Improving Read Access Time of High-Performance Solid-State Drives via Layered Coding Schemes

Hyegyeong Park, *Student Member*, *IEEE* and Jaekyun Moon, *Fellow*, *IEEE*
School of Electrical Engineering
Korea Advanced Institute of Science and Technology
Email: parkh@kaist.ac.kr, jmoon@kaist.edu

***Abstract*—We study 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 node failures. It provides a clear picture on the coding-overhead and read-access-time trade-offs given read failures and node failures. The node failure in the present context reflects various limitations on the memory element level such as page failures, block failures or channel failures that occur during the access of stored data from NAND flash memory chips. A strong motivation for this work is to understand the reliability requirement of NAND chip components given a layer of erasure protection across nodes, under the latency/storage-overhead constraints.**

I. 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) and well-tailored signal processing have been increasingly used to help relieve this burden on NAND manufacturing. For example, advanced low-density parity-check (LDPC) coding is now widely deployed to protect pages read off the NAND blocks, allowing raw bit error rates (RBERs) in accessing the NAND cells to drop to extremely low levels. 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 take an approach from a different direction. We investigate the power of ECC while bringing the read access time into the picture. We in particular explore coding-overhead and read-access-time trade-offs under the given RBER and memory element failure probability. To do this, we borrow the idea and analytical tools from the studies on erasure coding for distributed storage. Distributed storage coding is an active area of research. There have been studies on trade-offs among various key parameters such as storage overhead, repair bandwidth, access I/O, access speed and energy consumption [1]–[8]. The past work that is most relevant to our cause is the storage-overhead/download-time trade-off analysis of [6], [7]. In [6], [7], the authors discuss the possibility of using erasure coding to improve download time of distributed storage. The idea is to provide storage diversity via coding so that in the presence of varying delays in accessing local disks, fetching any k fastest data blocks out of n would complete the content download. This directly translates into a reduced download time of the target content stored across multiple storage nodes.

The approaches of [6], [7], however, are not directly applicable to the present problem, as the model and analysis there do not consider the SSD-specific read access time and the possibility of read/node failures, which are critical for our purposes. To this end, we modify the (n, k) fork-join model introduced in [6], [7] to include the two types of failures. One is the read failure in accessing individual NAND cells in spite of using ECCs, and the other is the NAND node failure event. The NAND node (or, simply node) here can be die/block/page which is similar to the silicon element in the *redundant array of independent silicon elements* (RAISE) [9]. Such node failure events are included as a higher priority job class in our analysis. While the multiple classes with different priorities are also considered in [8], there the priority is introduced to model the heterogeneity of the cloud storage data and the approach is not applicable to the problem at hand.

Like in [6]–[8], we derive bounds for the mean access time, while also considering SSD-specific read access time and node failure events. In [10], progressive memory-sensing/LDPC-decoding is utilized to reduce system latency in the face of read failure events. In the progressive memory sensing and decoding scheme, once the LDPC 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 paper.

In contrast to the work of [10], we also consider the node failures. The node failure, when occurs, is given the highest priority for service (i.e., reconstruction) under the preemptive resume priority. Note that the chip level memory failure is a growing concern and a variety of techniques related to the redundant-array-of-independent disks (RAID) already

This work was supported by the National Research Foundation of Korea under grant no. 2016R1A2B4011298 and ICT R&D program of MSIP/IITP. [2016-0-00563, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion]

exist to address this issue. For examples, *Chipkill* from IBM [11], *Advanced ECC* from HP [12] and *redundant array of independent NAND* (RAIN) from Micron [13] all provide fault tolerance on the chip level. RAISE from SandForce [9] further considers data protection on the die/block/page level.

