Cost Optimization for Cloud Storage from User Perspectives: Recent Advances, Taxonomy, and Survey

MINGYU LIU, LI PAN, and SHIJUN LIU, Shandong University

With the development and maturity of cloud storage, it has attracted a large number of users. Although cloud users do not need to concern themselves with the infrastructure used for storage, thus saving on equipment and maintenance costs, the sheer volume of data still generates significant cloud storage usage costs, which motivates cloud storage users to look for ways to further save costs. In this article, we analyze the whole process of using cloud storage to exhaustively explore opportunities, motivations, and challenges of cost optimization from user perspectives. Then we provide a comprehensive taxonomy and summary of recent advances in terms of storage efficiency (i.e., cost optimization by improving storage efficiency), cloud storage services (i.e., cost optimization by leveraging the features of cloud storage services), and emerging storage paradigms (i.e., cost optimization by leveraging emerging storage paradigms like edge storage). Finally, we present future directions for cost optimization from user perspectives and present our conclusion. This article offers a thorough survey of recent advances focusing on how to optimize the cost of using cloud storage for cloud users, and it has an opportunity to attract a broad audience in the cost-effective cloud storage market.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Information systems** → **Cloud based storage;**

Additional Key Words and Phrases: Storage-as-a-Service (STaaS), cloud storage, cost optimization

ACM Reference format:
Mingyu Liu, Li Pan, and Shijun Liu. 2023. Cost Optimization for Cloud Storage from User Perspectives: Recent Advances, Taxonomy, and Survey. *ACM Comput. Surv.* 55, 13s, Article 266 (July 2023), 37 pages.
https://doi.org/10.1145/3582883

1 INTRODUCTION

Cloud storage, or Storage-as-a-Service (STaaS), provides diverse types of storage such as object storage and block storage, and offers a convenient way to store a large volume of data with the advantages of nearly unlimited capacity, continuous availability, high scalability, and cost-effectiveness, which encourages numerous enterprises to move data out of their own local infrastructure to clouds [37]. According to the report released by Flexera [36], 46% of organizations'

This work was supported by the by the National Key R& D Program of China under grant 2017YFA0700601, the Key Research and Development Program of Shandong Province (2020CXGC010102), and project ZR2020LZH011 supported by Shandong Provincial Natural Science Foundation.
Authors' address: M. Liu, L. Pan (corresponding author), and S. Liu (corresponding author), School of Software, Shandong University, 1500 Shunhua Road, High-tech District, Jinan, Shandong, China; emails: my.liu@mail.sdu.edu.cn, {panli, lsj}@sdu.edu.cn.

0360-0300/2023/07-ART266 $15.00
https://doi.org/10.1145/3582883

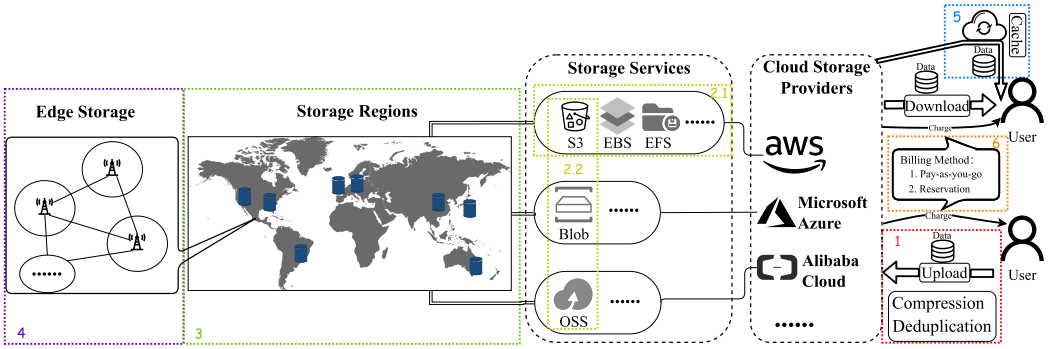


Fig. 1. A bird's eye view of opportunities for cost optimization in cloud storage.

data is in the public cloud today. With the increase of cloud storage users, the cloud storage market is gaining momentum and will continue to advance in the next few years—expected to grow to $154.21 billion by 2026 [37]. For cloud users, cost management is one of the top challenges, and they are struggling to handle growing expenses. Worst of all, the percentage of wasted expenses is about up to 30%, which drives users to explore cost-effectiveness for cloud storage [36].

The expenditure of cloud storage users mainly refers to the cost of renting cloud resources such as storage and bandwidth. Although the underlying infrastructure of cloud storage is transparent to users, they still have an opportunity to pursue cost-effectiveness from a higher level of service, which is based on the diversity of cloud resource pricing. When users host their data in the cloud, the process is fraught with multiple aspects rather than simply handling over their data to cloud providers. As shown in Figure 1, these aspects cover the whole process of using cloud storage from uploading data to the cloud to downloading data from the cloud, which hold a reservoir of opportunities for cost optimization. Specifically, in terms of the data itself in the cloud, users can take advantage compression and deduplication to save storage space, thus saving the cost of storage. In terms of the cloud, diverse billing methods, pricing schemes, and **Quality of Service (QoS)** metrics (e.g., availability and request latency) spawn several opportunities for cost optimization. The choice of billing methods exists when cloud users pay cloud providers for their usage. The differences in pricing schemes and QoS metrics exist widely across cloud storage providers, storage services, storage regions, and storage locations between the edge and the cloud. Users can take advantage of these diversities and dynamically adjust their adoption to achieve cost-effectiveness. These opportunities have motivated both industry and academia to explore more cost-effective ways to use cloud storage.

Currently, surveys on cloud storage mainly focus on managing data from perspectives of data availability [84, 103], data replication [97], data consistency [16, 54], data security [51, 104], and privacy [41]. These surveys do not investigate the cost management problem in cloud storage from user perspectives. Mansouri et al. [81] investigate cost management for cloud data based on ensuring QoS as a part of the whole-process data management in the cloud. In terms of cost optimization, there are some other investigated concerns including data deduplication [99] that removes duplicate data items to save storage costs and cloud service selection [100] that chooses appropriate services according to users' budget requirements. However, these investigations are not exclusively dedicated to cost optimization for cloud storage from user perspectives and omit some cost issues and optimization opportunities. There is a lack of comprehensive and systematic investigations on all possible opportunities for cost optimization for cloud storage users.

In this article, we review the recent advances, mainly after 2017, on cost optimization for cloud storage from the perspective of cloud storage users. Here, "user" refers to customers of cloud

Table 1. Comparison of Existing Surveys on Cloud Storage

Ref.	Deduplication	Single Cloud	Multi-Cloud	Edge	Serverless	User Perspective	Concern
[84]		✓	✓				Reliability
[103]		✓				✓	Availability
[97]	✓	✓	✓	✓			Replication
[16, 54]			✓				Consistency
[51, 104]		✓	✓			✓	Security
[41]		n.s.				✓	Privacy
[81]		✓	✓			✓	Data elements
[99]	✓	✓	✓				Secure deduplication
[100]		✓				✓	Cloud service
Ours	✓	✓	✓	✓	✓	✓	Users' cost

storage providers, including enterprises, individuals, and other service providers using cloud storage services. Table 1 shows the comparison between this article and existing surveys on cloud storage.

This article is organized as follows. First, we introduce the background and motivations for cost optimization in cloud storage (Section 2) so that all possible cost optimization opportunities can be explored. Then we provide an overview of cost optimization for cloud storage (Section 3). Furthermore, we review the efforts on improving cloud storage efficiency (Section 4) and optimizing costs for different cloud storage scenarios—that is, single-cloud storage (Section 5), multi-cloud storage (Section 6), edge-cloud collaborative storage (Section 7), and cloud-based caches (Section 8). Section 9 presents future directions for cost management of cloud storage. Section 10 concludes this survey.

2 BACKGROUND AND MOTIVATION

The growth of expenditure of cloud storage users motivates them to explore cost optimization. In this section, we detail the background and motivation for cost optimization from five aspects that correspond to the components bringing opportunities of cost-effectiveness in Figure 1. The boxes in different colors in Figure 1 indicate where cost optimization opportunities lie.

2.1 Efficient Storage Techniques

Today, the explosion of data leads to lots of challenges in both storage and retrieval. Efficient storage techniques are needed for users to handle vast data. Data compression and deduplication have been widely used in general storage systems for improving the efficiency of storage. These two techniques can be applied in cloud environments to lower storage and bandwidth costs [99, 126].

Data compression can reduce the size of data items, so it can save the resources for storing and transmitting data, thus optimizing the costs of storage and bandwidth. However, the processes of compression and decompression consume computational resources, which can lead to a negative impact on performance. Different from compression, deduplication, which is shown to be much more computationally efficient [120], intends to eliminate redundant data files in entire storage systems so that there is only one copy for each data file. The main concern of deduplication is to identify duplicate data files or chunks, which is based on the comparison of their fingerprints. Depending on when deduplication is performed, it consists of two categories: inline deduplication and offline deduplication. The process of inline deduplication for searching redundancies occurs before data are written, whereas offline deduplication removes redundancies after data have been stored at idle time. The former leads to a negative impact on performance, whereas the latter results in additional storage costs. In summary, the main challenge to utilize data compression and deduplication for cloud storage is to make tradeoffs between cost and performance.

2.2 Diversity of Cloud Storage Services

Generally, a cloud storage provider offers several storage services. Common services in the cloud storage market include *object storage*, *file storage*, and *block storage*. These services with different pricing schemes are aimed at satisfying various storage and QoS requirements of users, which motivates users to choose appropriate services to match the characteristics of their data as far as possible to avoid the cost increase caused by mismatching demands. Taking Amazon Web Services (AWS) [6] as an example, as shown in Figure 1, it provides storage services of object storage (i.e., Amazon S3), file storage (i.e., Amazon EFS), and block storage (i.e., Amazon EBS). Users can utilize these services in a cost-effective way by choosing appropriate ones on the basis of their requirements. Moreover, some services are featured with multiple storage classes that also have different pricing schemes from each other. For example, S3 consists of four storage classes (i.e., Standard, Standard (IA), Glacier, and Glacier Deep Archive), of which the storage cost decreases in order while the access cost increases in order. Apart from pricing differences, some other services also offer multiple storage classes with different performance. For example, EBS offers HDD-based and SSD-based volumes with different performance characteristics. Better performance means a higher price. It is a challenge for users to choose cost-effective storage services as well as storage classes in the face of time-varying requirements and workloads.

2.3 Diversity of Cloud Storage Billing Methods

In the current cloud storage market, there are two kinds of billing methods: pay-as-you-go (which is also known as on-demand) and reservation. Many famous cloud providers, such as Amazon, Alibaba, and Microsoft, support both billing methods. The pay-as-you-go method allows users to pay for their resource usage by the day, hour, or second with no long-term commitments, whereas the reservation method offers users lower prices in exchange for a long-term commitment to a consistent amount of usage, during which users are charged regardless of whether they use the committed resources or not. The former method is a more cost-saving option for data that have small sizes and short-term storage periods, whereas the latter billing method is cost-effective for data that have large sizes and long-term storage periods. Additionally, the reservation billing method requires users to prepay for their reservation capacity and period. To achieve the optimal cost, users need to have an exact knowledge of their data size and storage period in advance, but it is generally difficult for users to obtain the information due to the dynamics of data in clouds. Hence, whether or not to use the billing method of reservation and, if so, how to choose the appropriate reservation capacity and reservation period are challenges for users, which motivate them to pursue novel approaches to deal with the cost optimization problem resulting from multiple billing methods.

2.4 Abundance of Cloud Storage Regions

Those leading cloud storage providers, such as Amazon, Microsoft, Google, and Alibaba, host their own data centers around the world. They generally define data centers within a certain geographic area as a *region*. As shown in Figure 1, storage regions are spread over major geographic areas of the world. When users decide to store their data in clouds, they need to choose a region to store their data. Different regions are generally equipped with different pricing schemes. Users can choose appropriate regions according to their requirements such as request latency and cost limitation. They can store latency-sensitive data in the regions near them so that their requirement of latency is guaranteed. To reduce costs, they can store infrequently accessed data in the regions with lower access costs, and store frequently accessed data in the regions with lower storage costs.

Nevertheless, the access patterns of users' data are time varying, which means that a (or some) certain region(s) may not always be the most cost-effective during the life spans of their data. Cloud storage providers allow users to migrate their data from one region to another region, which brings

an opportunity for cost optimization. Therefore, explorations for cost-effectively migrating data among regions receive extensive attention.

2.5 Pricing Competition Among Cloud Storage Providers

For a certain storage service, cloud storage providers generally offer competitive pricing schemes to attract users. For example, as shown in Figure 1, in addition to Amazon, Microsoft and Alibaba respectively provide the service of object storage. The pricing schemes of object storage provided by these three providers are different, even if the same storage region (i.e., the U.S. East) is chosen. Meanwhile, for a certain storage service, different providers offer their own unique features to enhance their competitiveness in the market. Various pricing schemes and features offered by different providers of a certain storage service often make it challenging for users to choose the most cost-effective one. Users can accomplish the goal of cost savings by switching providers according to their requirements as well as the access patterns of their data [71]. However, this behavior is against the will of providers. Providers, with the aim of profit maximization, always expect users to only use their own services. Therefore, users encounter the problem of *vendor lock-in* [2], which aims to prevent them from switching providers and charges them an additional cost, when they want to migrate their data to another provider. This brings great challenges for users for cost optimization and encourages users to store their data across multiple providers.

2.6 Emergence of Edge Cloud

In recent years, edge computing has emerged as a new computing paradigm [95], which has spawned several mature services related to edge computing in the market, such as Alibaba ENS [27] and Edge Networking [1]. Edge computing provides resources of computing and storage at the edge of the network so that those resources are closer to users, which brings users better latency and experience [21, 56]. Edge servers are generally deployed at base stations or access points, and are connected via high-speed links to facilitate data transmissions between them [27, 56]. The high-speed links between edge servers allow users to migrate data between those edge servers. This kind of data transmission is generally of low cost compared with the data transmissions from cloud storage to edge servers because of the zero burden on the Internet backbone and the much shorter distance between adjacent edge servers. Moreover, as shown in Figure 1, edge computing can be considered as an extension of cloud computing, which means that data can be placed in edge servers so that the data volume transferred from the cloud to users can be reduced [139]. This extension has a positive impact on optimizing bandwidth costs [88].

However, a non-negligible issue of edge servers is that they are equipped with limited computing and storage capacity [22], which is a challenge for cost optimization. An edge server generally covers a specific geographical area. To make users in more areas reap the benefits of edge storage, data need to be replicated across edge servers in multiple areas, which incurs more storage costs and is also a challenge for cost optimization. In addition, although the cost of edge servers is generally low [1, 40], storing certain data, such as infrequently accessed data, in edge servers can also incur unnecessary costs. Therefore, users need to overcome these challenges to take advantage of edge storage cost-effectively.

2.7 Beneficial Cloud Caches

In general systems, caches are built on fast but size-limited devices that can store data temporarily by leveraging temporal and spatial localities. For cloud storage, users can cache their frequently accessed data or latency-aware data so that future requests for that data can be served faster. Cloud-based caching offers a cheaper solution compared with physical cache servers. Although using cloud caching services eliminates the cost of acquiring caching devices and maintenance for users,

there are still significant costs associated with long-term usage. The development of cloud storage services presents an opportunity for cost optimization for cloud-based caching. There exists a tradeoff for data caching, namely the tradeoff between performance and cost, which inspires users to optimize the cost of caching for cloud storage.

3 OVERVIEW OF COST OPTIMIZATION IN CLOUD STORAGE

Section 2 reveals the opportunities and challenges for cost optimization in cloud storage. These opportunities inspire cloud users to optimize costs for cloud storage from multiple aspects. We provide a taxonomy of recent studies focusing on cost optimization for cloud storage. For providing readers with a comprehensive understanding of the taxonomy (i.e., the various aspects of cost optimization), we give an overview in this section. To describe the overview in a clear and organized way, we use an example where a data object is uploaded to the cloud and then downloaded from the cloud, as shown in Figure 1, to help us describe this process. The numbers in the colored boxes of Figure 1 indicates the order of dataflow in this process.

Stage 1: Uploading. When the object is uploaded, as the red box with number 1 implies, it can be compressed or deduplicated before uploading. Thus, compression and deduplication have the opportunity to optimize costs. The recent advances related to this stage are discussed in Section 4.

