Reducing Repair-Bandwidth Using Codes Based on Factor Graphs

Dongwon Lee, *Student Member, IEEE*, Hyegyeong Park, *Student Member, IEEE,* and Jaekyun Moon, *Fellow, IEEE*

School of Electrical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, 305-338, Republic of Korea
Email: leedw1020@kaist.ac.kr, parkh@kaist.ac.kr, jmoon@kaist.edu

***Abstract*—Distributed storage systems suffer from significant repair traffic generated due to frequent storage node failures. This paper shows that properly designed low-density parity-check (LDPC) codes can substantially reduce the amount of required block downloads for repair thanks to the sparse nature of their factor graph representation. In particular, with a careful construction of the factor graph, both low repair-bandwidth and high reliability can be achieved for a given code rate. First, a formula for the average repair bandwidth of LDPC codes is developed. This formula is then used to establish that the minimum repair bandwidth can be achieved by forcing a regular check node degree in the factor graph. It is also shown that for a given repair-bandwidth overhead, LDPC codes can have substantially higher reliability than currently utilized Reed-Solomon (RS) codes. Our reliability analysis is based on a formulation of the general equation for the mean-time-to-data-loss (MTTDL) associated with LDPC codes. The formulation reveals that the stopping number is highly related to MTTDL. For code rates 1/2, 2/3, and 3/4, our results show that quasi-cyclic (QC) progressive-edge-growth (PEG) LDPC codes with variable node degree 2 allow $25\% \sim 50\%$ reduction in the repair bandwidth while maintaining higher MTTDL compared to currently employed RS codes.**

I. Introduction

Distributed storage has been introduced as a solution to storing and retrieving massive amounts of data. By using the MapReduce architecture [1], the distributed feature of recent storage systems enables data centers to store big data sets reliably while allowing scalability and offering high bandwidth. However, since distributed storage systems consist of commodity disks, failure events occur frequently. As a case in point, in the Google File System (GFS) "component failures are the norm rather than the exception" [2]. Simply replicating data multiple times can prevent data loss against the node failure events [2][3], but the associated costs in terms of storage overhead are rather high.

In order to reduce the large storage overhead of replication schemes, erasure codes have been introduced as alternatives [4]. Reed-Solomon (RS) codes [5] are typical erasure codes having the maximum distance separable (MDS) property that can tolerate the maximum number of erasures on the binary erasure channel (BEC) given a number of parity blocks. Typically, an (n, k) RS code splits a file to be stored into k blocks and encodes them into $n = k + m$ blocks including m parity blocks [6]. These n blocks of a code are referred to as a *stripe* in distributed storage. Any k out of n blocks can be used to reconstruct the original file, which is exactly how the MDS property is defined. In practice, a (14, 10) RS code is implemented on the Facebook clusters [6] whereas a (9, 6) RS code is used in the GFS [7]. Both of these codes have high storage efficiency as well as orders of magnitude higher reliability compared to 3-replication [4][8]. Hence, erasure coding schemes based on RS codes have become popular choices especially for archival storage systems where maintaining optimal tradeoff between data reliability and storage overhead is priority.

However, at issue is that MDS codes such as RS codes require high bandwidth overhead for the repair process. If a node failure event happens, the erased blocks need to be reconstructed in order to retain the same level of reliability; the amount of blocks to be downloaded for this repair task is defined as *repair bandwidth*. Since the repair bandwidth is a limited and expensive resource for data centers, bandwidth overhead generated from the repair job should be carefully managed. For a typical (n, k) RS code, k blocks are required to reconstruct a failed block whereas replication schemes need only one block. For instance, the (14, 10) RS code has a 10x repair bandwidth overhead relative to a replication scheme, consuming a significant amount of bandwidth during repair as confirmed by real measurements in the Facebook's clusters [6].

