Enabling Concurrent Failure Recovery for Regenerating-Coding-Based Storage Systems: From Theory to Practice (Supplementary File)

Runhui Li, Jian Lin and Patrick P. C. Lee

1 PROOFS

We present the proofs of the theoretical results in Section 3.3 of the main file.

1.1 Proof of Theorem 1

We can formally build our proof based on the analysis of the information flow graph as in [1]. Here, we only show the main idea. Let d be the number of surviving nodes from which the relayer downloads data for recovery. Let β be the amount of data downloaded (per stripe) from each of the d surviving nodes to recover t failed nodes. We assume that the reconstructed data will be stored on t new nodes, which contain a total of $d\beta$ units of information.

We first consider $t < k$. Due to the MDS property, we can restore the original data from any k out of n nodes, each storing $\frac{M}{k}$ units of data. For example, we can select a set of any $k-\hat{t}$ originally surviving nodes (denoted by set $\mathcal{X}$) and a set of any $\hat{t}$ new nodes (denoted by set $\mathcal{Y}$) for some $\hat{t} \leq t$. The total amount of useful information must be at least M in order for the original data to be restorable. However, $\mathcal{Y}$ contains $(k-\hat{t})\beta$ units of information derived from $\mathcal{X}$. By excluding the redundant information, we require:

$$\frac{M}{k}(k-\hat{t}) + (d\beta - (k-\hat{t})\beta) \geq M, \text{ for any } \hat{t} \leq t.$$

The left side is minimum when $\hat{t} = t$. Thus, the recovery bandwidth (i.e., $d\beta$) must be at least $\frac{M \times d \times t}{k(d-k+t)}$. To minimize the recovery bandwidth with respect to d, we set $d = n - t$ and the result follows.

When $t \geq k$, any k out of the t new nodes must be able to restore the original data due to the MDS property. Thus, the t new nodes must contain M units of useful information, which can be reconstructed by downloading data from any k surviving nodes as in erasure codes. The recovery bandwidth is M.

• *R. Li, J. Lin and P. Lee are with the Chinese University of Hong Kong, Shatin, N.T., Hong Kong (emails: {rhli, jlin, pclee}@cse.cuhk.edu.hk)*

1.2 Proof of Theorem 2

Since MSR codes achieve the lower bound of recovery bandwidth for single failure recovery, the amount of data downloaded from *each* surviving node is $\frac{M}{k(n-k)}$ [1] (see Equation (1) of the main file).

Consider $t < k$. CORE in essence performs t single failure recoveries based on MSR codes, and in each recovery we actually download $\frac{M}{k(n-k)}$ units of data from each of the $n-t$ surviving nodes. If the failure pattern is good, then we can recover the virtual packets and hence the lost data. The lower bound is achieved for $t < k$. For $t \geq k$, we can simply download M units of data from any k surviving nodes and *any* failure pattern can be recovered. The result follows.

2 COMPARISONS WITH BASELINE MINIMUM STORAGE REGENERATING (MSR) CODES

Baseline MSR code constructions (e.g., [3], [4]) can recover a single failure by downloading encoded packets from less than $n-1$ surviving nodes at the expense of higher recovery bandwidth. They can also recover concurrent failures by recovering each failure individually. In the following, we argue that CORE still provides benefits over the baseline MSR codes.

For the baseline MSR code constructions, let d (where $k \leq d \leq n-1$) be the number of surviving nodes that are connected for single failure recovery. We can express the minimum recovery bandwidth (denoted by γ_{MSR}) as a function of d as follows [1]:

$$\gamma_{MSR} = \frac{Md}{k(d+1-k)}. \quad (1)$$

To recover from t failures (where $t \geq 1$), MSR codes can connect to $d \leq n-t$ surviving nodes and recover each failure individually. Since γ_{MSR} decreases with increasing d, we set $d = n-t$ to minimize the recovery bandwidth. Note that the total recovery bandwidth of recovering each failure individually may be higher than M. In this case, MSR codes can resort to conventional recovery to recover all t failures, so the recovery bandwidth is upper-bounded by M. Thus, the minimum

recovery bandwidth of the baseline MSR codes for t-failure recovery (denoted by $\gamma_{MSR}(t)$) can be expressed as:

