**H-0210 November 23, 2003**
**Computer Science**

# IBM Research Report

## LDGM Codes for Storage Systems

**Ami Tavory, Vladimir Dreizin, Shmuel Gal, Meir Feder**
IBM Research Division
Haifa Research Laboratory
Haifa 31905, Israel

**Research Division**
**Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich**

# LDGM Codes for Storage Systems

Ami Tavory, Vladimir Dreizin, Shmuel Gal, and Meir Feder
IBM's Haifa Research Labs

*Abstract*— **Networked storage-systems and communication systems typically use codes to protect data from failures. As reliability settings in these two types of systems are different, codes developed for storage systems can out-perform communication codes applied to storage systems. In this work we show such codes, as well as their applications.**

## I. Introduction

To combat device failures, storage systems typically partition data into groups, and protect each data group by some form of an *ECC* (error-correcting code), *e.g.*, *RAID* (redundant array of independent disks) [1]. Coding theory has been extensively studied in the context of communications, where the important considerations are rate, error-correction capabilities, and computational complexity. Coding theory in the context of storage systems, might also have other considerations.

In communication systems, all of the data group is received by the receiver, regardless of error correction. Once a data group has been encoded, it is usually not modified. The bit error-probability of a data group is usually considered significant. There is little question on data-group location, as once a group is encoded, it is simply transmitted.

In storage systems, each data group has a *recovery penalty* and *update penalty*, which affect the number of storage devices actively involved in reliability-related operations. The recovery penalty is the number of bits which need be accessed to recover erased bits; The update penalty is the number of bits which need be accessed when bits are modified. The block error-probability of a data group is considered more significant, as it usually determines the *MTTDL* (mean time to data loss). Data-group location among devices is dynamic, as data can be corrected and migrated from failed devices to replacement devices (which blurs the distinction between coding theory and algorithm design).

In this work we study *LDGM* (low-density generation matrix) codes which are powerful, efficient, low-complexity codes, having low recovery and update penalties. We show how these codes can be applied for storage systems in two settings. We first show a simple application to large groups of *reference data*, which are data rarely modified. We then analyze a *random sparing* dynamic layout based on these codes.

**Related Work.** Reliability in storage systems was originally studied in the context of small-capacity systems [14], [15], [1], and in conjunction with performance improvement via parallelism, *e.g.*, RAID. The schemes were later extended in some directions. Concatenated codes were studied, *e.g.*, two-dimensional codes [16], and new RAID levels [17]. Questions on coding-group placement within devices were studied, *e.g.*, various distributed striping and sparing techniques [18], [19], [20], [21], [22], [23]. Differential coding based on data activity was studied in the context of very large, concrete systems [27], [28].

Later work in storage reliability has considered larger-capacity systems, and therefore a failure model taking into account multiple simultaneous errors. Some excellent analytical work can be found in [29], [30], [31].

The most relevant communication codes for this work are low-complexity codes, *e.g.*, [9], in particular, capacity-achieving codes, *e.g.*, Turbo codes [10], and LDPC and LDGM codes [11], [12], [13].

**Definitions and Notations.** We consider a system composed of $n$ storage devices. Each devices is composed of $c$ equal-sized blocks, and has a read/write bandwidth of $r$ blocks per time unit. We model the devices' failure laws as being distributed i.i.d. exponentially with mean $\frac{1}{\lambda}$ [1]. The failure of a device corresponds to the erasure of all data on it. Given a systematic erasure-code composed of $m$ data bits and $\alpha \cdot m$ check bits, we define the storage rate as $\frac{1}{1+\alpha}$. The *system MTTDL* is the mean time to data loss in the system.

For any $p$, we use $[p]$ to denote the set $\{1, \ldots, p\}$.

**Paper Organization.** The remainder of the paper is organized as follows. In Section II we describe the variant of LDGM codes we propose for storage systems. In Section III we discuss a simple application to reference data. In Section IV we discuss an application to a random-placement technique.

## II. LDGM Codes for Storage-Systems

In this section we discuss a variation of *LDGM* (low-density generation-matrix) codes. *LDPC* (low-density parity-check) and LDGM codes were extensively studied in the field of communication [33], [12], [11], [34], [35], due to their low computational-complexity and nearly-optimal rate. We propose a variation suitable for storage systems. The use of LDGM codes for storage was independently proposed previously [27], [32], [28], [35], for reasons of computational-complexity alone, and under the caveat that they are probabilistic.

