Aggrecode: Constructing Route Intersection for Data Reconstruction in Erasure Coded Storage

Jing Zhang[1,2], Xiangke Liao[1], Shanshan Li[1], Yu Hua[3], Xue Liu[2], Bin Lin[1]
[1]School of Computer Science and Technology, National University of Defence Technology, Changsha, China, {shanshanli,xkliao,binlin}@nudt.edu.cn
[2]School of Computer Science, McGill University, Montreal, Canada, {jingz,xueliu}@cs.mcgill.ca
[3]WNLO, School of Computer, Huazhong University of Science and Technology, Wuhan, China, csyhua@hust.edu.cn

Abstract*—Node failures often occur in large-scale data centers today. Erasure coded storage system provides high data reliability via data reconstruction. Existing work can improve reconstruction performance, while considering the transmission of recovery data as the main source of reconstruction overheads. Transmission costs are highly related with network topology, which is unfortunately overlooked. An ideal connected topology assumes that two nodes in a data center has a direct link. The unmatching design between the network model and the practical topology may lead to an underestimated transmission costs. In this paper, we propose an erasure coded storage system for data reconstruction, which uses the practical network topology to minimize the reconstruction transmission costs. First, we identify the aggregation feature of erasure coding reconstruction and propose Aggregation Decoding, which splits the decoding process into several sub-decoding operations during reconstruction routing to reduce overall recovery data to be transmitted. We further improve Aggrecode to construct efficient route basing on the location of participating nodes to exploit the aggregation feature of Aggregation Decoding. We formulate this routing problem as a relaxed Steiner Tree problem. We design two heuristic routing algorithms based on ant-colony optimization specialized for two failure recovery cases, e.g., node recovery and degraded read. Our analytical results demonstrate the important properties of Aggrecode. These properties are evaluated by extensive experiments deployed on popular data center topologies, such as Torus, Fat-tree, DCell and BCube. The results show that Aggrecode can reduce data transmission costs by at least 37.12% for all settings.

I. Introduction

Erasure coding for large scale data storage systems is becoming a promising technology due to its superiority on achieving much higher storage efficiency while ensuring data reliability. Compared with traditional replica solutions, erasure coding can achieve higher storage efficiency at the expense of slight extra coding and decoding overheads in data recovery. When failures occur, original data can be recovered from a subset of the surviving coded fragments as long as the amount of failures is smaller than a given upper bound. This process is called data reconstruction. During reconstruction, the corresponding surviving fragments, called participating fragments, are transmitted to an end node, called target node, to decode and recover the lost data fragments.

In the big data era, component failures in large scale clusters often occur [1]. Hence the reconstruction overhead is a significant consideration in erasure coded storage systems.

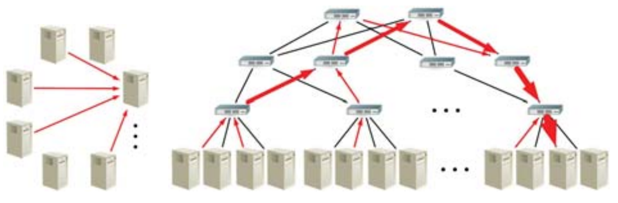

(a) Logical (b) Physical

Fig. 1: Network topology for erasure-coded storage systems

Normally, the reconstruction overhead is mainly reflected in transmission traffic costs. According to erasure coding mechanism, reconstruction process generates a burst of traffic in some specific links. On one hand, as data centers usually host lots of applications and services at one time [2], the transmission costs introduced by reconstruction will consume substantial bandwidth from other systems in the same cluster, especially when the load is heavy. On the other hand, from the view of "green" computing, the more traffic it generates, the more energy is consumed for transmitting these data [3].

Currently, more and more erasure coding schemes have been applied to the real world systems, including Facebook HDFS storage system [4] and Microsoft Windows Azure Storage [5]. Recent studies begin to consider reconstruction performance in practical scenarios, but most of them overlook the factors of practical network topology in reconstruction. Instead, they usually take the ideal connected graph as the logical topology, in which there is a directed path between every two vertexes, so that the distance between any two nodes is ideally one hop. Most current large-scale storage systems are deployed on a data center with specific network topology such as Torus [6], Fat-tree [7], DCell [8] and BCube [9]. The traffic patterns during reconstruction highly depend on the physical topology of the network infrastructure. The distance between target node and the host nodes of participating fragments in the practical reconstruction, therefore, is not ideally one hop. The frequently used ideal connected graph might have some mismatching in evaluating data transmission costs during reconstruction.

For example, a typical reconstruction data transmission in the ideal connected graph topology is shown as Figure 1(a). Assuming the practical network is a tree-based topology shown as Figure 1(b), we can observe that the logical topology cannot demonstrate the data transmission traffic pattern such as the congestion and the varying distances of different paths. The gap between logical and practical network topologies will lead to significant reconstruction performance degeneration in real systems compare with the the results of theoretical analysis.

Reconstruction with practical network topology introduces many challenges to system designers. To minimize the traffic bandwidth consumption, the optimal reconstruction routing should exploit the specific topology feature while carrying out the erasure coding mechanism, which is difficult for large-scale data centers. The various data center topologies make the routing problem even more challenging.

Transmission costs and latency are two fundamental concerns in reconstruction. Generally, reducing transmission costs contributes to latency decrease. But the aggregation feature that we demonstrate in the following analysis has a potential possibility of making a contradictory effect on these two factors. The tradeoff between these two concerns requires careful design.

