Efficient Routing for Cooperative Data Regeneration in Heterogeneous Storage Networks

Zhirong Shen[†], Patrick P. C. Lee[‡], and Jiwu Shu[†*]
[†]Department of Computer Science and Technology, Tsinghua University
[‡]Department of Computer Science and Engineering, The Chinese University of Hong Kong
zhirong.shen2601@gmail.com, pclee@cse.cuhk.edu.hk, shujw@tsinghua.edu.cn
[*]Corresponding author: Jiwu Shu (*shujw@tsinghua.edu.cn*)

***Abstract*—Large-scale storage systems often face node failures that lead to data loss. Cooperative regeneration has been extensively studied to minimize the repair traffic of simultaneously reconstructing the lost data of multiple failed nodes. However, existing cooperative regeneration schemes assume that nodes are homogeneous. They do not consider how to minimize the general regenerating cost when taking into account node heterogeneity.**

This paper presents the first systematic study on enhancing conventional cooperation regeneration (CCR) schemes in a heterogeneous environment. We formulate cooperative regeneration as a cost-based routing optimization model, and propose a new cost-based *heterogeneity-aware cooperative regeneration* (HCR) framework. The main novelty of HCR is to decompose CCR schemes into two stages (i.e., expansion and aggregation) that can be opportunistically carried out by different nodes depending on their costs. To efficiently select the nodes for expansion execution without exhaustive enumeration, we design two greedy algorithms based on the hill-climbing technique. We also formulate the routing problem in the aggregation stage as a Steiner Tree Problem. Finally, we conduct extensive trace-driven simulations and show that HCR can reduce up to 75.4% transmission time of CCR. Also, we demonstrate that HCR remains robust even when the heterogeneity information is not accurately measured.

I. Introduction

Large-scale distributed storage systems are widely deployed nowadays either in wide-area storage systems (e.g., Cleversafe [1]) or in peer-to-peer storage systems (e.g., PAST [6]). They often build on off-the-shelf storage nodes in a decentralized manner. Thus, node failures are prevalent due to various kinds of unexpected errors, such as disk crashes or silent data corruptions, and eventually the stored data is permanently lost. To maintain data availability, a storage system maintains a degree of data redundancy across the nodes, using replication or erasure coding. Recently, regenerating codes [5] have been proposed to reduce the repair traffic (i.e., the amount of data transferred to regenerate lost data), so as to speed up failure recovery and minimize the window of vulnerability.

Given the increasing scale of storage systems, repairing *multiple node failures* becomes critical. First, node failures are often correlated and co-occurring, such as in clustered storage [7] or wide-area storage [18]. Also, to avoid unnecessary repairs, lazy regeneration [2] may be adopted to recover multiple node failures in batch only when the number of accumulated failed nodes reaches a toleration limit. In view of this, *cooperative regeneration* has been extensively studied by previous work (e.g., [9], [10], [14], [21], [22]). It extends regenerating codes [5] to allow the coded data to be exchanged among the new replacement nodes (see Section II-B).

However, cooperative regeneration becomes complicated in a heterogeneous environment where nodes have different costs (e.g., monetary cost and transmission time) during regeneration. Node heterogeneity is common in practical storage systems due to regular upgrades of system components [25] or variance in communication bandwidths [4]. On the other hand, current studies on cooperative regeneration [9], [13], [14], [21], [22] mainly focus on the *star-structured* network topology with homogeneous connections, i.e., every new replacement node needs to *directly* receive data from the surviving nodes, and the new replacement nodes *locally* regenerate the lost data. As we show in this paper, the performance of cooperative regeneration remains unsatisfactory in a heterogeneous environment.

In this paper, we present the *first* systematic study on cooperative regeneration in heterogeneous distributed storage systems, and propose a novel cost-based *heterogeneity-aware cooperative regeneration* (HCR) framework. Specifically, we formulate a routing optimization problem that aims to minimize the regeneration cost of cooperative regeneration in a heterogeneous environment, where the *regeneration cost* can be defined as the total time or monetary cost incurred for the regeneration. The key novelty of HCR is to decompose the encoding operation for data regeneration on the new replacement node, which is actually the multiplication between a vector and a matrix, into two stages (i.e., expansion and aggregation), such that the regeneration cost can be further cut down by opportunistically executing these stages on appropriate nodes based on their costs. Note that HCR is applicable for general cooperative regeneration schemes.

To speed up the execution of HCR without exhaustive enumeration, we propose two greedy algorithms based on the hill-climbing (greedy) technique [20] to find the appropriate nodes and perform the expansion stage, so as to timely determine the near-optimal selection through iteratively replacing the currently chosen nodes with those that achieve less regeneration cost. We further formulate the routing problem in the aggregation stage as the Steiner Tree Problem [11], such that the optimal aggregation routing can be solved by finding the Steiner Minimal Tree [11].

Our contributions can be summarized as follows:

- We formulate a routing optimization problem for the cost-based cooperative regeneration in a heterogeneous scenario, and accordingly propose a new regeneration framework that separates the procedure of existing cooperative regeneration into two stages, i.e., expansion and aggregation.
- By considering node heterogeneity, we propose two greedy algorithms that can timely select the near-optimal solution to perform the expansion stage. We next prove that the establishment of aggregation routing can be formulated as the Steiner Tree Problem.
- We conduct a series of trace-driven simulations and demonstrate that HCR reduces up to 75.4% of transmission time of CCR. Compared with the brute-force enumeration, HCR is much more efficient while achieving near optimality. Moreover, HCR remains robust in regeneration cost reduction even when the information of heterogeneity cannot be accurately measured.

The remainder of this paper proceeds as follows. In Section II, we present the background of cooperative regeneration. In Section III, we formulate our new regeneration framework. In Section IV, we describe our HCR design. In Section V, we present the evaluation results. Finally, in Section VI, we conclude the paper.

II. Background and Related Work

A. Basics

We consider a distributed storage system composed of multiple storage nodes. To maintain data availability in the presence of node failures, a storage system maintains a degree of data redundancy across the nodes. Suppose that node failures happen, a storage system then introduces replacement nodes termed *"newcomers"* to the system. These newcomers then receive data from other surviving nodes termed *"providers"* and reconstruct the lost data.

Erasure coding is a redundancy approach that is well adopted in today's distributed storage systems. Specifically, we configure two parameters n and k (where $k < n$) for erasure coding. An (n, k) erasure coding scheme partitions a file of size F into k equal-size *chunks* of size $\frac{F}{k}$ each and encodes them into n coded chunks, such that any k out of n coded chunks suffices to reconstruct the original file. To recover the lost data of a failed node, conventional erasure coding sends the data with the same size of original file from the providers to the newcomer for data reconstruction [5]. We call this amount of data transferred for recovery to be *"repair traffic"*.

To balance between the storage efficiency and repair traffic, regenerating codes [5] allow the providers to have the computational capability to encode the stored data and send the encoded data, so that the repair traffic can be significantly reduced. Specifically, each provider divides a chunk into multiple pieces called *"packets"*. It then encodes the packets, say by linear combinations, and sends the encoded data for regeneration. Two specific optimal regenerating codes are proposed: *Minimum-Storage Regenerating (MSR)* codes [5], [23], which maintain the minimum storage redundancy while minimizing the repair traffic, and *Minimum-Bandwidth Regenerating (MBR)* codes [19], which allow more storage redundancy so as to further minimize the repair traffic.

B. Cooperative Regeneration

Regenerating codes are first proposed for minimizing the repair traffic for a single node failure. Given the prevalence of multiple node failures [7], [18], a new regeneration method to cope with multiple node failures is critical.