Stage 2: Provider and Service Selection. Then the object should choose a cloud provider and a storage service to reside in. *Service Selection.* In service selection, the object determines which storage service to use. Assuming that the object chooses Amazon to provide the storage service, as the yellow box with the number 2.1 shows, its optional storage services are S3, EBS, EFS, and so on. Through leveraging the differences in the features and pricing schemes of these storage services, the cost of using single-cloud storage can be optimized. The recent advances in this aspect are discussed in Section 5. *Provider Selection.* In provider selection, the object determines which provider to use. For a kind of storage service, it is offered by multiple cloud storage providers, as shown in the green box with the number 2.2. The object can capture cost benefits through leveraging the pricing differences for the same storage services caused by competition among cloud providers. The recent advances related to cost optimization for this case are discussed in Section 6.

Stage 3: Storage Region Selection. After choosing the provider and the storage service, as the green box with the number 3 indicates, the object also needs to choose a storage region. Different regions have different pricing schemes and have different distances from the requesters of the object, which gives the object an opportunity to optimize its costs according to its latency requirements. In this article, we refer to the scenarios where objects are stored among multiple storage regions as multi-cloud storage and discuss the related recent advances in Section 6.

Stage 4: Edge Storage. Further, the object can be stored in edge storage, as the violet box with the number 4 shows. Edge storage brings storage resources closer to users, and it has lower prices than cloud storage due to cheap storage devices and network resources. Thus, it brings an opportunity for cost optimization by collaborating with cloud storage. The recent advances focusing on edge-cloud collaborative storage are discussed in Section 7.

Stage 5: Downloading. When the object is downloaded, it can be transferred to users through cloud storage providers or caches. When the object is downloaded from caches, as the blue box with the number 5 shows, users can benefit in terms of latency and bandwidth cost, but can also suffer in terms of storage cost, which inspires users to explore cost-effective caches for cloud storage. We discuss the related recent advances in Section 8.

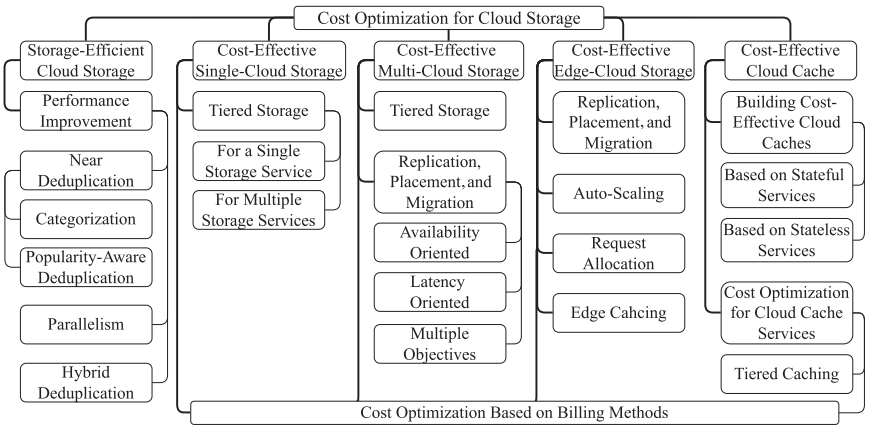


Fig. 2. Overview and organization of this article.

Stage 6: Billing. When users download the object from cloud or edge storage, storage providers charge them fees. As the orange box with the number 6 shows, there are two billing methods offered by the cloud. We integrate the cost optimization based on billing methods into multiple sections rather than using a separate section to discuss it.

Overview Summary. Finally, the overview of cost optimization for cloud storage is shown in Figure 2. As Figure 2 shows, we cover five aspects of cost optimization for cloud storage from user perspectives—that is, storage-efficient cloud storage, cost-effective single-cloud storage, cost-effective multi-cloud storage, cost-effective edge-cloud collaborative storage, and cost-effective cloud caches.

4 STORAGE-EFFICIENT CLOUD STORAGE

There are some studies focusing on the application of data compression and deduplication in cloud storage. The main concern of applying these techniques for cloud storage is I/O performance and request latency, because the techniques generally reduce costs at the expense of performance. Therefore, users often seek tradeoffs between performance and costs in using cloud storage.

4.1 Process of Deduplication

A generic deduplication process in cloud storage is shown in Figure 3. The process shows that data deduplication needs to calculate hash values and compare them with data already stored, which means that it may affect the performance of foreground requests.

4.2 Tradeoff Between Storage Efficiency and Performance

The entry points to improve performance are to reduce the number of hash calculations and comparisons, and to speed up the number of hash calculations and comparisons.

The former entry point is achieved by performing near-exact data deduplication by leveraging categorization and data popularity that trade storage costs for performance improvement. Both methods take advantage of locality. By exploiting locality, data can be deduplicated on a small scale, without having to perform deduplication across the entire storage system, which thus reduces the number of hash calculations and comparisons at the expense of deduplication accuracy. The main idea of categorization before deduplication is to classify data into different categories and then perform deduplication within each category. It comes in two main ways, namely content-aware and application-aware categorization. Content-aware categorization is based on the content

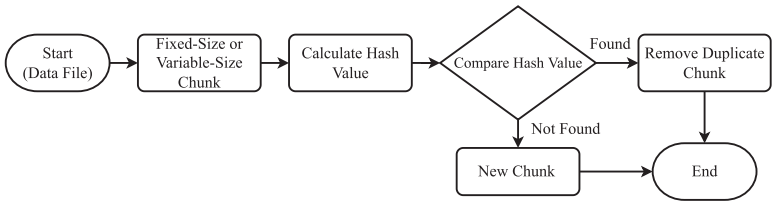
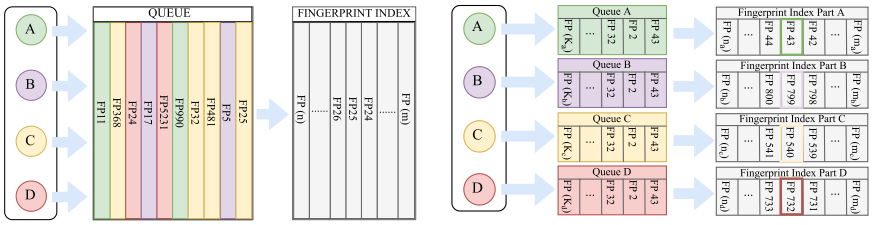


Fig. 3. The generic deduplication process in cloud storage.



(a) Serial lookups on the shared fingerprint index

(b) Parallel lookups on multiple independent fingerprint sub-indexes

Fig. 4. The parallelism chance of improving deduplication performance brought about by categorization [63].

bytes of data. Data with similar content bytes are grouped into the same category. This kind of categorization can be performed based on fixed-size [93] or variable-size [138] chunking. Compared with fixed-size chunking, variable-size chunking facilitates promoting deduplication efficiency. Different from content-aware categorization, application-aware categorization categorizes chunks according to where they come from. Applying application awareness in categorization is reasonable because it is based on the fact that data generated by the same applications own the same chunks in high probability, whereas data generated by different applications own the same chunks in low probability [38, 39, 124]. Hence, application-aware categorization instructs deduplication to compute and compare fingerprints of chunks generated by the same applications instead of entire chunks generated by all applications. Data popularity implies that different chunks of data have different occurrences. Those chunks with low frequency and a long time of appearance can be ignored so that the system stress can be eased and the throughput can be improved while ensuring the deduplication rate [73]. Further, Long et al. [73] combine categorization and data popularity as well as sampling and clustering techniques. Those data with frequent occurrences are categorized first and chunked with variable sizes. To reduce hash computations and comparisons, for each data file, several chunks are sampled as representations and clustered. Then data deduplication is performed for each cluster.

The latter entry point is achieved by performing data deduplication in parallel. Categorization not only improves the performance of data deduplication by alleviating hash calculations and comparisons but also brings a chance to further improve deduplication performance by utilizing parallelism [39, 63]. Figure 4 gives an illustration of the chance of parallelism.

These two entry points for performance improvement either sacrifice storage costs or require more computing resources. To pursue both cost-effectiveness and high performance without requiring more computing resources, hybrid deduplication is proposed. Hybrid deduplication combines inline and offline deduplication, which are distinguished by the time of execution. Inline deduplication deduplicates data before storing them, so duplicate data files or chunks are never stored in storage systems. Thus, inline deduplication maximizes cost savings at the expense of

Table 2. Comparison Summary of Various Data Deduplication Techniques

Ref.	Granularity	Parallelism	Exact	Near-Exact
[93]	Chunk (fixed size)			✓
[138]	Chunk (fixed size | variable size)			✓
[124]	Chunk (fixed size)			✓
[73]	Chunk (variable size)			✓
[39]	Chunk (variable size | fixed size)			✓
[63]	Chunk (variable size)	✓		✓
[116]	Chunk		✓	
[115]	Chunk		✓	
[133]	File and chunk	✓	✓	✓
Ref.	**Performance Improvement Means or Features**	**Categorization**	**Inline**	**Offline**
[93]	Content awareness	✓		✓
[138]	Content awareness	✓	✓	✓
[124]	Application awareness	✓	✓	
[73]	Content and application awareness	✓	✓	
[39]	Content and application awareness	✓	✓	
[63]	Application awareness	✓		✓
[116]	Temporal and spatial locality		✓	✓
[115]	Temporal and spatial locality		✓	✓
[133]	Content awareness, temporal and spatial locality	✓	✓	✓

performance such as I/O and latency. On the contrary, offline deduplication removes redundancies after data have been stored, so it results in additional storage costs due to waiting for deduplication. The fusion of these two forms of deduplication offers an opportunity to exploit both of their advantages and avoid both of their disadvantages so that the I/O performance and cost savings are balanced [115, 116, 133]. The process of hybrid deduplication is to first perform near-exact inline deduplication leveraging locality or popularity, then perform offline deduplication on the entire data [115, 116]. Moreover, locality can be used in the offline phase to perform near-exact offline deduplication so that the negative effects on performance can be further alleviated [133].

4.3 Comparison and Applications

Table 2 summarizes and compares the preceding methods on data deduplication in clouds. Through leveraging the existing data deduplication methods such as those mentioned earlier, some studies propose deduplication-assisted frameworks for cloud users so that users can achieve the cost benefits of deduplication in the cloud. HyCloud [33] makes it possible to optimize the transfer of data to Amazon S3 by utilizing data deduplication. Similarly, Duggal et al. [32] perform inline deduplication at variable-size chunk granularity so that the unique data can be transferred and stored in the cloud. EAD [130] applies content-defined chunking techniques just as Lin et al. [63] and inline deduplication that utilizes spatial and temporal locality for multi-tier storage that consists of **Random Access Memories (RAMs)**, **Solid-State Drives (SSDs)**, and **Hard Disk Drives (HDDs)** in the cloud with the aim of enhancing performance and reducing costs. For optimizing cost and improving performance in multi-cloud storage, DAC [117] applies fixed-size chunking methods to split data into chunks in terms of the computing overhead of chunking and the trade-off between performance and redundancy, then performs inline deduplication on those fixed-size chunks. PMCR [67] performs data compression or deduplication to optimize storage and bandwidth costs according to data popularity that includes hot, warm, and cold types classified based on several pre-defined thresholds. Data compression and deduplication are only performed on warm and cold data replicas.

Table 3. Classification of Tiered Storage

Classification		Reference
Tiered object storage	Hot and cold tiers	[12, 34, 68, 69, 80]
	Hot, cold, and archive tiers	[110]
Tiered file/block storage	HDDs and SSDs	[13, 14, 74, 105, 112]
	HDDs, SSDs, and RAMs	[130]
	HDDs and burstable SSDs	[87]
	Multiple types of SSDs	[129]

Further, DCStore [8], which is also faced with cost optimization and performance improvement in multi-cloud storage, respectively leverages fixed-size and variable-size chunking approaches for uneditable and editable data for deduplication so that a tradeoff between the deduplication ratio and deduplication throughput can be achieved. These frameworks generally offer various features such as high availability apart from storage efficiency, which will be surveyed in the following sections.

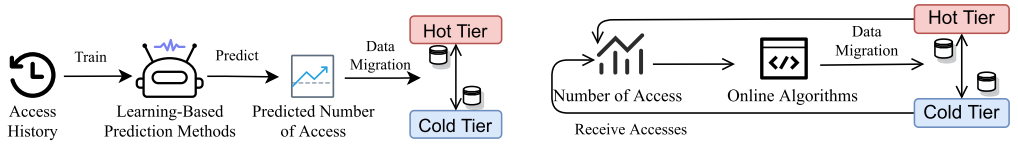
5 COST OPTIMIZATION FOR SINGLE-CLOUD STORAGE

Cloud storage providers have offered several tier options for a certain service such as object storage, with diverse pricing schemes, as discussed in Section 2.2, and they have also developed the same storage service with their own competitive pricing schemes, as discussed in Section 2.5. These diverse tier options and storage services motivate users to develop tiered storage for cost optimization by leveraging the diversity of pricing schemes. Common types of tiered storage include the utilization of multiple storage classes of a storage service, and the utilization of multiple cloud storage services. Each storage class in the former type and each storage service in the latter type can be considered as a storage tier. In the rest of this section, we discuss the two types of tiered storage mentioned earlier in Sections 5.1 and 5.2, respectively.

5.1 Tiering Inside a Single Storage Service

This kind of tiered storage is built on multiple storage classes (i.e., tier options) designed in a single service, of which Table 3 summarizes the references and classification.

5.1.1 Tiered Object Storage. Providers generally offer several storage tiers for the object storage service. Taking Amazon S3 as an example, as discussed in Section 2.2, it offers hot (Standard), cold (i.e., Standard (IA)), and archive (i.e., Glacier) tiers that are respectively designed for frequently and infrequently accessed data with almost the same performance metrics. Cloud storage users can migrate their data from one tier to the other according to the access patterns of their data with little performance penalty. Although users do not have any knowledge of the future access patterns, they have made some efforts on prediction of the future access patterns according to the history of accesses [23, 24, 49]. Their proposed prediction methods are generally learning based and need to be trained with exhaustive historical access records. Users can migrate their frequently accessed data to the hot tier in advance to prepare for the coming accesses according to the predicted large number of accesses in the near future, and vice versa. The common process of this kind of method is shown in Figure 5(a). However, at the initial stage, the lack of historical access records makes it impossible to train the methods effectively. In addition, the access patterns of data are constantly changing during the storage period and also vary depending on the type of data, so it is generally difficult to predict the future patterns precisely. Considering the preceding challenges, some



(a) Learning-based prediction methods (b) Online algorithms

Fig. 5. Process of learning-based prediction methods and online algorithms.

studies propose to develop online algorithms for data migration for hot and cold tiers with the aim of cost optimization. Online algorithms can handle the situations well in which the future is unknown and accesses arrive constantly. These studies generally develop their own algorithms based on some classic online problems. Mansouri and Erradi [80] apply the *Ski-Rental problem* to tiered storage services offering hot and cold tiers such as Amazon S3 mentioned previously and propose two online algorithms that migrate data to the hot tier for a period of time when the data receive any accesses, and Erradi and Mansouri [34] give the competitive analysis of these two algorithms. Liu et al. [68] extend the *Bahncard problem* to tiered storage services. They model the cost based on the pricing scheme of Google Storage and design their algorithm with guaranteed competitiveness based on the frequency of accesses at each slotted time. Their algorithm migrates a data object stored in the hot tier to the cold tier when the number of accesses to the data object in a time slot is below a certain level, and vice versa. Blamey et al. [12] adapt the *Secretary Hiring problem* for hot-cold tier placement under top-K stream workloads. They place the data items that are overwritten with a high probability in the hot tier and other data items in the cold tier so that the total cost is optimized. Liu et al. [69] propose a *randomized* online algorithm for user-generated data. They migrate data objects, of which access patterns have a long-tail phenomenon, from the hot tier to the cold tier at the appropriate time with competitive performance as the number of accesses declines so that incurred costs are optimized. The process of online methods is shown in Figure 5(b). Compared with Figure 5(a), the main difference between online methods and learning-based prediction methods on data migration is that the latter's migration decisions are based on predicted access patterns, whereas those of the former are based on temporal access situations. Moreover, some other studies leverage reinforcement learning, which learns from experiences gained through interactions with the environment to manage data migration among tiers. Wang et al. [110] propose a reinforcement learning based method to optimize costs in three-tier storage that includes hot, cold and archive tiers. The archive tier is designed for rarely accessed data. It can be used for backup, disaster recovery, and some industry-specific data, which are required for long-term storage, such as the data in highly regulated industries [6, 10]. Compared with hot and cold tiers, it offers lower costs that are competitive with or cheaper than on-premises solutions, and the main difference is that retrieving data from the archive tier needs to take several minutes or hours to complete.