A number of recent publications have dealt with the repair bandwidth issues. Dimakis et al. [9] showed repair models of MDS codes for functional repair and exact repair. Whereas exact repair restores the failed blocks by generating blocks having exact copies of the data, functional repair generates blocks that can be different from the failed blocks as long as the MDS property is maintained. They established optimal storage-bandwidth tradeoff for functional repair and coined the term *regenerating codes* for the codes that achieve optimality in this sense. Many researchers have since designed regenerating codes for exact repair that operate in some specific environments [10][11]. In contrast to existing works, this paper focuses on low-repair-bandwidth schemes that have both low code redundancy and reasonable encoding/decoding complexity. Locally repairable codes (LRCs) and piggybacked-RS codes are known methods aiming at reducing repair bandwidth. LRCs are non-MDS codes that add local parity symbols to existing RS codes to reduce repair bandwidth at the expense of an increased parity overhead [7][12]. Rashmi et al. [6] suggested piggybacked-RS codes which can reduce the repair bandwidth of the RS codes without using extra storage but at the expense of code complexity.

This paper specifically shows that LDPC codes [13] provide benefits in terms of both repair bandwidth and reliability given the same storage overhead. Since a variable node of LDPC codes is connected to a relatively small number of nodes, LDPC codes have inherent local repair property as LRCs. The repair bandwidth of an LDPC code does not depend on the length of the code. The reliability typically gets better with an increasing code length. Thus, in the case of LDPC codes, the code length can be allowed to grow to achieve excellent reliability without worrying about expanding repair bandwidth as in RS codes. The only limiting factor in growing the code length in the LDPC codes is the computation and buffer requirements, but compared to the RS codes, the implementation complexity/buffer requirements of the LDPC codes grow considerably slower with code length.

The key contributions of this paper are as follows. The average repair bandwidth of the LDPC codes is formulated which leads to the observation that a regular check node degree achieves the minimum repair bandwidth given a fixed total number of edges in the factor graph. For reliability analysis, a general formula for MTTDL for LDPC codes is derived. The formula shows how the stopping numbers of codes directly affect reliability. It is confirmed that increasing the stopping number of the factor graph enhances reliability. Regular QC-PEG LDPC codes with different code rates have been designed and compared against representative RS codes and their variants. The results indicate significant repair-bandwidth and reliability advantages of properly designed LDPC codes.

The rest of this paper is organized as follows. In Section II and III, repair bandwidth analysis and reliability analysis of LDPC codes are given. Approaches to reduce repair bandwidth and to increase reliability are introduced as well. In Section IV, some specific examples of LDPC codes are discussed which show great performance on distributed storage. Simulation results that compare LDPC codes with other schemes are also given in this section. Finally, the paper draws conclusions in Section V.

II. Repair Bandwidth Analysis of LDPC codes

LDPC codes have been considered as an alternative for conventional distributed storage coding schemes. However, most known works in this area have been about reducing the coding overhead factor of LDPC codes rather than the repair bandwidth [14][15]. Whereas Wei et al. [16] showed a low latency of LDPC codes and suggested that LDPC codes may have low repair bandwidth, there has been no rigorous analysis for repair bandwidth.

In this section, the repair bandwidth of LDPC codes is described. LDPC codes are similar to LRCs regarding the repair process since the parity blocks of both codes are made locally from a small portion of the data blocks. In Fig. 1, the factor graph of an LDPC code is illustrated which consists of check nodes (squares), variable nodes (circles), and edges (lines between squares and circles) [17]. As shown in Fig. 1, if a block represented by node VN1 is erased, repair job can be done by downloading adjacent blocks VN2 and VN3 connected to the same check node CN1. This simple example demonstrates that LDPC codes can reconstruct erased data by using a relatively small number of blocks.

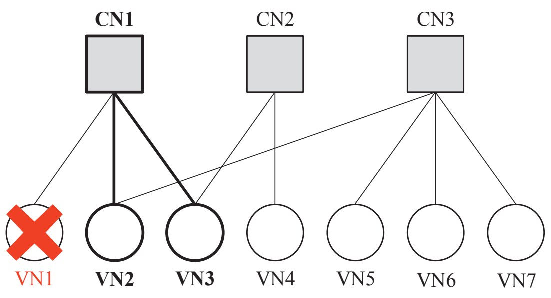

Fig. 1. A block erasure can be represented as a variable node erasure in factor graph. If a block is erased, the erased block can be recovered by downloading other blocks connected to the same check node. VN2 and VN3 are the blocks to be downloaded when VN1 fails.