In this paper, we explore a practical network topology-awared data reconstruction design for general erasure coded storage system. To the best of our knowledge, this is the first work to use practical network topology in erasure coded storage system recovery. Our goal is to minimize the reconstruction data transmission overhead in any data center network topology. We discover the aggregation feature in the practical reconstruction of erasure coded storage systems. Decoding process can be split into several sub-decoding operations. Based on this observation, we can perform an aggregation-based decoding on intermediate nodes rather than sending all to the target node, where the decoding can be executed. As a result, reconstruction routing could be carefully designed in data center network by exploiting this aggregation feature.

To address these issues, we propose Aggrecode, a novel design for erasure coded storage reconstruction. The main contributions of this paper are summarized below:

(i) **Aggregation Decoding**: To the best of our knowledge, Aggrecode is the first work to use topology-awared approach in reconstruction of erasure coded storage systems. We exploit the aggregation feature in the reconstruction process and propose Aggregation Decoding that splits the final decoding operation into sub-decodings on the route, which can reduce entire transmission costs.

(ii) **Reconstruction Routing**: We formulate the minimized Aggregating Reconstruction Tree problem as a Steiner Tree problem, which is NP-complete. We provide two heuristic algorithms based on ant-colony optimization for node recovery and degraded read. The approximation ratio of our algorithms can reach nearly 1.01, which is significantly better than classic solutions without the loss of computation complexity.

(iii) **Transmission Costs Saving**: We propose Aggrecode, an reconstruction approach for erasure coded storage system in large-scale data centers. We analyze and evaluate the performance and overhead on most promising practical network topologies including Torus, Fat-tree, DCell and BCube. Aggrecode can save transmission costs by more than 37.12% for all settings.

The rest of the paper is organized as follows. Section II introduces the background and motivation of our work. Section III formulates the problem. Section IV illustrates the Aggrecode design. Section V analyzes the performance and overhead. Section VI shows the evaluations and experiments. Section VII discusses the related work. We conclude the paper in Section VIII.

II. Background and Motivation

A. Background

We mainly focus on the large scale erasure coded storage systems deployed on data centers. Considering that erasure codes are mostly used for storing large block-size append-only data and providing permanent storage services such as archiving and backup, our design should have a good performance in these scenarios. We consider systematic erasure coding and focus on two typical reconstruction cases, node recovery and degraded read. Node recovery occurs when a server crash leads to the permanent lost of all the data stored on it. Then the system restarts the server and recovers all the lost data by surviving erasure coded redundant data. In this case, bandwidth costs are more critical than latency. Degraded read occurs when a request arrives at a current unavailable server, the system reconstruct the required data fragment on an available server and reply to the client. In this case, both bandwidth costs and latency should be concerned. On the network topology side, most of the data center networks can be classified into several typical topologies such as tree-based topology, recursively-defined topology, and torus-based topology. The aggregating reconstruction approach we provide should be broadly useful for most data center network topologies.

For a (n, k) MDS systematic erasure code scheme, the input data is split into k fragments first and then $n-k$ coded fragments are generated based on the k original fragments. The entire n fragments are stored on n nodes to tolerate up to arbitrary $n-k$ failures. The ability of fault tolerance depends on the specific erasure coding scheme and can be flexibly scaled by adjusting the ratio of redundancy. In reconstruction, k surviving data fragments are sent from k surviving participating nodes to the target node where decoding performed. The general decoding equation can be described by a matrix-vector product as shown in equation 1. From d_1 to d_k are the k original fragments. From r_1 to r_k are the k fragments that selected to participate in the reconstruction. The decoding matrix is a $k \times k$ matrix that derived from the coding matrix, and all the arithmetics are performed in Galois Field. In this way, the original data can be reconstructed.

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} & B_{14} \\ B_{21} & B_{22} & B_{23} & B_{24} \\ B_{31} & B_{32} & B_{33} & B_{34} \\ B_{41} & B_{42} & B_{43} & B_{44} \end{bmatrix} \times \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \quad (1)$$

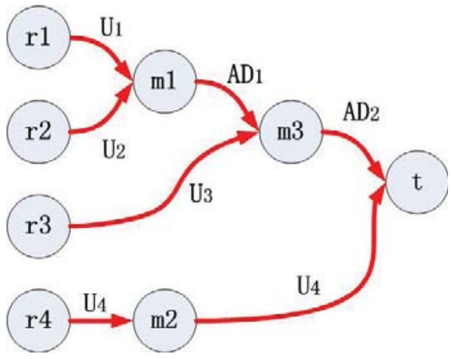


Fig. 2: Aggregation Decoding

B. Observation on Aggregation Decoding

In this part, we will illustrate some aggregation properties that we discover in the reconstruction decoding process, which shed light on our latter design. For simplicity, consider a (6, 4) erasure coding reconstruction in Figure 2. The nodes r_i on the left are participating nodes, node t is the target node, and the other nodes in the middle are intermediate nodes m_i. The reconstruction data transmission flow can be seen as a tree that consists of the routings from participating nodes to target node.

Property 1: The traditional decoding process can be split into several Aggregation Decoding operations.