5.1.2 Tiered File/Block Storage. HDDs and SSDs are two common storage types offered by file/block cloud storage such as Amazon EBS and have different performance metrics and pricing schemes from each other. Compared with HDDs, SSDs have some advantages regarding performance such as high throughput, high IOPS, and low latency, so the price of SSDs is higher as a matter of course [140]. The differences in performance and price between HDDs and SSDs offer an opportunity of building tiered storage to optimize costs. Cloud storage users can store their data in a combination of HDDs and SSDs with the goal of seeking a tradeoff between performance and cost. Some studies propose to leverage prediction-based [105] and heuristic algorithms [13, 14], which take periodical decisions to migrate data between HDDs and SSDs according to time-varying I/O workloads.

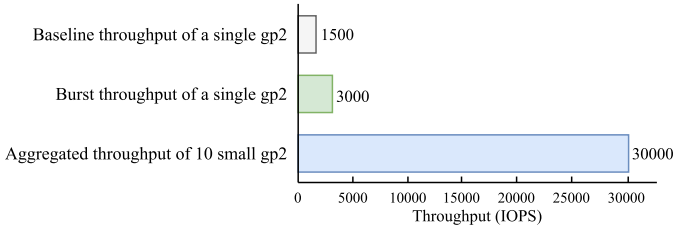


Fig. 6. Performance comparison of three storage configurations [87].

Additionally, it has been revealed that many storage systems are not well matched to workloads because the SLA of workloads can vary widely [105]. From the perspective of workload and performance matching, Zhou et al. [140] point out that only a small amount of data are hot, which should be stored on SSDs for the guarantee of performance requirements such as throughput and IOPS, and the remaining cold data should be stored on cheap HDDs because the performance of HDDs is usually adequate for data that are not accessed frequently. In the meantime, they also show that access sequence and access size have an impact on the performance of data access. The data that receive random accesses of small sizes should be stored on SSDs because SSDs achieve much better performance than HDDs. On the contrary, data with sequential accesses or large access sizes should be stored on HDDs because the performance difference is not significant in this case.

In the tiered storage environments consisting of HDDs and SSDs, the overuse of SSDs leads to more costs because of their higher prices. The phenomenon is remarkable for write-dominated tiered storage. In the face of the phenomenon, Wang et al. [112] find that HDDs can potentially provide write performance close to that of SSDs in the case of a series of sequential and continuous writes because of buffered writes in HDDs, which inspires them to propose a buffer-controlled write approach associated with a prediction model in order to take full advantage of HDDs. Based on the approach, they design a mixed IO scheduler to adaptively steer incoming data to SSDs and HDDs. Their method can give full play to the potential of HDDs in tiered storage and share the pressure of SSDs, so as to reduce the costs.

Different from the preceding studies that focus on storing data on either SSDs or HDDs, Lu et al. [74] designed a cost-effective tier-based hybrid storage scheme including four storage tiers, each of which consists of different proportions of HDDs and SSDs, for various requirements of data. The top tier is built entirely on SSDs and is suitable for storing data with high performance requirements (high throughput, high IOPS, etc.). The bottom tier is built entirely on HDDs and is suitable for storing data that have low cost requirements as well as low performance requirements. The two tiers between the top and bottom tiers are built on different proportions of HDDs and SSDs, of which the higher tier consists of 67% SSDs and the lower tier consists of 33% SSDs. Compared with the two-tier storage consisting of pure HDDs and SSDs, the four-tier storage offers more fine-grained cost-performance control.

Additionally, burstable storage, which is designed to burst to high performance levels for periods of time, brings better performance at some cost. Taking the *gp2* volume type of Amazon EBS with the size of 500 GB as an example, as shown in Figure 6, burst IOPS of a single *gp2* volume is twice the baseline IOPS, which is the standard performance for the *gp2* volume. Meanwhile, because burstable storage charges by capacity, there is an opportunity to obtain an additional performance improvement by splitting a burstable volume into multiple smaller volumes while the cost is the same. Figure 6 shows that the aggregated IOPS of 10 *gp2* volumes, each of which has a size of 50 GB, is about 20 times that of a single 500 GB volume. Hence, burstable storage is proposed to be utilized in the tiered storage to improve performance at low costs [87].

Table 4. Comparison of Amazon S3 and Amazon EFS (in the Same Region)

	Data Operation	Read Performance	Storage Price	Bandwidth Price	Data Transfer
Amazon S3	Simple, flat	Low	Low	Same	Free
Amazon EFS	Abundant, complex	High	High		

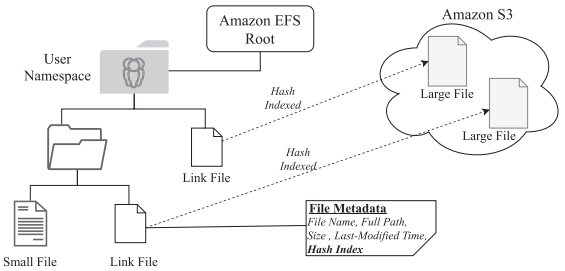


Fig. 7. The hybrid system design using the intuitive approach (large files stored in S3 are indexed by link files) [33].

Further, to pursue higher performance, Yang et al. [129] consider only using SSDs to store data. Cloud storage providers offer several SSD types with varying performance and pricing. Taking Amazon EBS as an example, it provides two types (i.e., *gp2* and *io2*) for SSDs with different performance and prices. *gp2* is cheaper but is equipped with lower performance, whereas *io2* is equipped with higher performance but is more expensive. To better take advantage of multiple types of SSDs, the authors build a tiered storage system in which each tier is built with one type of SSD. They formulate a mathematical optimization and obtain its approximate optimal solution by relaxing its constraints so that a tradeoff between performance and cost is accomplished.

5.2 Tiering by Multiple Storage Services

As described in Section 2.2, each cloud storage provider offers various storage services, such as object storage and file storage. These storage services are equipped with different pricing schemes and features, which gives an opportunity for users to take advantage of these diverse features through combining multiple services. For example, Amazon S3 offers a low unit storage price, whereas Amazon EFS offers a high unit storage price. However, Amazon EFS supports complex and hierarchical filesystem-like operations, whereas Amazon S3 only supports several simple flat operations. The details of the comparison of S3 and EFS are shown in Table 4. Users focusing on both cost and efficiency desire all the advantages of these services, which can be accomplished by fusing these services. An intuitive solution is that S3 is responsible for storing large data files, whereas EFS is responsible for storing small data files and the hierarchical directory structure that only hosts the links to those data files stored in S3 as shown in Figure 7.

However, E et al. [33] reveal that the preceding solution can encounter a read performance bottleneck because of the implementation defect of S3. In other words, although S3 is cost-effective for large data files, its inherent defect of load balance for bursty data requests impairs its performance for the delivery of large data files. The authors propose HyCloud, which ensures read performance as well as cost-effectiveness by fusing S3 and EFS. HyCloud is mainly inspired based on a fact, namely that data migration between Amazon S3 and Amazon EFS in the same region is quite rapid at no cost. Therefore, for the requested large data files, HyCloud employs EFS as a relay for speeding up the process of request. Specifically, it first migrates them from S3 to EFS in the same region through internal high-speed links, then delivers them to users. Figure 8 shows an overview and the typical process for a client to request a data file through HyCloud.

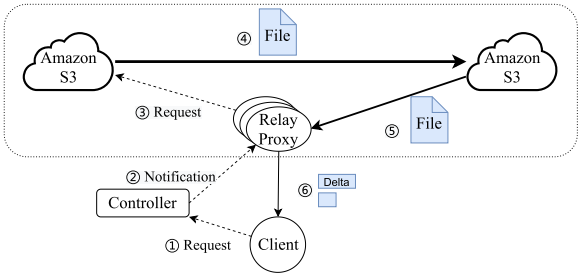


Fig. 8. Architectural overview of HyCloud, and a typical process of a client downloading a large file from the hybrid cloud storage services (steps ①–⑥) [33].

HyCloud shows a successful application case of utilizing Amazon S3 and EFS to optimize costs and performance, and it can be generalized to other cloud providers consisting of analogous object storage and file storage services.

Anna [114], a key-value store system, leverages Amazon EBS and RAM attached to Amazon EC2 as its storage tiers, and it provides interfaces that allow users to easily add additional tiers such as Amazon S3 and Glacier. Anna monitors the statistics of data to determine which tier data should be stored in. Specifically, it calculates the access frequency of data in the past T seconds, and migrates data to the RAM tier if these data files are stored in the EBS tier and the access frequency exceeds a configurable threshold. On the contrary, if these data files are stored in the RAM tier and the access frequency falls below the other threshold, Anna migrates them to the EBS tier.

Compared with utilizing distinct storage services across multiple cloud storage providers (which will be discussed in Section 6.1), the main attractive advance of utilizing distinct storage services offered by a single-cloud storage provider is that data migration between different services in the same region is cheap, which is a potential for cost optimization. Cloud storage users can be inspired to develop systems and methods for other storage services offered by a single provider, such as object storage service and cache service, according to their data characteristics, instead of being limited to object storage and file storage services.

6 COST OPTIMIZATION FOR MULTI-CLOUD STORAGE

As described in Sections 2.4 and 2.5, a cloud provider offers different pricing schemes for different storage regions, and a storage region or a storage service is priced differently by different providers. Users can take advantage of these pricing differences to accomplish cost savings by storing their data across regions or clouds. Apart from cost savings, high availability and low latency are two other significant advantages brought by multiple clouds, which are two main reasons that appeal to users [66]. Users can replicate their data across multiple clouds to ensure that their data are available although some clouds fail or crash, and they can lower their request latencies by requesting data from those clouds that are equipped with low latency. However, these benefits of multiple clouds come at a cost. High availability means more redundancy of data, so it incurs more storage costs. Low latency means a cloud is equipped with high-quality computing and storage resources, so users are charged more. Obviously, although multi-cloud storage has many advantages, they are often conflicting and none of the solutions provide a tunable choice of them in the same system, which means that users cannot enjoy all the advantages at a low cost at the same time. Hence, when users leverage multi-cloud storage, they can mainly focus on optimizing only one objective or make tradeoffs between their main concern and other requirements. In this section, we first pay attention to the cost optimization of tiered storage built on multiple storage services across clouds, and then concerning the cost optimization for clouds utilizing the reservation billing method.

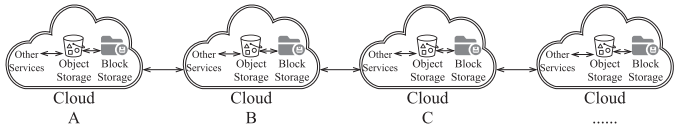


Fig. 9. An illustration of tiered storage built on the multi-cloud scenario, each cloud of which consists of multiple storage services.

After that, we discuss the cost optimization of multiple clouds when high availability or low latency is guaranteed, as well as the effort to facilitate multi-cloud data management.

6.1 Cost-Effectiveness Achieved by Tiered Storage

It can be obtained from Sections 2.2, 2.4, and 2.5 that there is a rich but complex cloud storage market where cloud storage services are diverse in terms of storage types, storage regions, pricing schemes, and performance metrics. As for cost optimization for cloud storage, these various storage services play a positive role by leveraging their pricing differences. Tiered storage can be built on these various pricing schemes, as shown in Figure 9. Different from Section 5.2, which focuses on tiered storage by leveraging multiple storage services provided by a single cloud, tiered storage in multi-cloud scenarios is typically built in two forms. One is to treat each cloud with different pricing as a different tier [79, 83]. The other kind of tiered storage is to build further tiered storage within each cloud [30, 85]. The former form of tiered storage is relatively simple, and its cost-effective migrations can be solved by heuristic algorithms, whereas the latter is relatively complex due to a variety of pricing schemes, and its cost-effective migrations are generally obtained by solving a mathematical programming problem subject to multiple constraints.

6.2 Cost-Effectiveness Achieved by Reservation Billing Method

As discussed in Section 2.3, appropriate utilization of the reservation billing method can lower costs. When leveraging the reservation billing method to optimize costs, the reservation capacity and period should be filled as much as possible, as otherwise there will be a risk of cost increase. In multi-cloud environments, users can choose different reservation capacities and periods for each cloud because the workloads of clouds are different from each other. Liu et al. [65] dynamically determine the reservation capacity and data allocation across multiple clouds at the end of each reservation period. To maximize the utilization of reservation capacity, they propose a heuristic algorithm that leverages historical and predicted workloads to find the optimal reservation capacity for each cloud. To allocate data objects cost-effectively, they analyze the dominant item of the cost for each data object and then migrate data objects based on the dominant item. For instance, the data objects whose costs are dominated by storage are classified as storage sensitive and migrated to the clouds that incur the cheapest storage costs. After data allocation, they perform a genetic algorithm based data adjustment method to adjust data allocation as workloads vary over time so that the utilization of reservation capacity can be further improved. Additionally, they apply dynamic request redirection that forwards requests from reservation-overutilized clouds to reservation-underutilized clouds and group requests for cost reduction.

6.3 Cost-Effective High Availability

Higher availability is provided by more redundancy, which inevitably imposes more storage costs on users. Fortunately, in the environment of multi-cloud storage, users have an opportunity to optimize costs by leveraging the pricing difference of multiple clouds while ensuring high availability. Different forms of redundancy have impacts on costs. The common method to create redundancy is replication including file-level replication and chunk-level replication. Meanwhile, RAID-like

Table 5. Comparison of the Work of Oh et al. [85] and Devarajan et al. [30]

Ref.	Multi-Cloud	Tiered Storage	Data Temperature	Dynamic	Compression
[85]	✓	✓	✓	✓	
[30]	✓	✓	✓	✓	✓
Ref.	Optimization target		Constraint features		Solution
[85]	Cost (get, put, storage, bandwidth)		Availability, latency, consistency		Mixed integer programming
[30]	Time (writing, (de-)compression)		Availability, latency, energy, cost durability, bandwidth, load balancing		Dynamic programming

data striping is another technique that is widely used to create redundancy in multi-cloud storage. We discuss cost optimization separately according to the form of redundancy.

6.3.1 Replication. For the cost optimization for replication, there are two main concerns: the number of replicas and the placement of replicas.

Cost-Effective Migration for a Fixed Number of Replicas. To ensure data availability, a certain number of replicas need to be placed in multiple clouds [65]. Three-way replication [67], which creates three replicas for a data object and places them based on random replication or copyset replication [26], is a commonly used method for availability guarantee. There is significant room for cost optimization in replica placement, which is based on two basic facts: replicas in different clouds have different time-varying access patterns, and pricing schemes vary from cloud to cloud. For the replicas of a data object, the reason for their different access patterns is that only one replica needs to be chosen for responding to requests. For example, users can choose the replica in the cloud that is closest to requests for responding. For various pricing schemes, some have lower storage prices that are suitable for storing infrequently accessed replicas, whereas others have lower request and bandwidth prices that are suitable for storing frequently accessed replicas. By leveraging these two facts, users can migrate replicas across multiple clouds to optimize costs. To this end, Mansouri et al. [82] propose two online algorithms, whose major concern is to compare the cost of storage and requests with the cost of migration, to dynamically determine the strategy of data placement and migration across multiple clouds as time goes on. The comparison means whether a replica should remain in the current cloud or be migrated to a cheaper cloud with the incurred cost of migration. For a replica, if the cost of storage and requests in its current cloud is greater than the costs of migration, storage, and requests of another cloud, it should be migrated to another cloud.

Cost-Effective Migration for a Variable Number of Replicas. However, the fixed number of replicas limits application scopes and creates a barrier to further cost optimization. Considering dynamically time-varying requirements and access patterns, a variable number of replicas can better optimize costs. Sun et al. [102] propose a heuristic algorithm to minimize the number of replicas while ensuring the time-varying requirement of data availability. Hsu and Kshemkalyani [48] propose to predict future access patterns that are used to calculate costs and to decide whether to create replicas according to whether the saved request and bandwidth costs are higher than the incurred storage costs. The cost-effective number of replicas as well as the placement can also be modeled by constrained optimization problems where the incurred cost is the optimization target or a constraint, and solved by mathematical programming techniques such as dynamic programming [30] and mixed integer programming [85]. A detailed comparison of the work of Oh et al. [30] and Devarajan et al. [85] is shown in Table 5.

