An Ant Colony Optimization Based Data Update Scheme for Distributed Erasure-Coded Storage Systems

YUPENG HU1, (Senior Member, IEEE), QIAN LI1, WEI XIE2, AND ZHENYU YE1

1College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
2School of Journalism and Communication Film and Television Arts, Hunan University, Changsha 410082, China

Corresponding author: Wei Xie (tntxie@hnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872130, Grant 61872133, Grant 61672221, and Grant 61632009, in part by the Science and Technology Project of Department of Communications of Hunan Provincial under Grant 201928, in part by the Key Research and Development Projects of Changsha under Grant qk1907103, in part by the Natural Science Foundation of Hunan Province under Grant 2019JJ50067 and Grant 2020JJ4008, and in part by the Fundamental Research Funds for the Central Universities under Grant 531118040067.

ABSTRACT Owing to the high availability and space-efficiency of erasure codes, they have become the de facto standard to provide data durability in large scale distributed storage systems. The update-intensive workloads of erasure codes lead to a large amount of data transmission and I/O. As a result, it becomes a major challenge to reduce the amount of data transmission and optimize the use of existing network resources so that the update efficiency of the erasure codes could be improved. However, very little research has been done to optimize the update efficiency of the erasure codes under multiple QoS metrics. In this paper, our proposed update scheme, the Ant Colony Optimization based multiple data nodes Update Scheme (ACOUS) employs a two-stage rendezvous data update procedure to optimize the multiple data nodes updates. Specifically, the two-stage rendezvous data update procedure performs the data delta collection and the parity delta distribution based on a multi-objective update tree which is built by the ant colony optimization routing algorithm. Under typical data center network topologies, extensive experimental results show that, compared to the traditional TA-Update scheme, our scheme is able to achieve a 26% to 37% reduction of update delay with convergence guarantee at the cost of negligible computation overhead.

INDEX TERMS Distributed storage system, data update, erasure codes, ant colony optimization.

I. INTRODUCTION
A. MOTIVATION
Owing to the high availability and space-efficiency of erasure codes, they have become the de facto standard to provide data durability in distributed storage systems and data centers [1]–[4]. In the erasure code schemes, a large data object is divided and then encoded into multiple data blocks as well as parity blocks, which may be deployed across the nodes of distinct storage clusters.

The data updates are common in distributed storage systems [5]–[8]. For many enterprise servers and network file systems, the update requests dominate the write workloads (often more than 90%) [9], [10]. To keep the data consistency, an update request to a data block inevitably involve \((n - k)\) updates to parity blocks in a typical \((n, k)\) Maximum Distance Separable (MDS) erasure-coded system. According to whether transmitting the whole data block, the update schemes could be divided into two categories: raid-based update and delta-based update. Raid-based update schemes need to transmit the entire block of data among data nodes and parity nodes, i.e., to finish an update on a data node, the data node needs to collect all data blocks and then recalculate all parity blocks which will be delivered to the parity nodes. In comparison, the delta-based update schemes can achieve more I/O and network bandwidth savings, because an update on a data node can be achieved by broadcasting the delta (the parts to be modified of data blocks) to all parity nodes. Consequently, frequent data updates incur tremendous I/O and bandwidth overhead in large scale cloud storage systems and distributed in-memory Key-Value (KV) stores, e.g., small-sized KV set to key/value data causes expensive I/O operations and network traffic [7], [11]–[13], [26], [27]. Therefore, it is significant to improve the update efficiency for erasure codes. Motivated thereby, a plethora of efforts have been devoted recently to optimize the updating performance both in reducing I/Os and network transmission latency [6], [14]–[16]. Existing update schemes for erasure...
codes, such as Azure [17] and CodFS [14], adopt a log-based data update or a hybrid of in-place data updates and log-based parity updates to reduce I/Os by sequentially appending updates. Alternatively, the authors in [6], [15], [16] try to mitigate the network transfer overhead by optimizing the update schedule and procedure.

It is desirable, from a network performance point of view, to ensure that the data delta are delivered along an optimal path from the data nodes to the parity nodes. The current routing algorithms can reliably handle large amounts of traffic for distributed storage systems [18], [19]. Nevertheless, in the data update scenario, the update routing techniques have not drawn abundant research attentions for heterogeneous storage systems, which are composed of nodes with unequal I/O throughput, link bandwidth, and other QoS constraints, leading to inefficient use of resources. The significance of the update routing for large scale distributed storage systems thereby motivates us to carry out in-depth investigation on the data update routing for the erasure-coded storage systems.

B. OUR CONTRIBUTIONS
In this paper, we are dedicated to optimizing the data update mechanism with a multi-objective routing algorithm in terms of the memory I/O throughput and other QoS metrics, such as the delay and bandwidth requirement. Our proposed Ant Colony Optimization based multiple data nodes Update Scheme (ACOUS) adopts a two-stage rendezvous data update procedure to optimize the multiple data nodes updates and routing, which performs the data delta collection and the parity delta distribution based on an update tree that is built by the Multi-objective Ant Colony Optimization Update routing algorithm (MACOU).

In its first stage of the two-stage rendezvous data update procedure, the update is triggered by the data nodes for updates. There may be multiple data nodes (up to \( k \)) for updates, while only one of them will be selected as the rendezvous node that collects data delta sent out from all other data nodes for update, and computes the parity delta which in turn will be distributed to the corresponding parity nodes in the second stage. As we will explain in Section IV, the benefit of choosing the rendezvous pattern is that each data node for update doesn’t have to distribute its \( \Delta d_i \) to \( r \) parity nodes. Moreover, Section V provides details on how to select the rendezvous node.

In the second stage, the rendezvous node delivers all parity delta, which are deduced from the all data delta, to the corresponding parity nodes based on a multi-objective update tree built by the MACOU.

