Coding improves read access time even when there exist node failures, in addition to enhancing the system reliability. The read access time improvement is larger for Case I with less frequent failure events. For example, in Case I we see up to about a 45 percent reduction in mean read access time in the RBER region around 0.006 to 0.008, while Case II gives about a 40 percent reduction. As the failure events become more frequent, the amount of reduction in read access time gets smaller. The read access time here also has a knee behavior.

Fig. 11 compares the normalized mean read access time under the IR and PR policies for given RBERs. If the failure events do not occur frequently, it can be easily predicted that the difference in read access time between the IR and PR policies will not be large. Thus we present the result only with $\lambda_{(1)}/\lambda_{(2)} = 0.1$ (i.e., the node failures occur 10 times less frequently than the read request arrivals). Recall that the PR policy reserves one node of the outer code for backup operations. For the sake of fairness, we compare the latency of the PR policy with a rate-$k/(n-1)$ outer code to the latency of the IR policy with a rate k/n outer code.

The advantage of PR policy is evident when we have four additional nodes (RS code rate 16/20 for IR and 16/19 for PR) for layered coding. The PR policy performs better than the IR policy in most RBER regimes that we are interested in. However, if there are two additional nodes (RS code rate 16/18 for IR and 16/17 for PR), the PR policy has poor performance compared to the IR policy in some regions where the latency increases rapidly according to the RBER. This is because, when using PR policy, one additional node is used to keep the read operation intact even if the node failure occurs. It would also be an interesting research topic to examine the latency gain through a more advanced model for PR policy that does not use a backup node later.

Figs. 9, 10 and 11 provide important insights into the role of redundant coding in improving the system's ability to tolerate degrading quality of individual NAND channels while maintaining the same level of read latency. Specifically, the figures reveal the nontrivial trade-offs between the read access time and the overhead of outer codes. Based on this, one can be guided to design the new practical SSD system maximizing the impact of the additional storage overhead imposed to improve the read access speed, as the effect of adding additional nodes varies depending on the RBER range. For example, in Fig. 9, it is seen that the impact of two additional nodes of outer coding is several-folds larger in the RBER region around 0.005 to 0.008 compared to the remaining RBER region. This observation leads us to be able to choose the proper number of parity nodes for the outer code balancing the desired level of read access time and the storage overhead of outer ECC. In other words, this type of analysis can provide guidance on the required outer ECC overhead in achieving a certain level of read access time improvement given an estimated operating RBER.

A concern may arise for the burden of using additional nodes to accommodate the parity symbols for the outer code. However, this layered coding structure offers design options for high-performance applications where top priority is placed on minimum latency at the expense of an overall code rate loss.

9 Conclusion

We have provided a queuing theoretic analysis for SSDs with parallel NAND channels with varying processing speeds. The impact of the read failure and the node failure events on the trade-off between read access time and coding overhead has been analyzed. An existing (n, k) fork-join model has been extended to include the NAND-specific read access time and node failures, and tight upper bounds on the mean read access time have been derived. Two different ways to mitigate the effect of node failures have been investigated using the notion of multi-class jobs with different priorities. Tolerable limits on the qualities of the physical NAND components under access time and storage space constraints can be investigated in this way.

Acknowledgments

The authors would like to thank Dr. Sukkwang Park for his helpful advice and comments. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) under Grant No. NRF-2016R1A2B4011298 and Grant NRF-2019R1I1A2A02061135, and in part by the BK21 Plus. This article was presented in part at the IEEE International Conference on Communications (ICC), 2017 [1].

References

[1] H. Park and J. Moon, "Improving read access time of high-performance solid-state drives via layered coding schemes," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–7.
[2] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," in *Proc. USENIX Conf. File Storage Technol.*, 2013, pp. 243–256.
[3] G. Dong, N. Xie, and T. Zhang, "On the use of soft-decision error-correction codes in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 429–439, Feb. 2011.

[4] G. Dong, N. Xie, and T. Zhang, "Enabling NAND flash memory use soft-decision error correction codes at minimal read latency overhead," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2412–2421, Sep. 2013.