Consider two coding alternatives with equal storage rates. In the first, all bits are encoded as a single group. In the second, the bits are partitioned into groups, and each group is coded. By the law of large numbers, it is clear that the former alternative is superior in terms of loss prob-

Email:{tavory,dreizin,gals}@il.ibm.com,meir@eng.tau.ac.il

ability to the latter (whose loss probability approaches 1, as the number of groups grows). The former alternative might have other drawbacks. Reed-Solomon coding [7], for example, would necessitate accessing nearly all bits for the recovery of nearly any failure configuration. We are interested in codes for large data-groups which have a recovery penalty determined by the number of errors which took place, rather than being always proportional to the group size.

The complementary question, of bounded update penalty (defined similarly), was previously addressed in an excellent paper [31], where an inverse relationship between recovery locality and update locality was also shown.

We first give the relevant definitions of LDGM codes. We consider a recursive LDGM code $C$ composed of $p$ levels, $C^{(1)},\ldots,C^{(p)}$, each of which can be *un-augmented* or *augmented*.

A un-augmented level of an LDGM code is described by a *Tanner-graph*, which is a bipartite graph $G=(V,E)=(L\bigcup\limits^{\cdot}R,E)$. Let $m=|L|$. Each node in $L$ represents a data bit; each node in $R$ represents a parity bit. The value of any node $v$ is denoted by $value\,(v)$. For any parity node $v^r\in R$, its *left-neighbor* set, $\overleftarrow{V}\,(v^r)\subseteq L$, is the set $\overleftarrow{V}\,(v^r)=\left\{v^\ell\in L\mid(v^\ell,v^r)\in E\right\}$. For any data node $v^\ell\in L$, its *right-neighbor* set, $\overrightarrow{V}\left(v^\ell\right)$, is defined similarly. The parity bit corresponding to $v^r$ is set to be the *XOR* (exclusive or) of the data bits corresponding to the nodes in $\overleftarrow{V}\,(v^r)$, *i.e.*,

$$value\,(v^r)=\bigoplus_{v^\ell\in\overleftarrow{V}(v^r)}value\left(v^\ell\right). \tag{1}$$

The edges in $E$ thus represent parity constraints. The graph is *sparse*, *i.e.*, $|E|=O\left(|L|\right)=O\left(|R|\right)$.

The average left-node and right-node degrees are

$$a_\ell=\frac{\sum_{v^\ell\in L}\left|\overrightarrow{V}\left(v^\ell\right)\right|}{|L|},\quad a_r=\frac{\sum_{v^r\in R}\left|\overleftarrow{V}\,(v^r)\right|}{|R|}, \tag{2}$$

respectively. The fractions of edges whose left and right node-degree is $i$, are

$$\begin{aligned}\lambda_i&=\frac{\left|\left\{e=(v^\ell,v^r)\in E\mid\left|\overrightarrow{V}\left(v^\ell\right)\right|=i\right\}\right|}{|E|},\\ \rho_i&=\frac{\left|\left\{e=(v^\ell,v^r)\in E\mid\left|\overleftarrow{V}\,(v^r)\right|=i\right\}\right|}{|E|},\end{aligned} \tag{3}$$

respectively. These degrees are usually put into the form of generating functions

$$\lambda(x)=\sum_{i=1}^{\infty}\lambda_i x^{i-1},\quad\rho(x)=\sum_{i=1}^{\infty}\rho_i x^{i-1}. \tag{4}$$

Note that $\lambda(x)$ and $\rho(x)$ completely define the un-augmented level.

Assume that all nodes in $R$ are valid, and that some nodes in $L$ have failed. The recovery process is iterative. In each step, a node $v^r\in R$ is chosen s.t. a single node $v^\ell\in\overleftarrow{V}\,(v^r)$ is failed. The value of $v^\ell$ is then corrected using (1). Of course, the recovery process terminates prematurely if such a node $v^r$ cannot be found. We denote by $\mathbf{P}_{\text{block}}\,(L)$ the block error-probability, which is the probability that the recovery process terminates while some bits in $L$ have not been recovered. We denote by $\mathbf{P}_{\text{bit}}\,(L,\alpha)$ the $\alpha$-bit error probability, which is the probability that the recovery process terminates while an $\alpha$ fraction of bits in $L$ have not been recovered.

We will use the following theorem from [11].

*Theorem 1:* Assume a fraction of $\delta$ nodes from $L$ originally fail.