When an erased block is connected to multiple check nodes, as is usually the case, we can choose a specific check node for repair. If the LDPC code is regular, any choice is equally good statistically. For an irregular LDPC code, however, the choice of a check node affects the amount of repair bandwidth since each check may have different degree. We thus define the bandwidth in average sense. If a variable node (VN) is erased, the repair bandwidth for that VN is the number of blocks downloaded averaged over all choices of check nodes the VN is connected to. Note that all other VNs are assumed intact in this definition. This value is then averaged over all VN erasure positions. This final average repair bandwidth is obtained by first considering all VNs connected to each check node (CN). For CN i with degree $d_{c,i}$, there are $d_{c,i}$ VNs attached to it, each of which will have a repair bandwidth of $d_{c,i}-1$, assuming the other VNs attached to CN i are downloaded for repair. The total repair bandwidth associated with CN i can be said to be equal to $d_{c,i}(d_{c,i}-1)$. Summing over all m CNs, we get $\sum_{i=1}^{m} d_{c,i}(d_{c,i}-1)$. To get to the per-VN repair bandwidth, we recognize that each VN is counted as many times as its node degree in the computation of $\sum_{i=1}^{m} d_{c,i}(d_{c,i}-1)$ since each VN is connected to multiple CNs in general. Thus, this sum should be divided by nd_v, where d_v is the average VN degree, to arrive at the per-VN average repair bandwidth we are looking for. But $nd_v = md_c = \sum_{i=1}^{m} d_{c,i}$, where d_c is the average CN degree. Note that nd_v also represents the total number of edges, E, in the factor graph. We establish a definition:

Definition: The average repair bandwidth or simply repair bandwidth of an LDPC code is defined as

$$RBW_{LDPC} = \frac{\sum_{i=1}^{m} d_{c,i}(d_{c,i}-1)}{E} \tag{1}$$

The following lemma subsequently tells us how the check node degrees should be distributed to minimize the average repair bandwidth of (1).

Lemma 1. *Given a fixed number E of edges on the factor graph, a regular check node degree minimizes the repair bandwidth of LDPC codes to $d_c - 1$.*

Proof. The repair bandwidth can be rewritten as

$$RBW = \frac{\sum_{i=1}^{m}(d_{c,i}-\frac{1}{2})^2 - \sum_{i=1}^{m}(\frac{1}{2})^2}{E}. \tag{2}$$

By using the Cauchy-Schwarz inequality, the choice $d_{c,1} = d_{c,2} = \cdots = d_{c,m} = E/m$ minimizes the average bandwidth. Thus, a regular check node degree minimizes the repair bandwidth and the corresponding minimum value is $RBW_{min} = d_c - 1$, one less than the CN degree. $\square$

Lemma 1 indicates that the LDPC code must be CN-regular in order to minimize the repair bandwidth. Since $E = nd_v = md_c$, for a CN-regular LDPC code we can write

$$RBW_{min} = d_c - 1 = \frac{d_v}{1-R} - 1 \tag{3}$$

where $R = \frac{k}{n} = \frac{n-m}{n}$.

From (3), it is clear that the repair bandwidth of the LDPC codes does not depend on the code length, but on d_c. This property makes the LDPC codes powerful options for distributed storage. Moreover, we see that for a given rate of the code, the average VN degree d_v must be made small to make the repair bandwidth small. For a fixed d_v, (3) also reveals an interesting relationship that RBW_{min} increases with increasing R, which is due to the fact that for a fixed d_v, increasing R must also mean increasing d_c.

In the sequel we assume that both VNs and CNs are regular and that d_v is fixed to 2. In comparing different coding schemes we consider three code rates: 1/2, 2/3 and 3/4.