Proof: In most reconstruction scenarios, it is not necessary to reconstruct all the k fragments, but only need to decode and recover the lost fragment at target node. Therefore, based on matrix multiplication rules, we derive a single data fragment reconstruction decoding equation from equation 1 as follows:

$$d_l = \sum_{i=1}^{k} (B_{li} \times r_i), \tag{2}$$

where d_l is the fragment we try to recover. For clarity, we define each $B_{li} \times r_i$ operation as a basic unit denoted by U_i. The summation of arbitrary multiple basic units is defined as an Aggregation Decoding operation denoted by AD_p. Based on the commutation rule, the result of one Aggregation Decoding can also participate in the following one, and the order of the operations will not influence the result of equation 2. Figure 2 is a simple example. Aggregation Decoding is executed in intermediate node m_1 and m_3, and the lost data d_l can be finally recovered. The entire Aggregation Decoding reconstruction process is shown as equation 3

$$\begin{aligned} AD_1 &= U_1 + U_2, \\ AD_2 &= AD_1 + U_3, \\ d_l &= AD_2 + U_4. \end{aligned} \tag{3}$$

Property 2: The data size of the result of an Aggregation Decoding is smaller than its input.

Proof: Galois Field is a finite field that has a limited number of elements. Results of any operations performed in Galois Field still lie in the field. All the elements in Galois Field have the same length. Given the size of an element m, since the original elements in equation 1 are in Galois Field, all the intermediate elements are also in Galois Field and have size of m. Therefore, from equation 3 we can see that the size of result of Aggregation Decoding is always m, which is smaller than the amount of inputs. See equation 4

$$Size(AD_p) = Size(U_i). \tag{4}$$

In other words, the entire amount of participating fragments is reduced after every Aggregation Decoding operation.

C. Motivation

In most current erasure coded storage systems (even in some named practical designs), the reconstruction costs are simply measured by the quantity of data that participate in the reconstruction. This is because data center network topology is overlooked and the distances between participating fragments and target node are ideally assumed to be one-hop. But under a practical data center network topology, the different transmission paths of participating fragments may highly influence the actual reconstruction costs. For example, in Figure 1(a), the reconstruction costs to recover m size data are $4m$, measured by the existing strategy. But in real network topology as Figure 1(b), the actual costs are $20m$.

In this paper, we propose Aggrecode to optimize the reconstruction costs, which use Aggregation Decoding and match the reconstruction tree with the practical network topology. If the aggregating reconstruction we proposed is employed, according to the Property 1, we can perform Aggregation Decoding when more than one participating fragments pass by the same node, such as m_1 and m_3 in Figure 2. Based on Property 2, the size of the result of each Aggregation Decoding is consistently as large as one participating fragment; hence the entire bandwidth consumed is lower than existing schemes. Therefore, if we take good use of the aggregating reconstruction and perfectly match the reconstruction tree to the topology, the entire transmission costs can be further saved.

III. Problem Formulation

In this section, we formulate the aggregating reconstruction routing problem. Given a practical (n, k) erasure coded storage system, we model its network topology as an undirected graph $G = (V, E)$. All the storage nodes compose the node set V in the graph. The edges between nodes compose the undirected edge set E. Each edge is assigned a weight to represent the bandwidth consumption. $t \in V$ is the target node of the reconstruction operation, $D \in V$ is the set of nodes that has the surviving fragments that correspond to the lost fragment. In other words, D is the set of nodes that are available for participating in reconstruction, called alternative node set, and obviously, $k \leq |D| < n$. As mentioned in section II, the reconstruction flow can be seen as an aggregation tree, thus the reconstruction transmission costs can be measured as the entire weight of the tree. The aggregating reconstruction routing problem can be formulated as follows:

Given: A (n, k) erasure coded storage deploy on a topology

$G = (V, E)$ with weight W for each edge; target node $t \in V$; corresponding group D.

Objective: To find a minimized aggregating reconstruction tree T that rooted at t, and spans k out of all the nodes in D such that the entire weight of the tree is minimized.

As mentioned in section II, based on property 2, the result of every Aggregation Decoding has the same size of a fragment. Therefore we assume the weight for every branch in T is consistent. Then the objective can be further simplified as $|ET|$ is minimized, where ET is the edge set of T. The k nodes that spanned by T are the participating nodes. We define the set of the k nodes as P, called participating node set, $P \in D$ and $|P| = k$. We summarize the notations which will be used through this paper in Table I.

TABLE I: Parameter Notation

Notations	Descriptions
G	Undirected graph modeled from network topology
V	Node set
E	Undirected edge set
t	Target node
D	Alternative node set, the set of surviving nodes that belongs to t's corresponding group
P	Participating node set, k nodes that spanned by T
T	Minimized aggregating reconstruction tree
ET	Edge set of T

IV. Aggrecode Design

In this section, we introduce the framework of our Aggrecode, including the data placement and the procedure of reconstruction. We propose the advanced MPH algorithm for node recovery routing and the advanced ADH algorithm for degraded read routing based on the ant-colony weighing algorithm.

A. Overview

We design the framework based on the most popular erasure coded storage architecture for general purpose, where there is a center master server for metadata maintain and system management. Assume the file is coded into n fragments and stored respectively into n nodes in the cluster, called a group. The same coding matrix is used for all files [5]. A group is only related to the current file. When a new file arrives, a new group will be chosen to store it. Since the nodes are not bounded to a group, a node can belong to different groups for different files.