Unlike in the RAID systems, we aim at both mitigating failures and improving data access time. In particular, we introduce a layered coding structure boosted by outer maximum distance separable (MDS) coding across nodes, 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 [14]. The highly parallelized structure of SSDs opens up the possibility of introducing queuing theoretic analysis on the node-level. In our case, reducing access time is possible via the outer-layer 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. We provide clear pictures on how the presence of failures affects the coding-overhead/read-latency trade-offs and how the coding allows the system to tolerate failures without compromising the system's read access speed. Compared to the recent work of [6]–[8], as mentioned above, we include the effects of the failures - both read failure and node failure - in the latency-storage trade-off analysis. This is critical in providing necessary insights into new trade-offs related to the tolerable level of physical memory failure rates and lays down a path toward more efficient utilization of storage overheads under the consideration of the target yields in the NAND manufacturing process.

II. Proposed Coding Structure

A. Layered Coding Structure

A failure of NAND elements such as the pages, blocks or dies is catastrophic in that it cannot be recovered by the ECC applied over a page. In order to provide additional data protection to correct full page/block/die failures, we introduce outer ECCs across NAND nodes based on a RAID viewpoint. Such a combination of an inner ECC and an outer ECC across the NAND nodes forms a layered coding structure in Fig. 1. Soft-decision ECCs commonly used in practical SSDs have significantly stronger error correction capability than a hard-decision ECC such as the Bose-Chaudhuri-Hocquenghem (BCH) code [10]. However, a latency loss caused by multiple retries of read-voltage sensing is inevitable to obtain stronger error performance. This structure invokes the outer ECC each time the inner ECC (e.g., LDPC codes) decoding step fails (see Fig. 2). Latency caused by consecutive read retries of the LDPC code can be reduced by the help of the outer ECC. We wish to improve read latency and node failure tolerance by leveraging the outer-code.

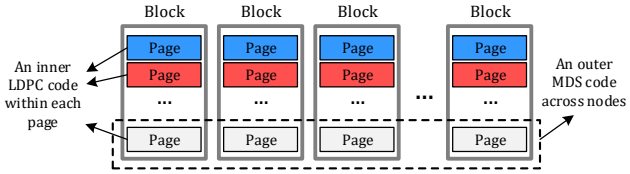


Fig. 1. Illustration of the layered coding structure. Each page is protected by an inner LDPC code. An outer MDS code is introduced across the nodes.

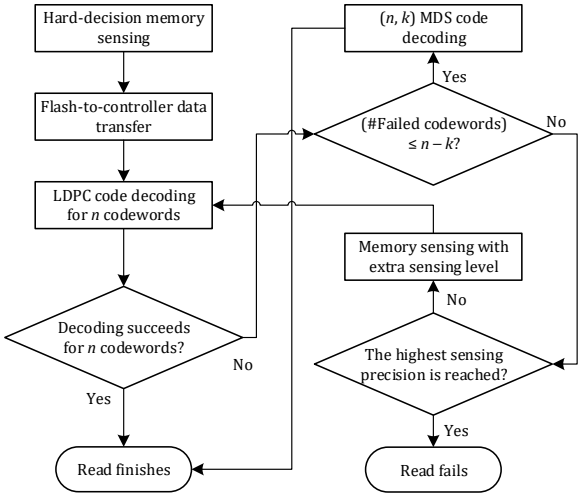
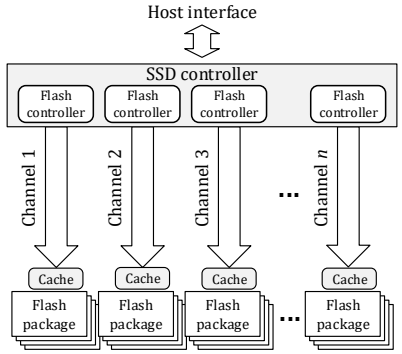