Hu et al. [9] investigate the regeneration for r ($r \geq 1$) nodes that fail concurrently. They propose a new regeneration approach called *cooperative regeneration*, which allows the newcomers to exchange the received information during the regeneration. It significantly decreases the repair traffic than the way that individually applies the traditional regeneration r times. As a variant of cooperative regeneration, the pipelined regeneration [13] reduces the number of involved nodes during the regeneration. The problem of cooperative pipelined regeneration scheme is also studied in [14]. Previous studies mainly realize *functional repair*, i.e., the reconstructed data may not be consistent with the lost data. Shum et al. [21] propose an *exact repair* solution that regenerates exactly the data lost. Li et al. [12] further develop a construction that *exactly* regenerates the data on any *two* newcomers based on exact MSR codes. Li et al. [17] design and implement a framework called CORE, which extends existing regenerating code constructions for single node failure recovery to support exact repair for multiple failures. However, the above studies do not consider node heterogeneity.

C. Heterogeneity

Heterogeneous regeneration takes into account the differences in the deployment environment, such as bandwidth, storage space of nodes, and the network topology. Li et al. [15] propose a tree-structured regeneration scheme to accelerate the regeneration process by seeking the regeneration tree with the maximum bounded bandwidth. However, this method may hurt data integrity because of transmitting insufficient data [24]. Aiming at this shortcoming, Wang et al. [24] suggest retrieving different amounts of data from different providers based on their heterogeneous bandwidths, so as to minimize the regeneration time. Zhang et al. [26] investigate the application of erasure codes in data centers with different network topologies, and consider the reduction on the amount of transmitted data during node reconstruction. However, previous studies still pay limited attention to the cooperative regeneration in a heterogeneous environment.

III. Problem Formulation

In this section, we first define the system framework of *conventional cooperative regeneration* (CCR), which covers many existing schemes (e.g., [9], [13], [14], [21]), either for the schemes (like [9]) with functional repair or the schemes (like [21]) with exact repair. We then propose a new framework

TABLE I
MAIN NOTATION IN THIS PAPER.

Notation	Descriptions
Defined in Section II	
(n, k)	erasure coding parameters ($k < n$)
Defined in Section III	
s	number of packets stored on a node
r	number of newcomers in regeneration
d	number of contacted providers in regeneration ($k \leq d \leq n-r$)
$\mathcal{X}$	set of possible providers $\{x_1, x_2, \cdots, x_{n-r}\}$
$\mathcal{Y}$	set of newcomers $\{y_1, y_2, \cdots, y_r\}$
$\beta_{h,i}$	retrieved packet from provider x_h to newcomer y_i ($1 \leq h \leq n-r$, $1 \leq i \leq r$)
$\beta'_{i,j}$	exchanged packet from newcomer y_i to newcomer y_j ($1 \leq i \neq j \leq r$)
$\mathbf{t}_{i,j}$	$d \times 1$ column vector of newcomer y_i to produce $\beta'_{i,j}$ ($1 \leq i \neq j \leq r$)
$\mathbf{M}_i$	$(d+r-1) \times s$ encoding matrix of newcomer y_i ($1 \leq i \leq r$)
$\mathcal{G}$	modeled graph
$\mathcal{V}$	set of vertices in $\mathcal{G}$, $\mathcal{V} = \mathcal{X} \cup \mathcal{Y}$
$\mathcal{E}$	set of edges in $\mathcal{G}$
$\mathcal{U}$	set of all valid regeneration routings
$w(v_i, v_j)$	weight function of an edge from v_i to v_j ($v_i, v_j \in \mathcal{V}$)
$\mathbf{e}_i$	a vector of all collected packets of newcomer y_i ($1 \leq i \leq r$)
$\mathbf{p}_i$	a vector of all retrieved packets of y_i
$\mathbf{q}_i$	a vector of all exchanged packets of y_i

based on CCR for a heterogeneous environment and formulate a routing optimization problem. Table I lists the main notation used in this paper.

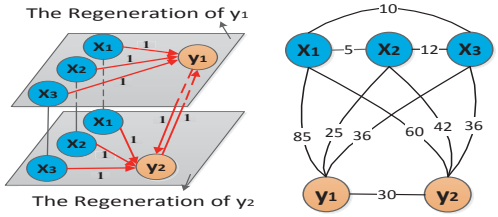
A. Framework of CCR

When the number of failed nodes reaches r in a system with n nodes, cooperative regeneration begins by introducing r newcomers for data regeneration. Without loss of generality, we let the newcomers be $\mathcal{Y} = \{y_1, \cdots, y_r\}$, the possible providers are $\mathcal{X} = \{x_1, \cdots, x_{n-r}\}$, and thus the nodes in the new storage systems are $\mathcal{V} = \{v_1, \cdots, v_n\} = \mathcal{X} \cup \mathcal{Y}$.

Preparation Step: Every file is first partitioned into k chunks, which are then encoded into n chunks ($n \geq k$) using (n, k) erasure coding. Each chunk is further partitioned into s packets and stored on a node, where the value of s and other coefficients in this framework depend on the specific regeneration scheme. For example, the scheme in [9] defines $s = n - k$ and $d = n - r$.

Retrieval Step: Suppose that there are $r \geq 1$ node failures. Each of the r newcomers retrieves packets from d providers ($d \leq n - r$). Without loss of generality, we assume every newcomer contacts the providers $\{x_1, \cdots, x_d\}$ for packet retrieval. Each requested provider then delivers a packet to the newcomer by encoding the stored s packets through a linear combination. We call this kind of packets retrieved from providers as *"retrieved packets"* and let $\beta_{i,j}$ denote the retrieved packet sent from the provider x_i ($1 \leq i \leq d$) to the newcomer y_j ($1 \leq j \leq r$). After this stage, each newcomer will obtain d retrieved packets.

Exchange Step: For every pair of newcomers y_i and y_j ($1 \leq i \neq j \leq r$), if y_i wants to exchange packets with y_j, it will encode its d retrieved packets into one coded packet by multiplying d retrieved packets with an exchanged vector $\mathbf{t}_{i,j}$, where $\mathbf{t}_{i,j}$ is a $d \times 1$ column vector. We call this kind of packets as *"exchanged packets"* and let $\beta'_{i,j}$ denote the exchanged packet issued from y_i to y_j ($1 \leq i \neq j \leq r$). After the exchange stage, each of the r newcomers receives $(r-1)$ exchanged packets.



(a) CCR Framework. (b) The Heterogeneity.

Fig. 1. The cost of CCR framework in a heterogeneous environment. The arrow shows the number of transmitted packets. The total cost is 344.

Encoding Step: Finally y_i encodes all the collected $(d + r - 1)$ packets $\mathbf{e}_i = \begin{pmatrix} \mathbf{p}_i & \mathbf{q}_i \end{pmatrix}$, where $\mathbf{p}_i = \begin{pmatrix} \beta_{1,i} & \cdots & \beta_{d,i} \end{pmatrix}$ and $\mathbf{q}_i = \begin{pmatrix} \beta'_{1,i} & \cdots & \beta'_{i-1,i} & \beta'_{i+1,i} & \cdots & \beta'_{r,i} \end{pmatrix}$, into s packets by multiplying $\mathbf{e}_i$ with the encoding matrix $\mathbf{M}_i$ whose size is $(d + r - 1) \times s$.

B. Limitations of CCR

In a heterogeneous environment, the direct deployment of CCR may cause a considerable amount of cost. Figure 1 illustrates an example, where the packet size is set as 1. Then the cost of CCR framework in this scenario is 344.