Our main contributions are summarized as follows:
1) To the authors’ best knowledge, the ACOUS is the first work to provide a thorough study of the multiple data nodes updates and routing under heterogeneous erasure-coded storage systems, in view of the unequal memory throughput and link bandwidth across nodes. ACOUS improves the overall data update efficiency based on the multi-objective ant colony optimization routing algorithm. Except for its wide application to solving QoS routing problems [31], the real advantage of adopting the ant colony optimization is that it can be used for our update routing search issue without any prior model information with fast convergence.
2) As shown in Table 1, we illustrate the multiple data nodes update routing problem by quantitatively analyzing the I/O and data transmission overhead of data update for the rendezvous and distributed update pattern, respectively.

### TABLE 1. The overhead of distributed and rendezvous update pattern when there are \( u \) data nodes for update.

| Update Pattern | Read(\text{Local}) | Transmission | Write(\text{Local}) |
|----------------|---------------------|--------------|---------------------|
| Distributed    | \( u + r \)         | \( u \times r \) | \( u + r \)         |
| Rendezvous     | \( u + r \)         | \( u - r - 1 \) | \( u + r \)         |

3) We develop a prototype and perform extensive experiments to evaluate the performance of the ACOUS under typical data center network topologies. Experimental results show that the ACOUS outperforms the traditional TA-Update scheme [5] and improves the update efficiency significantly with convergence guarantee.

The rest of this paper is organized as follows. Section II and Section III present the background and challenges and related work, respectively. Section IV illustrates the multiple data nodes updates routing problem. The ACOUS is introduced in Section V. We conduct extensive experiments to evaluate our scheme in Section VI. Section VII concludes this paper.

II. BACKGROUND AND CHALLENGES

A. BACKGROUND

Erasure codes: The need for both durability and storage efficiency makes erasure codes a new desirable research target. In the well-known erasure code \( RS(n, k) \) [20], a file of size \( D \) bytes is divided into \( k \) equal-sized data blocks \( d_i \) (\( 1 \leq i \leq k \)), each of which is of size \( [D/k] \) bytes. These data blocks are then encoded into a set (also called stripe) of \( k \) data blocks and \( (n-k) \) redundant parity blocks, which are distributed in \( n \) different storage nodes \( (D_1, \ldots, D_k; P_1, \ldots, P_{n-k}) \), belonging to different fault domains or clusters so as to maximize the system reliability. Each parity block \( p_j \) (\( 1 \leq j \leq (n-k) \)) could be calculated over a finite field according to the Eq.(1):

\[
p_j = \sum_{i=1}^{k} a_j^i \times d_i \tag{1}
\]

where \( a_j^i \) denotes the coefficient for \( p_j \) from \( d_i \). Based on this linear coding, any \( k \) blocks out the \( n \) blocks suffice to reconstruct the entire original file. An update on a data node can be obtained by broadcasting its delta to all parity nodes and letting them add the delta to parity with a predefined coefficient to keep the data consistency. Let \( \Delta d_i = d_i' \oplus d_i \) represent the data delta, which will be sent out by the node for update after overwriting the original \( d_i \) with new data \( d_i' \). Assume there are \( u(1 \leq u \leq k) \) data nodes for updates, each parity node computes the updated \( p_j' \) according to the Eq.(2)
where the probability of ants transferring from node $i$ to node $j$ at time $t$ will be affected by the amount of pheromone on this path as follows:

$$p_{ij} = \sum_{u=1}^{m} d_{ij}^u \times \Delta d_{ij} + p_j = \Delta p_{ij} + p_j \quad (2)$$

Ant colony optimization algorithm: Ant colony algorithm is a well-known heuristic optimization algorithm, which originates from the cooperation between ants when the environment changes. It has been employed to solve a number of combinatorial optimization problems, such as the shortest path problem, optimal task assignment problem, QoS routing problem, and so on [31]. The ant colony algorithm’s fundamental principle is that each ant will leave pheromone during searching to form a positive feedback mechanism. In this way, an ant could quickly find the shortest path to a food source by perceiving the pheromone left by other ants.

From time $t$ to $t+1$, the amount of pheromone on the path $(i,j)$ from node $i$ to node $j$ can be calculated according to the following Eq.(3) and Eq.(4):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \quad (3)$$

$$\Delta\tau_{ij}^k(t) = \frac{\tau_{ij}(t)\phi_{ij}^k(t)\eta_{ij}^k(t)}{\sum_{s \in \Phi_{ij}} \tau_{is}(t)\phi_{is}(t)\eta_{is}^k(t)} \quad (4)$$

where the $\rho$ is the volatilization coefficient of pheromone, $\Delta\tau_{ij}^k(t)$ denotes the amount of information left by the $k$th ant in the path from node $i$ to $j$, and the $\Delta\tau_{ij}$ is the increment of pheromone on path $(i,j)$ in certain generation $m$. The probability of ants $k$ transferring from node $i$ to $j$ at time $t$ will be affected by the amount of pheromone on this path and the global heuristic factor $\eta$, which is denoted by $p_{ij}^k(t)$ as follows:

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)\phi_{ij}^k(t)}{\sum_{s \in \Phi_{ij}} \tau_{is}(t)\phi_{is}(t)} \quad (5)$$

where the $\phi_{ij}(t)$ represents the set of next-hop candidates of node $i$, $\eta_{ij}$ is the expectation of node $i$ to node $j$, while the $\alpha$ and $\beta$ are used to adjust the effect of function of pheromone and heuristic function, respectively. In this way, the pheromone on each path will increase or decrease as time goes on. The more ants passing through the path, the more pheromone on this path will be over a period of time. Therefore, all ants can rapidly find an optimal path to the destination in a convergent and positively feedback manner.

B. CHALLENGES FOR DATA UPDATE

Executing erasure codes updates in large scale distributed storage systems may result in significant performance degradation. A data update involves updates to multiple parity nodes, inevitably leading to considerable I/O and bandwidth consumption. In particular, most of the write requests are long-time latency update operations for the distributed storage systems, such as cloud storage systems, distributed in-memory KV stores, and block storage systems. The well-known cloud storage systems Azure [27] has adopted the log-based update schemes to amortize the update overhead. Update operation in a distributed KV store may result in several small updates, which would introduce notable coding operations and network traffic. For example, in Memcached-based Cocytus [11], all value parities distributed among multiple parity nodes need to be frequently modified for the intensive small-sized update operations, i.e., set operation. For the erasure-coded disk array or block storage system files, such as structured database files and virtual machine file system (VMFS) volumes, an update to a file may lead to multiple blocks updating operations among different clusters simultaneously [8].