Fig. 2. Flow of the (n, k) MDS-coded layered coding structure

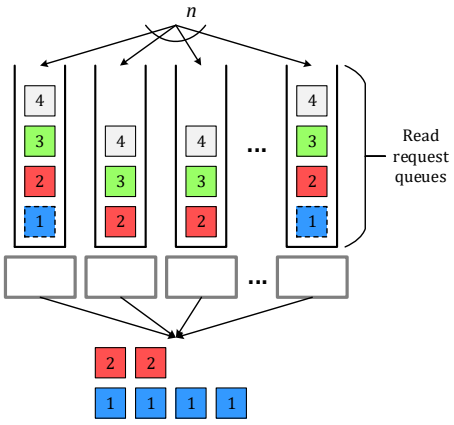
Incorporating the outer ECC in the memory system, we propose a framework wherein the NAND memory interfaces in an SSD act as *distributed data storage*. A distributed storage system consists of multiple storage nodes which contain information in a distributed manner. A simple replication or an erasure code is often applied across the nodes to improve data reliability throughout the system. Layered coding and distributed storage coding clearly have parallels in using ECCs across the nodes to provide node-level data reliability.

B. NAND Interface Modeling Using Queuing

In our modeling of the SSD system, there are multiple channels each of which is connected to one or more NAND flash chips. Each channel is assumed to have its own queue due to the presence of read/write processing speed variations across different memory chip interfaces (see Fig. 3a). 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; depending on whether the SSD architecture allows a separate hardware decoder for each channel the errors in the pages out of a given channel might have been suppressed. 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, each NAND flash



(a)



(b)

Fig. 3. NAND flash interface as a queuing model. (a) NAND interface with n channels combating latency variations. (b) Fork-join queuing model with n queues. For an $(n, k) = (10, 4)$ MDS code under consideration, accessing any four out of ten tasks complete the corresponding job.

channel can be interpreted as a distributed node with its own queue. This view is consistent with some of the existing high-performance enterprise SSD architectures [15], where separate NAND controllers or caches/buffers are employed that help smooth out the access speed variations across the parallel NAND channel interfaces [16].

We consider a storage system that consists of n distributed nodes or n NAND interface channels, which is modeled using the (n, k) fork-join system introduced in [6]. In this model, 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.

For illustrative purposes, consider the fork-join model in Fig. 3b corresponding to $(n, k) = (10, 4)$. Since four out of ten tasks for job 1 have already been served, the remaining

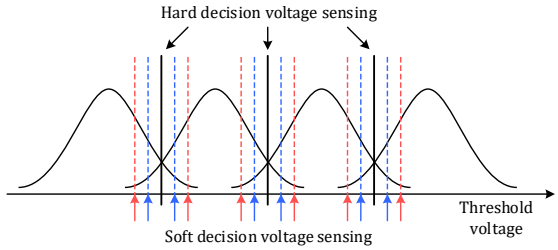


Fig. 4. Hard- and soft-decision voltage sensing for the MLC NAND flash

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

We assume that the read requests occur randomly according to the Poisson arrivals with rate λ. The read access time at each node is governed by the behavior of memory sensing and decoding as we shall see later in Section III-A. It is possible to assume that the read access time at each node is independent and identically distributed due to the wear-leveling technique.

III. Analysis of Read Access Time

A. Read Access Latency of SSDs

Soft-decision ECCs (e.g., LDPC codes) perform significantly better than hard-decision ECCs such as the BCH codes. 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. 4 illustrates progressive sensing and decoding [10] for the multi-level-cell (MLC) NAND flash channel. Once decoding fails following hard-decision sensing, the controller invokes soft-decision LDPC decoding. To do this, the sensing level between the adjacent storage states increases by one. The procedure is repeated until LDPC 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.

Based on the progressive sensing and decoding technique, the memory sensing and decoding latency to reach the i^{th} sensing level is given by

$$\tau_i = \tau_{\text{sen-ref}} + \tau_{\text{hard}} + \tau_{\text{dec}} + (i-1)(\tau_{\text{sen}} + \tau_{\text{soft}} + \tau_{\text{dec}}) \text{ for } i \in \mathbb{N}_+ \quad (1)$$

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. 4), τ_{hard} and τ_{soft} represent the latency of transferring one extra sensing level, and τ_{dec} indicates the decoding delay of an LDPC code.