The main idea of Aggrecode is to construct an optimal routing to minimize the transmission costs. According to the properties of Aggregation Decoding, when participating fragments intersect together during reconstruction, an Aggregation Decoding operation can be applied and a certain transmission costs can be saved. Therefore, the key problem is to construct an optimal routing that maximize the intersections of reconstruction data flow and make them occur as early as possible. We will introduce two heuristic algorithms in this section later.

In our design, for balancing the computing workload introduced by the aggregating reconstruction routing, we dynamically designate a temporary coordinator for reconstruction every time. Considering that lots of storage systems place data depending on applications' policy, our design does not add any extra requirement on group selection, so that preserving the data locality of the application can be maintained.

The reconstruction for node recovery includes several steps as follows. First, the master server designates the new restart node as the coordinator, and then sends it the metadata that is related to the data stored on it such as the data fragments list and the corresponding group for each fragment. Next the coordinator executes the aggregating reconstruction routing algorithm and obtains the Aggregation Decoding route. Finally, the participating fragments are sent to the target node following the route, and the Aggregation Decoding is performed.

Degraded read occurs when the node that hosts the read request is currently unavailable. The master server will designate a coordinator from the unavailable fragment's corresponding group. Then the coordinator runs aggregating reconstruction routing algorithm to choose an optimal node from the group as the target node and obtain the Aggregation Decoding route. The remaining part of the reconstruction is as same as the node recovery.

Aggregating reconstruction routing is the key part of Aggrecode. The main idea can be reduced to a Steiner Tree problem [10], which is a NP-complete problem. But due to the special features of our scenario, the classic Steiner Tree solutions become inefficient and even unavailable for our problem. Next, we propose an advanced ant-colony-based heuristic routing algorithm for our Aggregating Reconstruction Tree problem.

B. MPH Routing Algorithm for Node Recovery

Because of the particular properties of erasure coding reconstruction and the special characters of practical network topology, the classic solutions for Steiner Tree problem can not be used in erasure coding reconstruction routing directly. In the Steiner Tree problem, the set of objective nodes that are spanned by the tree is given and fixed. But in our Aggregating Reconstruction Tree problem, due to the properties of erasure coding, the objective set P is composed by choosing optimal k nodes out of D. Moreover, from the property 2 in section III, the transmission costs on each edge are consistently as large as one fragment, which makes all the edge weights become the same. As a result, the classic heuristic algorithms may have a high possibility of taking suboptimal edges which dramatically pull down the performance. On the other hand, the data center network topologies usually have much more redundant routings due to its specific geometrical features, which further raise the possibility of taking suboptimal edges in classic heuristic algorithms.

We propose an advanced heuristic algorithm for this Aggregating Reconstruction Tree problem for node recovery, MPH Routing Algorithm for Node Recovery(MPH-NR). Minimum cost path heuristic (MPH) algorithm [11] is a basic heuristic solution for Steiner Tree problem which has a balanced computation complexity and performance. In order to face the challenges in our scenario that edge weights are consistent, we propose an ant-colony-weighting algorithm to give the edges extra weight information to improve the heuristic routing algorithm's approximation rate.

Ant-colony optimization takes inspiration from the foraging behavior of some ant species, and use the pheromone deposited by these ants to find the favorable path [12] [13]. The basic idea of our ant-colony weighting algorithm is that, we simulate a number of artificial ants moving on the graph from the alternative node set D to the target node t, and use the presence of pheromone to express the potential value of an edge. The value of an edge represents the transmission costs saving if that edge is finally selected in the aggregating reconstruction routing tree. Each ant's route represents a potential solution for its corresponding alternative node. When an ant passes by an edge, it deposits pheromone on that edge. The more ants an edge be passed by, the more pheromone the edge present, which means the more it tend to be selected in the aggregating reconstruction routing tree.

In order to reduce the computation costs, we make some improvements based on the special features of our scenario. Taking advantage of data center topology is given, we calculate all the redundant shortest paths from every alternative node to target node first, and then use these paths as the ants' solutions (routes) instead of iteratively randomly generating the ant's solutions. Under these improvements, our algorithm is able to obtain an acceptable result with a significant saving on computation time. Also, we use an improved Shortest Path Faster (SPFA) algorithm for our scenario to calculate all the redundant shortest paths from alternative nodes to target node.

After weighting, we continue with an advanced MPH algorithm to calculate the final aggregating reconstruction routing tree, see Algorithm 1. The basic idea of our advanced MPH algorithm is, start the spanning tree T' from the target node t as root, for each iteration, pick up that node from D which closest to T' built so far, until T' spans k nodes out of D.

Algorithm 1: MPH Routing Algorithm for Node Recovery

```
Input: $G = (Vs, Vc, E)$, $n$, $k$, target node $t$, alternative
       node set $D$
Output: aggregating reconstruction routing tree $T$
$R \leftarrow SPFA(D, t)$ //$R$ is the redundant shortest paths set
from $D$ to $t$;
for $q \leftarrow 1$ to $|D|$ do
    //for every node in $D$ ;
    Set up ant[$q$], move through all the paths in $R$ that
    between $D[q]$ and $t$, insert the edge passed by in
    $phrm_q$;
end
for every edge(i,j) do
    for $q \leftarrow 1$ to $|D|$ do
        if $edge(i,j) \in phrm_q$;
        $Weight(i,j)$++;
    end
end
$Weight(i,j) = |g + 1 - Weight'(i,j)|$;
$T' \leftarrow t$ //Insert $t$ into the initial spanning tree and set up;
while $|T'| < k + 1$ do
    for $c_i \in D + t - T'$ do
        //for every node in $D$ that has not been inserted
        into the spanning tree;
        Dist[$i$] $\leftarrow$ weighted distance from node $c_i$ to $T'$;
        Path[$i$] $\leftarrow$ lowest weighted path from node $c_i$ to
        $T'$;
    end
    Choose node $c_i$ such that Dist[$i$] is the lowest;
    Insert $c_i$ and the nodes on Path[$i$] into the initial
    spanning tree $T'$;
end
$T \leftarrow T'$;
```