Therefore, below are two critical challenges that are not being addressed well by existing update schemes:

- The first challenge is that when there are multiple data nodes for updates, the collaboration among these nodes will incur a large amount of finite field arithmetic and network traffic. According to Eq.(2), the delta and coding computation can be executed efficiently at all data nodes for updates in a parallel nature. However, it is still challenging how to coordinate the data nodes for updates under specific update order/pattern to deliver their $\Delta p_{ij}$ to the corresponding parity nodes with minimum network traffic.

- The second challenge is how to find an optimal delivery path for each update delta in a large heterogeneous scale networked storage system, with regard to the unequal I/O throughput, link bandwidth across nodes, and other QoS constraints. Fundamentally, this problem is a multi-objective optimization for routing discovery which makes full use of network resources, to improve the data update efficiency.

III. RELATED WORK

In this section, we review some of the related works that are the starting point of our research.

Update schemes of erasure codes: Nowadays, there are two update schemes of erasure codes: Raid-based update scheme and delta-based update scheme. The raid-based-update scheme needs to transmit the whole block between data nodes and parity nodes, and the delta-based scheme only transfers the different parts of the original data block and the new data block. Delta-based update scheme is thus employed in our work due to its network efficiency compared with a raid-based update. There are three update patterns in delta-based update schemes: in-place update, log-based update, and hybrid update.

In-place update schemes realize real-time updates of data by overwriting the original data blocks and parity blocks with new data simultaneously. In-place update schemes can ensure data consistency and recovery efficiency, which are adopted widely in erasure-coded storage systems. However, it introduces considerable I/O and transmission overhead to complete the update of parity blocks [5].

In contrast, the log-based update schemes save I/O overhead of parity block and improve the update efficiency by appending the new data delta to both the data blocks and parity blocks, where the appended information will be merged at a given time. Many enterprise erasure-coded systems, such as GFS [25], HDFS [26], Azure [27], have adopted the log-based update schemes. However, the log-based update
schemes become the bottleneck for the storage systems that are dominated by the read requests since the data combination operations have to be executed once accessing data.

By allowing the asynchronous updates for data blocks and parity blocks, the hybrid update schemes could ensure the access efficiency for the data blocks and the update efficiency for the parity blocks. The hybrid update schemes, such as Parity-Logging (PL) [29] and Parity-Logging with Reserved space (PLR) [28], overwrite the data blocks immediately and append the parity delta to the parity blocks. To lower the update frequency and cost, the RAPID protocol [16] introduces an update window for data updates, which works by choosing a subset of parity blocks for updates based on the predicted number of failures.

To summarize, in the above three update patterns, the data nodes have to deliver the data delta to parity nodes for data consistency. Unlike existing methods, in addition to optimizing the I/O overhead in the updating process, the ACOUS is dedicated to improving the data transmission efficiency and save bandwidth concerning multiple QoS metrics.

**Ant colony based routing algorithm:** QoS routing with multiple constraints is an NP-C problem [31]. Ant colony algorithm has been successfully applied to solve many discrete optimization problems, owing to its robustness and convergence. The state-of-the-art ant colony optimization has been widely applied to solving QoS routing problems. In [24], the ant colony optimization based routing technology is used to adopt the node’s real-time location and load parameters as routing metrics to improve the QoS performance of the network. The authors in [22] propose an improved version of the well-known dynamic source routing scheme based on the ant colony optimization algorithm, which can produce a high data packet delivery ratio in low delay with low routing overhead and less energy consumption. In [23], the authors exploit the ant colony optimization scheme to improve the information-centric network routing performance. Also, the Aggrecode proposed in [33] aims to improve reconstruction performance for erasure-coded storage systems by adopting two heuristic routing algorithms based on ant colony optimization.

There are a few research works on update routing algorithms for erasure codes. Among available data update schemes, the most relevant but not identical one to ACOUS is the TA-Update proposed in [5], which adopted the hops-based minimum spanning tree to route the update information to all parity nodes for each data node for an update. In contrast, ACOUS resolves the multiple data nodes updates routing issue based on the multi-objective ant colony optimization routing algorithm.

**IV. MULTIPLE DATA NODES UPDATES ROUTING PROBLEM ANALYSIS FOR ERASURE CODES**

**A. TWO PATTERN OF MULTIPLE DATA NODES UPDATES**

With regard to the $RS(k + r, k)$ based storage system, as Figure 1 shows, there are two practical update patterns, i.e., distributed and rendezvous. Figure 1(a) shows the distributed pattern in which each data node for update distributes its $\Delta d_i (1 \leq i \leq k)$ to all parity nodes which perform a local update later according to Eq.(2). The Figure 1(b) depicts the rendezvous pattern, in which each data node $D_i$ for an update first sends its $\Delta d_i$ to the given rendezvous node, which subsequently calculates all $\Delta p_j(1 \leq j \leq r)$ to be sent to the corresponding parity node $P_j$ separately.

![Diagram](image.png)

**FIGURE 1.** Two update patterns for multiple data nodes.

According to Eq.(2), Table 1 presents the overhead of data transmission and I/O incurred by $RS(k + r, k)$ in the two update patterns respectively when there are $u(1 \leq u \leq k)$ data nodes for update and $r$ parity nodes. As shown in Table 1, the I/O cost of distributed pattern and rendezvous pattern are almost equal with regard to the parallel computing manner. Nevertheless, the data transmission overhead incurred in the distributed pattern is $a(u + r - 1)$ only if the given rendezvous node itself is also a data node for update. If $u > 1$ and $r > 1$, we have $u \times r > (u + r - 1)$, hence the data transmission overhead incurred in the rendezvous pattern will rapidly exceed the one in rendezvous pattern with the increase of $u$. Thereby, in this paper, we adopt the rendezvous pattern to accomplish the multiple nodes update.