The CCR framework has three rigid requirements that restrict it to achieve further reduction on the regeneration cost. First, newcomers should collect all the packets in preparation for encoding operation. Second, the whole of Encoding Step is forced to be *fully* executed on the newcomers. Third, CCR ignores the heterogeneity among connections and the data is delivered via the direct connection between the source and destination nodes. Based on these restrictions, we thus pose the following question: *Can we explore the regeneration cost reduction by relaxing these constraints?*

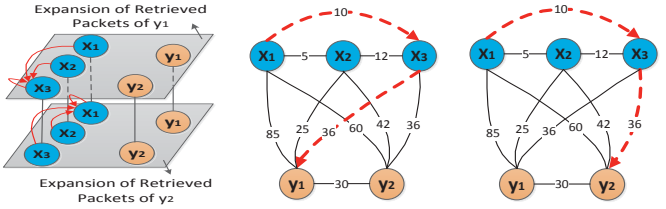
C. Motivation

The framework of CCR indicates that the regenerated packets of each newcomer are generated by performing linear combinations to all the retrieved packets. Motivated by this finding, we can decompose the linear combinations into several sub-operations, select appropriate nodes for execution according to the heterogeneity, and establish the regeneration routing, such that the regeneration cost can be reduced. Holding this motivation, we have the following toy example.

A Toy Example: Based on the heterogeneous environment in Figure 1(b), suppose the system adopts the cooperative regeneration scheme in [9], where $n = 5$, $k = 3$, $r = 2$, $d = 3$, $s = 2$. A regeneration routing under the new framework is shown in Figure 2, where the packet size is set as 1.

In this example, the retrieved packets of y_1 (i.e., $\mathbf{p}_1 = \begin{pmatrix} \beta_{1,1} & \beta_{2,1} & \beta_{3,1} \end{pmatrix}$) will be first sent to the node x_3 and the retrieved packets of y_2 (i.e., $\mathbf{p}_2 = \begin{pmatrix} \beta_{1,2} & \beta_{2,2} & \beta_{3,2} \end{pmatrix}$) will be delivered to x_1 (shown in Figure 2(a)).

We next describe how to regenerate the data on y_1 in the new regeneration framework (shown in Figure 2(b)). The



(a) Expansion: the cost is 37. Each link will transmit one packet.

(b) Aggregation of y_1: the cost is 92. Each link will transmit two packets.

(c) Aggregation of y_2: the cost is 92. Each link will transmit two packets.

Fig. 2. A toy example to regeneration y_1 and y_2 in the new regeneration framework ($n = 5$, $k = 3$, $r = 2$, $d = 3$, $s = 2$).

definitions of symbols can be reviewed in Table I. Suppose the encoding matrix $\mathbf{M}_1$ (with the size $(d + r - 1) \times s$) and the exchanged vector $\mathbf{t}_{2,1}$ (with the size $d \times 1$) are as follows.

$$\mathbf{M}_1 = \begin{pmatrix} m_{1,1} & m_{1,2} \\ \vdots & \vdots \\ m_{4,1} & m_{4,2} \end{pmatrix}, \mathbf{t}_{2,1} = \begin{pmatrix} t_{2,1,1} \\ t_{2,1,2} \\ t_{2,1,3} \end{pmatrix}$$

Stage 1: Each retrieved packets of y_2 (i.e., $\beta_{i,2}$ for $1 \le i \le 3$) can be "expanded" by performing the following calculation according to the Exchanged Step and Encoding Step in Section III-A.

$$\beta_{i,2} \cdot t_{2,1,i} \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix}$$

This expansion produces two packets (i.e., $\beta_{i,2} \cdot t_{2,1,i} \cdot m_{4,1}$ and $\beta_{i,2} \cdot t_{2,1,i} \cdot m_{4,2}$).

At the same time, each retrieved packets of y_1 (i.e., $\beta_{i,1}$ for $1 \le i \le 3$) resided on x_3 can also be expanded to two packets by running

$$\beta_{i,1} \cdot \begin{pmatrix} m_{i,1} & m_{i,2} \end{pmatrix}$$

Stage 2: In this stage, one has to add the packets that are relevant to the regenerated data on y_1. The expanded packets of $\beta_{i,2}$ $(1 \le i \le 3)$ on x_1 will be aggregated as follows.

$$\begin{aligned} (\sum_{i=1}^{3} \beta_{i,2} \cdot t_{2,1,i}) \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix} &= \mathbf{p}_2 \cdot \mathbf{t}_{2,1} \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix} \\ &= \beta'_{2,1} \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix} \end{aligned}$$

This result is composed of two packets and will be sent to x_3 for further aggregation (see Figure 2(b)). At the same time, the expanded packets of $\beta_{i,1}$ $(1 \le i \le 3)$ on x_3 will be aggregated as follows.

$$\sum_{i=1}^{3} (\beta_{i,1} \cdot \begin{pmatrix} m_{i,1} & m_{i,2} \end{pmatrix}) = \mathbf{p}_1 \cdot \begin{pmatrix} m_{1,1} & m_{1,2} \\ \vdots & \vdots \\ m_{3,1} & m_{3,2} \end{pmatrix}$$

When receiving $\beta'_{2,1} \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix}$, we can regenerate the data on y_1 by running

$$\mathbf{p}_1 \cdot \begin{pmatrix} m_{1,1} & m_{1,2} \\ \vdots & \vdots \\ m_{3,1} & m_{3,2} \end{pmatrix} + \beta'_{2,1} \cdot \begin{pmatrix} m_{4,1} & m_{4,2} \end{pmatrix} = \mathbf{e}_1 \cdot \mathbf{M}_1$$

As described in the Encoding Step of Section III-A, $\mathbf{e}_1 \cdot \mathbf{M}_1$ is the data to be regenerated on y_1, and the produced two packets will then be delivered to y_1. The regeneration of y_2 is similar (shown in Figure 2(c)).

We then analyze the regeneration cost of this routing established by the new regeneration framework. The routing in Figure 2(a) involves the edges $\{E(x_1, x_3), E(x_2, x_3), E(x_2, x_1), E(x_3, x_1)\}$, each of which will deliver 1 packet. Therefore, the regeneration cost in this stage is $C_1 = 37$. In the routing in Figure 2(b) and Figure 2(c) involves edges $\{E(x_1, x_3), E(x_3, y_1), E(x_3, y_2)\}$, where $E(x_1, x_3)$ transmits 4 packets and other two connections deliver 2 packets. Thus, the regeneration cost in this stage is $C_2 = 184$.

Finally, the regeneration cost of y_1 and y_2 will be $C_1 + C_2 = 221$, which is much less than that of CCR whose regeneration cost is 344 as shown in Figure 1.

D. A New Cooperative Regeneration Framework

Motivated by the above example, we present a new cooperative regeneration framework for a heterogeneous environment in the theoretical way. The new framework is driven by two observations.

Observation 1: *Every regenerated packet in each newcomer can be expressed as a linear combination of all the retrieved packets of r newcomers.*

In the Encoding Step of Section III-A, the newcomer y_i $(1 \le i \le r)$ finally collects packets $\mathbf{e}_i$ that is composed of the retrieved packets $\mathbf{p}_i$ and the exchanged packets $\mathbf{q}_i$, and regenerates the data $\mathbf{e}_i \cdot \mathbf{M}_i$, where $\mathbf{p}_i = \begin{pmatrix} \beta_{1,i} & \cdots & \beta_{d,i} \end{pmatrix}$, $\mathbf{q}_i = \begin{pmatrix} \beta'_{1,i} & \cdots & \beta'_{i-1,i} & \beta'_{i+1,i} & \cdots & \beta'_{r,i} \end{pmatrix}$, and

$$\mathbf{M}_i = \begin{pmatrix} \mathbf{m}_{i,1} \\ \vdots \\ \mathbf{m}_{i,d+r-1} \end{pmatrix} \tag{1}$$

Notice that $\beta'_{h,i} = \mathbf{p}_h \mathbf{t}_{h,i}$, where $\mathbf{p}_h$ is the retrieved packets of the newcomer y_h $(1 \le h \ne i \le r)$ and $\mathbf{t}_{h,i}$ is a $d \times 1$ exchanged vector (see the Exchanged Step of Section III-A). $\mathbf{m}_{i,j}$ denotes the j-th row vector of $\mathbf{M}_i$ $(1 \le j \le d + r - 1)$.