1. In the family of codes for which

$$\forall_{x\in(0,\delta]}\delta\cdot\lambda(1-\rho(1-x))<x, \tag{5}$$

a fraction of $1-o\,(1)$ of the codes have $\forall_{\alpha<1}\mathbf{P}_{\text{bit}}\,(L,\alpha)=o\,(e^{-\alpha m})$.

2. In the family of codes for which (5) holds and $\lambda_1=\lambda_2=0$, a fraction of $1-o\,(1)$ of the codes have $\mathbf{P}_{\text{block}}\,(L)=o\left(e^{-\Omega(m)}\right)$.

As seen in Theorem 1, only a code for which $\lambda_1=\lambda_2=0$, guarantees the recovery of all nodes. On the other hand, it was shown in [11] that if $\lambda_1=\lambda_2=0$, then the storage rate is suboptimal. For this reason, an augmented level is commonly used. In an augmented level, the nodes in $L$ are protected by both an LDGM code for which $\lambda_1=\lambda_2=0$ does not hold, and by an expander-based code [36]. The LDGM code corrects all but a negligible number of failed nodes. The expander-based code corrects the rest. Note also that it is assumed that all nodes in $R$ are valid. This is achieved by recursive encoding, obtaining $p$ levels, up to the point where some other erasure code (*e.g.*, a Reed-Solomon code) is used.

We first define a right-regular code which is a variation of one in [13], and analyze its properties.

*Definition 1:* Let $\hat{C}^{(i)}$ be an un-augmented level of an LDGM code, defined by the generating functions:

$$\begin{aligned}\hat{\lambda}(x)&=\frac{\sum_{k=2}^{\hat{q}-1}\binom{\hat{\alpha}}{k}(-1)^{k+1}x^k}{1-\frac{\hat{q}}{\hat{\alpha}}\binom{\hat{\alpha}}{\hat{q}}(-1)^{\hat{q}+1}-\hat{\alpha}},\\ \hat{\rho}(x)&=x^{\hat{a}-1},\end{aligned} \tag{6}$$

where $\hat{a}\geq2$ and $\hat{q}\geq3$, are arbitrary integers, and $\hat{\alpha}=1/(\hat{a}-1)$. Let the average left-node degree be denoted by $\hat{a}_\ell$ (the average right-node degree is obviously $\hat{a}$).

The following theorem can be proved, for the most part, by modifying proofs from [11] and [13].

*Theorem 2:* Let

$$\hat{\delta}_{\max}=\frac{1-\frac{\hat{q}}{\hat{\alpha}}\binom{\hat{\alpha}}{\hat{q}}(-1)^{\hat{q}+1}-\hat{\alpha}}{1-\hat{\alpha}}. \tag{7}$$

Then we have the following attributes of $\hat{C}^{(i)}$:

1. Let $\delta$ denote the fraction of errors in $L$.

$$\delta \leq \hat{\delta}_{\max} \Rightarrow \mathbf{P}_{\text{block}}(L) = o\left(e^{-\Omega(m)}\right). \tag{8}$$

2. • If the number of failed data-nodes is $|L_f|$ and the number of accessed data-nodes is $|L_a|$, then

$$|L_a| \leq \hat{a}\,|L_f|\,. \tag{9}$$

• Let $\hat{C}$ be a recursive LDGM code composed of $\hat{C}^{(i)}$s. Then the expected number of accesses per single (data or parity) failed-node recovery approaches $\hat{a}+1$.

3. • If the number of modified data-nodes is $|L_u| \ll |L|$ and the number of accessed parity-nodes is $|R_a|$, then

$$|R_a| \approx |R| - |R|\left(1 - \frac{\hat{a}_\ell\,|L_u|}{\hat{a}\,|R|}\right)^{\hat{a}} \tag{10}$$

• Let $\hat{C}$ be a recursive LDGM code composed of $\hat{C}^{(i)}$s, protecting $n$ original data-bits. Let a single data-node be modified. Then the ratio between the expected number of accessed nodes and total number of redundant nodes is

$$o\left(\frac{1}{n^{\Omega(1)}}\right), \tag{11}$$

where $n$ is the number of original data-bits.