**B. QoS METRICS OF UPDATE ROUTING**

Our design goal is to improve the efficiency of multiple data nodes updates for large scale erasure-coded storage systems with regard to the typical QoS metrics below.

**Distance:** The distance $D(s, d)$ between node $s$ and $d$ can be measured by the number of hops between them, which is usually utilized to construct a spanning tree [5] for transmission. The shorter network distance indicates the fewer hops of data transmission and the less transmission delay it takes.
**Bandwidth:** Bandwidth is the metric measuring the link capacity in the network. In general, the bottleneck of data transmission is determined by the minimum bandwidth over the delivery path. Since the available real-time bandwidth is difficult to obtain, we utilize the average bandwidth $B(e)$ as the link bandwidth of link $e$. The bottleneck bandwidth $B(s, d)$ on path $(s, d)$ is the available minimum average bandwidth given in Eq. (6) below:

$$B(s, d) = \min\{B(e), e \in \text{path}(s, d)\}$$

which needs to exceed the minimum delivery bandwidth requirement $B_{req}$ to avoid network congestion, i.e., $B(s, d) \geq B_{req}$. The practical $B_{req}$ can be determined according to the specific application traffic requirement detected via some network analyzer, e.g., Wireshark (https://www.wireshark.org/).

**Delay:** Delay is the key metric adopted in this paper to measure the update efficiency. Less delay indicates higher transmission efficiency. As shown in Eq. (7), the total delay $delay(s, d)$ over the path $\text{path}(s, d)$ is simply the sum of I/O processing delay $d_{\text{proc}}$, transmission delay $d_{\text{trans}}$, and propagation delay $d_{\text{prop}}$ over it. The transmission delay is measured by the duration of transmitting a given packet concerning the bandwidth constraint. The propagation delay is always negligible. It is worth noting that the queuing delay is omitted here and is beyond our discussion.

$$delay(s, d) = \sum_{e \in \text{path}(s, d)} (d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

**V. ANT COLONY OPTIMIZATION BASED MULTIPLE DATA NODES UPDATE SCHEME**

Based on the above QoS metrics, in this section, we illustrate the design of our proposed update scheme named ACOUS in detail. We first introduce the two-stage rendezvous data update procedure and then present the built-in multi-objective optimization update routing algorithm. Moreover, the rendezvous data selection mechanism is also discussed.

**A. THE TWO-STAGE RENDEZVOUS UPDATE SCHEME**

The main idea of the proposed ACOUS is to adopt a two-stage rendezvous update scheme to perform efficient data delta collection and parity delta distribution for RS$(k + r, k)$ via a multi-objective update tree.

The first stage: data delta collection. Figure 2(a) describes the first stage of the rendezvous update, given the Rendezvous Node (RN) $D_2$. If there are $u(u \leq k)$ data nodes for update, each one updates its own original data blocks $d_i(1 \leq i \leq k)$ as new data $d'_i$ directly, meanwhile calculating the data delta $\Delta d_i(\Delta d_i = d_i \oplus d'_i)$ which will be delivered to the rendezvous node $D_2$. It is worth noting that the RN is also a data node for update. To improve the update efficiency, each $\Delta d_i$ will be delivered to the RN via a delta collection tree formed by the proposed built-in MACOU routing algorithm introduced in the next subsection. In this way, it needs overall $(u - 1)$ data blocks transmission, $u$ local reads, and $u$ writes for the first stage to finish the data delta collection.

**FIGURE 2. The two-stage rendezvous update scheme.**

The second stage: parity delta distribution. Figure 2(b) describes the second stage of rendezvous update. Based on the $\Delta d_i$ in the first stage, the RN then is able to obtain each parity delta $\Delta p_j = \sum_{i=1}^{u} a'_i \times \Delta d_i, (1 \leq j \leq r)$ which will be distributed to the corresponding parity node $P_j$ in turn along with the multi-objective update tree constructed by the MACOU algorithm. Afterwards, each parity node updates its original parity $p_j$ as $p'_j = \Delta p_j + p_j$. In this way, it needs $r$ parity blocks transmission, $(r + 1)$ reads and $r$ writes for the second stage to accomplish the parity delta distribution.

**B. MULTI-OBJECTIVE ANT COLONY OPTIMIZATION UPDATE ROUTING ALGORITHM**

Algorithm 1 illustrates our proposed MACOU algorithm built in the ACOUS to search the path with the least delay from node $s$ to node $d$ subject to the bandwidth constraint. Table 2 shows the notations involved in Algorithm 1. The issue of searching the optimal update path between node $s$ and node $d$ can be defined as follows:

$$\arg \min_{e \in \text{path}(s, d)} \{delay(s, d), \ s.t. B(e) \geq B_{req}\}$$

As the Algorithm 1 shows, based on the inputs, the MACOU first initializes the values of each parameter in code line 1-2 and calculates the distance of each node to the destination node in code line 3. The key iterative search process of $\text{path}(s, d)$ is shown from code line 4 to 24. The ants are sent out in generations, and the number of each generation is called antnum. During the ant searching process, if current node $i$ is not the destination node, the ant will move to the next node $j$, which meets the available bandwidth $B(i, j) > B_{req}, j \in \phi(i)$, with a certain probability. The ants leave pheromone on their travel path, depending on the path quality. According to the Eq.(5), if the node $v(v \in \phi(i))$ is the next node to be visited, the expectation $\eta_{ij}$ can be obtained by
Table 2. Ant colony algorithm parameters.

| Notation | Representation |
|----------|----------------|
| $W_i$    | The weight of node $i$ |
| $W_e$    | The weight of edge $e$ |
| $p_{ij}$ | The probability of moving from node $i$ to $j$ |
| $\tau_{ij}$ | The pheromone quantities on edge $e(i,j)$ |
| $\alpha$ | The distance between node $i$ and $j$ |
| $\beta$  | The pheromone effect adjustment parameter |
| $\lambda$ | The delay effect adjustment parameter |
| $M$      | The number of ants per generation (antnum) |
| $K$      | The number of generations |
| $D_{i,j,k}$ | The distance between node $j$ and $d$ |
| $pt(m, k)$ | The path of $k$th ant in the $m$th generation |
| $sd(m, k)$ | The delay of $k$th ant in the $m$th generation |
| $\phi(i)$ | The neighbors of node $i$ |