We can divide the encoding matrix $\mathbf{M}_i$ into a $d \times s$ submatrix $\mathbf{M}_{i_1}$, and a $(r - 1) \times s$ submatrix $\mathbf{M}_{i_2}$. Then the regenerated packets on y_i are

$$\begin{aligned} \mathbf{e}_i \mathbf{M}_i &= \begin{pmatrix} \mathbf{p}_i & \mathbf{q}_i \end{pmatrix} \begin{pmatrix} \mathbf{M}_{i_1} \\ \mathbf{M}_{i_2} \end{pmatrix} \\ &= \mathbf{p}_i \mathbf{M}_{i_1} + \mathbf{q}_i \mathbf{M}_{i_2} \\ &= \mathbf{p}_i \mathbf{M}_{i_1} + \begin{pmatrix} \beta'_{1,i} \cdots \beta'_{i-1,i} \, \beta'_{i+1,i} \cdots \beta'_{r,i} \end{pmatrix} \mathbf{M}_{i_2} \\ &= \mathbf{p}_i \mathbf{M}_{i_1} + \sum_{1 \le h < i} \beta'_{h,i} \mathbf{m}_{i,d+h} + \sum_{i < h \le r} \beta'_{h,i} \mathbf{m}_{i,d+h-1} \\ &= \mathbf{p}_i \mathbf{M}_{i_1} + \sum_{1 \le h \ne i \le r} \mathbf{p}_h \mathbf{Z}_{h,i} \end{aligned} \tag{2}$$

$$\text{where} \quad \mathbf{Z}_{h,i} = \begin{cases} \mathbf{t}_{h,i} \mathbf{m}_{i,d+h} & 1 \le h < i \\ \mathbf{t}_{h,i} \mathbf{m}_{i,d+h-1} & i < h \le r \end{cases} \tag{3}$$

Like $\mathbf{M}_{i_1}$, $\mathbf{Z}_{h,i}$ is also a matrix whose size is $d \times s$. Therefore, Equation (2) indicates that the regenerated data $\mathbf{e}_i \mathbf{M}_i$ on y_i

is actually linear combinations of all newcomers' retrieved packets $\{\mathbf{p}_1, \cdots, \mathbf{p}_r\}$ realized by the linear multiplications between r vectors and r matrices. Specifically, to regenerate the data on y_i, the retrieved packets of y_i (i.e., $\mathbf{p}_i$) should multiply with $\mathbf{M}_{i_1}$, while other newcomer's retrieved packets (i.e., $\mathbf{p}_h$ for $1 \leq h \neq i \leq r$) should multiply with $\mathbf{Z}_{h,i}$.

Observation 2: *A linear multiplication between a vector and a matrix can be partitioned into two sub-operations.*

We take $\mathbf{p}_i$ and $\mathbf{M}_{i_1}$ in Equation (2) as an example, where

$$\mathbf{p}_i = \begin{pmatrix} \beta_{1,i} & \cdots & \beta_{d,i} \end{pmatrix}, \mathbf{M}_{i_1} = \begin{pmatrix} \mathbf{m}_{i,1} \\ \vdots \\ \mathbf{m}_{i,d} \end{pmatrix}_{d \times s} \quad (4)$$

To calculate the multiplication $\mathbf{p}_i \cdot \mathbf{M}_{i_1}$, we can first *expand* every packet $\beta_{j,i}$ $(1 \leq j \leq d)$ to a vector called "*packet vector*" with s values by multiplying $\beta_{j,i}$ with every value of $\mathbf{m}_{i,j}$, and produce $\beta_{j,i}\mathbf{m}_{i,j}$. We call this sub-operation as "*expansion*".

After the expansion, we can perform another sub-operation called "*aggregation*" by adding the produced d vectors and finally obtain $\sum_{j=1}^{d} \beta_{j,i}\mathbf{m}_{i,j} = \mathbf{p}_i \cdot \mathbf{M}_{i_1}$.

Based on the above observations, we propose a novel cost-based *heterogeneity-aware cooperative regeneration* (HCR) framework, which has two key properties. First, we can partition the linear multiplications performed on the newcomer into the expansion and aggregation stages. Second, we can select appropriate nodes to execute the two stages according to the node heterogeneity, so as to decrease the regeneration cost. The expansion and aggregation stages are described as follows.

1) ***Expansion:*** This stage will let each retrieved packet of r newcomers appoint a node (called "expansion node") to perform the expansion operation to assist the regenerations of r newcomers. The delivery will always choose the *shortest path* from the providers to the expansion nodes.

For a retrieved packet, it will be expanded in different ways in different newcomers' regenerations. To regenerate y_i $(1 \leq i \leq r)$, its retrieved packets $\mathbf{p}_i$ will be expanded based on $\mathbf{M}_{i_1}$ (referred to Equation (2)). Once to regenerate y_j $(1 \leq j \neq i \leq r)$, $\mathbf{p}_i$ should be expanded based on $\mathbf{Z}_{i,j}$ (referred to Equation (2)). Therefore, the aggregation stage of each newcomer is *independent*.

2) ***Aggregation:*** In this stage, each newcomer y_i $(1 \leq i \leq r)$ will establish an aggregation routing connecting all the expansion nodes and regenerating the data on y_i. A node is treated to be farther from the newcomer if it should go through more nodes along the routing to reach the newcomer. The aggregation principle is that a node will receive the packet vectors from its connected nodes that are much farther from the newcomer, combine the received packet vectors, and send the result to the next node that is closer to the newcomer along the routing. For example, Figure 2(b) shows an aggregation routing of the newcomer y_1, where x_1 is much farther from y_1 compared with x_3 and it should send the packet vector to x_3 for packet aggregation. The final aggregation result of y_i will correctly output $\mathbf{e}_i\mathbf{M}_i$.

Summary: From above descriptions, we can observe that for any scheme covered by the CCR framework, it can also be realized under the new regeneration framework without any modification. The difference lies in that the new regeneration framework separately executes the Encoding Step based on the node heterogeneity.

The new regeneration framework lets the intermediate node along the regeneration routing to assist in the regeneration. It is more scalable than CCR, in which all the packets are processed by newcomers only. Besides, the new regeneration model does not restrict the parallelism in the regeneration, as the computations and data transfers on the nodes that assist in the regeneration can be executed in a pipelined manner.

E. Optimization Regeneration Routing Model

Given the new regeneration framework, a subsequent question is how to find the regeneration routing to achieve the minimal cost. Specifically, a regeneration routing should include the selected providers, the selected expansion nodes, and the aggregation routing. We can formulate this problem as follows.

Suppose $\mathcal{V} = \{v_1, \cdots, v_n\}$ denotes the set of nodes and $\mathcal{E} = \{E(v_i, v_j) | v_i, v_j \in \mathcal{V}\}$ is the set of connections, where $E(v_i, v_j)$ is the connection between v_i and v_j. Then the system topology is modeled as a complete connected undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$, where w is the weight function that maps $\mathcal{E}$ to a set of nonnegative numbers. A weight value $w(v_i, v_j)$ denotes the cost to transmit a unit of data (e.g., byte) via $E(v_i, v_j)$.