Assume that 1) the observations of read access time are scattered about τ_i and are represented by N equally spaced values with a maximum dispersion of α to reflect the data's spread from the mean in reality (see Fig. 5), and that 2)

each i^{th} set of N read access time samples representing the i^{th} 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.

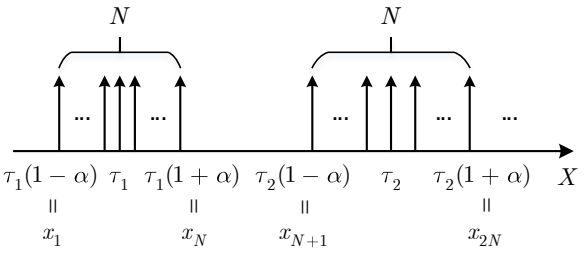


Fig. 5. Illustration of the read access time distribution

We now present the analysis of read access time considering following two types of failures.

- Read request failure: LDPC decoding failures invoke additional voltage sensing
- NAND node failure: occasional node failures require data reconstruction.

B. 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 [7]. We also focus on the bounds on the mean read access time. We confine our interest to the upper bounds which can give enough insights into the read latency.

The probability that LDPC decoding succeeds at the i^{th} sensing level is given by

$$P(T = \tau_i) = P_{\text{fail}}^{(i-1)}(1 - P_{\text{fail},i}) \text{ for } i \in \mathbb{N}_+ \tag{2}$$

where $P_{\text{fail},i}$ denotes the probability of read request failure at the i^{th} sensing level, $P_{\text{fail}}^{(i-1)} \triangleq P_{\text{fail},1} P_{\text{fail},2} \cdots P_{\text{fail},i-1}$ and $P_{\text{fail}}^{(0)} \triangleq 1$.

From (2) we define the probability mass function of read time distribution X with the j^{th} read-try for the proposed system:

$$p_j \triangleq P(X = x_j) = \frac{1}{N} P_{\text{fail}}^{(i-1)}(1 - P_{\text{fail},i}) \text{ for } j \in \mathbb{N}_+ \tag{3}$$

where $x_j = \tau_i[1-\alpha+\frac{2\alpha}{N-1}\{j-N(i-1)-1\}]$ and $i = \lceil \frac{j}{N} \rceil$.

In order to obtain an upper bound, we resort to the split-merge system as in [17], [18], which is a degraded version of the fork-join system where all n nodes are kept from continuing on to the next job until k out of n requests are served. The mean access time of the split-merge system is modeled as a $M/G/1$ system[1] with the service time governed by k^{th} order statistic. The k^{th} order statistic is defined as the k^{th} smallest sample of n random variables [20].

[1]An $M/G/1$ queue is a single-server queuing system where arrivals are Markovian and the service times have a general distribution [19].

Let $X_1, \cdots, X_n$ be random samples from a discrete distribution with the probability mass function $f_X(x_j) = p_j$, where $x_1 < x_2 < \cdots$ are the possible samples of X. For a sample of size n, let $X_{1:n}, \cdots, X_{n:n}$ denote the order statistics from the sample. Then the k^{th} order statistic from the sample is given by [21]:

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

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

For the read time distribution of (3),

$$r_j = 1 - P_{\text{fail}}^{(i-1)} \left[1 - \left\{ \frac{j}{N} - (i-1) \right\} (1 - P_{\text{fail},i}) \right] \tag{4}$$

where $i = \lceil \frac{j}{N} \rceil$.