Cost-Effective Correlated Failures. To improve availability in multi-cloud environments, most studies (e.g., [30, 85]) only consider non-correlated node failures—that is, failures that happen

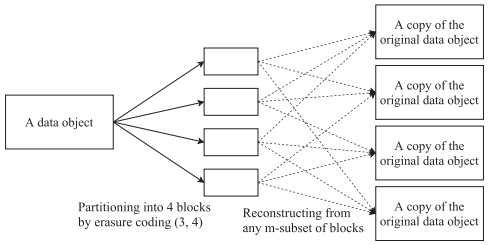


Fig. 10. An illustration of erasure coding (3, 4) [70].

independently. However, the reasons that result in data unavailability are complex. It not only consists of non-correlated node failures but also includes correlated node failures. Correlated node failures indicate that nodes fail or crash (nearly) simultaneously, and they have a significant impact on data availability. Taking both non-correlated and correlated node failures can improve data availability more realistically. To this end, recent advances [66, 67] derive the number of replicas that accomplishes a tradeoff between costs and availability using nonlinear integer programming, then disperse replicas across multiple nodes based on copyset replication [26].

Variable-Size Chunking. The strategy of chunk-level replication in other works [30, 66, 67] divides data into chunks of fixed size. This strategy is traditional and widely used (e.g., [42, 64]). Although it can obtain good results in some conditions, such as when pricing schemes, network environments, and workloads are the same and stable, it has an inherent drawback in which it treats different clouds equally. In practice, different clouds are equipped with various performance and pricing schemes, and they also encounter different network environments and workloads. Hence, it is possible to pursue more cost-effectiveness by assigning clouds with chunks of varying sizes according to the differences between these clouds by analyzing workloads and cloud environments [77].

6.3.2 Erasure Coding. Erasure coding is a RAID-like technique used in modern storage systems [20, 44, 86]. Its most attractive nature is that, for a data object, several arbitrary lost chunks do not have an impact on availability. Specifically, when erasure coding (m, n) where $m \leq n$ is performed, it codes data into n chunks, any m chunks of which can be used to recover the original data. Figure 10 shows an example of erasure coding (3, 4).

Similar to RAID that stripes data across multiple disks, erasure coding can be used to stripe data across multiple clouds, each of which holds one or several chunks. For instance, for an erasure coded data object, its n chunks can be placed in n distinct clouds. Compared with data replication, erasure coding is more storage efficient, which leads to lower costs. Through erasure coding data across multiple clouds, the amount of data in each cloud is reduced while data availability is guaranteed. Hence, erasure coding can mitigate the negative impact of vendor lock-in on costs when ensuring data availability by properly designing m and n.

Cost-Effective Migration for Erasure Coding. Despite the benefits of erasure coding, cost savings are an unswerving pursuit for cloud users. Since m erasure coded chunks out of n can be used to construct original data, users only need to request m chunks from the m cheapest clouds rather than requesting all chunks from n clouds [8, 70], which brings cost-effectiveness. However, the cheapest m clouds are not always set in stone. The costs of clouds vary with data workloads and pricing models, and they can be time varying. Thus, the m cheapest approach results in different access frequencies for chunks in different clouds. Based on the phenomenon, tiered storage can be used for cost optimization. Effeclouds [70] builds a tiered storage scheme that leverages the pricing

differences between storage classes offered by object storage services such as Amazon S3 to further optimize costs according to time-varying access workloads. Additionally, dynamically migrating erasure coded chunks among clouds according to access frequencies brings cost-effectiveness [106]. The frequently accessed chunks can be migrated to those clouds with lower access prices and higher performance, whereas the infrequently accessed chunks can be migrated to those clouds with lower storage prices.

Cost-Effective Data Protection. Apart from availability, data protection can also benefit from erasure coding. Through storing n erasure coded chunks of a data object across n clouds, the object leakage occurs when chunk leakage occurs simultaneously in more than m of n clouds. Users can consider reducing the risk of data leakage while lowering costs, so as to find cost-effective placement and migration of chunks [91, 127].

Cost-Effective Erasure Coding Approaches for Clouds. Additionally, for cloud environments, more cost-effective erasure coding approaches are sedulously pursued. As existing approaches have different focuses such as improving storage efficiency and lowering computation complexity, users can perform different erasure coding approaches according to various data characteristics rather than treating all data identically regardless of data characteristics, which is an opportunity to lower costs. Approximate Code [50] treats important and unimportant data in different ways. Important data generally need high availability because their losses can have serious consequences, whereas unimportant data do not require the same availability as important data so they can be stored at a low redundancy level. Through this double-standard approach, Approximate Code reduces storage costs while ensuring availability. The idea of double standards is also considered by EC-Fusion [92]. EC-Fusion combines two different erasure coding approaches, and dynamically chooses appropriate approaches according to data workload, so that storage costs and computational overheads can be optimized simultaneously. In addition to reducing storage costs, Bao et al. [11] focus on bandwidth cost optimization for multi-cloud environments when repairing erasure coded data. Data repair occurs in scenarios where a cloud loses its data due to failures or outages, and to recover the lost data, encoded chunks of data need to be transferred from other clouds to construct the lost data. The repair process incurs significant bandwidth costs. The authors propose an adaptive erasure coding method that encodes data according to the network environment and determines data placement among multiple clouds to optimize bandwidth costs for data repair.

6.3.3 Hybrid Replication. As Sections 6.3.1 and 6.3.2 described, we can conclude that replication and erasure coding have their own advantages and limitations. Both replication and erasure coding can guarantee low latency and high throughput by leveraging parallelism. Replication is not as storage efficient as erasure coding. To guarantee the same fault tolerance level, replication requires more storage space [132]. Erasure coding requires more computational resources to perform encoding and decoding compared with replication, which has a negative impact on read/write efficiency. Hence, replication provides better performance while erasure coding provides better storage efficiency, which inspires users to build hybrid redundancy schemes by combining their advantages. Hybrid redundancy schemes have different designs due to different goals. In general, hybrid redundancy schemes pursue to take full advantage of the performance of replication and the storage efficiency of erasure coding. Hence, one design is to utilize fully replicated data objects to receive requests and erasure coded data objects to ensure availability, such as ASSER [132]. For a data object, ASSER structures it in two forms: a full replica and a group of erasure coded chunks. The replica and the chunks are dispersed into multiple clouds. The replica serves all read requests, whereas the erasure coded chunks are responsible for recovering the replica when it is lost. One thing to note is that the erasure coded chunks are not responsible for receiving any requests.

Another hybrid design is to take advantage of the idea of data tiering, such as what DAC [117] does, which classifies data according to data characteristics and then chooses appropriate redundancy methods for each classification. DAC analyzes data popularity and then classifies data into hot and cold groups according to request frequencies. For hot data, which is requested frequently, its performance requirements are generally high. Thus, it is better to use replication for hot data to ensure availability while guaranteeing request performance. The characteristics of cold data are the opposite of hot data, which means that cold data does not have strong performance requirements in most cases. Thus, DAC utilizes replication for hot data to ensure request performance and erasure coding for cold data to improve storage efficiency.

6.3.4 Discussion. File-level replication and chunk-level replication are two types of granularity for replication. Erasure coding is another kind of redundancy that differs from these two replication methods. The main difference among these three methods is the ways of ensuring data availability, which results in different ways of optimizing costs. For file-level replication, it improves data availability by replicating data into more clouds. Its cost optimization can be achieved by minimizing the number of data replicas and migrating replicas among price-varying clouds according to data workloads. For chunk-level replication, it faces more challenges in availability improvement, because a data object is only available when all its chunks stored in different clouds are available. Taking these challenges into consideration, its cost optimization is achieved by similar ideas to cost optimization for file-level replication—that is, minimizing the number of replicas and adjusting the placement of replicas among multiple clouds, but chunk-level replication brings finer-grained cost-effectiveness. Chunks of a data object can separately select the number and placement of replicas as well as other cost optimization techniques such as data compression, depending on the situation. For example, suppose a data object is divided into three chunks, the number of replicas for each of which is 3, 3, and 5. This object using chunk-level replication incurs less cost compared with using file-level replication, which creates five replicas for it. Erasure coding is at the same granularity as chunk-level replication, but it is more complex. It ensures the availability of a data object when several chunks of the object are unavailable, which is its advantage over chunk-level replication. Its cost optimization can also be achieved by replica placement for each chunk, but it should consider more factors compared with chunk-level replication because each erasure coded chunk of a data object has different access frequencies.

Due to the coarsest granularity of file-level replication, some cost optimization techniques for file-level replication can be used for chunks. For a chunk, whether obtained by chunk-level replication or by erasure coding, it can be regarded as a separate data object when availability is ignored. Thus, cost optimization for chunks can be achieved by some file-level replication optimization techniques. Taking the work of Mansouri et al. [82] as an example, this study achieves cost optimization for file-level replication for each data object by migrating data objects among price-varying clouds. When this research is applied to chunk-level replication or erasure coding, it can treat a chunk as a separate data object and then migrate it across multiple clouds according to its access patterns. Note that not all cost optimization methods for file-level replication can be applied to chunks, because the way in which the availability of data objects is calculated can change due to data chunking or erasure coding.

6.4 Cost-Effective Low Latency

Apart from data availability, latency can also benefit from multi-cloud storage. Through replicating data in multiple clouds, users can choose the closest one that incurs the lowest latency to retrieve data. Generally, the locations of user requests for data are difficult to predict accurately in advance. Therefore, the data had better be replicated in as many clouds as possible and be migrated between clouds to offer low latency for requests from various locations [131, 137]. Nevertheless,

more replicas and migrations result in more costs [78]. Additionally, request latency is not only simply affected by physical distances but also impacted by network instability. Network instability is common and has a greater effect on latency than distance, which is a challenge for users to request a replica with the lowest latency. Some studies [29, 118] reduce request latency by issuing redundant requests for each data request, which means that they issue a request for each cloud that stores a replica of requested data and consider the cloud that responds first to be of low latency. This strategy can overcome the impact of network instability that is time varying, and it can always find the cloud with the lowest latency. Although this strategy is simple, effective, and accurate, it comes with significant costs, especially bandwidth costs. The reason is that each cloud responds with a replica of requested data, and these responses arrive on request sides one by one, instead of terminating transmissions from other clouds once users receive the response from the cloud with the lowest latency. Accordingly, the pursuit of low latency inevitably comes at a high cost.

6.4.1 Data Popularity-Aware Replication. In fact, not all data need to be replicated across multiple clouds. The popularity of different data is various, which means the number of requests for different data objects varies. The data objects requested frequently are more likely to violate the latency constraint, because the power of a single cloud to process requests is limited. Therefore, data objects can be dynamically replicated or deleted across multiple clouds in terms of their popularity to lower latency. DRAPP [62] considers replicating data to avoid latency violations when the estimated latency is greater than a threshold, and determines to remove low-popularity replicas by monitoring the number of requests for each replica. Similarly, DPRS [77] also provides a dynamic popularity-aware replication strategy that replicates frequently requested data. Different from DRAPP, the replication strategy of DPRS is based on the 80/20 principle, meaning that it replicates only a small amount (i.e., 20%) of frequently requested data. Meanwhile, DPRS predicts the access patterns of replicas to prepare for incoming requests in advance to satisfy latency constraints.

6.4.2 Node Load Aware Replication. Nodes can be considered as servers in clouds responsible for storing data. The performance of each node is limited, so the number of requests it can serve is limited. Hence, new replicas should be created in other nodes when a certain node is close to the overload so that the node can relieve its request pressure. Through considering the load of nodes, users can better perceive latency, because high latency is due to insufficient node performance. Node load aware replication can save costs and ensure low latency by maximizing node performance without waste. RPMSP [134] takes the loads of nodes into account to guide replica placement while ensuring low latency. It aims to assign a high proportion of requests to the clouds with good performance and vice versa, then it concludes a tradeoff among latency, cost, and availability.

6.4.3 Replication with Hybrid Awareness. Considering both data popularity and node load, it is possible to control more finely how many replicas are placed on each node to reduce costs as much as possible while meeting latency requirements. Popular data should be placed or replicated in the nodes with low load or better performance to ensure the latency requirements and make full use of node performance, thus avoiding cost wastage due to idle node performance [78, 101]. Instead of focusing on the popularity of each individual replica and the load of each individual node, coarser granularity (i.e., replication based on the popularity of a group of replicas and the centrality of multiple nodes) delivers easier management [76]. Replicas in a group are highly related, which can be found by frequent pattern mining of historical requests. The centrality of a node is used to describe its distance from other nodes. A node with good centrality indicates low latency and bandwidth consumption from other nodes to that node. Therefore, replicating data to those nodes with good centrality incurs fewer costs. Through replicating the grouped data with high popularity into those nodes with good centrality, users can achieve low request latency while reducing the

cost of replication. The preceding kind of dynamic replication that is based on data popularity or node load avoids replicating all data objects across multiple clouds so that the unnecessary storage cost incurred by storing replicas that are requested infrequently is saved while the request requirement is satisfied.

6.4.4 QoS-Aware Replication. QoS includes several measurements such as availability and latency. By measuring QoS, users can detect whether their requirements are met. QoS is mainly influenced by the computing and storage resources allocated by cloud providers, whereas users' perception of QoS is also indirectly influenced by their behavior. For example, a service that promises a QoS response time of 150 ms for users with no more than 1,000 requests can result in high latency when users initiate 10,000 requests. Hence, QoS reflects node load and data popularity to some extent. By measuring the QoS represented by latency, users can calculate the number of replicas needed to meet QoS and then find cost-effective replica placement [52]. There is a tradeoff between cost and QoS [119]. To improve cost-effectiveness, the number of replicas should be moderate rather than a greater number of replicas in pursuit of lower latency or a smaller number of replicas in pursuit of lower costs. Thus, for QoS-aware replication, specifically latency-aware replication, users can increase the number of data that encounter high latency and decrease the number of data whose latency is lower than users' expectation so that the latency of requests is just right for users' requirements, neither too high nor too low. Then replicas can be assigned to the clouds with low latency or minimal costs or both by a greedy algorithm [136]. Additionally, dynamic replica migration can further lower costs as time-varying requirements and performance. From a cost-effective perspective, a migration is performed when the candidate clouds that satisfy QoS requirements can achieve cost savings [79].

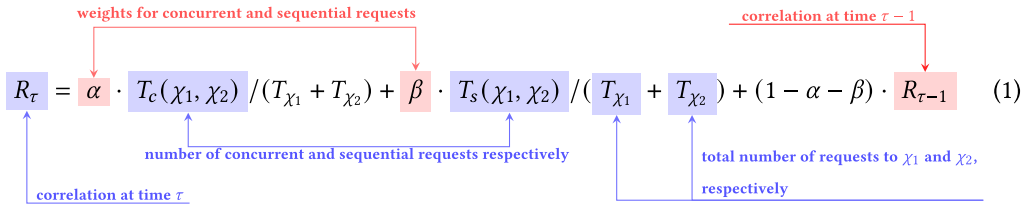
6.4.5 Parallelism Technique. The redundancy methods of chunk-level replication and erasure coding are inherently conducive to parallelism because these two methods divide data into chunks that can be stored in different clouds separately. Hence, they are born with low latency. To complete a data request, they concurrently obtain different data chunks and then construct the original data, which reduces request latency exponentially. Especially, DPRS [77] takes into account the diversity in network environments of different clouds and divides each replica into multiple chunks of varying sizes depending on the network environment. The chunks with large sizes are allocated to those clouds with good network environments, whereas the chunks with small sizes are allocated to those clouds with poor network environments. This kind of chunk-level replication of varying sizes further lowers latency compared with traditional chunk-level replication that divides replicas into equal-sized chunks. Regardless of redundancy methods, multiple replicas can be leveraged to improve request performance by simultaneously issuing multiple requests, each of which fetches a different portion of the replica [109]. This parallel request strategy can expedite the data retrieving process without incurring additional bandwidth costs.