Eq.(9),

$$
\eta_{ij} = (1/D(j, d))^\beta \times (1/(W_j + W_e))^\lambda, \quad e \in \text{path}(i, j) \quad (9)
$$

where the weight of node $i$ is its $d_{\text{proc}}$, the weight of edge $e$ is the sum of its $d_{\text{trans}}$ and $d_{\text{prop}}$, and the distance $D(j, d)$ is the key parameter indicating the hops and latency. Then the node $v$ will be added into the path $\text{path}(m, k)$ in code line 12. The code line 16 is used to judge if the search is convergent or not. An indicator of search convergence is that the path is constant in a number of iterations. Afterwards, the pheromone is regularly updated in code lines 18-19, according to Eq.(3) and Eq.(4). Consequently, the multi-objective update tree for the RN can be set up based on the MACOU.

Algorithm 1 Multi-Objective ant Colony Optimization Update Routing Algorithm (MACOU)

Require:
- Network topology $G(V, E)$
- The $W_{i}(i \in V)$ and $W_{e}(e \in E)$
- Bandwidth $B(i, j)$ of every edge and minimum bandwidth $B_{\text{req}}$ constraint
- Source node $s$ and destination node $d(s, d \in V)$

Ensure:
- $\text{path}(s, d)$ with minimum delay subject to the minimum bandwidth constraint;

1: initialize $\alpha$, $\beta$, $\lambda$, $m$, $i \leftarrow s$, converge $\leftarrow$ false;
2: initialize the pheromone $\tau$ on each path;
3: calculate the distance $D(i, d)$, $(i \in V)$ from each node to the destination node;
4: for $m = 1$ to $M$ do
5:     for $k = 1$ to $K$ do
6:         while $! = d$ or $\phi(i)$ is not empty do
7:             for each $j, k \in $ do
8:                 calculate $P_{ij}$;
9:             end for
10:         end while
11:     end for
12:     select max $P_{iv}, v \in \phi(i)$ from $P_{ij}$ with $B_{\text{req}}$;
13:     if $v = = d$ then
14:         save $P_{m, k}$ and $sd(m, k)$;
15:         converge $\leftarrow$ isconverge($pt$);
16:         if converge $\leftarrow$ true then
17:             update pheromone on all edges;
18:             $i \leftarrow v$;
19:         else
20:             return $P_{m, k}$ and $sd(m, k)$;
21:         end if
22:     end if
23:     end for
24: end for
25: end for

The weight value on each edge represents its available average bandwidth (Mbps). Assume the minimum delivery bandwidth requirement is $B_{\text{req}}=50$Mbps, we need to find an update tree with minimum delay, each edge $e$ of which should satisfy $B(e) \geq 50$Mbps. Our proposed update routing algorithm MACOU could be undertaken during the network initialization. Hence, each data node can keep the optimal routing to each parity node for some time until the next network reconfiguration. Thereby, as we can see from Figure 3(a), the edges in bold and all nodes construct the multi-objective update tree for $V_1$.

Specifically, the Figure 3(b) depicts the search steps of path $\text{path}(V_1, V_7)$ in our scheme. Let the ant start from $V_1$, in the same case of the initial pheromone, since the $B(V_1, V_2)$ is less than 50Mbps, the next hop is $V_3$. Similarly, the ant then starts from $V_3$, the distance $D(V_2, V_7)$ is 2, distance $D(V_4, V_7)$

A case study: A case shown in Figure 3 illustrates the efficiency of the MACOU algorithm. Figure 3(a) shows the multi-objective update tree for $V_1$. $V_1$ is the rendezvous node and the other nodes are parity nodes. The weight value on each node denotes the distance (hops) and I/O processing delay. In practical distributed storage systems, all the nodes are connected with the tree topology, where some nodes are connected with a lower layer switch, and the lower layer switches are connected with the higher layer switches. All the paths between nodes are all relayed through the switches.
and distance $D(V_5, V_7)$ is 1. Meanwhile, the delay of $V_5$ is greater than $V_4$. As a result, $V_4$ is selected as the next hop. Finally, the $V_4$ can reach $V_7$ directly. Based on the positive feedback mechanism of the proposed MACOU, the more number of ants that choose the path: $V_1 \rightarrow V_3 \rightarrow V_4 \rightarrow V_7$, the more pheromone on this path will be. Therefore, in several iterations, the path from $V_1$ to $V_7$ ultimately converges to $V_1 \rightarrow V_3 \rightarrow V_4 \rightarrow V_7$ satisfying the minimum update delay and the bandwidth constraint. Similarly, the remaining optimal update paths for other parity nodes ($V_2$ to $V_6$) can then be found out to construct the update tree in Figure 3(a).

It is worth noting that the data delta collection tree in the first stage of ACOUS also can be constructed in the same way as the distribution tree by MACOU.

Compared with the update tree in red dashed edges formed by the Prim’s algorithm-based TA-Update scheme in Figure 3(a), the MACOU can save more update delay. For instance, due to the wider bandwidth, the path $(V_1, V_2)$ searched by our MACOU will be $V_1 \rightarrow V_3 \rightarrow V_2$, which is more efficient than the delivery path $V_1 \rightarrow V_2$ selected by TA-Update, and the overall delay gap between the two paths is monotonously increasing along with the size of data packet, e.g., the delay of delivering data packets of size 1Mb to 9Mb on path $V_1 \rightarrow V_3 \rightarrow V_2$ and $V_1 \rightarrow V_2$ is 0.03ms to 0.27ms and 0.033ms to 0.3ms, respectively. It is worth noting that if the node $V_3$ is congested to some degree by multiple parities delta, MACOU will select a better path to deliver owing to its adaptability, e.g., $V_1 \rightarrow V_2$.