C. ADH Routing Algorithm for Degraded Read

The reconstruction routing for degraded read is even more challenging because the target node is not given. The exciting methods do not consider the choosing of target node. But our experiments show that a carefully choosing of target node may significantly save the transmission costs and even the latency. Therefore, we use the strategy deriving from Average Distance Heuristic (ADH) algorithm [14] [15] and propose ADH Routing Algorithm for Degraded Read (ADH-DR).

First, we scale our improved SPFA to calculate all the redundant shortest paths between every two nodes in D. Then, instead of choosing the closest node to insert into the spanning tree T' in MPH, we initialize D as a forest and choose two sub-trees which have the lowest weighted distance to merger together in every iteration. When T' spans k nodes out of D, the aggregating reconstruction routing tree is created. After this, target node selection is performed to select a node located in the center of the tree as root. The other parts use the similar technologies as MPH-NR.

V. Theoretical Analysis

In this section, we give a detailed analysis of performance and costs of Aggrecode including reliability, transmission costs saving, latency and computation overhead.

The reliability our design provides is based on the erasure code scheme of the storage system. We can flexibly adjust the parameters n and k of the (n, k) erasure code to achieve a desired reliability. In other words, our Aggrecode design will not have any negative influence on the reliability of the original erasure coded storage system.

Now we analyze the expectative transmission costs saving. Assuming in a (n, k) erasure coded storage system, the entire transmission costs without Aggrecode are

$$C = C_1 + C_2 + \cdots + C_k, \tag{5}$$

where C_x is the transmission costs from the x^{th} node spanned by T to the target node. In our design, the basic strategy is to span the closest node from the existing aggregating reconstruction tree. When the tree spans a new node x, the entire transmission costs of the tree increases by d_x. So the transmission costs with Aggrecode are

$$C_A = C_1 + (d_2 + d_3 + \cdots + d_k). \tag{6}$$

Because of C_1 is the closest node to the target node, transmission costs of x^{th} new spanning node can be descried as follows:

$$C_x = C_1 + e_x, e_x \geq 0. \tag{7}$$

Then the transmission costs without Aggrecode can be described as:

$$\begin{aligned} C &= C_1 + (C_1 + e_2) + (C_1 + e_3) + \cdots + (C_1 + e_k) \\ &= k \times C_1 + (e_2 + e_3 + \cdots + e_k). \end{aligned} \tag{8}$$

Therefore the expectative transmission costs saving S is

$$\begin{aligned} S &= 1 - \frac{C_A}{C} \\ &= 1 - \frac{C_1 + (d_2 + d_3 + \cdots + d_k)}{k \times C_1 + (e_2 + e_3 + \cdots + e_k)}. \end{aligned} \tag{9}$$

Based on equation 9 we can see:

(1) S increases as k increases. k expresses the reliability of the system. Hence, our scheme has better performance when in high-reliability-required systems.
(2) S increases as C_1 increases. C_1 indirectly reflects the scale of the topology. So our scheme has better performance for large scale systems.

The reconstruction latency is very important for degraded read, but not for node recovery. Node recovery usually not occurs right after the failures. Instead, it waits until the amount of failed nodes in the system reach a preset upper limit or the waiting time over a preset bound, and then recover the failed nodes by batch to reduce overhead. Obviously, the latency from reconstruction process is insignificant comparing to the waiting time, which means the latency is not very critical in node recovery. But the latency is critical for degraded read since the clients are waiting for response online.

In aggregating reconstruction, the latency mainly depends on the farthest participating node, in another word, the depth of the reconstruction routing tree. The Stainer Tree algorithm has a possibility of growing deeper for achieving minimized tree. To trade off the transmission costs and latency in our Aggrecode, in the weighting part, we find out the redundant shortest paths to the target node, and increase the value of the edges that belong to those paths. Another point may lead to latency is the extra computation costs that introduced by Aggrecode, which mainly comes from two parts, Aggregation Decoding and aggregating reconstruction routing. In Aggregation Decoding, all the decoding operations we perform can be reduced to linear combinations and matrix inversions. From equation 2 we can see, comparing to general erasure coding schemes, our Aggregation Decoding does not increase the amount of operations. Furthermore, the Aggregation Decoding can be performed in parallel, which may reduce the computation time and avoid hot spot. Therefore, our design does not introduce extra computation costs in Aggregation Decoding part.

For the computation costs of routing algorithm we do not worry about the MPH-NR because (1) as we explained before, latency is not very critical in node recovery; (2) coordinator is new restarted node, which has no current jobs and can afford full load computation for routing algorithm. Moreover, the computation costs of ADH-DR are higher than MPH-NR because of the target node selection part. So we mainly analyze ADH-DR. We will show an evaluation result in the next section.