6.4.6 Prefetching Technique. Instead of gaining the low-latency strategy according to the current replica placement status and then updating replica placement based on request history, some studies propose adjusting replica placement in advance to prepare to serve incoming requests with lower latency. The former is passive and belated, whereas the latter is active and prophetic. The latter is also known as prefetching and provides the possibility to further pursue cost-effective low latency. Prefetching enables migration or replication of data in multi-cloud environments in advance according to some factors such as bandwidth and latency so that the cost or latency can be optimized. Cloud-based systems are no strangers to this technique. For example, JointCloud [111], as a cross-cloud cooperation framework, integrates prefetching techniques to improve request performance. Prefetching techniques are generally based on the correlations of data objects. In

Table 6. Comparison of Multi-Objective Optimization for Multi-Cloud Environments

Ref.	Replication Form	Optimization Goals	Constraints	Solution
[134]	File	Cost, latency, availability	n.a.	Meta-heuristic algorithm (self-similarity propagation of plants)
[78]	File	Energy, latency, availability, storage load, etc.	n.a.	Fuzzy inference, meta-heuristic algorithm (self-defense mechanism of plants)
[66]	Chunk	Cost, availability, consistency	Resource capacity limitation	Nonlinear integer programming, heuristic algorithm
[25]	File	Cost, latency	Resource capacity limitation	Mixed integer programming, meta-heuristic algorithm (NSGA-II)

practice, every data object is not completely independent and usually has some kind of association with other data objects, which is usually reflected in the request [75]. In other words, multiple data objects can be requested simultaneously. For example, users generally request video objects along with their preview images, or request a group of video or image objects in the same category at the same time. This phenomenon can be exploited by prefetching techniques. The main focus of prefetching techniques is to analyze the correlations of data objects, which can be accomplished by frequent pattern mining [76] and graph-based model [66, 75] techniques. Frequent pattern mining can discover correlations from historical requests. The graph-based model creates a weighted undirected graph to show correlations. Each node in the graph represents a data object, and each edge in the graph represents the relationship between two data objects. The weights of edges can be derived by counting the number of requests for each pair of replicas during a short interval, and they are dynamically updated [75]. To obtain a finer correlation, requests are distinguished into concurrent and sequential requests. Then, the weights can be calculated based on the number of these two kinds of requests, as shown in Equation (6.4.6) [66]. Finally, the graph is broken into several subgraphs by applying a minimum-cut algorithm or removing the edges whose weights are smaller than a threshold. Each subgraph is a group of highly correlated replicas.



6.5 Multi-Objective Optimization

Facing the complexity of multi-cloud environments, users usually pursue several optimization targets, some of which even conflict with each other. To pursue tradeoffs between several conflicting goals, users can formulate multi-objective optimization problems. Common optimization targets include improved availability, reduced request latency, reduced cost, and consistency, among others. A general method to solve multi-optimization problems is to leverage heuristic or meta-heuristic algorithms to obtain near-optimal solutions. For cost-effectiveness, the multi-objective optimizations can take costs as one of their optimization targets or constraints [25, 66, 78, 134]. Table 6 shows a comparison of these studies in terms of optimization targets, constraints, solutions, and so forth.

6.6 Applications

The diversity of multi-cloud storage in pricing and performance attracts many applications that are built on multiple clouds. Intuitively, due to geo-distributed clouds, multi-cloud storage has

an inherent advantage in content delivery. By leveraging the pricing advantage of multiple clouds, users can achieve economical content delivery, which is beneficial for those small to medium enterprises and government agencies to achieve low-cost content delivery rather than renting expensive content delivery networks [94]. Despite the cost benefit of cloud-based content delivery, efficient content placement strategies remain as open problems, which has an direct impact on cost. Hence, the studies on content delivery based on multi-cloud storage mainly focus on pursuing low cost and high performance by considering coverage and budget [15, 35]. Additionally, cloud-based content delivery can face some specific problems, which also capture the attention of researchers. For example, Cui et al. [29] identify the high tail latency problem in cloud-based content delivery and focus on optimizing cost and tail latency.

Moreover, multi-cloud storage is also beneficial for the cost and performance of social networks where worldwide users share various contents such as images and videos. When cloud-based social networks meet optimization, it is necessary to consider not only the characteristics of the data such as access locality but also the relationship of users in social networks that affects access patterns [52, 64, 137].

7 COST OPTIMIZATION FOR EDGE-CLOUD COLLABORATIVE STORAGE

The emergence of edge computing gives birth to the scheme of edge-cloud storage. Edge nodes have cheaper pricing of storage and bandwidth than cloud nodes, but their performance, such as computation, storage, and energy, is limited. Properly storing data in edge nodes can effectively share the pressure of storage and access in the cloud and reduce the cost of data storage [9]. For example, edge nodes store data with low access load or significant geographical locality of accesses, whereas cloud nodes store data with high access load or significant geographic distribution of accesses. Apart from cheap computing and storage resources, edge storage is also friendly for latency-sensitive data [3]. By placing data closer to users, users have an opportunity to achieve lower access latency. Therefore, edge storage can be a complement to cloud storage, and it gives users an opportunity to optimize costs and improve latency through edge-cloud collaborative storage.

7.1 Cost-Effective Data Replication

Cost optimization can be accomplished by exploring cost-effective data replication for edge-cloud collaborative storage. Edge nodes can be considered as performance-limited cloud nodes when modeling the problem of data replication in the edge-cloud storage environment. Then the problem of cost-effective data replication and replica placement in edge-cloud storage can be regarded as an extension of multi-cloud storage by taking into account some performance-related constraints of nodes. In other words, the problem can be formulated and solved by considering both data-related factors (which refers to the characteristics of data like popularity, size, and locality) and the node-related factors (which refers to the loads of nodes like the utilization of memory, bandwidth, CPU, and storage capacity).

To achieve cost-effectiveness, the number and placement of replicas should be addressed. To this end, different studies have different emphases and consider different factors. Some studies (e.g., [60]) solve the cost-effective number of replicas, whereas other studies (e.g., [9, 17, 59, 61, 98]) solve both the cost-effective number and placement of replicas. The cost-effective number and placement of replicas can be solved by one method at the same time or by two different methods taking into account different factors, respectively. We refer to the former as a one-stage solution and the latter as a two-stage solution. Later, Table 8 summarizes and compares the factors they consider.

Additionally, similar to multi-cloud storage, which is discussed in Section 6, edge-cloud storage replicates data at three redundancy levels consisting of file, chunk, and erasure coding, which

Table 7. Comparison Summary of the Studies Focusing on Data Replication and Replica Placement for Edge-Cloud Collaborative Storage in Terms of Redundancy Level and Considered Factors

Ref.	Redundancy Level	Considered Factors		Solutions
		Data Popularity	Node Load	
[98]	Chunk	✓	✓	Heuristic and meta-heuristic algorithms (ITO)
[9]	File	✓		Approximation algorithm
[58]	Chunk	✓	✓	Prediction-based and heuristic algorithms
[57]	Chunk	✓	✓	Prediction-based and heuristic algorithms
[61]	Chunk	✓	✓	Meta-heuristic algorithm (NSGA-II)
[60]	Chunk	✓	✓	Heuristic algorithm
[59]	Chunk	✓	✓	Heuristic and meta-heuristic algorithms (NSGA-II)
[17]	Erasure coding	✓		Meta-heuristic algorithm (NSGA-II)
Ref.	**Features**			
[98]	Low latency, cost-effectiveness, few migrations, guaranteed SLAs			
[9]	Low latency, low storage cost, low bandwidth cost, temporal locality, spatial locality, geographical locality			
[58]	Low latency, low cost, low energy consumption, automatic scaling			
[57]	Low latency, low cost, guaranteed availability, guaranteed reliability, parallelism, automatic scaling			
[61]	Low latency, high availability, low bandwidth consumption, load balance, consistency			
[60]	Low storage cost, low bandwidth cost, high reliability, consistency			
[59]	Low latency, low bandwidth consumption, high availability, load balance, consistency, improved throughput, improved reliability, effective storage utilization			
[17]	Low latency, low cost (including storage, bandwidth, and operation), high availability			

respectively are addressed by Shao et al. [98], Li et al. [57–61], and Cao et al. [17]. In particular, Cao et al. [17] disperse erasure coded data chunks across storage nodes in the edge and the cloud, and they pay attention to the availability of popular data in regions (i.e., the coverage area of edge nodes) while taking into account the factors of cost and latency simultaneously.

Table 7 summarizes a comparison of these studies focusing on data replication and placement for edge-cloud collaborative storage in terms of redundancy level, considered factors, solutions, and features. More detailed considered factors will be given in Table 8.

7.2 Cost-Effective Auto-Scaling

Further, when using edge-cloud collaborative storage, the incurred cost can be reduced not only by dynamically adjusting data placement but also by dynamically scaling storage nodes, which leads to studies on automatic scaling for storage nodes in the collaborative storage environment [57, 58]. Instead of cost-effectively placing data on provisioned storage nodes so that the cost related to data such as storage cost, bandwidth cost, and request cost is optimized, automatic scaling allows users to rent or release some nodes according to their workloads during subsequent usage. When a storage node gets overloaded, its performance declines. Therefore, it is necessary to rent more nodes to share the workload pressure of the node to guarantee users' experiences such as latency. In edge-cloud collaborative storage, obviously, newly rented nodes can come from two sources, namely the edge and the cloud. For an overloaded edge node, newly rented nodes from the edge can be further divided into two subsources—that is, these nodes can be rented from the region where the overloaded node is located or from the neighbor regions. After automatic scaling, data placement should be adjusted due to changes in the number and layout of storage nodes. For newly rented nodes, they are empty. Data in other nodes should be migrated or replicated to these nodes. For released nodes, the data stored in them should be migrated to other nodes to ensure data reliability. Additionally, the number and placement of replicas can be re-computed according to data popularity and node load, when the number and layout of storage nodes are stable, which means no auto-scaling is required. The factors considered by the re-computation of number and placement are similar to other works [59, 60, 98] discussed previously.

Auto-scaling and migration have their exclusive considerations of factors. The cost incurred by renting storage nodes is considered in addition to the workloads of nodes when performing auto-scaling. When performing migration, the node load, migration time, and migration cost are considered.

7.3 Cost-Effective Request Allocation

How user requests are allocated among storage nodes is also an important factor affecting costs [56]. If the nodes in a region are overloaded and cannot guarantee the QoS of users, common practice is to redirect the requests to remote cloud nodes and then lease more nodes in the region to cope with the possible high workload of the following requests. This is not what users expect because remote cloud storage nodes generally lead to high bandwidth costs. In the meantime, lazy renting of edge storage nodes also has a risk of lowing cost-effectiveness because the requests in the region where new nodes are rented may keep a low level after renting. Therefore, to pursue maximum cost-effectiveness, all requests had better be allocated to edge nodes. However, a number of provisioned edge storage nodes are also not cost-effective because these nodes are idle when the workloads of requests are kept low but still charge users. Hence, the number of edge nodes should be minimized while ensuring the QoS of users. Lai et al. [56] solve the cost-effective user allocation problem by allocating as many requests as possible to edge nodes and minimizing the number of edge nodes while guaranteeing the QoS of users.

7.4 Cost-Effective Edge Caching

Another perspective of utilizing edge storage as an extension for cloud storage is to cache data at the edge with low latency and cost. The cost mainly concerned with edge caching is bandwidth cost. Through taking advantage of lower bandwidth costs, caching data at the edge brings an opportunity to optimize cost. However, edge caching also brings a risk of cost escalation due to additional storage costs. Hence, it is challenging to find cost-effective cost schemes for the edge.

7.4.1 Cost Optimization on Storage and Bandwidth. In some scenarios, the storage cost incurred by edge caching is competitive to the saved bandwidth cost resulting from edge caching and cannot be ignored. Therefore, the benefits of edge caching come with a cost. The related studies aim to achieve a tradeoff between the incurred costs and the benefits brought by caching [72, 89, 121, 122]. Their optimization objectives, solutions, and other features are summarized in Table 9.

7.4.2 Cost Optimization on Bandwidth Only. In other scenarios, the storage cost is generally nothing compared to the expensive bandwidth, because the resources of the edge are cheaper than that of the cloud and only popular data that incur exhaustive bandwidth costs are cached in the edge. Hence, studies [47, 123] that focus on these scenarios consider minimizing bandwidth costs. In summary, the cost-effectiveness of edge caching is worth the efforts of users. When caching data at the edge, the replica of the data object to be cached can come from the cloud or other storage nodes of the edge. It is a natural choice to get the data to be cached from the cloud when there is no cached replica of the data at the edge. However, when there exist replicas of the data to be cached in other storage nodes of the edge, some challenges appear. Users have to determine whether to get the data to be cached from the cloud or from the other storage nodes of the edge. When getting data from other storage nodes of the edge, one thing that should be noted is that the data can come not only from neighbor nodes but also from nodes multiple hops away. The cost and latency resulting from edge caching are two important factors that users should take into account. The optimization objectives, solutions, and other features of the studies are summarized in Table 9.

Table 8. Classification of Studies on Data Replication and Replica Placement for Edge-Cloud Collaborative Storage as Well as Their Factor Comparison When Making Decisions

Classification	Ref.	Considered Factors	
		When Determining Number	
Number determination	[60]	Storage cost, communication cost, data size, data availability, data reliability, data popularity (historical), node load (CPU, memory, bandwidth, storage space)	
		When Determining Number	*When Determining Placement*
1-Stage determination	[9]	Latency, storage cost	
	[17]	Availability, latency, storage cost, bandwidth cost, request cost	
	[61]	Bandwidth cost, data availability, node load (CPU, memory, bandwidth, storage space)	
2-Stage determination	[98]	Data popularity (historical), node load (storage space)	Storage cost, bandwidth, distance, transmission cost, multicast cost
	[59]	Latency, data size, data popularity (historical and predicted)	Distance, node load (I/O performance, CPU, memory)
		When Determining Scaling	*When Determining Migration*
Auto-scaling	[57] [58]	Node lease cost, overload cost and load (CPU, memory, bandwidth, storage space)	Migration time and cost, node load (CPU, memory, bandwidth, storage space)
		When Determining Number	*When Determining Placement*
		Bandwidth, data size, data popularity (predicted)	Data availability, bandwidth cost, data popularity (predicted), node load (CPU, memory, bandwidth, storage space)
		When Determining the Allocation Between Users and Nodes (or Servers)	
Allocation	[56]	Node coverage area, number and load (CPU, memory, bandwidth, storage), QoS	

7.4.3 Consideration of Heterogeneous Pricing Schemes in the Edge. Heterogeneous pricing schemes mean that the storage unit price and the bandwidth unit price of storage nodes in the edge are different based on data and node characteristics, whereas homogeneous pricing schemes mean the storage unit price of these nodes and the bandwidth unit price between two nodes in the edge are identical. Han et al. [45] propose to leverage online algorithms to solve the cost-effective edge caching problem by minimizing storage and bandwidth costs for homogeneous and heterogeneous pricing schemes of storage nodes in the edge, respectively.

7.5 Comparison Summary

Table 8 shows the classification of studies in Sections 7.1, 7.2, and 7.3, and a comparison of considered factors. Table 9 shows a comparison of cost-effective edge caching discussed in Section 7.4.

8 COST-EFFECTIVE CLOUD-BASED CACHES

Through utilizing cloud-based caching, the amount of data that are transferred frequently can be reduced, which leads to several benefits such as reduced bandwidth costs and improved request performance [8]. Nevertheless, the benefits of cloud-based caching are not free. Instead, they come at a cost. It is generally a tough decision for users to trade cost-effectiveness for performance.

8.1 Cost Optimization for Cloud Cache Services

Cloud providers provide cloud cache services, such as Amazon ElastiCache [4], which give an opportunity to optimize the cost of data caching.