C. THE RENDEZVOUS NODE SELECTION

The optimal node selection is an NP-hard problem with regard to the multiple QoS metrics. The ACOUS simplifies the rendezvous node selection in a random or delay efficient manner to achieve the Random Rendezvous Node (RRN) or the delay Optimal Rendezvous Node (ORN), respectively.

In our implementation, we randomly assign a stripe head in each stripe to perform the node selection algorithm. According to the node address, the RRN is randomly selected from the data nodes for updates by the stripe head. The RRN may not be the optimal one; nevertheless, its selection process is more efficient than the ORN’s. The ORN is the optimal node that performs the data delta collection and parity delta distribution at the cost of minimum delay when there are $u$ nodes for updates, as the Algorithm 2 shows. The problem of choosing ORN can be defined as arg min$_{i,j \in V}$ \{sumdelay$[i]$, where the sumdelay$[i]$ is given as follows:

$$\text{sumdelay}[i] = \sum_{k=1}^{u} \text{delay}(D_i, D_k) + \sum_{j=1}^{r} \text{delay}(D_i, P_j) \quad (10)$$

As shown in Figure 4, we experiment with our prototype to verify the update delay efficiency of the ORN. As we can see, the ORN becomes more efficient than RRN with the increase of the number of nodes for updates at the cost of rendezvous node selection computation. It takes no more than 0.01s to perform the selection procedure in Algorithm 2 in our $30+$ nodes prototype storage system based on the routing information obtained by MACOU. It is worth noting that the routing information obtained by MACOU for each data node will not be frequently updated during a period until the network reconfiguring. Thus, it would be preferable to adopt the ORN for the large scale storage system in view of the universal available computing resources. Given the negligible computation cost, more evaluations on the update delay of the two different rendezvous nodes are provided in section VI. Moreover, considering the random update sequence of the nodes, the ORNs are substantially uniformly distributed on different nodes and won’t become the bottleneck.

![FIGURE 4. Comparisons of update delay under different RNs.](source:https://github.com/599020328/acous)

**Algorithm 2 ORN Selection Process**

**Require:**
- Network topology $G(V, E)$;
- The $W_i (i \in V)$ and $W_e (e \in E)$;
- Bandwidth $B(i, j)$ of every edge;
- Updated node set $D'$ and parity node set $P$;

**Ensure:**
- ORN
1. initialize $\text{sumdelay}[i] \leftarrow 0$
2. for each $D_i$ in $D'$ do
3. for each $D_k (D_k \neq D_i)$ in $D'$ do
4. calculate $\text{delay}(D_i, D_k)$ with Algorithm 1 MACOU;
5. $\text{sumdelay}[i] \leftarrow \text{sumdelay}[i] + \text{delay}(D_i, D_k)$;
6. end for
7. for each $P_j$ in $P$ do
8. calculate $\text{delay}(D_i, P_j)$ by Algorithm 1 MACOU;
9. $\text{sumdelay}[i] \leftarrow \text{sumdelay}[i] + \text{delay}(D_i, P_j)$;
10. end for
11. end for
12. select minimum $\text{sumdelay}[i]$;
13. $\text{ORN} \leftarrow D_i$;
14. return $\text{ORN}$;

VI. PERFORMANCE EVALUATIONS

We implement our update scheme and develop a prototype on Tencent cloud virtual machines based storage clusters to

1. source code:https://github.com/599020328/acous
evaluate our scheme; meanwhile, we perform extensive simulations on larger scale network topologies (≥ 200 nodes) for further investigation. The data nodes and parity nodes belong to the same stripe are randomly deployed in different clusters. Since the delay of rendezvous node selection operations in ORN are typically in microseconds and becoming neglectable compared to the data update delay, we ignore the computation time in our evaluation below. The key parameters are listed in Table 3.

A. PROTOTYPE EVALUATIONS

Our prototype storage clusters consist of up to 35 Tencent cloud virtual machines (Ubuntu Server 14.04.1 LTS, 4GB RAM and dual-core Intel Xeon Cascade Lake (2.5 GHz)) that all are connected through 1.5Gbps network in the same subnet. We exploit the performance monitor tool of Tencent cloud to estimate the average bandwidth. This prototype’s experimental results are based on the workloads collected from various online datasets, such as small-sized pdf files, transaction logs, and large-sized NoSQL database files and videos.

Figure 5 provides extensive comparisons of the update delay under different schemes. Figure 5(a) shows how the update delay increases along with $k$ when $r = 3$. Figure 5(b) presents how the update delay increases along with $r$ when $k = 5$. As the number of data nodes or parity nodes increases, there are more data delta and parity delta that will be delivered efficiently along the optimal update tree in our scheme. Thus, the saving of update delay of our scheme becomes more apparent when the $k$ or $r$ increases. Our schemes show a 30% to 32% delay decrease compared to the TA-Update.

Figure 5(c) shows the update delay under a varying number of nodes of our prototype system when there are only 5 data nodes and 5 parity nodes for updates, which are randomly deployed in distinct storage clusters. As the number of nodes increases, we arrange them into more storage clusters for load balance, leading to more delivery hops and data packets relaying. Therefore, the delay saving achieved by our scheme with ORN and MACOU increases rapidly from 28% to 37% or so as the storage system scales out. Figure 5(d) clearly illustrates the advantage of our scheme when the size of the data delta grows. The delay reduction that our scheme can obtain is increasing almost linearly along with the size of data delta.

B. LARGE SCALE NETWORK SIMULATIONS

We perform extensive experiments on the well-known component-based simulator OPNET [32] to evaluate our update scheme in a large heterogeneous scale distributed storage network system, which implements the update periodically. In addition to the random topology with a random rate of I/O and bandwidth, we evaluate our update scheme on the other two typical data center network topologies, as shown in Figure 6, i.e., Fat tree and DCell, both of which has high network capacity and robust connectivity. We also implement the TA-Update scheme as the baseline under the same conditions. It is worth noting that the workloads run on OPNET are based on the traces of SETI@home Desktop Clouds.