For a cooperative regeneration scheme (e.g., [9], [13], [21]), suppose $U(\mathcal{V}_U, \mathcal{E}_U, w, l_U)$ is a regeneration routing under the new framework, where $\mathcal{V}_U$ and $\mathcal{E}_U$ are the involved vertices and edges in U, respectively. Let l_U be a function that records the size of delivered data through $\mathcal{E}_U$ in U. For example, $l_U(v_i, v_j)$ denotes the size of transmitted data along $E(v_i, v_j)$ in U. Our objective is *to find the regeneration routing that minimizes the cost for the cooperative regeneration schemes deployed under the new framework*. Specifically, for a cooperative regeneration scheme, if $\mathcal{U}$ denotes the set of all the possible regeneration routings, then the optimization problem can be formulated as follows:

$$\text{Minimize} \quad C_U = \sum_{E(v_i,v_j) \in \mathcal{E}_U} w(v_i, v_j) l_U(v_i, v_j), \quad U \in \mathcal{U}.$$

In general, the physical meaning of the regeneration cost depends on how $w(v_i, v_j)$ is defined. We show several examples of C_U under different interpretations of $w(v_i, v_j)$.

- If $w(v_i, v_j) = 1$ for all v_i, then C_U denotes the number of packets transmitted for regeneration.
- If $w(v_i, v_j)$ is the inverse of the bandwidth between v_i and v_j, then the regeneration cost C_U denotes the total amount of transmission time, which can be treated as the total elapsed time for the system to serve the regeneration.
- If $w(v_i, v_j)$ is the monetary cost to transmit a unit of data along $l_U(v_i, v_j)$, then C_U denotes the total monetary cost for regeneration.

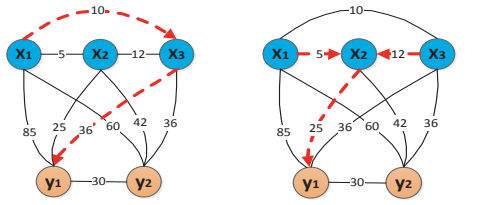


Fig. 3. Two Steiner Trees constructed over $\{y_1, x_1, x_3\}$, where $\{x_1, x_3\}$ are expansion nodes and y_1 is a newcomer.

We require the weight values to be periodically probed [4]. In this case, the weight values may not be always stable, which adds requirements as follows.

1) **Efficiency.** This optimization model should be solved *timely* before the weight values become outdated.
2) **Insensitivity.** The optimal routing should own the *insensitivity property*, i.e., a slight deviation of the measured weight values should not affect the optimality of the selected routing in the real scenario.

Discussion: Although some of the links (resp. nodes) of the optimal regeneration routing may be transmissive (resp. computational) overload, the minimum regeneration cost (e.g., the minimum regeneration time or monetary cost) will be achieved. Besides, we can also add an extra restricted condition on the maximum load allowed on each link to avoid the overload case.

IV. Heterogeneous Cooperative Regeneration

Given that the new regeneration framework can significantly decrease the regeneration cost, a remaining question is how to efficiently solve the optimization model by selecting appropriate nodes to execute the two stages. To answer this question, we first map the minimal aggregation routing to the Steiner Tree Problem, and then propose two greedy algorithms to timely find the contacted providers and expansion nodes, so as to obtain the routing with near-minimal regeneration cost.

A. Seeking of Minimal Aggregation Routing

In this section, we first discuss how to determine the aggregation routing of each newcomer y_i $(1 \leq i \leq r)$ when given the expansion nodes, which is the cornerstone in the seeking of expansion nodes and contacted providers. Actually, the seeking of aggregation routing is equivalent to the Steiner Tree Problem. We begin with the definition of Steiner Tree.

Definition 1. *(Steiner Tree [11]) Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, any tree in $\mathcal{G}$ spanning over a given set of nodes $\mathcal{A} \subseteq \mathcal{V}$ is called a Steiner Tree for $\mathcal{A}$ and $\mathcal{G}$.*

Figure 3 shows two Steiner Trees spanning over $\{y_1, x_1, x_3\}$, where we assume $\{x_1, x_3\}$ are expansion nodes, and y_1 is a newcomer. In Figure 3(a), the Steiner Tree is strictly constructed over the given nodes $\{y_1, x_1, x_3\}$ and its weight is 46. Another Steiner Tree shown in Figure 3(b) includes an extra node x_2 and its weight is 42.

Theorem 1. *In the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, suppose the collection of expansion nodes is $\mathcal{N} \subseteq \mathcal{V}$ and y_i $(1 \leq i \leq r)$ is a newcomer. Any Steiner Tree T_i for $\mathcal{G}$ and $\mathcal{N} \cup \{y_i\}$, whose root is y_i and leaves are the expansion nodes, corresponds to one aggregation routing in the regeneration of y_i.*

Proof. On one hand, given a Steiner Tree T_i whose root is a newcomer y_i $(1 \leq i \leq r)$ and leaf nodes are expansion nodes, we first prove that it establishes an aggregation routing of y_i. For every leaf node, as it is the expansion node, it should aggregate the packet vectors it keeps for the regeneration of y_i, and forward the produced packet vector to its parent (e.g., x_1 transmits the packet vector to x_3 in Figure 3(a)). For any non-leaf node except the root node y_i, there are two possible cases. If it is not the expansion node (e.g., x_2 in Figure 3(b)), it then simply aggregates the received packet vectors from its children and forwards the result to its parent. If it is an expansion node, it should aggregate the received packet vectors with those it expands and forward the result to its parent (e.g., x_3 receives packet vectors from x_1, aggregates them with its packet vectors, and sends the result to y_1 in Figure 3(a)). Finally, the newcomer y_i (e.g., y_1 in Figure 3) will receive the aggregation of all the packet vectors on the expansion nodes.

On the other hand, given an aggregation routing T_i that covers the involved expansion nodes and y_i, we will prove that it corresponds to a Steiner Tree whose root is the newcomer y_i $(1 \leq i \leq r)$ and leaves are expansion nodes $\mathcal{N}$. First, for any two nodes in T_i, both of them connect the newcomer y_i, therefore T_i is connected. Second, each node will only send the data to a node that is closer to the newcomer, therefore the cycle will not exist in T_i. As a summary, T_i is a tree. For T_i, the expansion nodes in T_i should be leaves, since other nodes do not keep any packet vector for delivery. Meanwhile, y_i is the final destination of all the related packet vectors for its regeneration. Therefore, for T_i, y_i is the root and expansion nodes are leaves. Since T_i should always include the expansion nodes $\mathcal{N}$ and y_i, T_i is a Steiner Tree for $\mathcal{G}$ and $\mathcal{N} \cup \{y_i\}$. □

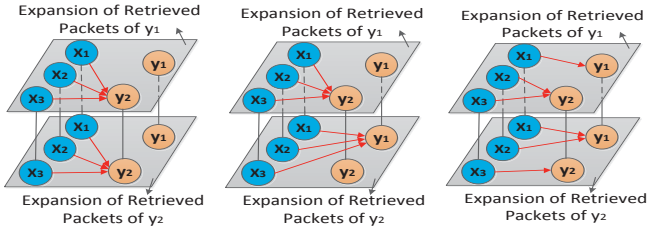
Based on Theorem 1, we can derive the Lemma 1.

Lemma 1. *Given $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, the optimal aggregation routing of y_i $(1 \leq i \leq r)$, is a Steiner Minimal Tree for $\mathcal{N} \cup \{y_i\}$ and $\mathcal{G}$, where $\mathcal{N}$ is the collection of expansion nodes.*

Steiner Minimal Tree is a NP-Complete problem [11] and can be approximately solved by using an existing greedy algorithm. A typical greedy algorithm initializes the tree T_i by first randomly selecting a node from $\mathcal{N} \cup \{y_i\}$. Every time it selects another node from $\mathcal{N} \cup \{y_i\}$ which has the shortest distance from the tree T_i, records the corresponding nodes in the shortest path, and extracts related nodes to T_i from $\mathcal{N} \cup \{y_i\}$. This procedure repeats until T_i covers all the nodes of $\mathcal{N} \cup \{y_i\}$. The complexity of this algorithm is $O(|\mathcal{N}|n^2)$.