The Pollaczek-Khinchin (P-K) mean-value formula [22] gives the mean access time of an $M/G/1$ system S in terms of its first two moments.

$$S = \mathbb{E}[X_{k:n}] + \frac{\lambda \mathbb{E}[X_{k:n}^2]}{2(1 - \lambda \mathbb{E}[X_{k:n}])} \tag{5}$$

Let $N_{\max}$ denote the maximum number of read tries. Then m^{th} moment of the read access time distribution is given by

$$\mathbb{E}[X_{k:n}^m] = \sum_{j=1}^{N_{\max}} x_j^m P(X_{k:n} = x_j) \tag{6}$$

$$= \binom{n}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} I_j^m(x_j, N, P_{\text{fail},\{1,i\}}, l, n) \tag{7}$$

where $i = \lceil \frac{j}{N} \rceil$, $P_{\text{fail},\{1,i\}} \triangleq \{P_{\text{fail},1}, \cdots, P_{\text{fail},i}\}$ and $I_j^m(x_j, N, P_{\text{fail},\{1,i\}}, l, n) \triangleq x_j^m \Big[[1 - P_{\text{fail}}^{(i-1)} \{1 - (\frac{j}{N} - i + 1)(1 - P_{\text{fail},i})\}]^l \cdot [P_{\text{fail}}^{(i-1)} \{1 - (\frac{j}{N} - i + 1)(1 - P_{\text{fail},i})\}]^{n-l} - [1 - P_{\text{fail}}^{(i-1)} \{1 - (\frac{j-1}{N} - i + 1)(1 - P_{\text{fail},i})\}]^l \cdot [P_{\text{fail}}^{(i-1)} \{1 - (\frac{j-1}{N} - i + 1)(1 - P_{\text{fail},i})\}]^{n-l} \Big]$.

Therefore, the read access time is upper bounded by

$$S = \binom{n}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} I_j^1(x_j, N, P_{\text{fail},\{1,i\}}, l, n) + \frac{\lambda \binom{n}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} I_j^2(x_j, N, P_{\text{fail},\{1,i\}}, l, n)}{2[1 - \lambda \binom{n}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} I_j^1(x_j, N, P_{\text{fail},\{1,i\}}, l, n)]} \tag{8}$$

provided $\lambda \binom{n}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n} I_j^1(x_j, N, P_{\text{fail},\{1,i\}}, l, n) < 1$. The condition is from the stability requirement for the $M/G/1$ queue, namely: $\lambda \mathbb{E}[X_{k:n}] < 1$.

IV. Analysis of Read Access Time in the Presence of Node Failures

A. Repairing the Node Failure

In this section, we give an analysis of the read access time including the presence of node failures in addition to considering only read request failures as in Section III. Queuing with

breakdown or failure of queues can be interpreted as queuing with *preemptive resume priority* [23]. Under the preemptive resume priority policy, the service is interrupted when a higher priority request arrives (see Fig. 6). Assume two classes of priorities ($c \in \{1, 2\}$) and denominate the normal job to access the node as a *regular* read job. When a node 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 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., node failure event) arrives. Read requests resume from the point of interruption once the repair requests have been served.

Once a node failure is detected, the neighbors of the failed node embark on a reconstruction phase to maintain the desired redundancy level. In order to repair the failed node, contents of the failed node need be duplicated to another location. The repair process takes the k fastest copying requests out of the $n-1$ nodes. Both the repair job and the read job can be described essentially as a collection of the k fastest requests and thus be represented by the k^{th} 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 node failure since it is the dominant failure pattern. Multiple node failures can be handled by consecutive reconstructions from single node 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.

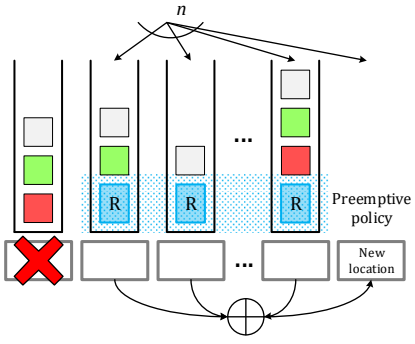


Fig. 6. The reconstruction process from a NAND node failure. When a NAND node fails, regular read jobs denoted by the square boxes are interrupted, and the virtual repair job requests appear (represented by the rectangular boxes marked by "R") and move to the head of queues (under preemptive priority). 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.