III. Reliability Analysis of LDPC codes

A. The Mean Time to Data Loss

We provide reliability analysis for regular LDPC codes. In particular, we show that increasing the girth of the factor graph can enhance reliability. A Markov model is introduced to estimate reliability of coding schemes. Continuous-time Markov models have been used commonly to compare reliability of storage systems in terms of the MTTDL [8]. Unlike the bit-error-rate (BER) or the-word-error-rate (WER) performance metric, the MTTDL metric based on the Markov model considers the repair speed, which is our main interest in this paper.

Fig. 2 shows a Markov model example of the (14, 10) RS code [12]. The MTTDL is mainly influenced by the number of failures which can be tolerated before data loss as well as by the repair rate. Here, λ indicates the failure rate of a node and μ represents the repair rate of the nodes. Typically, $\mu \gg \lambda$ for storage applications. We can assume that each node fails independently at rate λ if the blocks are stored in different racks (physically separated storage units in data centers). Then, it is reasonable to ignore the possibility of burst failures. Also, the adoption of a continuous-time Markov model presupposes that only a single node failure is allowed at a given instance. Each state of the Markov model represents the number of erased blocks in a stripe. For the (14, 10) RS code, state 5 is the data loss (DL) state since five erasures in a stripe cannot be decoded. Whereas the failure rates depend on the state, the repair rates are all the same since the number of blocks to be downloaded for repair is always 10. The MTTDL can be obtained from this Markov model by calculating the mean arrival time to the DL state. The MTTDL of the MDS codes are well-established [8][18]. The MTTDL analysis for MDS codes can be modified and extended for the LDPC codes, as discussed next.

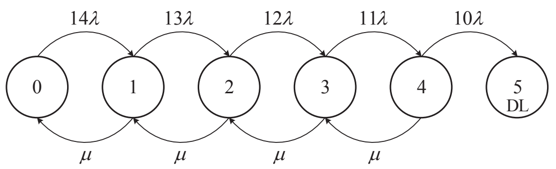

Fig. 2. Markov model of the (14, 10) RS code

B. MTTDL of LDPC codes

In this section, details of calculating the MTTDL for non-MDS codes are described. While the Markov model is already discussed for the LDPC codes in [19], the general formula for the MTTDL of the LDPC codes has not been given. We provide such a formula here. We also develop insights into how the MTTDL of the LDPC codes is affected by the stopping number.

Before presenting the Markov model of LDPC codes, some key terms are clarified. On factor graphs, the girth indicates the shortest cycle. A stopping set [20] is a subset of variable nodes such that all check nodes connected to it are connected by at least two edges, and the stopping number is the size of the smallest stopping set.

The derivation process is similar to that for MDS codes. However, as shown in Fig. 3, LDPC codes can directly go to the data loss state with only a small number of erasures. For instance in Fig. 1, if VN6 and VN7 fail, it is impossible to repair those nodes unlike in MDS codes. To model this behavior, probability parameters are introduced to the Markov model. Probability p_i is the conditional probability that a stripe of a given code can tolerate an additional node failure given state i. This means that the code has already survived from i failures and can tolerate one more failure with probability p_i. In general, LDPC codes are designed to guarantee $p_0 = 1$ and

$p_1 = 1$ since length-4 cycles are prohibited; however, other probabilities depend on the parity-check matrix of the code. If the parity-check matrix of the LDPC code is given, p_i values can be obtained by the relationship, $p_i = q_{i+1}/q_i$, where q_i denotes the unconditional probability that a given code can tolerate i failures [19]. These unconditional probabilities can be estimated by decoding simulation of LDPC codes on the erasure channel.