[5] J. Hsieh, C. Chen, and H. Lin, "Adaptive ECC scheme for hybrid SSD's," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3348–3361, Dec. 2015.

[6] R. Liu, M. Chuang, C. Yang, C. Li, K. Ho, and H. Li, "Improving read performance of NAND flash SSDs by exploiting error locality," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1090–1102, Apr. 2016.

[7] J. Moon, J. No, S. Lee, S. Kim, S. Choi, and Y. Song, "Statistical characterization of noise and interference in NAND flash memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 8, pp. 2153–2164, Aug. 2013.

[8] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," in *Proc. Conf. Des. Autom. Test Europe*, 2013, pp. 1285–1290.

[9] Q. Wu and T. Zhang, "OFWAR: Reducing SSD response time using on-demand fast-write-and-rewrite," *IEEE Trans. Comput.*, vol. 63, no. 10, pp. 2500–2512, Oct. 2014.

[10] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[11] N. B. Shah, K. Lee, and K. Ramchandran, "The MDS queue: Analysing the latency performance of erasure codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2014, pp. 861–865.

[12] S. Chen *et al.*, "When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1042–1050.

[13] Y. Xiang, T. Lan, V. Aggarwal, and Y. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2443–2457, Aug. 2016.

[14] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Proc. Annu. Allerton Conf. Commun. Control, Comput.*, 2012, pp. 326–333.

[15] A. Kumar, R. Tandon, and T. C. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 221–233, Apr. 2017.

[16] Y. Xiang, T. Lan, V. Aggarwal, and Y. Chen, "Optimizing differentiated latency in multi-tenant, erasure-coded storage," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 1, pp. 204–216, Mar. 2017.

[17] DuraClass technology, [Online]. Available: https://www.ampinc.com/wp-content/uploads/2013/09/RAISE.pdf

[18] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Operating Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed File System," in *Proc. IEEE Symp. Mass Storage Syst. Technol.*, 2010, pp. 1–10.

[20] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.

[21] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[22] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[23] I. Tamo, Z. Wang, and J. Bruck, "Access versus bandwidth in codes for storage," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2028–2037, Apr. 2014.

[24] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.

[25] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, Aug. 2014.

[26] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[27] H. Park, D. Lee, and J. Moon, "LDPC code design for distributed storage: Balancing repair bandwidth, reliability, and storage overhead," *IEEE Trans. Commun.*, vol. 66, no. 2, pp. 507–520, Feb. 2018.

[28] K. V. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth," in *Proc. USENIX Conf. File Storage Technol.*, 2015, pp. 81–94.

[29] J. Kim, E. Lee, J. Choi, D. Lee, and S. H. Noh, "Chip-level RAID with flexible stripe size and parity placement for enhanced SSD reliability," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1116–1130, Apr. 2016.

[30] Y. Li, P. P. C. Lee, and J. C. S. Lui, "Analysis of reliability dynamics of SSD RAID," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1131–1144, Apr. 2016.

[31] T. J. Dell, "A white paper on the benefits of Chipkill-correct ECC for PC server main memory," *IBM Microelectronics Division*, vol. 11, pp. 1–23, 1997.

[32] HP advanced memory protection technologies, 2008. [Online]. Available: ftp://ftp.hp.com/pub/c-products/servers/options/c00256943.pdf

[33] S. Shadley, "NAND flash media management through RAIN," 2011. [Online]. Available: https://www.micron.com/ /media/documents/products/technical-marketing-brief/brief_ssd_rain.pdf

[34] Micron Technology, Inc., "Memory management in NAND flash arrays (TN-29–28)," 2005. [Online]. Available: https://www.micron.com/∼/media/documents/products/technical-note/nand-flash/tn2928.pdf/

[35] C. Dirik and B. Jacob, "The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization," *Proc. ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 279–289, 2009.

[36] J.-U. Kang, H. Jo, J.-S. Kim, and J. Lee, "A superblock-based flash translation layer for NAND flash memory," in *Proc. ACM & IEEE Int. Conf. Embedded Softw.*, 2006, pp. 161–170.

[37] F. Chen, R. Lee, and X. Zhang, "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2011, pp. 266–277.