B. Analysis of Read Access Time

Under the additional consideration of node 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 arrivals 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 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 i^{th} sensing level is given by

$$\begin{aligned}\tau_{(1),i} &= \tau_i + \tau_{\text{prog}} \\ &= \tau_{\text{sen-ref}} + \tau_{\text{hard}} + \tau_{\text{dec}} \\ &\quad + (i-1)(\tau_{\text{sen}} + \tau_{\text{soft}} + \tau_{\text{dec}}) + \tau_{\text{prog}} \text{ for } i \in \mathbb{N}_+ \end{aligned} \tag{9}$$

where τ_{prog} denotes the programming latency.

In the same sense as Section III, we define the probability mass function of the read time distribution considering the node failure events:

$$p_{(c),j} \triangleq P(X = x_{(c),j}) = \frac{1}{N} P_{\text{fail}}^{(i-1)}(1 - P_{\text{fail},i}) \text{ for } j \in \mathbb{N}_+ \tag{10}$$

where $x_{(c),j} = \tau_{(c),i}[1 - \alpha + \frac{2\alpha}{N-1}\{j - N(i-1) - 1\}]$ and $i = \lceil \frac{j}{N} \rceil$.

The m^{th} moment of $X_{k:n_{(c)}}$ under the read time distribution is given by

$$\mathbb{E}[X_{k:n_{(c)}}^m] = \sum_{j=1}^{N_{\max}} x_{(c),j}^m P(X_{k:n_{(c)}} = x_{(c),j}) \tag{11}$$

$$= \binom{n_{(c)}}{k} \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(c)}} I_j^m(x_{(c),j}, N, P_{\text{fail},\{1,i\}}, l, n_{(c)}) \tag{12}$$

where $i = \lceil \frac{j}{N} \rceil$, $P_{\text{fail},\{1,i\}} \triangleq \{P_{\text{fail},1}, \cdots, P_{\text{fail},i}\}$ and $I_j^m(x_{(c),j}, N, P_{\text{fail},\{1,i\}}, l, n_{(c)}) \triangleq x_{(c),j}^m \Big[[1 - P_{\text{fail}}^{(i-1)}\{1 - (\frac{j}{N} - i + 1)(1 - P_{\text{fail},i})\}]^l \cdot [P_{\text{fail}}^{(i-1)}\{1 - (\frac{j}{N} - i + 1)(1 - P_{\text{fail},i})\}]^{n_{(c)}-l} - [1 - P_{\text{fail}}^{(i-1)}\{1 - (\frac{j-1}{N} - i + 1)(1 - P_{\text{fail},i})\}]^l \cdot [P_{\text{fail}}^{(i-1)}\{1 - (\frac{j-1}{N} - i + 1)(1 - P_{\text{fail},i})\}]^{n_{(c)}-l} \Big]$.

Let $S_{(c)}$ $(c \in \{1, 2\})$ be the service time distribution for priority class c. Then the average delay for class c under preemptive resume priority policy is written as [19]:

$$\begin{aligned} S_{(c)} = &\frac{\mathbb{E}[X_{k:n_{(c)}}]}{1 - \sum_{v=1}^{c-1} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}]} \\ &+ \frac{\sum_{v=1}^{c} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}^2]}{2(1 - \sum_{v=1}^{c-1} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])(1 - \sum_{v=1}^{c} \lambda_{(v)} \mathbb{E}[X_{k:n_{(v)}}])} \end{aligned} \tag{13}$$

Note that (13) reduces to a form similar to (5) for $c = 1$, since a job of class 1 cannot see any other job of higher priority than itself while it stays in the queue. Substituting (12) into (13) yields the upper bounds of read access time $S_{(c)}$ for class c ($c \in 1, 2$):

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

$$S_{(1)} = \binom{n_{(1)}}{k} J^1(1) + \frac{\lambda_{(1)} \binom{n_{(1)}}{k} J^2(1)}{2[1 - \lambda_{(1)} \binom{n_{(1)}}{k} J^1(1)]} \quad (14)$$