Computation complexity is also very important here because our design should fit the large scale systems. The Computation complexity of MPH-NR is $O(NM)$, N is the cluster scale, M is the edges number. The Computation complexity of ADH-DR is $O(N^3)$. Comparing to classic MPH and ADH algorithm respectively, our two algorithms do not increase the computation complexity.

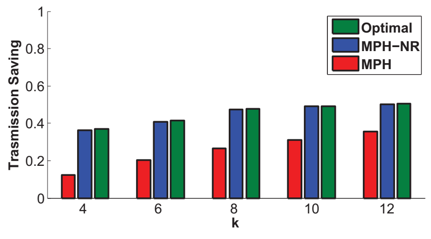
VI. Performance Evaluation

In this section, we evaluate the performance and overhead of Aggrecode including transmission costs saving, approximation ratio, latency, computation costs and scalability, and compare it with existing solutions. We also evaluate Aggrecode on different network topologies including Torus, Fat-tree, DCell and BCube. All results in our experiments are averaged over 1000 runs.

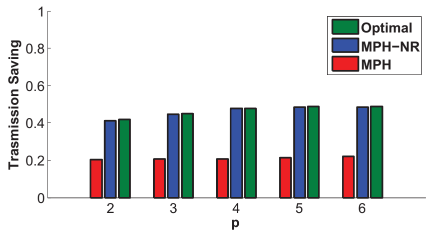
For the most evaluations below we choose Torus-based topology as the host topology for three reasons. First, Torus-based topology is one of the most promising topology in future data center design especially for large scale clusters due to its high multiple paths, resilient in server failures and efficient wiring. Second, Torus-based topology has a very successful and mature prototype cluster design, Camcube [16] [17], that expose the physical topology directly to the services. Camcube support flexible physical topology based custom self-defining routing service, which gives a natural interface to Aggrecode. So the torus-based topology is most likely to be the first beneficiary of Aggrecode in industrial circles. Third, torus-based topology has a strong and stable geometrical regulation when the server number in the cluster increases, which can show the relationship between performance and system scale more clearly. Moreover, we also evaluate Aggrecode on other different network topologies.

A. Transmission Costs

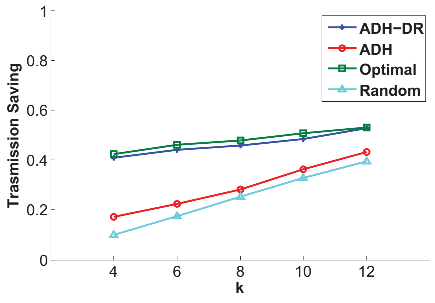
Figure 3(a) compares the transmission costs saving of MPH-NR with classic MPH and optimal algorithm in node recovery when k increases from 4 to 12 in a 27 nodes cluster. Apparently, optimal algorithm has the best performance but cannot be solved in polynomial time; classic MPH Algorithm is not very efficient in this scenario. Our MPH-NR performs a dramatic efficiency grow than classic MPH with polynomial time computation complexity and almost has the same performance as optimal algorithm. The transmission costs saving



(a) Node recovery with different k



(b) Node recovery with different p



(c) Degraded read with different k

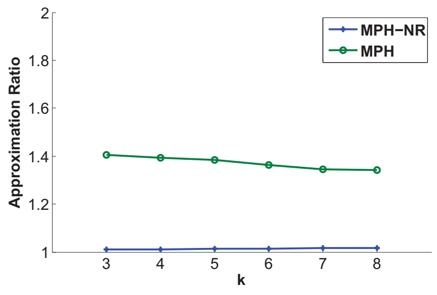
Fig. 3: Transmission saving

increases with k because more participating nodes offer more optional routing combinations during Aggregation Decoding. Figure 3(b) illustrates the transmission costs with different parity number, and our MPH-NR can save more transmission costs when p is larger. Figure 3(c) shows the transmission costs saving in degraded read. “Random” means randomly pick k nodes as participating nodes and one node as target node, and then build an optimal reconstruction route. The result shows that our ADH-DR has the best performance.

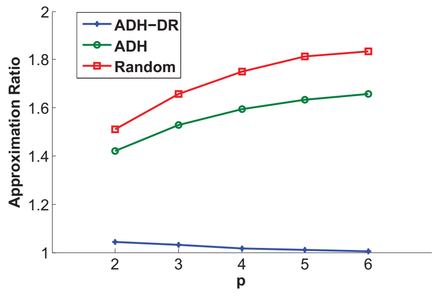
In order to further demonstrate the superiority of Aggrecode, we evaluate the approximation ratio in Figure 4. The approximation ratio r here is obtained by:

$$\mathrm{r} = \frac{\text{transmission costs of objective algorithm}}{\text{transmission costs of optimal algorithm}} \tag{10}$$

From Figure 4(a) we can see, in node recovery, MPH-NR's approximation ratio is close to 1, which is significantly better than classic MPH. Parameter k is related to the storage overhead of the erasure coded storage system. When k increases, the classic MPH has some performance gain due to the rising possibility of choosing the right k participating nodes out of $k+p-1$ alternative nodes. However for MPH-NR, because the participating nodes selection function has already integrated in, no further gain can be obtained from



(a) Node recovery



(b) Degraded read

Fig. 4: Approximation ratio

the increasing of k. The slightly performance fall on MPH-NR comes from the interference of other alternative nodes during the weighting of MPH-NR, and it is marginal comparing to the gained performance.