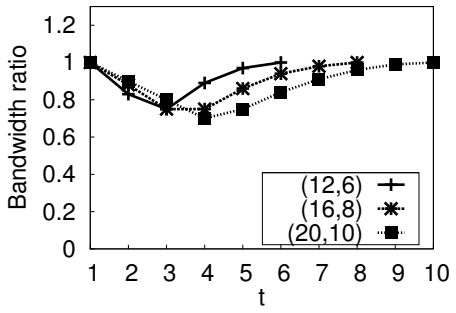
$$\gamma_{MSR}(t) = \min\left(\frac{M(n-t)t}{k(n-t+1-k)}, M\right). \quad (2)$$

We now compare CORE with the baseline MSR codes. Similar to Section 3.4 of the main file, we compare the bandwidth ratio, which we now define as the ratio of recovery bandwidth of CORE to the minimum recovery bandwidth of the baseline MSR codes. We vary (n, k) and the number t of failed nodes to be recovered. We also consider two cases of CORE: (i) recovering a good failure pattern and (ii) recovering a bad failure pattern.

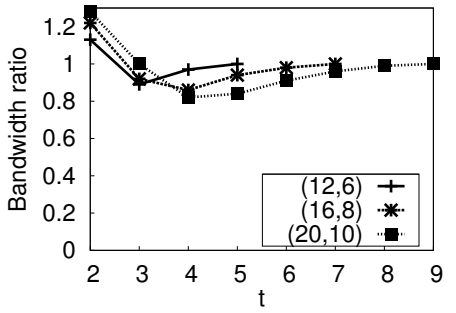
Figure 1(a) shows the bandwidth ratios for good failure patterns. For $t = 1$, both CORE and the baseline MSR codes perform recovery in the same way, so they have the same recovery bandwidth; for $t = k$, both CORE and the baseline MSR codes are the same as conventional recovery, and hence have the same recovery bandwidth as well. For $1 < t < k$, CORE achieves bandwidth savings over the baseline MSR codes, for example, by 10-17%, 20-25%, and 11-30% for $t = 2$, $t = 3$ and $t = 4$, respectively. The reasons are two-fold. First, CORE operates based on the MSR codes with $d = n - 1$, while the baseline MSR codes operate with $d = n - t$. A larger d implies lower recovery bandwidth (see Equation 1). Second, instead of recovering each failure individually, CORE recovers all failures concurrently, thereby avoiding redundant downloads.

Figure 1(b) shows the bandwidth ratios for bad failure patterns. We see that CORE has higher recovery bandwidth than the baseline MSR codes in the cases when t is small. For example, for $t = 2$, the recovery bandwidth of CORE is 13-28% higher than that of the baseline MSR codes. The reason is that in order to recover a bad t-failure pattern, CORE actually downloads data as in $(t+1)$-failure recovery. Nevertheless, CORE still achieves bandwidth savings for most cases, say by 3%-18% over the baseline MSR codes for $t = 4$. Considering that only less than 1.6% of failure patterns are bad failure patterns (see Section 3.2 of the main paper), CORE shows significant bandwidth savings over the baseline MSR codes in general.

It is worth noting that when constructing the baseline MSR codes, we must fix d in advance. However, the "best" value of d is difficult to determine, since it depends on the number t of failed nodes and t is a variable in practice. To elaborate, suppose that we fix $d = n - t^*$ for some constant t^*. If $t < t^*$, the recovery bandwidth is sub-optimal as we can further reduce the recovery bandwidth by setting $d = n - t$; if $t > t^*$, we perform conventional recovery, which is again sub-optimal. On the other hand, CORE operates on the MSR codes with $d = n - 1$, and supports the optimal recovery for a general number of failures in most cases (i.e., for good failure patterns).