4. The storage rate of the code is greater than

$$1 - \frac{2\left(1 - \frac{\hat{q}}{\hat{\alpha}}\binom{\hat{\alpha}}{\hat{q}}(-1)^{\hat{q}+1} - \hat{\alpha}\right)}{1 - \frac{2}{\hat{\alpha}}\binom{\hat{\alpha}}{\hat{q}}(-1)^{\hat{q}+1} - \hat{\alpha}}. \tag{12}$$

5. The ratio between the fraction of errors corrected by $\hat{C}^{(i)}$, and the fraction corrected by an optimal code of the storage rate in (12), is

$$\frac{1}{2} - \frac{1}{q^{\hat{\alpha}+1}(1-\hat{\alpha})}. \tag{13}$$

6. The computational complexity of encoding and decoding is linear.

The following right-regular code is from [13].

*Definition 2:* Let $\tilde{C}^{(i)}$ be an un-augmented level of an LDGM code, defined by the generating functions:

$$\begin{aligned} \tilde{\lambda}(x) &= \frac{\tilde{\alpha}\sum_{k=1}^{\tilde{q}-1}\binom{\tilde{\alpha}}{k}(-1)^{k+1}x^k}{\tilde{\alpha} - \tilde{q}\binom{\tilde{\alpha}}{\tilde{q}}(-1)^{\tilde{q}+1}}, \\ \tilde{\rho}(x) &= x^{\tilde{a}-1}, \end{aligned} \tag{14}$$

where $\tilde{a} \geq 2$ and $\tilde{q} \geq 2$, are arbitrary integers, and $\tilde{\alpha} = \frac{1}{\tilde{a}-1}$.

We now define a code that combines the previous two.

*Definition 3:* For some $\epsilon$, let $\acute{C}^i = \acute{C}^i(\epsilon)$ be an LDGM level defined by $\tilde{C}^i$, and augmented by $\hat{C}^i$ coded at storage rate

$$\frac{2\epsilon}{\frac{1}{2} - \frac{1}{q^{\hat{\alpha}+1}(1-\hat{\alpha})}}. \tag{15}$$

By its construction, the code defined by $\acute{C}^i$ has similar recovery and update penalties and computational complexity, as the codes which compose it.

In terms of error correction, it works similarly to an LDGM code augmented by an expander-based code [11]. Assume that at most

$$\acute{\delta}_{\max} = \frac{\tilde{\alpha} - \tilde{q}\binom{\tilde{\alpha}}{\tilde{q}}(-1)^{\tilde{q}+1}}{\tilde{\alpha}} \tag{16}$$

errors occurred. The probability that $\tilde{C}^i$ leaves more than an $\epsilon$ fraction of nodes uncorrected, decreases exponentially in $n$. By Theorem 2, $\hat{C}^i$ corrects these uncorrected bits with high probability.

The ratio between the fraction of errors corrected by the code and the fraction corrected by an optimal code of the same storage rate is

$$1 - \frac{1}{q^{\tilde{\alpha}+1}}\frac{1}{1 + \frac{2\epsilon}{\frac{1}{2} - \frac{1}{q^{\tilde{\alpha}+1}(1-\tilde{\alpha})}}}. \tag{17}$$

## III. Applications to Reference Data

The first simple application of the LDGM code is for protecting large amounts of reference data (*i.e.*, data which is not often updated). In this application, the bounded proportionality of accessed bits per erased bits, comes into effect. *E.g.*, consider a three-site reference-data system storing a copy of each datum in two of the three sites. While this system can survive a disaster obliterating a single site, or ongoing failures affecting some devices in all three sites, the MTTDL can be shown to decrease linearly in the number of devices. Using an LDGM code to protect all devices, would result in a system with similar performance, but with MTTDL increasing in the number of devices. Using a classic Reed-Solomon code to protect all devices, would result in a system whose devices would be engaged in much of the time in recovery-related operations.

## IV. Applications to Random Sparing

Data which are active (as opposed to reference data), are not best protected by the method in Section IV; The update penalty is too high. Commonly, active data-groups are distributed in some manner over devices, in an attempt to parallelize request handling [18], [19], [20], [21], [22], [23]. We consider a *random-sparing* scheme, apparently similar to an independent OceanStore solution [27], but differing in the random location-decision and use of bounded-penalty codes.

We first describe the scheme.

Initially, there are $n$ devices in the system. The data is divided into data groups. For simplicity, we assume the size of each group is $\ell$ blocks, of which $\ell'$ can be corrected. We assume the code used has bounded recovery-locality. Specifically, we assume that the ratio of accessed blocks per failed blocks of a group, is at most $k$. We assume an *average fill-ratio* of $\beta$, *i.e.*, of the $nc$ blocks in the system, $\beta nc$ are taken (and so the number of groups is $\frac{\beta nc}{\ell}$). We

place the restriction that a fraction of at most $\mu$ of all operations can be dedicated to recovery operations.