8.1.1 Cost Optimization Based on Billing Methods. Cloud cache services generally offer two billing methods, namely pay-as-you-go and reservation billing methods as discussed in Section 2.3, which give users an opportunity to optimize costs by combining them appropriately. Because it is cost-effective to use the pay-as-you-go billing method when users require short-term data storage and the reservation billing method when users require long-term data storage. For this purpose,

Table 9. Comparison of Cost-Effective Edge Caching

Ref.	Objective	C0C0C0Solution	Features
[72]	Minimizing cost (storage, bandwidth), maximizing caching benefit	Integer programming	Prediction
[89]	Minimizing latency and cost (VM, storage, migration (bandwidth, latency))	Online algorithm, dual decomposition, randomized algorithm	Request routing, resource allocation
[121]	Minimizing cost (storage), maximizing caching benefit	Lexicographic goal programming	Graph-based model
[122]	Minimizing latency and cost (storage, bandwidth, QoS penalty)	Lyapunov optimization	Graph-based model
[47]	Minimizing cost (bandwidth)	Transfer learning, greedy algorithm, K-means algorithm	Prediction
[123]	Minimizing cost (bandwidth)	Integer programming, approximation algorithm	Graph-based model
[45]	Minimizing cost (storage, bandwidth)	Online algorithm	Homogeneous pricing, heterogeneous pricing
Ref.	**Considered Factors**		
[72]	Data popularity, storage cost, bandwidth cost, node coverage		
[89]	Latency, node capacity (connection, bandwidth, storage), bandwidth cost, storage cost, node deployment cost		
[121]	Distance, latency, node coverage, node adjacency, node capacity		
[122]	Distance, QoS, node coverage, node adjacency, node capacity, storage cost, bandwidth cost, latency		
[47]	Data popularity, distance, node capacity		
[123]	Distance, latency, node adjacency		
[45]	Storage cost, transmission cost		

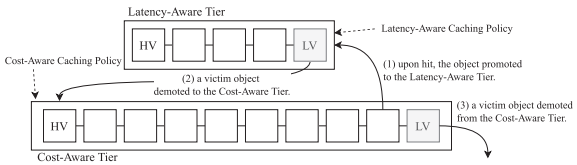


Fig. 11. Illustration of two-tier structure for caching and data migration between tiers [46].

the studies of Wang et al. [113] and Yang et al. [128], which leverage online algorithms to optimize costs for pay-as-you-go and reservation billing methods, can be applied for cloud cache services such as Amazon ElastiCache to dynamically choose cost-effective billing methods for users as workloads vary over time.

8.1.2 Cost Optimization Based on Tiered Caches. To pursue simultaneous optimization of cost-effectiveness and performance, tiered caches attract attention. Hou and Chen [46] adopt a logical two-tier design: each tier is designed for a dedicated optimization objective (either low latency or low cost), as shown in Figure 11. Accordingly, the two tiers respectively execute latency-aware and cost-aware optimization algorithms. Amazon ElastiCache provides a service of data tiering for caching by taking advantage of SSDs [96]. The data tiering service places the data that are infrequently accessed or latency insensitive on SSDs, and the data that are frequently accessed or latency sensitive in memory. Obviously, the tiering service expands the capacity of cloud cache services while reducing the incurred costs. Apart from SSDs, various types of memories such as ersistent memory or **Non-Volatile Memory (NVM)** [28] are another option for building tiered cache schemes. Compared to general **Dynamic Random-Access Memory (DRAM)**, NVM has a low cost and correspondingly low performance but is better than SSD. Thus, NVM is somewhere between DRAM and SSD in price and performance. The tiered cache schemes based on DRAM and NVM are addressed by Mnemo [31], which aims to achieve a trade-off between performance and costs. For this trade-off, it determines the capacity ratio between DRAM and NVM by analyzing data workload, so as to maximize the cost-effectiveness without significantly affecting performance.

8.2 Cost-Effective Caches Built on Cloud Services

The cloud cache services natively offered by cloud providers sometimes cannot satisfy users' requirements. Thus, users leverage existing cloud services to build their own caches optimized for their requirements. To further pursue cost-effectiveness when using caching for cloud-based systems, some studies leverage some features of cloud services to develop cost-effective caching schemes and frameworks. An effective way to achieve the goal of cost-effectiveness is to prevent the cold data from replacing cached hot data, and dynamically resize the cache space to avoid unnecessary costs of idle space [135]. The elasticity of cloud services provides an opportunity to easily and dynamically resize the cache space. Through utilizing the elasticity of cloud services, storage resources can scale with users' requirements, which inspires an opportunity to optimize costs.

8.2.1 Caches Built on Elastic Computing Instances. The elasticity offered by elastic computing instances of cloud services is a good choice to be leveraged for cost optimization [18, 19]. Users can increase or decrease their compute capacity depending on the demands of their application and only pay the specified per hourly rates for the instance they use.

Compared with on-demand instances, spot instances enjoy a price discount of up to 90% at the same performance [7]. The main feature of spot instances is that their renting is based on bidding, and the instances can be interrupted by cloud providers at any time due to bidding failures. Users achieve success in the bid when their bidding price is higher than the spot price, and fail when the spot price rises and exceeds the bidding price. Hence, spot instances only are suitable for various stateless, fault-tolerant, or flexible applications, although they have a very attractively low price. A possible cost-effective cache scheme based on spot instances is to place relatively unpopular or stateless caches in spot instances and relatively popular or state-changing caches in on-demand instances [46, 108, 125], because the cost of caching missing due to the revocation of spot instances is not high for those unpopular or stateless caches. Additionally, to cost-effectively avoid performance degradation due to spot instance revocations, the caches in spot instances can be backed up to cheap burstable instances [108] or replicated to the spot instances in other regions [125].

8.2.2 Caches Built on Stateless Cloud Services. **Function-as-a-Service (FaaS)** is a burgeoning computing paradigm and provides a new way of deploying applications, which has been offered as a proven service by several major cloud providers, such as Amazon Lambda [5]. Through utilizing FaaS, developers do not need to care about resource scaling, which is all supported by FaaS providers. Functions are executed on demand, and their states are cached by FaaS providers, so the cached data in their memory are retained between functions, which offers a sufficient lifetime for data caching. Users are charged only when they invoke a function, which brings a cost-effective opportunity. Even so, FaaS introduces non-trivial challenges for caching data because it has limited resource capacities and can be revoked by providers at any time, which creates a risk of losing cached data.

Distinctive from other works [108, 125], InfiniCache [107] is completely built and deployed atop ephemeral serverless functions by leveraging FaaS services. InfiniCache uses several policies in terms of connection management, anticipatory control of the billing cycle, and a function state monitor to work around the inherent capacity limitations of FaaS with cost-effectiveness, and it leverages erasure coding to provide fault tolerance against data loss resulting from function revocations and improve performance such as latency by utilizing parallelism. Meanwhile, it periodically performs a delta backup mechanism for functions to further avoid data loss. InfiniCache achieves performance comparable to Amazon ElastiCache and improves the cost-effectiveness by 31 to 96×.

InfiniCache strives to avoid the risk of data loss between serverless functions. If there is a storage solution for data between serverless functions with high availability and performance, the cache

Table 10. Classification of Cloud-Based Cache Schemes

Ref.	Infrastructure
[96]	VM instances (memory and SSD)
[31]	Memory optimized VM instances (DRAM and NVM)
[18, 19]	On-demand instances
[125]	On-demand and spot instances
[108]	On-demand, spot, and burstable instances
[107]	Function-as-a-Service (FaaS)

schemes based on FaaS can free up hands to focus on cost-effective data management. The purpose of building cost-effective ephemeral data storage for serverless functions spawns several studies, such as those of Klimovic et al. [55] and Pu et al. [90]. Both studies achieve cost-effectiveness by building tiered storage.

8.3 Comparison Summary

Table 10 summarizes the classification of the preceding cloud-based cache schemes in terms of what cloud services they are built on.

9 OPEN CHALLENGES AND FUTURE DIRECTIONS

Given the recent advances, the future directions of cost optimization for cloud storage from user perspectives are becoming clear. This section presents a summary and discusses future directions as well as challenges of different aspects of cost optimization.

9.1 Cost-Effective Deduplication

First, data deduplication can be used to save storage and bandwidth costs for cloud storage but at the expense of sacrificing read and write performance. Hence, the main direction of data deduplication is to pursue tradeoffs between cost and performance. Additionally, when performing data deduplication for some storage services such as Amazon S3, users need to lease computing instances from cloud providers or maintain local servers to pre-process data uploading to S3, which incurs costs. In this case, there exists a direction of cost optimization for performing data deduplication, which strives to maximize storage cost savings in the cloud at the lowest cost of renting computing instances or maintaining local servers.

9.2 Cost-Effective Single-Cloud Storage

For users who store their data in a single cloud, the direction of cost optimization is to construct tiered storage. Tiered storage can be built on multiple pricing options of a storage service. Recent advances on cost optimization for single-cloud storage mainly focus on building cost-effective two- or three-tier storage. For future directions, building tiered storage based on more tiers is an opportunity to further optimize costs. Meanwhile, tiered storage built on multiple services is another direction for cost optimization. Through storing data in storage services with different pricing and different features, costs can be optimized while complex requirements of users can be satisfied. As the variety of storage services offered by a cloud storage provider grows, so will the variety of tiered storage, which is the main direction for cost optimization for single-cloud storage.

9.3 Cost-Effective Multi-Cloud Storage

For users storing data in multiple clouds, cost optimization is mainly achieved by leveraging the pricing and performance differences among multiple storage services and storage providers.

9.3.1 Tiered Multi-Cloud Storage. One future direction is to build tiered storage on multiple clouds, which is more complex than that based on storage services offered by a single cloud.

First, complex pricing schemes across multiple clouds make cost optimization more challenging. Even for the same type of storage service, pricing schemes vary widely from provider to provider. For example, both Alibaba Cloud and Amazon provide object storage services, but their pricing schemes and features are different from each other. How to make the most of these differences is a future direction to realize cost optimization.

Second, migrating data among clouds may not be as easy as migrating data among multiple services in a single cloud. Users can encounter vendor lock-in and other obstructions due to competition among cloud providers. Although some studies have made some attempts in this direction, users still face difficulties when it comes to concrete implementation. For example, when migrating a data object from a cloud to another cloud, how to handle requests cost-effectively during the migration process is a problem faced by users. A trivial solution is to keep the data object in the original cloud until the migration is complete, but it incurs more storage costs due to redundancy during the process of migration. Hence, for this cost optimization direction, users should be dedicated to more realistic and effective algorithms.

9.3.2 Tradeoffs in Multi-Cloud Storage. Another direction is to cost-effectively replicate data across multiple clouds while ensuring the requirements of users. The main purpose for users to disperse data across clouds is to improve data availability and reduce request latency. Both high availability and low latency can be achieved by redundancy. There is a tradeoff among availability, latency, and cost. Therefore, this direction pursues more cost-effective redundancy approaches so as to optimize storage costs while ensuring high availability and low latency.

Apart from redundancy approaches, the degree of redundancy and the placement of redundant data (e.g., the number and placement of replicas) also have a significant impact on cost optimization. As pricing varies from cloud to cloud, it is challenging for users to find cost-effective degree and placement of redundancy data according to time-varying data workloads and user requirements, which gives rise to one of the future directions of cost optimization.

Additionally, when users pay attention to latency optimization, the future direction is to consider on the one hand latency-oriented replica placement to find a tradeoff between latency and cost, and on the other hand request techniques for data in multiple clouds to reduce request latency.

In addition to latency and availability, there are some other optimization goals that may also conflict with cost. Users need to make tradeoffs between cost and other optimization goals based on their requirements.

9.3.3 Multi-Cloud Storage Management. Further, as clouds are independent and compete with each other, users need to spend great efforts to manage multiple clouds and the data stored in them. Hence, frameworks or middlewares that facilitate the management of multiple clouds are another pursuit in future directions.

9.4 Cost-Effective Edge-Cloud Collaborative Storage

For the future directions of cost optimization for edge-cloud collaborative storage, users can follow the following aspects.

First, edge-cloud collaborative storage can be considered as multi-cloud storage with limited resources of computing and storage and coverage of service. Due to the limited resources and coverage of the edge, users need to determine the values of placing data in the edge. If cost savings and performance improvements are not as expected, it is not worth placing data at the edge. Users need to design efficient algorithms to decide data placement in edge-cloud environments.

Second, auto-scaling for computing and storage resources of the edge brings an opportunity to store more data in inexpensive resources and release idle resources. Scaling up has a risk that the cost savings of storing more data at the edge cannot outweigh the cost of leasing more resources. Similarly, scaling down has a risk that the saved cost of releasing edge resources is lower than the incurred cost of requesting and storing data in clouds. Hence, users need to design cost-effective auto-scaling algorithms for edge resources in edge-cloud collaborative environments.

Third, in complex edge environments, request allocation also has an impact on cost optimization since the regions covered by different edge servers may overlap. Reasonable allocation of requests within overlapping coverage allows the full utilization of edge resources, resulting in cost savings.

Moreover, evolving edge computing is expected to introduce more mature edge services in the future market, which may attract users to design more cost-effective algorithms to leverage them.

9.5 Cost-Effective Cloud-Based Caches

The future directions of cost optimization for data caching are summarized in two aspects. First, users need to design algorithms to decide whether data objects are cache-worthy or not, which means that users need to make a tradeoff between incurred costs and improved performance or between incurred storage costs and potential bandwidth cost savings.

Second, users need to design algorithms to utilize cloud cache services cost-effectively. The main idea behind optimizing the costs of cloud services is to take advantage of differences in pricing schemes and billing methods as well as performance. Additionally, apart from caching data in off-the-shelf cloud services such as Amazon ElastiCache, users can lease cloud resources to build cloud-based caches to further optimize costs. It is challenging for users to determine which kind of cloud resources to lease to build cost-effective caches according to their requirements, which spawns a future direction of cost optimization for caches.

9.6 Emerging AI-Based Solutions

AI-based strategies are becoming a trend to solve complex optimization problems in cloud and edge computing environments [43, 53]. This kind of strategies has the potential to optimize search over a large parameter space so that the objective of cost can be solved optimally. When AI meets the cost optimization problem in the cloud, it will face the following future directions:

- Cost optimization in the cloud is an online problem. Lack of future access information is its major challenge, which can lead to laziness and misjudgments in decision making. To overcome this challenge, it is imperative to develop more exact prediction methods based on **Machine Learning (ML)** and **Deep Learning (DL)** to support proactive decision making.
- Reinforcement learning focuses on online learning, which achieves objectives through the trial-and-error approach. The reinforcement theory continues to evolve, and its latest methods can be used in the area of cost optimization in the cloud.
- For tiered and multi-cloud storage, ML- or DL-based methods can be used to proactively select cost-effective tiers and data placement according to access patterns and QoS requirements.
- For edge storage, edge intelligence is a worthy direction. ML- and DL-based methods can be deployed at the edge to reduce data transmissions. Due to limited computing resources of the edge, it is crucial to develop high-precision and lightweight AI-based methods to enable real-time performance. In terms of collaboration between the edge and the cloud, federated learning is a concern to take advantage of the diversity of data in multiple edge regions to achieve cost performance improvement.

- For FaaS-based storage, the major challenge is how to manage data among ephemeral functions. AI-based methods can be used for state management between serverless functions and dynamic capacity expansion to avoid data loss.

10 CONCLUSION

In this article, we presented a survey on cost optimization for cloud storage from users' perspectives, with the aim of inspiring users of the increasingly prosperous cloud storage to save costs. First, we described the background of cost optimization for cloud storage, and then we detailed several opportunities and motivations for cost optimization. The utilization of these opportunities should be careful, or inappropriate utilization can backfire—that is, incurring more costs rather than saving costs. After that, we gave an overview of the aspects of cost optimization for cloud storage, then exhaustively discussed these aspects including high storage efficiency, cost optimization for single-cloud storage, cost optimization for multi-cloud storage, cost optimization for edge-cloud collaborative storage, and cost-effectiveness supported by caches, respectively.

Looking ahead, with the evolving cloud storage market and changing user requirements, new storage services are on their way. The cost of data storage will remain an inevitable topic. In the future, cost optimization of data storage will also evolve with storage paradigms. The pursuit of cost-effective cloud storage will go hand in hand with cloud storage and be pursued by a wide range of users. It is our hope that this article can give users some insights into cost optimization for cloud storage and inspire them to develop more cost-effective solutions.

REFERENCES

[1] Edge. 2022. Edge Networking. Retrieved June 22, 2022 from https://edge.network.

[2] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. 2010. RACS: A case for cloud storage diversity. In *Proceedings of ACM SoCC*. 229–240.

[3] Ahmed Ali-Eldin, Bin Wang, and Prashant J. Shenoy. 2021. The hidden cost of the edge: A performance comparison of edge and cloud latencies. In *Proceedings of SC*. Article 23, 12 page.

[4] Amazon. 2022. Amazon ElastiCache. Retrieved February 15, 2022 from https://aws.amazon.com/elasticache.