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

$$S_{(2)} = \frac{\binom{n_{(2)}}{k} J^1(2)}{1 - \lambda_{(1)} \binom{n_{(1)}}{k} J^1(1)} + \frac{\sum_{v=1}^{2} \lambda_{(v)} \binom{n_{(v)}}{k} J^2(v)}{2[1 - \lambda_{(1)} \binom{n_{(1)}}{k} J^1(1)][1 - \sum_{v=1}^{2} \lambda_{(v)} \binom{n_{(v)}}{k} J^1(v)]} \quad (15)$$

where $J^m(c) \triangleq \sum_{j=1}^{N_{\max}} \sum_{l=k}^{n_{(c)}} I_j^m(x_{(c),j}, N, P_{\text{fail},\{1,i\}}, l, n_{(c)})$ for simplicity, provided $\sum_{v=1}^{2} \lambda_{(v)} \binom{n_{(v)}}{k} J^1(v) < 1$. The condition is from the stability requirement for the $M/G/1$ queue, namely: $\lambda_{(1)} \mathbb{E}[X_{k:n_{(1)}}] + \lambda_{(2)} \mathbb{E}[X_{k:n_{(2)}}] < 1$.

The upper bound on the mean read access time of the (n, k) MDS-coded memory system under read request failure and node failure, S_{nf}, 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

$$S_{\text{nf}} = \frac{\lambda_{(1)} S_{(1)} + \lambda_{(2)} S_{(2)}}{\lambda_{(1)} + \lambda_{(2)}}. \quad (16)$$

V. Quantitative Results

For simulation, 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 [24]. As an outer ECC with the MDS property, a Reed-Solomon (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, 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 $(\tau_{\text{sen-ref}}, \tau_{\text{sen}}, \tau_{\text{hard}}, \tau_{\text{soft}}, \tau_{\text{dec}}) = (96\ \mu\text{s}, 96\ \mu\text{s}, 5\ \mu\text{s}, 5\ \mu\text{s}, 8\ \mu\text{s})$ based on the measurement results on 25 nm MLC NAND flash memory chips [10]. Each value is normalized by $\tau_{\text{sen-ref}} + \tau_{\text{hard}} + \tau_{\text{dec}}$ 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% [25], [26] from τ_i's.

Fig. 7 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 nodes gives improved access time. There are 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 with one or more parity nodes, we see up to about 70% 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 latency gain up to 10% 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 exists 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. 8 presents the results as the read access time versus

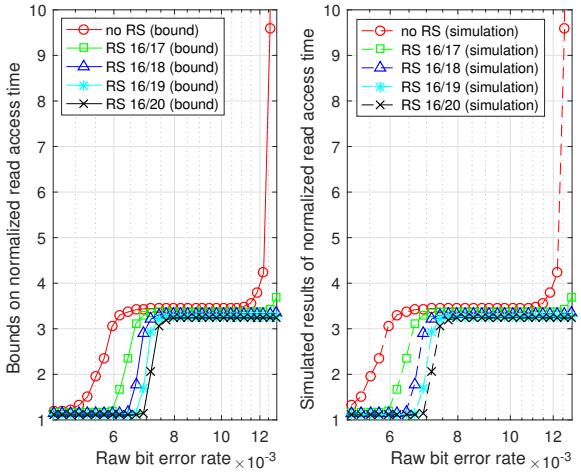


Fig. 7. Read access time for the layered RS codes with 1-KB LDPC codes without node failures. RS codes of different rates k/n are used for comparison.