B. Selection of Expansion Nodes

After the seeking of minimal aggregation routing when the expansion nodes are fixed, a subsequent question is how to



(a) Stage 1 (b) Stage 2 (c) Stage 3

Fig. 4. The selection of expansion nodes ($r = 2$ and $d = 3$). (a) y_2 is the common expansion node of all the retrieved packets. (b) The retrieved packets of a newcomer will choose the same expansion node. (c) Different retrieved packets can have different expansion nodes.

choose the appropriate expansion nodes when given the selected providers, so as to achieve the minimal regeneration cost. The straightforward enumeration tries all possible expansion nodes for each retrieved packet and incurs n^{dr} trials, which greatly overwhelms the search. To efficiently provide an online regeneration routing, we now propose a greedy algorithm as shown in Algorithm 1. The algorithm is composed of three stages, where the search space of stage i ($i = 1, 2$) is the subset of stage $i+1$. The **main idea** is that this algorithm will work in a smaller search space first. If there is no better routing, the algorithm will jump to the next stage with a larger search space for deep seeking. This design significantly reduces the search effort and iteratively approaches the optimal routing.

Algorithm Details: In stage 1 (shown in Figure 4(a)), all the rd retrieved packets select a random node as the common expansion node. The cost g^* (i.e., the minimal delivery cost in expansion stage and the minimal aggregation cost to regenerate r newcomers) under this selection is recorded (step 3).

In stage 2, the retrieved packets of a same newcomer will select a common expansion node. For example, the retrieved packets of y_1 will select y_2 as the common expansion node as shown in Figure 4(b). At first, it repeatedly calculates the regeneration cost, when setting each candidate in $\mathcal{V}$ to act as the universal expansion node of retrieved packets that belong to a common newcomer (step 6∼8). If this setting owns less regeneration cost compared to the current minimal cost g^*, then g^* is updated and the setting is recorded (step 9∼10). Finally, the replacement with the minimal cost will be performed and the iteration increases by 1 (step 11∼12).

Conversely, in stage 3, each retrieved packet can have its own expansion node (see Figure 4(c)). During the expansion node replacement of each retrieved packet, it always records the replacement information that brings less cost and accordingly updates the current minimal cost g^* (step 16∼20). This stage finally performs the replacement with the minimal cost (step 21∼22). The last two stages will repeat until the iteration is t_1 or there is no cost saving for any potential replacement (step 13, step 23).

C. Selection of Providers

After investigating the selection of expansion nodes when the selected providers are given, in this section, we then turn to the provider selection. For r newcomers, to choose d providers ($d \leq n - r$) from $(n - r)$ surviving nodes for every newcomer, the enumeration of selecting all possible providers will incur up to $\binom{n-r}{d}^r$ tests. To improve the search efficiency, we propose a greedy algorithm for provider selection in Algorithm 2.

Algorithm Details: This algorithm starts with a primary set of selected providers $\mathcal{S}_i$ for each newcomer y_i ($1 \leq i \leq r$) (step 1) and initializes the set of candidate providers $\mathcal{R}_i$ (step 2). For

Algorithm 1: Expansion Nodes Selection

Input: A connected graph $\mathcal{G}$, the vertices $\mathcal{V}$, the expected steps t_1, and the selected providers $\mathcal{S}$.

1 set iteration=0
2 ▷ `Stage1: Retrieved packets have a common expansion node.`
3 randomly set a node as the expansion node of all rd retrieved packets, calculate the cost g^*
4 ▷ `Stage2: Retrieved packets of the same newcomer have a common expansion node.`
5 set $f = 0$
6 **for** *each newcomer* **do**
7 **for** *each candidate* $v_c \in \mathcal{V} = \mathcal{X} \cup \mathcal{Y}$ **do**
8 compute the cost g if setting v_c as the expansion node of the retrieved packets of this newcomer
9 **if** $g < g^*$ **then**
10 $f = 1$, $g^* = g$, record the setting
11 **if** $f = 1$ **then**
12 perform the recorded replacement with the cost g^*, iteration++
13 repeat step 5 ∼ 12 until $f = 0$ or iteration=t_1
14 ▷ `Stage3: Different retrieved packets have different expansion nodes.`
15 set $f = 0$
16 **for** *each retrieved packet* **do**
17 **for** *each candidate* $v_c \in \mathcal{V} = \mathcal{X} \cup \mathcal{Y}$ **do**
18 calculate the cost g once setting v_c as the expansion node of this retrieved packet
19 **if** $g < g^*$ **then**
20 $f = 1$, $g^* = g$, record the setting
21 **if** $f = 1$ **then**
22 perform the setting with the cost g^*, iteration++
23 repeat step 15∼22 until $f = 0$ or iteration=t_1

Algorithm 2: Provider Selection

Input: The connected graph $\mathcal{G}$, and the expected steps t_2.

1 set $\mathcal{S}_i = \{x_1, \cdots, x_d\}$ for each newcomer y_i ($1 \leq i \leq r$)
2 set $\mathcal{R}_i = \{x_{d+1}, \cdots, x_{n-r}\}$ for y_i ($1 \leq i \leq r$)
3 set $f = 0$, step=0, record current cost g^*
4 **for** *each newcomer* y_i **do**
5 **for** *each provider* $x_p \in \mathcal{S}_i$ **do**
6 **for** *each candidate* $x_c \in \mathcal{R}_i$ **do**
7 calculate g if replacing x_p with x_c in $\mathcal{S}_i$
8 **if** $g < g^*$ **then**
9 $g^* = g$, $f = 1$, record the replacement
10 **if** $f = 1$ **then**
11 find the replacement pair (x'_p, x'_c) that reaches g^*
12 $\mathcal{S}_i = \mathcal{S}_i - x'_p + x'_c$, $\mathcal{R}_i = \mathcal{R}_i - x'_c + x'_p$, step++
13 repeat steps 3∼12 until $f = 0$ or step=t_2

each newcomer y_i, it repeatedly executes the replacement by substituting every provider in $\mathcal{S}_i$ with each candidate in $\mathcal{R}_i$ and calculates the cost (step 4∼7). If the replacement brings cost saving over the optimal routing currently found (step 8), then it records the replacement, updates the current minimal cost g^*, and sets f to 1 (step 9). It then performs the replacement that brings the minimal cost, and increases the iteration step by 1 (step 10∼step 12). This process repeats until either there is no saving by any replacement or the iteration step reaches the expected value t_2 (step 13). Note that the cost of given providers $\{\mathcal{S}_i\}_{i=1}^r$ is the minimum regeneration cost when $\{\mathcal{S}_i\}_{i=1}^r$ are selected by r newcomers. We can calculate it by utilizing the selections of expansion nodes in Section IV-B and the seeking of minimal aggregation routing in Section IV-A.

D. Complexity Analysis of Nodes Selection

We mainly analyze the computation complexity of provider selection and expansion node selection. Firstly, for provider selection, the number of replacements in Algorithm 2 is at most $t_2 r(n-r-d)d$. For expansion nodes selection of r newcomers, the number of replacements in Algorithm 1 is at most $t_1 rdn$. Therefore, the computation complexity to select the appropriate providers and expansion nodes with near-optimal regeneration cost in HCR is less than $O(t_1 t_2 r^2 d^2 (n-r-d)n)$, where $r + d \leq n$. In real storage systems, r, d, and n are usually assigned with small values [3], [7]. Meanwhile, the selection of t_1 and t_2 can also be adjusted by system administrators based on the preference on search time or regeneration cost.

V. Evaluation

A. Evaluation Setup

We compare the regeneration routings found by HCR with both CCR and the Enumeration scheme. Compare with HCR, CCR will randomly select the providers. Enumeration scheme will test every possible trial of providers and expansion nodes and select the optimal routing.