[5] Amazon. 2022. AWS Lambda. Retrieved February 15, 2022 from https://aws.amazon.com/lambda.

[6] Amazon. 2022. Cloud Storage on AWS: S3. Retrieved June 22, 2022 from https://aws.amazon.com/products/storage.

[7] Amazon. 2022. Amazon EC2 Spot Instances. Retrieved February 15, 2022 from https://aws.amazon.com/ec2/spot.

[8] Bo An, Yan Li, Junming Ma, Gang Huang, Xiangqun Chen, and Donggang Cao. 2019. DCStore: A deduplication-based cloud-of-clouds storage service. In *Proceedings of IEEE ICWS*. 291–295.

[9] Atakan Aral and Tolga Ovatman. 2018. A decentralized replica placement algorithm for edge computing. *IEEE Trans. Netw. Serv. Manag.* 15, 2 (2018), 516–529.

[10] Microsoft Azure. 2022. Azure Blob Storage. Retrieved June 22, 2022 from https://azure.microsoft.com/services/storage/blobs.

[11] Han Bao, Yijie Wang, and Fangliang Xu. 2020. An adaptive erasure code for JointCloud storage of Internet of Things big data. *IEEE Internet Things J.* 7, 3 (2020), 1613–1624.

[12] Ben Blamey, Fredrik Wrede, Johan Karlsson, Andreas Hellander, and Salman Toor. 2019. Adapting the secretary hiring problem for optimal hot-cold tier placement under top-K workloads. In *Proceedings of IEEE/ACM CCGRID*. 576–583.

[13] Djillali Boukhelef, Kamel Boukhalfa, Jalil Boukhobza, Hamza Ouarnoughi, and Laurent Lemarchand. 2017. COPS: Cost based object placement strategies on hybrid storage system for DBaaS cloud. In *Proceedings of IEEE/ACM CCGRID*. 659–664.

[14] Djillali Boukhelef, Jalil Boukhobza, Kamel Boukhalfa, Hamza Ouarnoughi, and Laurent Lemarchand. 2019. Optimizing the cost of DBaaS object placement in hybrid storage systems. *Future Gener. Comput. Syst.* 93 (2019), 176–187.

[15] James Broberg, Rajkumar Buyya, and Zahir Tari. 2009. MetaCDN: Harnessing 'storage clouds' for high performance content delivery. *J. Netw. Comput. Appl.* 32, 5 (2009), 1012–1022.

[16] Robson A. Campêlo, Marco A. Casanova, Dorgival O. Guedes, and Alberto H. F. Laender. 2020. A brief survey on replica consistency in cloud environments. *J. Internet Serv. Appl.* 11, 1 (2020), 1.

[17] E. Cao, P. Wang, C. Yan, and C. Jiang. 2020. A cloudedge-combined data placement strategy based on user access regions. In *Proceedings of BigDIA*. 243–250.

[18] Damiano Carra, Giovanni Neglia, and Pietro Michiardi. 2019. TTL-based cloud caches. In *Proceedings of IEEE INFOCOM*. 685–693.
[19] Damiano Carra, Giovanni Neglia, and Pietro Michiardi. 2020. Elastic provisioning of cloud caches: A cost-aware TTL approach. *IEEE/ACM Trans. Netw.* 28, 3 (2020), 1283–1296.
[20] Ceph. 2022. Ceph Docs. Retrieved February 15, 2022 from https://docs.ceph.com.
[21] Lixing Chen, Sheng Zhou, and Jie Xu. 2018. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. *IEEE/ACM Trans. Netw.* 26, 4 (2018), 1619–1632.
[22] Min Chen, Yixue Hao, Kai Lin, Zhiyong Yuan, and Long Hu. 2018. Label-less learning for traffic control in an edge network. *IEEE Netw.* 32, 6 (2018), 8–14.
[23] Yingying Cheng, Fan Zhang, Gang Hu, Yiwen Wang, Hanhui Yang, Gong Zhang, and Zhuo Cheng. 2021. Block popularity prediction for multimedia storage systems using spatial-temporal-sequential neural networks. In *Proceedings of ACM MM*. 3390–3398.
[24] Giovanni Cherubini, Yusik Kim, Mark Lantz, and Vinodh Venkatesan. 2017. Data prefetching for large tiered storage systems. In *Proceedings of IEEE ICDM*. 823–828.
[25] Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, and Jalil Boukhobza. 2021. *StorNIR*, a multi-objective replica placement strategy for cloud federations. In *Proceedings of ACM/SIGAPP SAC*. 50–59.
[26] Asaf Cidon, Stephen M. Rumble, Ryan Stutsman, Sachin Katti, John K. Ousterhout, and Mendel Rosenblum. 2013. Copysets: Reducing the frequency of data loss in cloud storage. In *Proceedings of USENIX ATC*. 37–48.
[27] Alibaba Cloud. 2022. Edge Node Service. Retrieved June 22, 2022 from https://www.alibabacloud.com/product/edge-node-service.
[28] Alibaba Cloud. 2022. Persistent Memory Usage. Retrieved February 15, 2022 from https://www.alibabacloud.com/help/en/doc-detail/188251.htm.
[29] Yong Cui, Ningwei Dai, Zeqi Lai, Minming Li, Zhenhua Li, Yuming Hu, Kui Ren, and Yuchi Chen. 2019. TailCutter: Wisely cutting tail latency in cloud CDNs under cost constraints. *IEEE/ACM Trans. Netw.* 27, 4 (2019), 1612–1628.
[30] Hariharan Devarajan, Anthony Kougkas, and Xian-He Sun. 2020. HReplica: A dynamic data replication engine with adaptive compression for multi-tiered storage. In *Proceedings of IEEE Big Data*. 256–265.
[31] Thaleia Dimitra Doudali and Ada Gavrilovska. 2018. Mnemo: Boosting memory cost efficiency in hybrid memory systems. In *Proceedings of ACM SoCC*. 523.
[32] Abhinav Duggal, Fani Jenkins, Philip Shilane, Ramprasad Chinthekindi, Ritesh Shah, and Mahesh Kamat. 2019. Data domain cloud tier: Backup here, backup there, deduplicated everywhere! In *Proceedings of USENIX ATC*. 647–660.
[33] Jinlong E, Yong Cui, Zhenhua Li, Mingkang Ruan, and Ennan Zhai. 2020. HyCloud: Tweaking hybrid cloud storage services for cost-efficient filesystem hosting. *IEEE/ACM Trans. Netw.* 28, 6 (2020), 2629–2642.
[34] Abdelkarim Erradi and Yaser Mansouri. 2020. Online cost optimization algorithms for tiered cloud storage services. *J. Syst. Softw.* 160 (2020), 110457.
[35] Ghazaleh Eslami and Abolfazl Toroghi Haghighat. 2017. A new surrogate placement algorithm for cloud-based content delivery networks. *J. Supercomput.* 73, 12 (2017), 5310–5331.
[36] Flexera. 2021. Flexera Releases 2021 State of the Cloud Report. Retrieved November 22, 2022 from https://www.flexera.com/about-us/press-center/flexera-releases-2021-state-of-the-cloud-report.
[37] Market Data Forecast. 2021. Cloud Storage Market Research Report. Retrieved June 22, 2022 from https://www.marketdataforecast.com/market-reports/cloud-storage-market.
[38] Yinjin Fu, Nong Xiao, Tao Chen, and Jian Wang. 2022. Fog-to-multicloud cooperative ehealth data management with application-aware secure deduplication. *IEEE Trans. Dependable Secur. Comput.* 19, 5 (2022), 3136–3148.
[39] Yinjin Fu, Nong Xiao, Hong Jiang, Guyu Hu, and Weiwei Chen. 2019. Application-aware big data deduplication in cloud environment. *IEEE Trans. Cloud Comput.* 7, 4 (2019), 921–934.
[40] Ammar Gharaibeh, Abdallah Khreishah, Bo Ji, and Moussa Ayyash. 2016. A provably efficient online collaborative caching algorithm for multicell-coordinated systems. *IEEE Trans. Mob. Comput.* 15, 8 (2016), 1863–1876.
[41] Amal Ghorbel, Mahmoud Ghorbel, and Mohamed Jmaiel. 2017. Privacy in cloud computing environments: A survey and research challenges. *J. Supercomput.* 73, 6 (2017), 2763–2800.
[42] Navneet Kaur Gill and Sarbjeet Singh. 2016. A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. *Future Gener. Comput. Syst.* 65 (2016), 10–32.
[43] Sukhpal Singh Gill, Minxian Xu, Carlo Ottaviani, Panos Patros, Rami Bahsoon, Arash Shaghaghi, Muhammed Golec, et al. 2022. AI for next generation computing: Emerging trends and future directions. *Internet of Things* 19 (2022), 100514.
[44] Gluster. 2022. GlusterFS Documentation. Retrieved February 15, 2022 from https://docs.gluster.org.
[45] Xinxin Han, Guichen Gao, Yang Wang, Hing-Fung Ting, Ilsun You, and Yong Zhang. 2021. Online data caching in edge computing. *Concurr. Comput.* Early view, July 11, 2021.

[46] Binbing Hou and Feng Chen. 2017. GDS-LC: A latency- and cost-aware client caching scheme for cloud storage. *ACM Trans. Storage* 13, 4 (2017), Article 40, 33 pages.
[47] Tingting Hou, Gang Feng, Shuang Qin, and Wei Jiang. 2018. Proactive content caching by exploiting transfer learning for mobile edge computing. *Int. J. Commun. Syst.* 31, 11 (2018), e3706.
[48] Ta Yuan Hsu and Ajay D. Kshemkalyani. 2019. A proactive, cost-aware, optimized data replication strategy in geo-distributed cloud datastores. In *Proceedings of IEEE/ACM UCC*. 143–153.
[49] Ying-Feng Hsu, Ryo Irie, Shuuichirou Murata, and Morito Matsuoka. 2018. A novel automated cloud storage tiering system through hot-cold data classification. In *Proceedings of IEEE CLOUD*. 492–499.
[50] Huayi Jin, Chentao Wu, Xin Xie, Jie Li, Minyi Guo, Hao Lin, and Jianfeng Zhang. 2019. Approximate code: A cost-effective erasure coding framework for tiered video storage in cloud systems. In *Proceedings of ICPP*. Article 95, 10 pages.
[51] Nesrine Kaaniche and Maryline Laurent. 2017. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. *Comput. Commun.* 111 (2017), 120–141.
[52] Hourieh Khalajzadeh, Dong Yuan, Bing Bing Zhou, John C. Grundy, and Yun Yang. 2020. Cost effective dynamic data placement for efficient access of social networks. *J. Parallel Distrib. Comput.* 141 (2020), 82–98.
[53] Tahseen Khan, Wenhong Tian, Guangyao Zhou, Shashikant Ilager, Mingming Gong, and Rajkumar Buyya. 2022. Machine learning (ML)-centric resource management in cloud computing: A review and future directions. *J. Netw. Comput. Appl.* 204 (2022), 103405.
[54] Abdennacer Khelaifa, Saber Benharzallah, Laïd Kahloul, Reinhardt Euler, Abdelkader Laouid, and Ahcène Bounceur. 2019. A comparative analysis of adaptive consistency approaches in cloud storage. *J. Parallel Distributed Comput.* 129 (2019), 36–49.
[55] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. 2018. Pocket: Elastic ephemeral storage for serverless analytics. In *Proceedings of OSDI*. 427–444.
[56] Phu Lai, Qiang He, John Grundy, Feifei Chen, Mohamed Abdelrazek, John G. Hosking, and Yun Yang. 2022. Cost-effective app user allocation in an edge computing environment. *IEEE Trans. Cloud Comput.* 10, 3 (2022), 1701–1713.
[57] Chunlin Li, Jingpan Bai, Yuan Ge, and Youlong Luo. 2020. Heterogeneity-aware elastic provisioning in cloud-assisted edge computing systems. *Future Gener. Comput. Syst.* 112 (2020), 1106–1121.
[58] Chunlin Li, Jun Liu, Bo Lu, and Youlong Luo. 2021. Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment. *J. Netw. Comput. Appl.* 180 (2021), 103017.
[59] Chunlin Li, Mingyang Song, Min Zhang, and Youlong Luo. 2020. Effective replica management for improving reliability and availability in edge-cloud computing environment. *J. Parallel Distrib. Comput.* 143 (2020), 107–128.
[60] Chunlin Li, Chengyi Wang, Hengliang Tang, and Youlong Luo. 2019. Scalable and dynamic replica consistency maintenance for edge-cloud system. *Future Gener. Comput. Syst.* 101 (2019), 590–604.
[61] Chunlin Li, YaPing Wang, Hengliang Tang, and Youlong Luo. 2019. Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud. *Future Gener. Comput. Syst.* 100 (2019), 921–937.
[62] Said Limam, Riad Mokadem, and Ghalem Belalem. 2019. Data replication strategy with satisfaction of availability, performance and tenant budget requirements. *Clust. Comput.* 22, 4 (2019), 1199–1210.
[63] Chuan Lin, Qiang Cao, Jianzhong Huang, Jie Yao, Xiaoqian Li, and Changsheng Xie. 2018. HPDV: A highly parallel deduplication cluster for virtual machine images. In *Proceedings of IEEE/ACM CCGRID*. 472–481.
[64] Guoxin Liu, Haiying Shen, and Harrison Chandler. 2016. Selective data replication for online social networks with distributed datacenters. *IEEE Trans. Parallel Distrib. Syst.* 27, 8 (2016), 2377–2393.
[65] Guoxin Liu, Haiying Shen, and Haoyu Wang. 2017. An economical and slo-guaranteed cloud storage service across multiple cloud service providers. *IEEE Trans. Parallel Distrib. Syst.* 28, 9 (2017), 2440–2453.
[66] Jinwei Liu, Haiying Shen, Hongmei Chi, Husnu S. Narman, Yongyi Yang, Long Cheng, and Wingyan Chung. 2021. A low-cost multi-failure resilient replication scheme for high-data availability in cloud storage. *IEEE/ACM Trans. Netw.* 29, 4 (2021), 1436–1451.
[67] Jinwei Liu, Haiying Shen, and Husnu S. Narman. 2019. Popularity-aware multi-failure resilient and cost-effective replication for high data durability in cloud storage. *IEEE Trans. Parallel Distrib. Syst.* 30, 10 (2019), 2355–2369.
[68] Mingyu Liu, Li Pan, and Shijun Liu. 2019. To transfer or not: An online cost optimization algorithm for using two-tier storage-as-a-service clouds. *IEEE Access* 7 (2019), 94263–94275.
[69] Mingyu Liu, Li Pan, and Shijun Liu. 2021. Keep hot or go cold: A randomized online migration algorithm for cost optimization in STaaS clouds. *IEEE Trans. Netw. Serv. Manag.* 18, 4 (2021), 4563–4575.
[70] Mingyu Liu, Li Pan, and Shijun Liu. 2022. Effeclouds: A cost-effective cloud-of-clouds framework for two-tier storage. *Future Gener. Comput. Syst.* 129 (2022), 33–49.
[71] Songbin Liu, Xiaomeng Huang, Haohuan Fu, and Guangwen Yang. 2013. Understanding data characteristics and access patterns in a cloud storage system. In *Proceedings of IEEE/ACM CCGRID*. 327–334.