Figure 4(b) shows the approximation ratio in degraded read. Obviously, ADH-DR has overwhelming performance gain compare with other existing algorithms. The algorithm without target node selection function has the worst performance. Parameter p is the number of parity nodes, which is related to the reliability of the erasure coded storage system. We notice that the gap between ADH-DR and other two algorithms is increasing when p increase, since the options for target node choosing is growing, makes the effect of target node selection in ADH-DR more significant.

B. Latency and Computation Costs

We simulate reconstruction on a Torus-based cluster with 64 nodes and 1Gb net work port. We evaluate the latency in different network environments and the computation overhead.

First, we deploy proper background workload on the cluster that average RTT(Round-Trip Time) is from 0.01 ms to 10ms [18] [19]. We perform degraded read on 64MB fragment data set with (6, 3) Erasure code. As shown in Table II, when the workload is light, Aggrecode is not very efficient comparing to existing solutions because the computation and synchronization overhead take a higher proportion out of the overall response time. But when the workload turns to heavy, Aggrecode's advantage becomes more significant. That is because Aggrecode saves transmission traffic, which reduces the congestion on heavy workload links. Column “computation” expresses the proportion of computation time of ADH-DR,

TABLE II: Average response time for degraded read

RTT(ms)	Baseline(s)	Aggrecode(s)	Computation
0.01	1.142	1.215	4.42%
0.1	2.656	2.221	2.42%
1	25.463	20.916	0.26%
10	253.647	207.864	0.03%

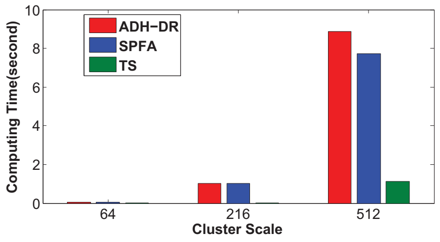


Fig. 5: Computation time

which is obtained from the ratio of ADH-DR execution time and overall response time of degraded read. The result from experiments demonstrates our analysis in section IV that the computation costs are insignificant comparing to transmission latency. Since the computation time is not raising with the workload increase, the computation overhead is even more insignificant in heavy workload network environment.

We try to find out which part of our algorithm contributes most latency. Figure 5 shows the proportion of computation overhead of different parts in ADH-DR. "ADH-DR" is the overall computation time of the algorithm; "SPFA" represents the redundant shortest paths calculation part; "TS" is the time costs of the tree spanning part. Figure 5 illustrates that the main contributor of computation costs is the redundant shortest paths calculation by SPFA. Therefore, an alternative improvement can be made by pre-executing SPFA and store the redundant shortest paths on data nodes, which trades off storage overhead for response time.

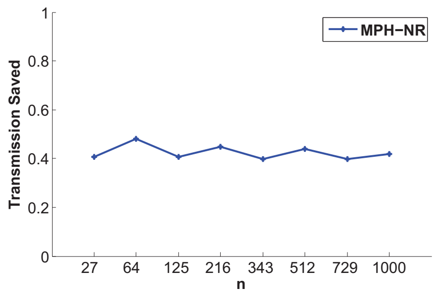
C. Scalability

Here we define scalability at two dimensions: the elasticity of performance when the cluster scales out and the generality that how our approach applies on different network topologies.

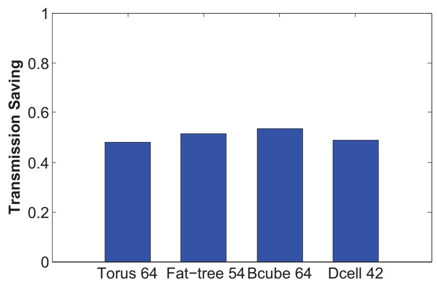
Figure 6(a) shows the transmission saving when cluster scale out from 27 nodes to 1000 nodes. Basically, the performance gain is stable between 40% and 50%. In order to demonstrate the generality of Aggrecode, Figure 6(b) illustrates the performance in Fat-tree, DCell and BCube, all of which are most promising data center topologies. We can see that Aggrecode can broadly fit these topologies very well and even has better performance than Torus.

VII. Related Work

Erasure coded storage systems have been widely studied, especially on reconstruction performance. Most of the existing work focus on reducing the amount of data that take participate in reconstruction. The intuitive think is to retrieval less fragments. Pyramid code proposed by Huang et al. allows



(a) Elasticity



(b) Generality

Fig. 6: Scalability

flexible reconstruction efficiency in the trade-off of storage space [20]. LRC also proposed by Huang et al. exploits non-uniform parity degrees, where some fragments retrieval less participating parities in reconstruction than others, to optimize the most occurred recovery cases in Windows Azure Storage [5]. Duminuco et al. propose Hierarchical code, that organizes the fragments as different groups hierarchically, and the parity degrees for each group are different, which is more efficient when reconstruction occurs on lower level groups [21]. On the other hand, instead of retrievaling from fewer fragments, some other solutions retrieval more fragments but less data from each. Regenerating code introduces network coding to code the data into smaller fragments and retrieval more fragments but less data totally [22] [23]. Huang et al. exploit Regenerating code's strategy on Hierarchical code to reduce the amount of participating data [24].

Another strategy arises recently is to improve the reconstruction process base on the practical system behaviors and I/O patterns. Khan et al. propose Rotated Reed-Solomon code for RAID in cloud file systems, which consider the read patterns and make the parities split across adjacent rows to reduce the unnecessary penalty during most reconstruction cases [25]. Li et al. proposes a cooperative pipelined regeneration process that reconstructs multiple data losses cooperatively [26]. A single pipelined regeneration process is also proposed for a single failure for Minimum-Storage Regenerating codes [27].