We divide the operating time into *rounds* of duration $t_s$, and *epochs*, each consisting of $s$ rounds. For simplicity, we assume that $st_s = \frac{c}{\mu r}$, *i.e.*, each epoch lasts the time that would be required to sequentially read a device from start to end, normalized by $\mu$.

Naturally, a request to a device can be blocked due to its servicing previous requests. We assume that each device has a queue of read and write requests, which it services subject to its bandwidth and constraint on fraction of recovery-operations. Memory is required for write requests in the queue, and for read requests which have been completed but whose contents are still needed. We later analyze the system-wide amount of such memory.

At some points, the scheme requires writing a (recovered) group element to a one of the operating devices. The selection of the operating device is random, but differs from the standard uniform selection between all operating devices which are not full [27]. Rather, a uniform selection is made between all operating devices (full or non-full), which does not contain a member of the data group. If a full device is selected, a *reassignment* takes place; A random element from the device is chosen to be written someplace else, and the process continues recursively. We explain the rationale for this later.

The scheme is composed of two concurrent processes, a *contracting* process and an *expanding* process, which we describe next.

The contracting process works as follows. In each round, the system observes the set of devices which have failed in the round. For each data group, the system identifies the members that are on failed devices. If these members cannot be recovered, then data has been lost at this time. Otherwise, the system randomly selects, for each failed member, a subset of $k$ devices out of all subsets of $k$ devices which can recover the element. For each of these devices, it inserts into its queue a request to read the required element. If the system has enough elements to recover an element, it recovers it, randomly selects a device, and inserts into its queue a request to write the element (possibly triggering reassignment).

We term this a contracting process, since it attempts to maintain the data groups into a continually contracting group of devices (those which are still operating).

The expanding process works as follows. At the beginning of each epoch, replacement devices are inserted into the system. The number of replacement devices is determined s.t. the expected number of operating devices in the end of the epoch will remain $n$. In each round, the system chooses, for each replacement device, $\frac{\beta c}{\ell t_s}$ random data groups which are not represented in the device. For each such group, it randomly selects a non-replacement device containing a member of the group, and inserts into its queue a request to read the required element. When the element has been read, the replacement devices writes it.

At the end of an epoch, all replacement devices which have not failed during an epoch, become (new) operating devices. For each element written to a replacement device during an epoch, the system modifies its marked location. It is now marked as being located in the (new) operating device. Its old location is marked as empty.

We term this a contracting process, since it attempts to maintain the data groups into a continually expanding group of devices.

From the above description, it is clear that the number of operating devices (originally $n$), and the average fill ratio of each device (originally $\beta$), change with time. In an arbitrary point in an epoch, we denote these sizes by $n'$ and $\beta'$, respectively.

The main result is the following theorem.

*Theorem 3:* Assume

$$\ell \gg \log(n), \quad (18)$$

$$c\left(1-\beta\left(1+2\lambda st_s+2(\lambda st_s)^2\right)\right) \gg 1, \quad (19)$$

$$\frac{1}{\lambda} \gg \frac{c}{\mu r}. \quad (20)$$

For some $\delta \gtrsim 0$, let the storage rate of each coding group be

$$\approx 1-(1+\delta)\lambda\left(t_s+\frac{k+1}{\mu r}\right). \quad (21)$$

Then the MTTDL of random sparing is

$$\frac{\frac{c}{s\mu r}}{e^{\left(-\frac{1}{2\frac{\beta nc}{\ell}e^{-\lambda\left(t_s+\frac{k+1}{\mu r}\right)}\frac{\ell\delta^2}{2}}\right)}}. \quad (22)$$

Following is the proof-outline of Theorem 3. We first cite some known points on *negatively-dependent* random variables. In Lemma 2 we bound the expectation of the number of full devices in a round. In Lemma 3 we bound the probability of a failure in a round, given the effective number of failed devices in the round. In Lemma 4 we approximate the number of pending recovery-requests, using Lemma 2. Using the number of pending requests, we approximate the effective number of failed devices in a round. For brevity, we omit much of the proofs.

The analysis of random sparing is complicated by the fact that the devices' and groups' states are not independent, *e.g.*, if some data-groups have very many representatives in some set of devices, then the number of representatives of other data-groups is probably not very large. This makes it difficult to apply directly the Chernoff bound. For this reason, we use in some places in the analysis, the notion of *negative dependence* [37].