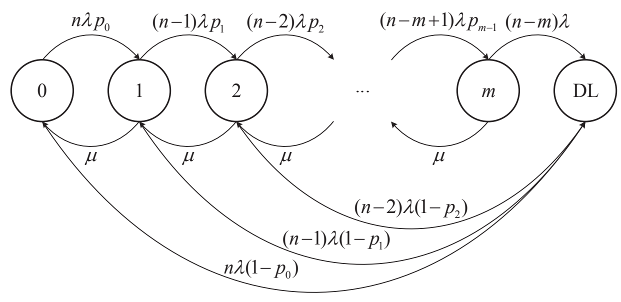

Fig. 3. Markov model of LDPC codes with m parity blocks

For m parity blocks (Fig. 3), the MTTDL equation is given by Lemma 2 below. We omit the proof due to lack of space.

Lemma 2. *For an arbitrary number m of the parity blocks and $\mu \gg \lambda$, the MTTDL of LDPC codes is:*

$$MTTDL \simeq \frac{\mu^m}{f(n, m, \lambda, \mu, p_0, \ldots, p_{m-1})} \tag{4}$$

where

$$\begin{aligned} &f(n, m, \lambda, \mu, p_0, \ldots, p_{m-1}) \\ &= n\lambda(1-p_0)\cdot\mu^m \\ &\quad + \sum_{j=1}^{m-1}[\{\prod_{i=0}^{j}(n-i)\lambda^{j+1}\}\cdot\{\prod_{i=0}^{j-1}p_i(1-p_j)\cdot\mu^{m-j}\}] \quad (5) \\ &\quad + \{\prod_{i=0}^{m}(n-i)\lambda^{m+1}\}\cdot\{\prod_{i=0}^{m-1}p_i\} \\ &= n\lambda(1-p_0)\cdot\mu^m + n\lambda\cdot(n-1)\lambda(1-p_1)p_0\cdot\mu^{m-1} \\ &\quad + n\lambda\cdot(n-1)\lambda\cdot(n-2)\lambda(1-p_2)p_0p_1\cdot\mu^{m-2} + \cdots \end{aligned} \tag{6}$$

From (6) it is seen that with all other parameters fixed, making p_i values large increases the MTTDL (decreases the denominator of the right-hand side of (4)). This observation is the key to designing factor graphs that enhance reliability. Especially, since p_0 is the coefficient of μ^m which is the most influential factor in the denominator, setting $p_0 = 1$ is critically important. Thus, setting as many p_i's for small i as possible to 1 is crucial to increase the MTTDL. It is already known that increasing the stopping number can drive more p_i's to 1 since the stopping number is the smallest number of erasures that cannot be corrected under iterative decoding. Therefore, increasing the stopping number of the factor graph can enhance the MTTDL. Proposition 1 establishes the relationship between the MTTDL and the stopping number. Again we omit the proof.

Proposition 1. *Define $s(\geq 2)$ be the stopping number of the given factor graph. For an arbitrary number m of the parity blocks and $\mu \gg \lambda$, the MTTDL for the LDPC codes is:*

$$MTTDL \simeq \frac{\mu^m}{\sum_{j=s}^{m-1}[\{\prod_{i=0}^{j}(n-i)\lambda^{j+1}\}\cdot\{\prod_{i=0}^{j-1}p_i(1-p_j)\cdot\mu^{m-j}\}]} \tag{7}$$

Especially, for the VN degree of 2, the stopping number is equal to $g/2$ where g is the girth of the graph [20]. As a result, to increase reliability of the regular LDPC codes with $d_v = 2$, the girth should be increased. This observation motivates LDPC code design by PEG, which is an effective search method for factor graphs with good girth properties.

IV. Simulation Results

From the repair bandwidth analysis in Section II, it is shown that a regular CN degree minimizes the average repair bandwidth of LDPC codes. For regular LDPC codes, it is also shown that $d_v = 2$ can minimize repair bandwidth overhead for a given code rate. In addition, from the MTTDL analysis in Section III, it is verified that regular LDPC codes with $d_v = 2$ should have large girth which helps to improve reliability. We shall focus on PEG-LDPC codes with $d_v = 2$ in this section. PEG is a well-known algorithm which can construct factor graphs having large girth [21]. However, a concern that may arise for setting $d_v = 2$ is a potentially poor erasure correction capability since each VN is protected by only two sets of checks. We first plot the data loss probability of a $d_v = 2$ regular LDPC code in Fig. 4. The results indicate that even short LDPC codes show erasure correction behavior similar to the 3-replication and (15, 10) RS codes asymptotically at low erasure probabilities.