We first define the system configuration (n, d, r), where n is the number of storage nodes, d denotes the number of providers, and r is the number of newcomers. We choose the scheme [21] for cooperative regeneration and the erasure coding in [21] defines $k = d < n - r$ and $s = r$ in the regeneration. We set $t_1 = t_2 = 100$ and run our simulations on a server with a quad-core Intel Xeon X5472 CPU at 3GHz and 8GB RAM. The operating system is Ubuntu 10.04.3 LTS.

Existing storage systems often prefer a small parameter of n to avoid huge repair traffic. For example, HDFS-RAID [3] adopts the ($n = 10$, $k = 4$) erasure coding scheme. Therefore, our tests range n from 5 to 15.

Trace-driven Simulation: We run trace-driven simulations under a PlanetLab-like environment constructed by uniformly choosing bandwidth values between all node pairs from the range [0.3Mbps, 120Mbps]. This simulation method is also used in [15], [16]. The inverse of bandwidth values will be served as the weights. Therefore the regeneration cost reflects the transmission time of data in the regeneration. We also utilize the clean trace *app-cleaned.avt* of "PlanetLab all pairs ping" [8], which monitors the availability of hundreds of nodes from Jan. 2004 to Jan. 2005. This trace records the session availability of every node in the continuous form [up-time,down-time]. The up-time/down-time is the moment when a node is first detected to be up/down for an available session. A node is more unstable if it has more up/down events.

Evaluation Method: We sort the nodes according to the number of sessions and select n most unstable nodes that are up at time 0, which indicates that the system is intact at time 0. As time goes by, the system will become unstable as some nodes are down. We assume that once a node is detected to be down, it will be regarded as failed as its stored data becomes unavailable. Once the number of failed nodes reaches r, a cooperative regeneration will be activated, which will follow the routings established by HCR and CCR respectively. The next r nodes (following the n most unstable nodes that are selected) will replace the failed nodes, and serve as the newcomers. We assume each node in the original storage system stores data files with the size of 1GB. The test is run for many times and the final results are averaged.

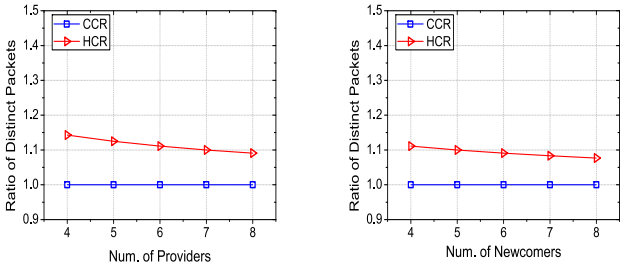
B. Evaluation Results

Experiment 1 (Ratio of Distinct Packets): We first compare the number of distinct packets that are needed to be transmitted during the regeneration between CCR and HCR, where the number of distinct packets of CCR is normalized to be 1.

Figure 5(a) first compares the ratio of distinct packets under different number of providers, where $n = 12$ and $r = 4$. We can find that HCR needs a bit more distinct packets compared with CCR. However, the ratio will decrease when the number of contacted providers increases. When $d = 8$, HCR needs to transmit about 9.1% more distinct packets.

Figure 5(b) also compares the ratio of distinct packets under different number of newcomers, where $n = 14$ and $d = 6$. When $r = 8$, HCR only needs to transmit 7.7% extra distinct packets compared with CCR. We can learn that HCR only causes a bit more distinct packets compared with CCR, and seeks the better regeneration routing with huge cost saving according to the node heterogeneity (see Experiment 3). It can be treated as a tradeoff between repair traffic and regeneration cost.

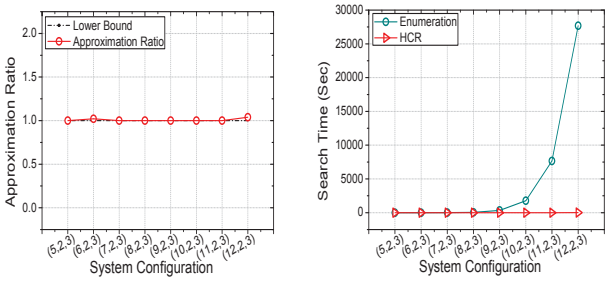
Experiment 2 (Approximation Ratio and Search Time): Figure 6 compares HCR with Enumeration on approximation ratio and search time. We evaluate the *"approximation ratio"* by the ratio of the transmission time of the optimal routing found by HCR to that of the optimal one sought by Enumeration. Therefore, the approximation ratio reflects the optimality of the routing found by HCR. We see that the optimal routing found by HCR achieves the same transmission time compared with that obtained by Enumeration for most tested system configurations (shown in Figure 6(a)). Even when the system configuration is $(n = 12, d = 2, r = 3)$, HCR only incurs an extra of 4.0% transmission time over Enumeration.



(a) Under different number of providers.

(b) Under different number of newcomers.

Fig. 5. The comparison on number of distinct packets.



(a) Approximation ratio.

(b) Search time.

Fig. 6. The comparison on approximation ratio and search time.

Meanwhile, HCR greatly reduces the required time to find the optimal regeneration routing as shown in Figure 6(b). For example, when the system configuration is $(n = 12, k = 2, r = 3)$, Enumeration needs about 7.7 hours to identify the optimal regeneration routing. It may be prohibitive especially when the regeneration routing needs to be determined online based on the current system loads. On the contrary, HCR merely needs 4.4s under the same system configuration.

Experiment 3 (Transmission Time): We also compare HCR with CCR on the transmission time under different system parameters.

We first evaluate the transmission time under different system scales as presented in Figure 7(a), where $d = 5$ and $r = 3$. First, HCR significantly decreases the transmission time compared with CCR. For example, under the system configuration ($n = 15$, $d = 5$, $r = 3$), the transmission time of HCR is 75.4% less than that of CCR. Second, with the system scales up, the transmission time of HCR will decrease. This is because there are more candidate providers to be selected in HCR, the transmission time will then be potentially reduced.

We then investigate the transmission time under different number of providers, where $n = 12$ and $r = 4$. The evaluated results are shown in Figure 7(b). We can find that the transmission time of both HCR and CCR will increase with the number of providers. Meanwhile, HCR can still retain its efficiency when the number of providers varies. The saving on transmission time brought by HCR is 59.0% when $d = 4$, and will reach 59.3% when $d = 8$.

We finally let $n = 12$ and $d = 4$, and measure the transmission time when the number of newcomers varies. The final results are illustrated in Figure 7(c). We can observe that both HCR and CCR need more transmission time when r increases. This is because more data are needed to be transmitted and regenerated when there are more newcomers.

This comparison also indicates that *HCR though needs a bit more distinct packets to be transmitted, it can greatly reduce the regeneration time by utilizing the node heterogeneity*.

Experiment 4 (Connection Sensitivity Analysis): In most practical scenarios, the weight values usually cannot be accurately measured. Thus, it is critical to ensure that the optimal routing we obtain under inaccurate weights is still close to the optimal one under the actual weights. We call it *insensitivity*. We first define the concept of deviation ratio:

$$\text{deviation ratio} = \frac{\text{the measured weight value}}{\text{the real weight value}} \tag{5}$$

In this experiment, we will evaluate the insensitivity of HCR when the deviation ratio is: (1) 1 (i.e., the weight is accurately measured); (2) uniformly selected from the range of $[0.9, 1.1]$, $[0.8, 1.2]$, and $[0.7, 1.3]$, respectively. We conduct the experiment under different system parameters in Figure 8.