*Definition 4:* Let $X = \left\{x_1, \ldots, x_{|X|}\right\}$ be an ordered set of random variables. The elements of $X$ are *negatively dependent* if for every disjoint index-sets $I \bigcup J \subseteq [|X|]$,

and any functions $f : \mathcal{R}^{|I|} \to \mathcal{R}$ and $g : \mathcal{R}^{|J|} \to \mathcal{R}$ that are both non-increasing or non-decreasing,

$$\mathbf{E}\left[f\left(x_i, i \in I\right) \cdot g\left(x_j, j \in J\right)\right] \leq \mathbf{E}\left[f\left(x_i, i \in I\right)\right] \cdot \mathbf{E}\left[g\left(x_j, j \in J\right)\right]. \tag{23}$$

The following lemma contains useful properties of negatively-associated random variables which we will use. The statements of the lemma appear in [37], or are slight variations of them.

*Lemma 1:* Let $X$ be an ordered set of random variables. Then
1. If $f : \mathcal{R} \to \mathcal{R}$ is a non-increasing or non-decreasing function, then the Chernoff bound can be applied to $\sum_{x \in X} f\left(x\right)$.
2. If $Y$ is a set of negatively-associated random variables, and $X$ and $Y$ are independent, then $X \bigcup Y$ is negatively dependent.

We first bound the expectation of the number of full devices in a round. This in turn, serves to bind the expected number of reassignments performed.

*Lemma 2:* At some round, let the fraction of operating devices' blocks be $\beta'$. Then, with high probability, the fraction of full devices is at most

$$\left(\frac{2}{\beta'\left(\frac{1}{\beta'}-1\right)^3 c}\right)^{\frac{1}{3}}. \tag{24}$$

Let the set of operating devices be $S'$. Let $n' = |S'|$. For some $\delta' \leq 1$ and $\beta'' \leq \beta'$, let there be a subset $S'' \subseteq S'$ of devices s.t. $|S''| \geq \delta' n'$, and $\forall_{i \in [n']'} \left|\underline{c}[i]\right| \geq \beta'' c$. A straightforward calculation shows that the maximum fraction of full devices is at most

$$\delta \leq \frac{\frac{\beta' n' c - \delta' n' \beta'' c}{c - \beta'' c}}{n'} = \frac{\beta' - \delta'\beta''}{1 - \beta''}. \tag{25}$$

From (25), to show that $\delta$ is small, we can show that $\delta' \overset{\beta'' \approx \beta'}{\approx} 1$.

To do so, consider a new, different, process which inserts data groups into devices, subject to the constraint that no two members of a group are in the same device, but with $c \to \infty$, which means that reassignments are not performed. Clearly, a device in the new process has smaller chances of being relatively non-full than in the original process. Define the vector $\underline{w}$ as the indicator of $\underline{c}[i]$ containing less than $(1-\epsilon)\beta' c$ elements, *i.e.*, $\underline{w}[i] = I\left(\underline{c}[i] \leq (1-\epsilon)\beta' c\right)$. By the Chernoff bound, it can be shown that for $i \in [n']$,

$$\mathbf{P}\left(\underline{w}[i] = 1\right) \leq e^{-\frac{\beta' c \epsilon^2}{2}}. \tag{26}$$

Unfortunately, we cannot directly deduce from the low probability of the event $\underline{w}[n'] = 1$ in (26), the high-probability of a low-fraction of entries of $\underline{w}$ being 1; The elements of $\underline{w}$ are neither independent, nor are they negatively dependent. Rather than using Lemma 1 directly, we show the Chernoff bound applies, by applying the Harris inequality in a slightly different way than used in [37]. Having done so, we obtain

$$\frac{\beta' - \left(1 - e^{-\frac{\beta' c \epsilon^2}{2}}\right)\beta'(1-\epsilon)}{1 - \beta'(1-\epsilon)} \overset{(a)}{\approx} \frac{e^{-\frac{\beta' c \epsilon^2}{2}} + \epsilon}{\frac{1}{\beta'} - 1} \overset{(b)}{\leq} \left(\frac{2}{\beta'\left(\frac{1}{\beta'}-1\right)^3 c}\right)^{\frac{1}{3}}, \tag{27}$$

where (a) follows from neglecting the second order term $\epsilon \cdot e^{-\frac{\beta' c \epsilon^2}{2}}$, and (b) follows from taking $\epsilon = \left(\frac{2}{\beta' c}\right)^{\frac{1}{3}}$.