Having ensured a reasonable erasure correction capability, the metrics considered for comparison are storage overhead (code rate inverse), repair bandwidth overhead and MTTDL. For the MTTDL simulation, the following normalized equation is used for fair comparison among codes having different lengths.

$$MTTDL = \frac{MTTDL_{stripe}}{C/nB} \tag{8}$$

where $MTTDL_{stripe}$ is the MTTDL given in Section III for a stripe. Here, the MTTDL for a stripe is normalized by the number of stripes, C/nB, in storage system. The parameters used for MTTDL simulation are given in Table I. These values are chosen according to the existing literature [7][12]. Note that for the repair rate, both the triggering time and the downloading time are included; the downloading time depends on the repair bandwidth (BW) overhead of the coding scheme.

For LDPC code simulations, using specific QC-PEG parity-check matrices, p_i's are first obtained from decoding simulation and the MTTDL values are calculated from (4). Table II

TABLE I
PARAMETERS USED FOR MTTDL SIMULATION

Parameter	Value	Description
C	40 PB	Total amounts of data
B	256 MB	Block size
N	2000	Number of disk nodes
S	20 TB	Storage capacity of a disk
r	1 Gbps	Network bandwidth on each node
$1/\lambda$	1 year	MTTF (mean-time-to-failure) of a node
μ	$\frac{1}{T_t+T_r}$	Repair rate
T_t	15 min	Detection and triggering time for repair
T_r	$\frac{S\cdot BW_{cost}}{r\cdot(N-1)}$	Downloading time of blocks
BW_{cost}		Repair BW overhead of the given code
n		Number of total coded blocks in a stripe
k		Number of data blocks in a stripe
m		Number of parity blocks in a stripe

TABLE II
PERFORMANCE OF QC-PEG LDPC CODES WITH $d_v = 2$, $R = 2/3$

Scheme	Storage overhead	Repair BW overhead	MTTDL (days)
3-replication	3x	1x	1.20E+3
(15, 10) RS	1.5x	10x	2.13E+10
(60, 40) LDPC	1.5x	5x	7.34E+5
(120, 80) LDPC	1.5x	5x	8.38E+7
(180, 120) LDPC	1.5x	5x	5.52E+9
(240, 160) LDPC	1.5x	5x	2.67E+11

shows performance of the QC-PEG LDPC codes with $d_v = 2$ for $R = 2/3$. Here, the (15, 10) RS code is chosen for comparison. For a given storage overhead, LDPC codes in Table II have a 5x repair bandwidth overhead, relative to replication, whereas the RS code has a 10x overhead. Thus, compared to the RS code, these LDPC codes require only one half of the repair bandwidth given the same storage overhead. Moreover, LDPC codes maintain the same repair bandwidth even as the code length is increased. So LDPC codes can get better MTTDLs than the (15, 10) RS code when longer codes are used. The table shows specifically that the (240, 160) LDPC code has better performance in terms of both repair bandwidth and MTTDL. This is at the expense of a longer code length. However, the complexity of erasure decoding of LDPC codes is quite reasonable for the code lengths discussed here.

For rates 3/4, 2/3 and 1/2, various coding schemes are compared in Fig. 5. Here we only consider codes that have higher MTTDLs than the (14, 10) RS code used in the Facebook cluster. The MTTDL of the (14, 10) RS codes is 1.61E+7. For the three storage overhead factors (code rate inverses), it is shown that LDPC codes have consistently better repair-bandwidth/storage-space tradeoffs compared to other codes. As the storage overhead is forced to decrease, LDPC codes enjoy a bigger performance gap relative to other codes with the exception of the LRC codes that perform similar to the LDPC codes as well.