FIGURE 6. Two typical data center network topologies.

1) UPDATE DELAY UNDER VARYING AMOUNT OF DATA TRANSMISSION

This subsection presents the simulation results on update delay under the varying amount of data transmission. We first compare different update schemes under the varying size
of data delta $\Delta d$. As shown in Figure 7, the update delay increases along with the size of data delta. Our scheme utilizing ORN and MACOU outperforms the other two schemes and can save about 26% delay time compared to the traditional TA-Update. Figure 7 shows an 18% or so improvement of the scheme adopting ORN and MACOU over the scheme using RRN and MACOU.

Figure 8 illustrates how the update delay increases along with $r$ when $k = 9$. Figure 9 shows how the update delay increases along with $k$ when $r = 5$. Consistent with the evaluation results of our prototype in Figure 5, as the number of data nodes or parity nodes increases, our scheme's advantage becomes more apparent. As we can see, the scheme adopting ORN and MACOU shows a 30% latency decrease compared to the TA-Update.

2) UPDATE DELAY UNDER VARYING SCALE OF STORAGE SYSTEM
Figure 10 depicts the update delay under different storage system scales when there are 5% data nodes and 5% parity nodes for updates. These nodes for updates are randomly deployed in the storage system, which initiates the update procedure periodically. As the system scale increases, the TA-Update delay increases faster compared to our scheme adopting ORN and MACOU. Figure 11 shows the delay under a varying number of nodes of the system when there are only 5 data nodes and 5 parity nodes for updates. Following
the evaluation results of the prototype shown in Figure 5(c), in Figure 10 and Figure 11, our scheme adopting ORN and MACOU outperforms the TA-Update algorithm rapidly as the system scales up. When the number of nodes exceeds 250, our scheme is able to achieve 28% to 30% reduction of update delay.

3) UPDATE DELAY UNDER DIFFERENT NETWORK TOPOLOGIES

To further investigate the adaptability of our proposed scheme, we conduct extensive experiments below under two typical data center network topologies shown in Figure 6, i.e., Fat tree and DCell. Figure 12 and Figure 13 present the experimental results of the update delay under a variety of data center topologies. As Figure 12 and Figure 13 show, similar to the aforementioned evaluation results, our schemes also can achieve almost the same saving of delay, which shows the favorable adaptability of our proposed scheme.

4) CONVERGENCE ANALYSIS

Periodical updating the pheromone enables our update scheme to satisfy the convergence constraint in many ants sent out. The Figure 14 evaluates the convergence of our search algorithm MACOU, when we set $\alpha = 1$, $\beta = 1$, $\lambda = 1.6$, $\rho = 0.4$. As shown in Figure 14, the update delay decreases sharply along as the number of ants increases and reaches a lower limit at the best ant number, implying the convergence of the search. For instance, the best ant number is about 500 for the storage network of 300 nodes in Figure 14(a).

Figure 14(a), (b), and (c) show that a larger scale of network topology generally leads to slower convergence. In particular, the results show the approximate linear increase of best ant numbers in different topology scales. For example, the best ant numbers are nearly 250 and 500 when the number of pods is 6 and 8, respectively, while it needs about 1000 ants to finish the update path search when the number of pods is 10. Therefore, our scheme is able to obtain robust convergence.

Figure 14(d) compares the convergence speed of our update scheme under three network topologies consisting of 200 nodes or so. As we can see, in DCell topology, our scheme achieves the fastest convergence speed at the best ant number of 230 or so due to its efficient global connectivity. In contrast, the random network topology leads to the slowest convergence speed of our update scheme owing to the complexity and redundancy of random edges.

VII. CONCLUSION

In this paper, we make the first attempt to optimize the multiple data nodes updates of the erasure codes under multiple QoS metrics for erasure-coded storage systems. Our proposed update scheme, ACOUS, employs a two-stage rendezvous data update procedure to optimize the multiple data nodes updates. Specifically, the two-stage rendezvous data update procedure performs the delta collection and the parity delta distribution data based on a multi-objective update tree built by the ant colony optimization routing algorithm. Under typical data center network topologies, extensive experimental results show that, compared to the traditional TA-Update scheme, our scheme can achieve a 26% to 37% reduction of update delay with convergence guaranteed at the cost of negligible computation overhead.

REFERENCES

[1] Z. Zhang, A. Deshpande, E. T. X. Ma, and D. Narayanan, “Does erasure coding have a role to play in my data center?”, Microsoft, Redmond, WA, USA, Tech. Rep. MSR-TR-2010-52, 2010.
[2] Q. Liu, D. Feng, H. Jiang, Y. Hu, and T. Jiao, “Z-codes: General systematic erasure codes with optimal repair bandwidth and storage for distributed storage systems,” in Proc. IEEE 34th Symp. Reliable Distrib. Syst. (SRDS), Sep. 2015, pp. 212–217.
[3] J. Tan, W. Liu, T. Wang, M. Zhao, A. Liu, and S. Zhang, “A high-accuracy content popularity prediction computational modeling for mobile edge computing using matrix completion technology,” Trans. Emerg. Telecommun. Technol., Jan. 2020, doi: 10.1002/ett.3871.
[4] Y. Hu, Y. Liu, W. Li, K. Li, K. Li, N. Xiao, and Z. Qin, “Unequal failure protection coding technique for distributed cloud storage systems,” IEEE Trans. Cloud Comput., early access, Dec. 19, 2017, doi: 10.1109/TCC.2017.2785396.
[5] Y. Wang, X. Pei, X. Ma, and F. Xu, “TA-update: An adaptive update scheme with tree-structured transmission in erasure-coded storage systems,” IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 8, pp. 1893–1906, Aug. 2018.
[6] F. Zhang, J. Huang, and C. Xie, “Two efficient partial-updating schemes for erasure-coded storage clusters,” in Proc. IEEE 7th Int. Conf. Netw., Archit., Storage, (NAS), Jun. 2012, pp. 21–30.
[7] J. Xia, J. Huang, X. Qin, Q. Cao, and C. Xie, “Revisiting updating schemes for erasure-coded in-memory stores,” in Proc. Int. Conf. Netw., Archit., Storage (NAS), Aug. 2017, pp. 1–6.
[8] J. Shen, K. Zhang, J. Gu, Y. Zhou, and X. Wang, “Efficient scheduling for multi-block updates in erasure coding based storage systems,” IEEE Trans. Comput., vol. 67, no. 4, pp. 573–581, Apr. 2018.
[9] I. F. Adams, M. W. Storer, and E. L. Miller, “Analysis of workload behavior in scientific and historical long-term data repositories,” ACM Trans. Storage, vol. 8, no. 2, pp. 1–27, May 2012.