We next bound the error probability in a round, assuming that all groups failing up to the round's start have been recovered.

*Lemma 3:* At some round, let the number of functioning devices be $n'$, and let the effective number of failed device in the round be $n'_f$. Assume that for some $\delta$, each group can be recovered if at most

$$\ell' = (1+\delta) n'_F \frac{\ell}{n'} \tag{28}$$

blocks of it are lost.

Then the probability of failure in the round, is

$$e^{\left(-\frac{1}{2\frac{\beta n c}{\ell} e^{-\frac{n'_F \frac{\ell}{n'} \cdot \delta^2}{2}}}\right)}. \tag{29}$$

The proof follows from the fact that given a set of devices, the number of representatives of the various data groups, are negatively dependent. This is achieved by the insertion and reassignment methods.

A block is *pending* if it is waiting to be read from or written to some device. The number of pending blocks affects the effective number of failures per round.

**Check what to do in next lemma regarding lamdda and**

*Lemma 4:* Let $m$ be the total number of system-wide pending blocks in steady state. Then

$$m \lessapprox n\left(\lambda\beta c t_s + \frac{e^{\frac{\Lambda}{\Xi}} - 1}{2 - e^{\frac{\Lambda}{\Xi}}}\right) \approx n\beta c\lambda\left(t_s + \frac{k+1}{\mu r}\right), \tag{30}$$

where

$$\Lambda \approx \lambda\beta c\left(k+1\right), \quad \Xi = \mu r. \tag{31}$$

The proof follows from queueing theory. The number of pending requests of a device can be bound by an M/D/1 queue. The birth process of the queue is determined by the

contracting process's read and write requests, the expanding process's read requests, and by reassignment requests. The first two can be shown to have approximately poisson distribution, and the latter can be shown to be negligible. The death rate is determined by the bandwidth of the device, and by the factor $\mu$. The steady-state state distribution can thus be found.

It can be shown that the queue lengths of the different devices are negatively dependent. This is used to project the steady-state average queue-length of the entire system.

## V. Conclusions and Future Work

In this work we have presented and analyzed codes for storage systems. We have shown that it is easy to construct LDGM codes which have bounded recovery penalty. We have shown algorithms which incorporate codes of this type. Finding bounds and optimal codes subject to bounded penalty constraints, is left to future research.

## VI. Acknowledgements

Thanks to Dana Ron and David Burshtein of the Dept. of EE-Systems at Tel Aviv University, and Alain Azagury, Michael Factor, Kalman Meth, Julian Satran, and Dafna Sheinwald of IBM, for useful discussions.

## References

[1] G. A. Gibson, *Redundant Disk Arrays: Reliable Parallel Secondary Storage*, ser. ACM Distinguished Dissertations. Cambridge, MA: MIT Press, 1992.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications. New York, NY, USA: John Wiley & Sons, 1991.

[3] P. Elias, "Coding for two noisy channels," in *Information Theory Third London Symposium*. London: Buterworth's Scientific Publications, Sept. 1955, pp. 61–76.

[4] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[5] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," in *Proceedings: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico*. 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA: IEEE Computer Society Press, Nov. 1994, pp. 604–612.

[6] S. Boucheron and K. Salamatian, "About priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 46, pp. 697–705, Mar. 2000.

[7] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, ser. North-Holland Mathematical Library. North-Holland, 1977, vol. 16.

[8] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, June 1960.

[9] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEETIT: IEEE Transactions on Information Theory*, vol. 42, 1996.

[10] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding:turbo-codes," in *Proceedings of IEEE ICC'93*.

[11] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC '97)*. New York: Association for Computing Machinery, May 1997, pp. 150–159.

[12] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEETIT: IEEE Transactions on Information Theory*, vol. 47, 2001.

[13] A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," pp. 65–76, 1999.

[14] J. A. Katzman, "System architecture for nonstop computing," in *14th IEEE Cpmputer Society International Conference (COMPCON)*, Feb. 1977, pp. 77–80.

[15] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proceedings of the ACM Conference on Management of Data (SIGMOD)*, June 1988, pp. 109–116.

[16] Q. M. Malluhi and W. E. Johnston, "Coding for high availability of a distributed-parallel storage system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 12, pp. 1237–1252, Dec. 1998.

[17] Q. Xin, E. Miller, D. Long, S. Brandt, T. Schwarz, and W. Litwin, "Reliability mechanisms for very large storage systems," in *In Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies. IEEE*, Apr. 2003.