[72] Ying Liu, Qiang He, Dequan Zheng, Mingwei Zhang, Feifei Chen, and Bin Zhang. 2019. Data caching optimization in the edge computing environment. In *Proceedings of IEEE ICWS*. 99–106.
[73] Saiqin Long, Zhetao Li, Zihao Liu, Qingyong Deng, Sangyoon Oh, and Nobuyoshi Komuro. 2020. A similarity clustering-based deduplication strategy in cloud storage systems. In *Proceedings of IEEE ICPADS*. 35–43.
[74] Yung-Feng Lu, Chin-Fu Kuo, Shih-Chun Chou, Jhong-Syuan Li, and Yan-Wei Lai. 2017. Cost-aware software-defined hybrid object-based storage system. In *Proceedings of PDCAT*. 477–482.
[75] Najme Mansouri and Mohammad Masoud Javidi. 2018. A new prefetching-aware data replication to decrease access latency in cloud environment. *J. Syst. Softw.* 144 (2018), 197–215.
[76] Najme Mansouri, Mohammad Masoud Javidi, and Behnam Mohammad Hasani Zade. 2020. Using data mining techniques to improve replica management in cloud environment. *Soft Comput.* 24, 10 (2020), 7335–7360.
[77] Najme Mansouri, M. Kuchaki Rafsanjani, and Mohammad Masoud Javidi. 2017. DPRS: A dynamic popularity aware replication strategy with parallel download scheme in cloud environments. *Simul. Model. Pract. Theory* 77 (2017), 177–196.
[78] Najme Mansouri, Behnam Mohammad Hasani Zade, and Mohammad Masoud Javidi. 2020. A multi-objective optimized replication using fuzzy based self-defense algorithm for cloud computing. *J. Netw. Comput. Appl.* 171 (2020), 102811.
[79] Yaser Mansouri and Rajkumar Buyya. 2019. Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers. *J. Parallel Distrib. Comput.* 126 (2019), 121–133.
[80] Yaser Mansouri and Abdelkarim Erradi. 2018. Cost optimization algorithms for hot and cool tiers cloud storage services. In *Proceedings of IEEE CLOUD*. 622–629.
[81] Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. 2018. Data storage management in cloud environments: Taxonomy, survey, and future directions. *ACM Comput. Surv.* 50, 6 (2018), Article 91, 51 pages.
[82] Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. 2019. Cost optimization for dynamic replication and migration of data in cloud data centers. *IEEE Trans. Cloud Comput.* 7, 3 (2019), 705–718.
[83] Ahmed Awad Mohamed, Rashed K. Salem, Hatem Abdel Kader, and Mustafa Abdul Salam. 2021. A novel intelligent approach for dynamic data replication in cloud environment. *IEEE Access* 9 (2021), 40240–40254.
[84] Rekha Nachiappan, Bahman Javadi, Rodrigo N. Calheiros, and Kenan M. Matawie. 2017. Cloud storage reliability for big data applications: A state of the art survey. *J. Netw. Comput. Appl.* 97 (2017), 35–47.
[85] Kwangsung Oh, Nan Qin, Abhishek Chandra, and Jon B. Weissman. 2020. Wiera: Policy-driven multi-tiered geo-distributed cloud storage system. *IEEE Trans. Parallel Distrib. Syst.* 31, 2 (2020), 294–305.
[86] Openstack. 2022. Swift Docs. Retrieved February 15, 2022 from https://docs.openstack.org/swift.
[87] Hojin Park, Gregory R. Ganger, and George Amvrosiadis. 2020. More IOPS for less: Exploiting burstable storage in public clouds. In *Proceedings of USENIX HotCloud*.
[88] Milan Patel, Brian Naughton, Caroline Chan, Nurit Sprecher, Sadayuki Abeta, and Adrian Neal. 2014. *Mobile-Edge Computing*. Introductory Technical White Paper. Mobile-Edge Computing (MEC) Industry Initiative.
[89] Lingjun Pu, Lei Jiao, Xu Chen, Lin Wang, Qinyi Xie, and Jingdong Xu. 2018. Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks. *IEEE J. Sel. Areas Commun.* 36, 8 (2018), 1751–1767.
[90] Qifan Pu, Shivaram Venkataraman, and Ion Stoica. 2019. Shuffling, fast and slow: Scalable analytics on serverless infrastructure. In *Proceedings of NSDI*. 193–206.
[91] Han Qiu, Hassan Noura, Meikang Qiu, Zhong Ming, and Gerard Memmi. 2021. A user-centric data protection method for cloud storage based on invertible DWT. *IEEE Trans. Cloud Comput.* 9, 4 (2021), 1293–1304.
[92] Han Qiu, Chentao Wu, Jie Li, Minyi Guo, Tong Liu, Xubin He, Yuanyuan Dong, and Yafei Zhao. 2020. EC-fusion: An efficient hybrid erasure coding framework to improve both application and recovery performance in cloud storage systems. In *Proceedings of IEEE IPDPS*. 191–201.
[93] Shweta Saharan, Gaurav Somani, Gaurav Gupta, Robin Verma, Manoj Singh Gaur, and Rajkumar Buyya. 2020. QuickDedup: Efficient VM deduplication in cloud computing environments. *J. Parallel Distrib. Comput.* 139 (2020), 18–31.
[94] Mohammad A. Salahuddin, Jagruti Sahoo, Roch H. Glitho, Halima Elbiaze, and Wessam Ajib. 2018. A survey on content placement algorithms for cloud-based content delivery networks. *IEEE Access* 6 (2018), 91–114.
[95] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.
[96] Amazon Web Services. 2022. ElastiCache Data Tiering. Retrieved February 15, 2022 from https://aws.amazon.com/elasticache/pricing/#Data_tiering.
[97] Ali Shakarami, Mostafa Ghobaei-Arani, Ali Shahidinejad, Mohammad Masdari, and Hamid Shakarami. 2021. Data replication schemes in cloud computing: A survey. *Clust. Comput.* 24, 3 (2021), 2545–2579.
[98] Yanling Shao, Chunlin Li, Zhao Fu, Leyue Jia, and Youlong Luo. 2019. Cost-effective replication management and scheduling in edge computing. *J. Netw. Comput. Appl.* 129 (2019), 46–61.

[99] Young-Joo Shin, Dongyoung Koo, and Junbeom Hur. 2017. A survey of secure data deduplication schemes for cloud storage systems. *ACM Comput. Surv.* 49, 4 (2017), Article 74, 38 pages.

[100] Le Sun, Hai Dong, Farookh Khadeer Hussain, Omar Khadeer Hussain, and Elizabeth Chang. 2014. Cloud service selection: State-of-the-art and future research directions. *J. Netw. Comput. Appl.* 45 (2014), 134–150.

[101] Sheng-Yao Sun, Wenbin Yao, and Xiaoyong Li. 2018. DARS: A dynamic adaptive replica strategy under high load cloud-P2P. *Future Gener. Comput. Syst.* 78 (2018), 31–40.

[102] Sheng-Yao Sun, Wenbin Yao, Baojun Qiao, Ming Zong, Xin He, and Xiaoyong Li. 2019. RRSD: A file replication method for ensuring data reliability and reducing storage consumption in a dynamic cloud-P2P environment. *Future Gener. Comput. Syst.* 100 (2019), 844–858.

[103] Choon Beng Tan, Mohd. Hanafi Ahmad Hijazi, Yuto Lim, and Abdullah Gani. 2018. A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends. *J. Netw. Comput. Appl.* 110 (2018), 75–86.

[104] Jun Tang, Yong Cui, Qi Li, Kui Ren, Jiangchuan Liu, and Rajkumar Buyya. 2016. Ensuring security and privacy preservation for cloud data services. *ACM Comput. Surv.* 49, 1 (2016), Article 13, 39 pages.

[105] Keitaro Uehara, Yu Xiang, Yih-Farn Robin Chen, Matti A. Hiltunen, Kaustubh Joshi, and Richard D. Schlichting. 2018. SuperCell: Adaptive software-defined storage for cloud storage workloads. In *Proceedings of IEEE/ACM CCGRID*. 103–112.

[106] Philipp Waibel, Johannes Matt, Christoph Hochreiner, Olena Skarlat, Ronny Hans, and Stefan Schulte. 2017. Cost-optimized redundant data storage in the cloud. *Serv. Oriented Comput. Appl.* 11, 4 (2017), 411–426.

[107] Ao Wang, Jingyuan Zhang, Xiaolong Ma, Ali Anwar, Lukas Rupprecht, Dimitrios Skourtis, Vasily Tarasov, Feng Yan, and Yue Cheng. 2020. InfiniCache: Exploiting ephemeral serverless functions to build a cost-effective memory cache. In *Proceedings of FAST*. 267–281.

[108] Cheng Wang, Bhuvan Urgaonkar, Aayush Gupta, George Kesidis, and Qianlin Liang. 2017. Exploiting spot and burstable instances for improving the cost-efficacy of in-memory caches on the public cloud. In *Proceedings of EuroSys*. 620–634.

[109] Haoyu Wang, Haiying Shen, Zijian Li, and Shuhao Tian. 2021. GeoCol: A geo-distributed cloud storage system with low cost and latency using reinforcement learning. In *Proceedings of IEEE ICDCS*. 149–159.

[110] Haoyu Wang, Haiying Shen, Qi Liu, Kevin Zheng, and Jie Xu. 2020. A reinforcement learning based system for minimizing cloud storage service cost. In *Proceedings of ICPP*. Article 30, 10 pages.

[111] Huaimin Wang, Peichang Shi, and Yiming Zhang. 2017. JointCloud: A cross-cloud cooperation architecture for integrated Internet service customization. In *Proceedings of IEEE ICDCS*. 1846–1855.

[112] Shucheng Wang, Ziyi Lu, Qiang Cao, Hong Jiang, Jie Yao, Yuanyuan Dong, and Puyuan Yang. 2020. BCW: Buffer-controlled writes to HDDs for SSD-HDD hybrid storage server. In *Proceedings of FAST*. 253–266.

[113] Wei Wang, Ben Liang, and Baochun Li. 2015. To reserve or not to reserve: Optimal online multi-instance acquisition in IaaS clouds. *IEEE Trans. Parallel Distrib. Syst.* 26, 12 (2015), 3407–3419.

[114] Chenggang Wu, Vikram Sreekanti, and Joseph M. Hellerstein. 2021. Autoscaling tiered cloud storage in Anna. *VLDB J.* 30, 1 (2021), 25–43.

[115] Huijun Wu, Chen Wang, Yinjin Fu, Sherif Sakr, Kai Lu, and Liming Zhu. 2018. A differentiated caching mechanism to enable primary storage deduplication in clouds. *IEEE Trans. Parallel Distrib. Syst.* 29, 6 (2018), 1202–1216.

[116] Huijun Wu, Chen Wang, Yinjin Fu, Sherif Sakr, Liming Zhu, and Kai Lu. 2017. HPDedup: A hybrid prioritized data deduplication mechanism for primary storage in the cloud. In *Proceedings of MSST*.

[117] Suzhen Wu, Kuan-Ching Li, Bo Mao, and Minghong Liao. 2017. DAC: Improving storage availability with deduplication-assisted cloud-of-clouds. *Future Gener. Comput. Syst.* 74 (2017), 190–198.

[118] Zhe Wu, Curtis Yu, and Harsha V. Madhyastha. 2015. CosTLO: Cost-effective redundancy for lower latency variance on cloud storage services. In *Proceedings of NSDI*. 543–557.

[119] Qiufen Xia, Zichuan Xu, Weifa Liang, Shui Yu, Song Guo, and Albert Y. Zomaya. 2019. Efficient data placement and replication for QoS-aware approximate query evaluation of big data analytics. *IEEE Trans. Parallel Distrib. Syst.* 30, 12 (2019), 2677–2691.

[120] Wen Xia, Hong Jiang, Dan Feng, Fred Douglis, Philip Shilane, Yu Hua, Min Fu, Yucheng Zhang, and Yukun Zhou. 2016. A comprehensive study of the past, present, and future of data deduplication. *Proc. IEEE* 104, 9 (2016), 1681–1710.

[121] Xiaoyu Xia, Feifei Chen, Qiang He, Guangming Cui, Phu Lai, Mohamed Abdelrazek, John Grundy, and Hai Jin. 2020. Graph-based data caching optimization for edge computing. *Future Gener. Comput. Syst.* 113 (2020), 228–239.

[122] Xiaoyu Xia, Feifei Chen, Qiang He, John Grundy, Mohamed Abdelrazek, and Hai Jin. 2021. Online collaborative data caching in edge computing. *IEEE Trans. Parallel Distrib. Syst.* 32, 2 (2021), 281–294.

[123] Xiaoyu Xia, Feifei Chen, Qiang He, John C. Grundy, Mohamed Abdelrazek, and Hai Jin. 2021. Cost-effective app data distribution in edge computing. *IEEE Trans. Parallel Distrib. Syst.* 32, 1 (2021), 31–44.

[124] Jiwei Xu, Wenbo Zhang, Shiyang Ye, Jun Wei, and Tao Huang. 2014. A lightweight virtual machine image deduplication backup approach in cloud environment. In *Proceedings of IEEE COMPSAC*. 503–508.
[125] Zichen Xu, Christopher Stewart, Nan Deng, and Xiaorui Wang. 2016. Blending on-demand and spot instances to lower costs for in-memory storage. In *Proceedings of IEEE INFOCOM*. 1–9.
[126] Chi Yang and Jinjun Chen. 2017. A scalable data chunk similarity based compression approach for efficient big sensing data processing on cloud. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1144–1157.
[127] Jie Yang, Haibin Zhu, and Tieqiao Liu. 2019. Secure and economical multi-cloud storage policy with NSGA-II-C. *Appl. Soft Comput.* 83 (2019), 105649.
[128] Shengsong Yang, Li Pan, Qingyang Wang, Shijun Liu, and Shuo Zhang. 2018. Subscription or pay-as-you-go: Optimally purchasing IaaS instances in public clouds. In *Proceedings of IEEE ICWS*. 219–226.
[129] Zhengyu Yang, Morteza Hoseinzadeh, Allen Andrews, Clay Mayers, David Thomas Evans, Rory Thomas Bolt, Janki Bhimani, Ningfang Mi, and Steven Swanson. 2017. AutoTiering: Automatic data placement manager in multi-tier all-flash datacenter. In *Proceedings of IEEE IPCCC*. 1–8.
[130] Zhengyu Yang, Yufeng Wang, Janki Bhamini, Chiu Chiang Tan, and Ningfang Mi. 2018. EAD: Elasticity aware deduplication manager for datacenters with multi-tier storage systems. *Clust. Comput.* 21, 3 (2018), 1561–1579.
[131] Zhen Ye, Shanping Li, and Xiaozhen Zhou. 2013. GCplace: Geo-cloud based correlation aware data replica placement. In *Proceedings of ACM SAC*. 371–376.
[132] Jianwei Yin, Yan Tang, ShuiGuang Deng, Ying Li, Wei Lo, Kexiong Dong, Albert Y. Zomaya, and Calton Pu. 2017. ASSER: An efficient, reliable, and cost-effective storage scheme for object-based cloud storage systems. *IEEE Trans. Comput.* 66, 8 (2017), 1326–1340.
[133] Jianwei Yin, Yan Tang, Shuiguang Deng, Bangpeng Zheng, and Albert Y. Zomaya. 2021. MUSE: A multi-tiered and SLA-driven deduplication framework for cloud storage systems. *IEEE Trans. Comput.* 70, 5 (2021), 759–774.
[134] Xingjia Yuan and Yuelong Zhao. 2019. RPMSP: A novel replica placement method inspired by self-similarity propagation of plants. In *Proceedings of IEEE ISPA/BDCloud/SocialCom/SustainCom*. 596–601.
[135] Victor Zakhary, Lawrence Lim, Divy Agrawal, and Amr El Abbadi. 2021. Cache on Track (CoT): Decentralized elastic caches for cloud environments. In *Proceedings of EDBT*. 217–228.
[136] Lingfang Zeng, Shijie Xu, Yang Wang, Kenneth B. Kent, David Bremner, and Cheng-Zhong Xu. 2017. Toward cost-effective replica placements in cloud storage systems with QoS-awareness. *Softw. Pract. Exp.* 47, 6 (2017), 813–829.
[137] Lei Zhang, Xuejun Li, Hourieh Khalajzadeh, Yan Yang, Ruiyue Zhu, Xia Ji, Chuanhui Ju, and Yun Yang. 2018. Cost-effective and traffic-optimal data placement strategy for cloud-based online social networks. In *Proceedings of IEEE CSCWD*. 110–115.
[138] Panfeng Zhang, Ping Huang, Xubin He, Hua Wang, and Ke Zhou. 2017. Resemblance and mergence based indexing for high performance data deduplication. *J. Syst. Softw.* 128 (2017), 11–24.
[139] Wei Zhao, Jiajia Liu, Hongzhi Guo, and Takahiro Hara. 2018. ETC-IoT: Edge-node-assisted transmitting for the cloud-centric Internet of Things. *IEEE Netw.* 32, 3 (2018), 101–107.
[140] Jiang Zhou, Yong Chen, Wei Xie, Dong Dai, Shuibing He, and Weiping Wang. 2020. PRS: A pattern-directed replication scheme for heterogeneous object-based storage. *IEEE Trans. Comput.* 69, 4 (2020), 591–605.

Received 22 June 2022; revised 15 January 2023; accepted 30 January 2023