[38] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and exploiting the multilevel parallelism inside SSDs for improved performance and endurance," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1141–1155, Jun. 2013.

[39] S. Im and D. Shin, "Flash-aware RAID techniques for dependable and high-performance flash memory SSD," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 80–92, Jan. 2011.

[40] D. Ryu, "Solid state disk controller apparatus," U.S. Patent 8 159 889, Apr. 17, 2012.

[41] C. Kim and A. K. Agrawala, "Analysis of the fork-join queue," *IEEE Trans. Comput.*, vol. 38, no. 2, pp. 250–255, Feb. 1989.

[42] R. Nelson and A. Tantawi, "Approximate analysis of fork/join synchronization in parallel queues," *IEEE Trans. Comput.*, vol. 37, no. 6, pp. 739–743, Jun. 1988.

[43] E. Varki, A. Merchant, and H. Chen, "The M/M/1 fork-join queue with variable sub-tasks," 2009. [Online]. Available: Unpublished-http://www.cs.unh.edu/ varki/publication/2002-nov-open.pdf

[44] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.

[45] H. A. David and H. N. Nagaraja, *Order Statistics*. New York, NY, USA: Wiley, 2003.

[46] G. Casella and R. L. Berger, *Statistical Inference*, vol. 2. Pacific Grove, CA, USA: Duxbury, 2002.

[47] K. S. Trivedi, *Probability & Statistics With Reliability, Queuing and Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2008.

[48] L. S. C. Harrison White, "Queuing with preemptive priorities or with breakdown," *Operations Res.*, vol. 6, no. 1, pp. 79–95, 1958.

[49] J. Kim, S. Seo, D. Jung, J. Kim, and J. Huh, "Parameter-aware I/O management for solid state disks (SSDs)," *IEEE Trans. Comput.*, vol. 61, no. 5, pp. 636–649, May 2012.

[50] Q. Wei, C. Chen, and J. Yang, "CBM: A cooperative buffer management for SSD," in *Proc. IEEE Symp. Mass Storage Syst. Technol.*, 2014, pp. 1–12.

[51] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[52] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proc. USENIX Annu. Tech. Conf.*, 2008, vol. 8, pp. 57–70.

[53] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of NAND flash memory," in *Proc. USENIX Conf. File Storage Technol.*, 2012, pp. 2–2.

[54] P. Desnoyers, "What systems researchers need to know about NAND flash," in *Proc. USENIX Workshop Hot Topics Storage File Syst.*, 2013, Art. no. 6.

Hyegyeong Park (Member, IEEE) received the PhD degree in electrical engineering from the Korea Advanced Institute of Science and Technology, South Korea, in 2018, and was a postdoctoral researcher at Korea Advanced Institute of Science and Technology, South Korea. She is currently a postdoctoral fellow with the Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania. Her research interests include the field of coding and information theory for distributed systems with the current focus on coding for distributed computing and distributed machine learning.

Jaekyun Moon (Fellow, IEEE) received the PhD degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania. He is currently a professor of electrical engineering at Korea Advanced Institute of Science and Technology, South Korea. From 1990 through early 2009, he was with the faculty of the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities. He was consulted as chief scientist for DSPG, Inc. from 2004 to 2007. He also worked as chief technology officer at Link-A-Media Devices Corporation. His research interests are in the area of channel characterization, signal processing and coding for data storage and digital communication. He received the McKnight Land-Grant Professorship from the University of Minnesota, Minneapolis, Minnesota. He received the IBM Faculty Development Awards as well as the IBM Partnership Awards. He was awarded the National Storage Industry Consortium (NSIC) Technical Achievement Award for the invention of the maximum transition run (MTR) code, a widely used error-control/modulation code in commercial storage systems. He served as program chair for the 1997 IEEE Magnetic Recording Conference. He is also past chair of the Signal Processing for Storage Technical Committee of the IEEE Communications Society. He served as a guest editor for the 2001 IEEE JSAC issue on Signal Processing for High Density Recording. He also served as an editor for the *IEEE Transactions on Magnetics* in the area of signal processing and coding for 2001 to 2006.