[18] M. Holland, G. A. Gibson, and D. P. Siewiorek, "Architectures and algorithms for on-line failure recovery in redundant disk arrays," *Journal of Distributed and Parallel Databases*, vol. 2, no. 3, pp. 295–335, July 1994.

[19] J. Menon and D. Mattson, "Distributed sparing in disk arrays," in *Proceedings of the COMPCOM Conference*, Feb. 1992, pp. 410–421.

[20] J. M. Menon and R. L. Mattson, "Comparison of sparing alternatives for disk arrays," in *Proceedings the 19th Annual International Symposium on Computer Architecture,ACM SIGARCH*, Gold Coast, Australia, May 1992, pp. 318–329.

[21] A. Thomasian and J. Menon, "RAID5 performance with distributed sparing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 6, pp. 640–657, June 1997.

[22] X. Wu, J. Li, and H. Kameda, "Reliable analysis of disk array organizations by considering uncorrectable bit errors," in *Proceedings of The 16th Symposium on Reliable Distributed Systems (SRDS '97)*. Washington - Brussels - Tokyo: IEEE, Oct. 1997, pp. 2–9.

[23] J. Chandy and A. L. N. Reddy, "Failure evaluation of disk array organizations," in *Proceedings of the 13th International Conference on Distributed Computing Systems*, R. Werner, Ed. Pittsburgh, PA: IEEE Computer Society Press, May 1993, pp. 319–327.

[24] R. G. S. Kirkpatrick, W. Wilcke and H. Huels, "Percolation in dense storage arrays," in *Messina Symposium,*.

[25] F. Chang, M. Ji, S.-T. Leung, J. MacCormick, S. Perl, and L. Zhang, "Myriad: Cost-effective disaster tolerance," in *Proceedings of the FAST '02 Conference on File and Storage Technologies (FAST-02)*. Berkeley, CA: USENIX Association, Jan. 28–30 2002, pp. 103–116.

[26] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system," in *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, H. Jin, T. Cortes, and R. Buyya, Eds. New York, NY: IEEE Computer Society Press and Wiley, 2001, pp. 90–106.

[27] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao, "OceanStore: An architecture for global-scale persistent storage," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 190–201, Nov. 2000.

[28] H. Weatherspoon, M. Delco, and S. Zhuang, "Typhoon: An archival system for tolerating high degrees of file server failure," 1999, available through http://www.cs.berkeley.edu/~hweather/Typhoon/TyphoonReport.html.

[29] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson, "Failure correction techniques for large disk arrays," *Third Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, p. 123, Apr. 1989.

[30] L. Hellerstein, G. A. Gibson, R. M. Karp, R. H. Katz, and D. A. Patterson, "Coding techniques for handling failures in large disk arrays," *Algorithmica*, vol. 12, no. 2/3, pp. 182–208, Aug./Sept. 1994.

[31] Y. Chee, Colbourn, and Ling, "Asymptotically optimal erasure-resilient codes for large disk arrays," *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, vol. 102, 2000.

[32] D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, and J. Kubiatowicz, "Oceanstore: An extremely wide-area storage system,"

in *Proceedings of the Nine International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, Nov. 2000.

[33] M. Luby, "LT codes," in *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[34] A. Shokrollahi, "An introduction to low-density parity-check codes," *Lecture Notes in Computer Science*, vol. 2292, pp. 175–197, 2002.

[35] J. A. Cooley, J. L. Mineweaser, L. D. Servi, and E. T. Tsung, "Software-based erasure codes for scalable distributed storage," in *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03)*. IEEE, April 2003, p. 157.

[36] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory (special issue devoted to coding theory)*, pp. 1710–1722, 1996.

[37] D. Dubashi and D. Ranjan, "Balls and bins: A study in negative dependence," *BRICS*, 1996.

[38] T. E. Harris, "A lower bound for the critical probability in a certain percolation process," in *Proc. Cam. Phil. Soc.*, vol. 56, 1960, pp. 13–20.

[39] S. M. Ross, *Introduction to Probability Models*, 8th ed. Harcourt Publishers Ltd, Dec. 2002.

[40] B. A. Kozlov and I. A. Ushakov, *Reliability Handbook*. New York: Holt, Rinehart and Winston Inc., 1970.

[41] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system," *ACM Transactions on Computer Systems*, vol. 14, no. 1, pp. 108–136, Feb. 1996.