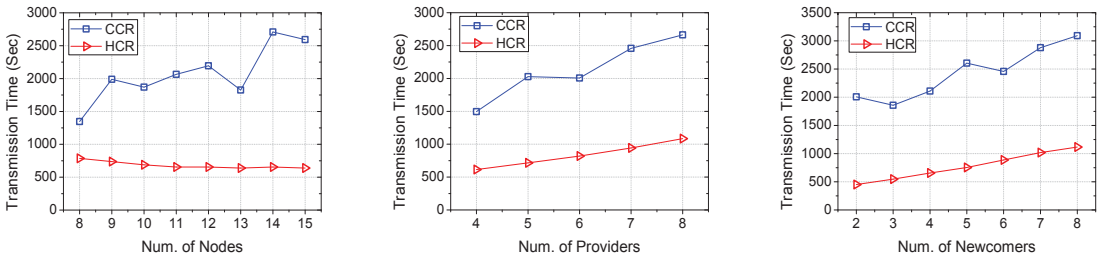
Figure 8(a) first gives the insensitivity of HCR under different system scales, where $d = 5$ and $r = 3$. It indicates that HCR can retain its insensitivity where there are different numbers of nodes in a system. For example, when the deviation ratio ranges from $[0.7, 1.3]$, the found routing at the inaccurate bandwidth will cause 5.0% more transmission time compare with that obtained at the accurate bandwidth under the system configuration ($n = 12$, $d = 5$, $r = 3$).

Figure 8(b) shows the insensitivity of HCR under different number of providers, where $n = 12$ and $r = 4$. We can observe that the optimal regeneration routing found by HCR in the environment with inaccurate bandwidth will still close to the optimal one found in real environment. Generally, the environment with the larger deviation ratio will exert more influence on the optimality of the found routing.

Figure 8(c) gives the insensitivity of HCR under different number of newcomers, where $n = 12$ and $d = 4$. HCR owns good insensitivity property when the deviation ratio is in the range of both $[0.9, 1, 1]$ and $[0.8, 1.2]$. When the deviation ratio is $[0.7, 1.3]$, the gap between the optimal routing found in this inaccurate environment and that of real environment is still very narrow. For example, when the system is ($n = 12$, $d = 4$, $r = 8$), the optimal regeneration routing in this inaccurate scenario merely needs 4.9% more transmission time.

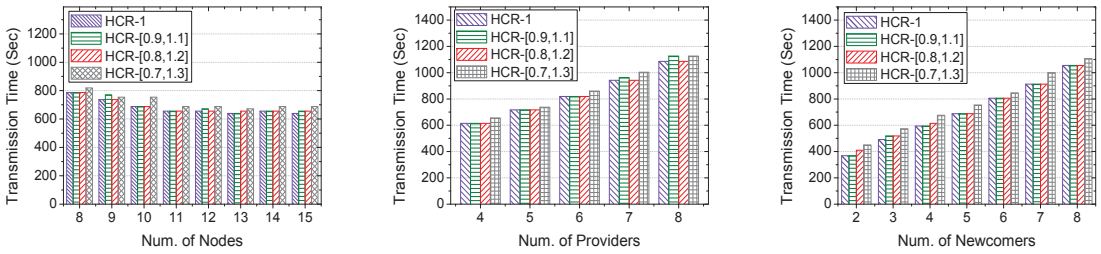
VI. Conclusion

In this paper, we present HCR, an efficient routing that helps the existing cooperative regeneration schemes to decrease the regeneration cost in the heterogeneous environment. We first analyze the procedure in CCR, and propose two stages to explore the regeneration cost saving. We further design two greedy algorithms to select the near-optimal execution nodes, and prove that the aggregation routing in the newcomer's regeneration is equivalent to the construction of Steiner Tree. Finally, through a series of trace-driven simulations, we justify both the effectiveness and the insensitivity of HCR.



(a) Under different system scales. (b) Under different number of providers. (c) Under different number of newcomers.

Fig. 7. The comparison on transmission time.



(a) Under different system scales. (b) Under different number of providers. (c) Under different number of newcomers.

Fig. 8. The comparison on sensitivity.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61232003, 61327902), the Beijing Municipal Science and Technology Commission of China (Grant No. D151100000815003), the State Key Laboratory of Highend Server and Storage Technology (Grant No. 2014HSSA02), University Grants Committee of Hong Kong (Grant No. AoE/E-02/08), and Research Grants Council of Hong Kong (Grant No. ECS CUHK419212).

REFERENCES

[1] Cleversafe dispersed storage. *http://www.cleversafe.org/downloads*, 2008.

[2] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker. Total recall: System support for automated availability management. In *Proc. of USENIX NSDI*, 2004.

[3] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling. Hdfs raid. In *Hadoop User Group Meeting*, 2010.

[4] M. Chowdhury, S. Kandula, and I. Stoica. Leveraging endpoint flexibility in data-intensive clusters. In *Proc. of ACM SIGCOMM*, 2013.

[5] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proc. of INFOCOM*, 2007.

[6] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Prof. of USENIX HotStorage*, 2001.

[7] D. Ford, F. Labelle, F. Popovici, M. Stokely, V. Truong, L. Barroso, C. Grimes, and S. Quinlan. Availability in globally distributed storage systems. In *Proc. of USENIX OSDI*, 2010.

[8] B. Godfrey. Repository of availability traces, 2006.

[9] Y. Hu, Y. Xu, X. Wang, C. Zhan, and P. Li. Cooperative recovery of distributed storage systems from multiple losses with network coding. *IEEE JSAC*, 28(2):268–276, 2010.

[10] A.-M. Kermarrec, N. Le Scouarnec, and G. Straub. Repairing multiple failures with coordinated and adaptive regenerating codes. In *Proc. of NetCod*. IEEE, 2011.

[11] E. Keyder and H. Geffner. Trees of shortest paths vs. steiner trees: Understanding and improving delete relaxation heuristics. In *Proc. of IJCAI*, 2009.

[12] J. Li and B. Li. Cooperative repair with minimum-storage regenerating codes for distributed storage. In *Proc. of IEEE INFOCOM*, 2014.

[13] J. Li, X. Wang, and B. Li. Pipelined regeneration with regenerating codes for distributed storage systems. In *Proc. of NetCod*, 2011.

[14] J. Li, X. Wang, and B. Li. Cooperative pipelined regeneration in distributed storage systems. In *Proc. of IEEE INFOCOM*, 2013.

[15] J. Li, S. Yang, X. Wang, and B. Li. Tree-structured data regeneration in distributed storage systems with regenerating codes. In *Proc. of IEEE INFOCOM*, 2010.

[16] J. Li, S. Yang, X. Wang, X. Xue, and B. Li. Tree-structured data regeneration with network coding in distributed storage systems. In *Proc. of IWQoS*, 2009.

[17] R. Li, J. Lin, and P. P. C. Lee. Enabling concurrent failure recovery for regenerating-coding-based storage systems: From theory to practice. *IEEE Trans. Computers*, 64(7):1898–1911, 2015.

[18] S. Nath, H. Yu, P. Gibbons, and S. Seshan. Subtleties in tolerating correlated failures in wide-area storage systems. In *Proc. of NSDI*, 2006.

[19] 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. Inf. Theory*, 2011.

[20] S. Russell and P. Norvig. Artificial intelligence: A modern approach. 2009.

[21] K. Shum. Cooperative regenerating codes for distributed storage systems. *arXiv preprint arXiv:1101.5257*, 2011.

[22] K. Shum and Y. Hu. Existence of minimum-repair-bandwidth cooperative regenerating codes. In *Proc. of NetCod*. IEEE, 2011.

[23] C. Suh and K. Ramchandran. Exact-repair mds codes for distributed storage using interference alignment. In *Proc. of ISIT*, 2010.

[24] Y. Wang, D. Wei, X. Yin, and X. Wang. Heterogeneity-aware data regeneration in distributed storage systems. In *Proc. of IEEE INFOCOM*, 2014.

[25] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In *Proc. of USENIX OSDI*, 2008.

[26] J. Zhang, X. Liao, S. Li, Y. Hua, X. Liu, and B. Lin. Aggrecode: Constructing route intersection for data reconstruction in erasure coded storage. In *Proc. of IEEE INFOCOM*, 2014.