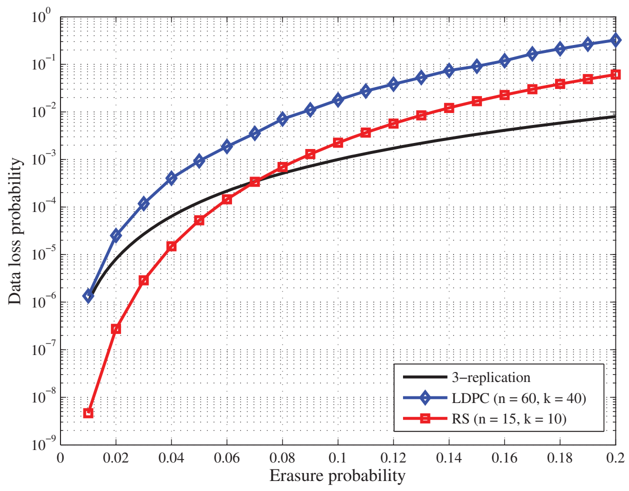

Fig. 4. Data loss probability of the (60, 40) LDPC code with $d_v = 2$ compared to the 3-replication and (15, 10) RS codes

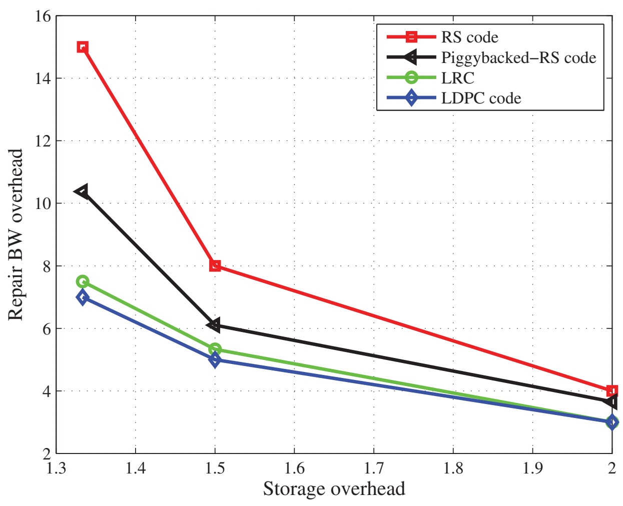

Fig. 5. Tradeoffs between repair bandwidth overhead and storage overhead for different codes. Coding schemes having higher reliability than the (14, 10) RS code are considered.

TABLE III
PARAMETERS OF CODES USED IN FIG.5. FOR LRC, (k, l, r) DENOTES THE NUMBER OF DATA BLOCKS, THE SIZE OF THE LOCAL GROUPS AND THE NUMBER OF GLOBAL PARITIES, RESPECTIVELY. FOR OTHER CODES, (n, k) DENOTES THE NUMBER OF CODE BLOCKS AND THE NUMBER OF DATA BLOCKS, RESPECTIVELY.

Scheme	Rate = 3/4	Rate = 2/3	Rate = 1/2
RS	(20, 15)	(12, 8)	(8, 4)
Piggybacked-RS	(20, 15)	(12, 8)	(8, 4)
LRC	(18, 3, 3)	(12, 3, 3)	(6, 3, 3)
LDPC	(240, 180)	(120, 80)	(56, 28)