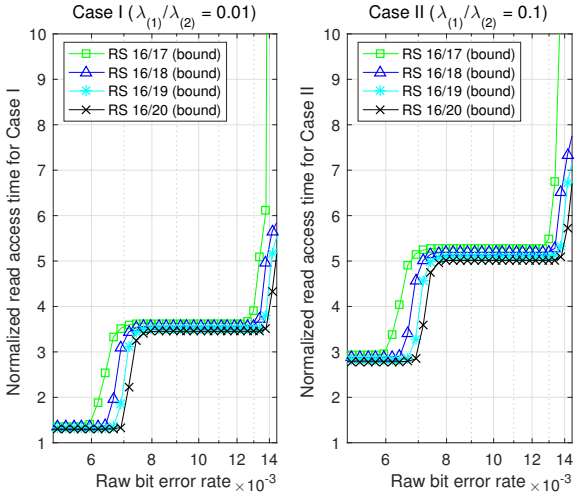


Fig. 8. Read access time of the combinations of RS codes and 1-KB LDPC codes with node failures. RS codes of different rates k/n are used for comparison.

RBER, considering the node failures for 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 $\tau_{\text{prog}} = 1.57$ ms. This parameter is again normalized as in the 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 just as the cases without the node 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 65% reduction in mean read access time in the RBER region around 0.006 to 0.008, while Case II gives about 50% 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.

Figs. 7 and 8 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. 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. This type of analysis can also provide guidance on the required outer ECC overhead in achieving a certain level of read access time improvement given an estimated operating RBER.

VI. Conclusion

We provided a queuing theoretic analysis for SSDs with parallel NAND channels with varying processing speeds. The impact of the read failure as well as 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 service time and node failures, and tight upper bounds have been derived 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.

References

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4539–4551, Sept 2010.

[2] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE, 2009, pp. 1243–1249.

[3] C. Suh and K. Ramchandran, "Exact-repair mds codes for distributed storage using interference alignment," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, June 2010, pp. 161–165.

[4] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: Mds array codes with optimal rebuilding," *Information Theory, IEEE Transactions on*, vol. 59, no. 3, pp. 1597–1616, March 2013.

[5] ——, "Access versus bandwidth in codes for storage," *Information Theory, IEEE Transactions on*, vol. 60, no. 4, pp. 2028–2037, April 2014.

[6] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, Oct 2012, pp. 326–333.

[7] ——, "On the delay-storage trade-off in content download from coded distributed storage systems," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 5, pp. 989–997, May 2014.

[8] A. Kumar, R. Tandon, and T. Clancy, "On the latency and energy efficiency of distributed storage systems," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[9] "Duraclass technology," http://www.seagate.com/tech-insights/duraclass-technology-master-ti/.

[10] 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 *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*, 2013, pp. 243–256.

[11] T. J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," *IBM Microelectronics Division*, pp. 1–23, 1997.

[12] "Hp advanced memory protection technologies," ftp://ftp.hp.com/pub/c-products/servers/options/c00256943.pdf, p. 7, 2008.

[13] S. Shadley, "Nand flash media management through rain," https://www.micron.com/~/media/documents/products/technical-marketing-brief/brief_ssd_rain.pdf, 2011.

[14] 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 *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 2011, pp. 266–277.

[15] S. Im and D. Shin, "Flash-aware raid techniques for dependable and high-performance flash memory ssd," *Computers, IEEE Transactions on*, vol. 60, no. 1, pp. 80–92, Jan 2011.

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

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

[18] E. Varki, A. Merchant, and H. Chen, "The m/m/1 fork-join queue with variable sub-tasks," *Unpublished–http://www. cs. unh. edu/varki/publication/open. pdf*, 2008.

[19] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.

[20] H. A. David and H. N. Nagaraja, *Order statistics*. Wiley Online Library, 1970.

[21] G. Casella and R. L. Berger, *Statistical inference*. Duxbury Pacific Grove, CA, 2002, vol. 2.

[22] K. S. Trivedi, *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons, 2008.

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

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

[25] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of nand flash memory," in *Proceedings of the 10th USENIX conference on File and Storage Technologies*. USENIX Association, 2012, pp. 2–2.

[26] P. Desnoyers, "Empirical evaluation of nand flash memory performance," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 50–54, 2010.