The interesting of recent work is moving from reducing the participating data by theoretical study to optimizing reconstruction process considering practical system behaviors. However, so far, none of the existing work consider the practical network topology for reconstruction performance. Therefore, to the best of our knowledge, our work is the first to use practical network topology in erasure-coded storage system data reconstruction.

VIII. Conclusion

We propose Aggrecode, an erasure coded storage system reconstruction approach that provides efficient data recovery with low transmission costs. We discover Aggregation Decoding that performs decoding in disperse and parallel to reduce overall recovery data transmitting. Aggrecode considers the practical network topology to construct efficient route for best exploiting Aggregation Decoding. We formulate the Aggregating Reconstruction Tree problem and propose two ant-colony-based heuristic routing algorithms MPH-NR and ADH-DR. Aggrecode can be deployed on most data center network topologies such as Trous, Fat-tree, DCell and BCube, and achieve a significant transmission costs saving.

Acknowledgment

We would like to thank the anonymous reviewers for their comments. This paper is supported in part by National Nature Science Foundation of China (NSFC) NO. 61379146, National Basic Research 973 Program of China NO. 2011CB302301, NSFC NO. 61272483, 61232003, 61173043, and Training Abroad Program of China Scholarship Council (CSC).

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, ser. SOSP, 2003, pp. 29–43.

[2] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC, 2010, pp. 267–280.

[3] D. Talbot, “A smarter algorithm could cut energy use in data centers by 35 percent,” Technical Article, 2013. [Online]. Available: http://www.technologyreview.com/news/513656/a-smarter-algorithm-could-cut-energy-use-in-data-centers-by-35-percent/,

[4] D. Borthakur, “The hadoop distributed le system: Architecture and design,” Technical Report, 2009. [Online]. Available: http://hadoop.apache.org/common/docs/current/hdfs-design.html,

[5] 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 ATC*, 2012.

[6] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O’Shea, and A. Donnelly, “Symbiotic routing in future data centers,” *SIGCOMM*, vol. 40, no. 4, pp. 51–62, 2010.

[7] C. E. Leiserson, “Fat-trees: universal networks for hardware-efficient supercomputing,” *Computers, IEEE Transactions on*, vol. 100, no. 10, pp. 892–901, 1985.

[8] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, “Dcell: a scalable and fault-tolerant network structure for data centers,” in *SIGCOMM*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[9] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, “Bcube: a high performance, server-centric network architecture for modular data centers,” *SIGCOMM*, vol. 39, no. 4, pp. 63–74, 2009.

[10] D.-Z. Du, B. Lu, H. Q. Ngo, and P. M. Pardalos, “Steiner tree problems.” *Encyclopedia of optimization*, vol. 5, pp. 227–290, 2009.

[11] H. Takahashi and A. Matsuyama, “An approximate solution for the steiner problem in graphs,” *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.

[12] C. Blum, “Ant colony optimization: Introduction and recent trends,” *Physics of Life reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[13] Y. Liu, M. Wu, and J.-x. Qian, “A novel distributed multicast routing algorithm based on ant colony algorithm,” *Journal of Circuits and Systems*, vol. 5, p. 024, 2008.

[14] V. Rayward-Smith, “The computation of nearly minimal steiner trees in graphs,” *International Journal of Mathematical Education in Science and Technology*, vol. 14, no. 1, pp. 15–23, 1983.

[15] V. J. Rayward-Smith and A. Clare, “On finding steiner vertices,” *Networks*, vol. 16, no. 3, pp. 283–294, 1986.

[16] P. Costa, A. Donnelly, G. Oshea, and A. Rowstron, “Camcube: a key-based data center,” Technical Report MSR TR-2010-74, Microsoft Research, Tech. Rep., 2010.

[17] P. Costa, A. Donnelly, A. Rowstron, and G. OShea, “Camdoop: Exploiting in-network aggregation for big data applications,” in *NSDI*, vol. 12, 2012.

[18] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data center tcp (dctcp),” *SIGCOMM*, vol. 40, no. 4, pp. 63–74, 2010.

[19] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: measurements & analysis,” in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 202–208.

[20] C. Huang, M. Chen, and J. Li, “Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems,” in *Network Computing and Applications*. IEEE, 2007, pp. 79–86.

[21] A. Duminuco and E. Biersack, “Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems,” in *P2P*. IEEE, 2008, pp. 89–98.

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

[23] Y. Hu, H. C. Chen, P. P. Lee, and Y. Tang, “Nccloud: Applying network coding for the storage repair in a cloud-of-clouds,” in *FAST*, 2012.

[24] Z. Huang, E. Biersack, and Y. Peng, “Reducing repair traffic in p2p backup systems: exact regenerating codes on hierarchical codes,” *TOS*, vol. 7, no. 3, p. 10, 2011.

[25] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, “Rethinking erasure codes for cloud file systems: Minimizing i/o for recovery and degraded reads,” in *FAST*, 2012.

[26] J. Li, X. Wang, and B. Li, “Cooperative pipelined regeneration in distributed storage systems,” in *INFOCOM*.

[27] ——, “Pipelined regeneration with regenerating codes for distributed storage systems,” in *NetCod*. IEEE, 2011, pp. 1–6.