[10] D. Narayanan, A. Donnelly, and A. Rowston, “Write off-loading: Practical power management for enterprise storage,” ACM Trans. Storage, vol. 4, no. 3, pp. 1–23, Nov. 2008.

[11] H. Zhang, M. Dong, and H. Chen, “Efficient and available in-memory KV-store with hybrid erasure coding and replication,” in Proc. 14th USENIX Conf. File Storage Technol. (FAST), 2016, pp. 167–180.

[12] M. M. T. Yiu, H. H. W. Chan, and P. P. C. Lee, “Erasure coding for small objects in in-memory kv storage,” in Proc. 10th ACM Int. Syst. Storage Conf. (SYSTOR), May 2017, pp. 1–12.

[13] D. Shankar, X. Lu, and D. K. Panda, “High-performance and resilient key-value store with online erasure coding for big data workloads,” in Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jun. 2017, pp. 527–537.

[14] J. Chan, Q. Ding, P. Lee, and H. Chan, “Parity logging with reserved space: Towards efficient updates and recovery in erasure-coded clustered storage,” in Proc. 12th USENIX Conf. File Storage Technol. (FAST), 2014, pp. 163–176.

[15] X. Pei, Y. Wang, X. Ma, and F. Xu, “Efficient in-place update with grouped and pipelined data transmission in erasure-coded storage systems,” Future Gener. Comput. Syst., vol. 69, pp. 24–40, Apr. 2017.

[16] G. J. Akash, O. T. Lee, S. D. M. Kumar, P. Chandran, and A. Cuzzocrea, “RAPID: A fast data update protocol in erasure coded storage systems for big data,” in Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID), May 2017, pp. 890–897.

[17] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjelsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, and J. Haridas, “Windows Azure Storage: A highly available cloud storage service with strong consistency,” in Proc. ACM Symp. Operating Syst. Princ., 2011, pp. 143–157.

[18] L. Guillon, S. Izumi, T. Abe, T. Suganuma, and H. Muraoka, “SDN implementation of multipath discovery to improve network performance in distributed storage systems,” in Proc. 13th Int. Conf. Netw. Service Manage. (CNSM), Nov. 2017, pp. 1–4.

[19] P. P. Ponnamamy and S. Abinaya, “Virtual network routing in cloud computing environment,” in Proc. Int. Conf. Comput. Commun. Informat. (ICCCI), Jan. 2016, pp. 1–6.

[20] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, Jun. 1960.

[21] J. Li, S. Yang, X. Wang, and B. Li, “Tree-structured data regeneration in distributed storage systems with regenerating codes,” in Proc. IEEE INFOCOM, Mar. 2010, pp. 1–9.

[22] S. Chatterjee and S. Das, “Ant colony optimization based enhanced dynamic source routing algorithm for mobile ad-hoc network,” Inf. Sci., vol. 295, pp. 67–90, Feb. 2015.

[23] J. Lv, X. Wang, K. Ren, M. Huang, and K. Li, “ACO-inspired information-centric networking routing mechanism,” Comput. Netw., vol. 126, pp. 200–217, Oct. 2017.

[24] J. Brito, F. J. Martínez, J. A. Moreno, and J. L. Verdegay, “An ACO hybrid metaheuristic for close-open vehicle routing problems with time windows and fuzzy constraints,” Appl. Soft Comput., vol. 32, pp. 154–163, Jul. 2015.

[25] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” ACM SIGOPS Operating Syst. Rev., vol. 37, no. 5, pp. 29–43, Dec. 2003.

[26] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, “A tale of two erasure codes in HDF5,” in Proc. USENIX Conf. File Storage Technol., 2015, pp. 213–226.

[27] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, “Erasure coding in windows azure storage,” in Proc. USENIX Conf. Tech., 2015, pp. 15–26.

[28] J. C. W. Chan, Q. Ding, K. Ren, P. P. C. Lee, and H. H. W. Chan, “Parity logging with reserved space: Towards efficient updates and recovery in erasure-coded clustered storage,” in Proc. USENIX Conf. File Storage Technol., 2014, pp. 163–176.

[29] C. Jin, D. Feng, H. Jiang, and L. Tian, “RAID6L: A log-assisted RAID6 storage architecture with improved write performance,” in Proc. IEEE 27th Symp. Mass Storage Syst. Technol., May 2011, pp. 1–6.

[30] J. Yang, R. Ding, Y. Zhang, M. Cong, F. Wang, and G. Tang, “An improved ant colony optimization (I-ACO) method for the quasi travelling salesman problem (Quasi-TSP),” Int. J. Geographical Inf. Sci., vol. 29, no. 9, pp. 1534–1551, Sep. 2015.

[31] G. Li, L. Boulkhatem, and J. Wu, “Adaptive quality-of-service-based routing for vehicular ad hoc networks with ant colony optimization,” IEEE Trans. Veh. Technol., vol. 66, no. 4, pp. 3249–3264, Jun. 2016.

[32] X. Chang, “Network simulations with OPNET,” in Proc. 31st Conf. Winter Simulation Simulation A, Bridge to Future (WSC), 1999, pp. 307–314.

[33] J. Zhang, X. Liao, S. Li, Y. Hua, X. Liu, and B. Lin, “Aggcode: Constructing route intersection for data reconstruction in erasure coded storage,” in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Apr. 2014, pp. 2139–2147.

** */ **