(a) Good failure patterns



(b) Bad failure patterns

Fig. 1. Ratio of recovery bandwidth of CORE to that of the baseline MSR codes.

3 Comparisons of Immediate Recovery and Lazy Recovery

CORE allows a storage system to defer the immediate recovery of a single failure, but instead recover concurrent failures in batch. We now show that CORE reduces the recovery bandwidth of immediate recovery in general. [1]

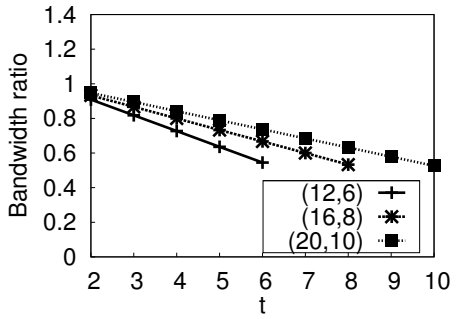
Suppose that t failures show up one by one. In immediate recovery, we recover each (single) failure using MSR codes with $d = n-1$, which achieves the minimum recovery bandwidth for single failure recovery. Thus, the total recovery bandwidth of immediate recovery for t failures (denoted by $\gamma_{immediate}$) is:

$$\gamma_{immediate} = t \times \frac{M(n-1)}{k(n-k)}. \quad (3)$$

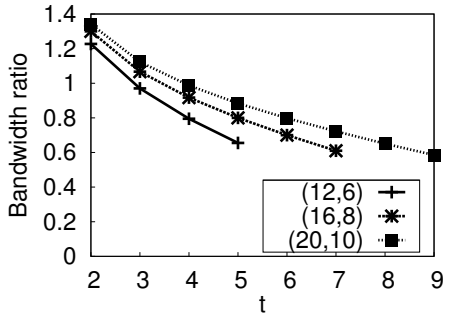
We now compare the recovery bandwidth of CORE with that of immediate recovery. We now define the bandwidth ratio as the ratio of the recovery bandwidth of CORE to that of immediate recovery. We vary (n, k) and the number t of failed nodes to be recovered. We again consider two cases of CORE: (i) recovering a good failure pattern and (ii) recovering a bad failure pattern.

Figure 2(a) shows the bandwidth ratios for good failure patterns. In all cases, CORE achieves bandwidth savings over immediate recovery, and the bandwidth

1. The authors of [2] show that cooperative MSR codes achieve the same recovery bandwidth in both immediate and deferred recoveries, but their definition of "recovery bandwidth" differs from ours as the former also accounts for the amount of data transferred among new nodes.



(a) Good failure patterns



(b) Bad failure patterns

Fig. 2. Ratio of recovery bandwidth of CORE to that of immediate recovery.

ratios drop (i.e., the savings increase) as t increases. For example, for (20,10), CORE achieves bandwidth savings by 5.3-47.4% for $2 \leq t \leq 10$.

Figure 2(b) shows the bandwidth ratios for bad failure patterns. For small t, CORE has higher recovery bandwidth (by up to 34.2%) than immediate recovery, mainly because the recovery of a t-node bad failure pattern is done by a $(t+1)$-failure recovery. For large t, CORE achieves bandwidth savings (by up to 41.5%) over immediate recovery. Given that bad failure patterns only account for a small fraction of all failure patterns, CORE achieves bandwidth savings over immediate recovery in general.

REFERENCES

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran. Network Coding for Distributed Storage Systems. *IEEE Trans. on Information Theory*, 56(9):4539–4551, Sep 2010.

[2] A. Kermarrec, N. Le Scouarnec, and G. Straub. Repairing Multiple Failures with Coordinated and Adaptive Regenerating Codes. In *Proc. of NetCod*, Jun 2011.

[3] K. Rashmi, N. Shah, and P. Kumar. Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction. *IEEE Trans. on Information Theory*, 57(8):5227–5239, Aug 2011.

[4] C. Suh and K. Ramchandran. Exact-Repair MDS Code Construction using Interference Alignment. *IEEE Trans. on Information Theory*, 57(3):1425–1442, Mar 2011.