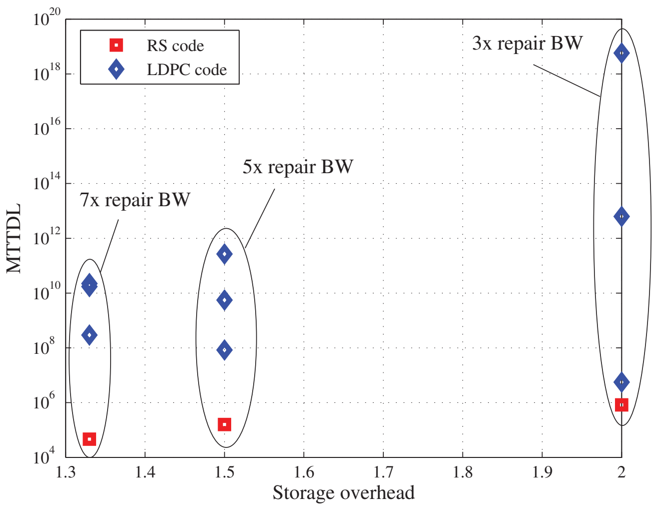

Fig. 6. MTTDL comparison of LDPC and RS codes under different storage overhead and repair bandwidth constraints

TABLE IV
PARAMETERS OF CODES USED IN FIG.6. LDPC1 REPRESENTS THE LDPC CODES WITH THE LOWEST MTTDLS.

Scheme	Rate = 3/4	Rate = 2/3	Rate = 1/2
RS	(10, 7)	(8, 5)	(6, 3)
LDPC1	(240, 180)	(120, 80)	(52, 26)
LDPC2	(280, 210)	(180, 120)	(60, 30)
LDPC3	(320, 240)	(240, 160)	(100, 50)

For given storage and repair bandwidth overheads, LDPC codes can achieve better MTTDL by increasing the code length, compared to the LRC and other codes. Fig. 6 shows such MTTDL comparison between the RS and LDPC codes, where for a given storage overhead, the MTTDL advantage of the LDPC codes is evident. Since the MTTDL of the LRC is known to be similar to that of the RS codes [7], LDPC codes will have definite reliability advantages over the LRCs.

While no analysis has been conducted for multiple erasures, we note that the repair bandwidth overhead of the LRC and piggybacked-RS codes approaches that of the RS code whereas the LDPC codes maintain the same overhead.

V. CONCLUSION

For distributed storage applications, this paper shows that LDPC codes can have great performance in terms of storage overhead, repair bandwidth and reliability. Unlike the RS code, the repair bandwidth of the LDPC codes does not increase with the code length. As a result, the LDPC codes can be designed to enjoy both low repair bandwidth and high reliability compared to the RS code and its known variants. It has been specifically shown that for a given number of edges in the factor graph, CN-regular LDPC codes minimize the repair bandwidth. The MTTDL analysis for LDPC codes have also been provided that relate the code's stopping set size with its MTTDL. Interesting future works include MTTDL analysis on VN-irregular LDPC codes as well as non-binary LDPC codes.

REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[4] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Peer-to-Peer Systems*. Springer, 2002, pp. 328–337.

[5] 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, 1960.

[6] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," *Proc. USENIX HotStorage*, 2013.

[7] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin *et al.*, "Erasure coding in windows azure storage." in *Usenix annual technical conference*. Boston, MA, 2012, pp. 15–26.

[8] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems." in *OSDI*, 2010, pp. 61–74.

[9] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[10] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *Information Theory, IEEE Transactions on*, vol. 57, no. 8, pp. 5227–5239, 2011.

[11] 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, 2013.

[12] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in *Proceedings of the VLDB Endowment*, vol. 6, no. 5. VLDB Endowment, 2013, pp. 325–336.

[13] R. G. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[14] J. S. Plank and M. G. Thomason, "A practical analysis of low-density parity-check erasure codes for wide-area storage applications," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 115–124.

[15] J. S. Plank, A. L. Buchsbaum, R. L. Collins, and M. G. Thomason, "Small parity-check erasure codes-exploration and observations," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005, pp. 326–335.

[16] Y. Wei, Y. W. Foo, K. C. Lim, and F. Chen, "The auto-configurable ldpc codes for distributed storage," in *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 1332–1338.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 150–159.

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

[19] J. L. Hafner and K. Rao, "Notes on reliability models for non-mds erasure codes," *IBM Res. rep. RJ10391*, 2006.

[20] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of tanner graphs," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*. IEEE, 2002, p. 2.

[21] Z. Li and B. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2. IEEE, 2004, pp. 1990–1994.