A Data Replication Placement Strategy for the Distributed Storage System in Cloud-Edge-Terminal Orchestrated Computing Environments

Peng Chen, Mengke Zheng, Xin Du, Muhammad Bilal, Zhihui Lu, *Member, IEEE* Qiang Duan, *Senior Member, IEEE,* and Xiaolong Xu, *Senior Member, IEEE*

***Abstract*—Cloud-edge-terminal orchestrated computing, as an expansion of cloud computing, has sunk resources to the edge nodes and terminal equipment, which can provide high-quality services for delay-sensitive applications and reduce the cost of network communication. Due to the high volume of data generated by Internet of Things (IoT) devices and the limited storage capacities of edge nodes, a significant number of terminal devices are now being considered for utilization as storage nodes. However, because of the heterogeneous storage capacity and reliability of these hardware devices and the different data requirements of user services, the performance and storage reliability of applications deployed in cloud-edge-terminal orchestrated computing environments have become urgent problems to be solved. Especially, for a distributed storage system in these environments, it is required to ensure reliable storage of the generated data and its' replications. In this paper, we first implement a distributed storage system and construct a data replication placement model. Then, based on the constructed model, we formulate the data replication placement problem and design a data replication placement strategy called DRPS to solve it. The DRPS covers a ranks-based replication storage node selection algorithm and a greedy load balancing algorithm, which can select appropriate hardware devices for different data requirements of services and is implemented in the data storage system to store replications and balance loads. We design extensive experiments to verify the effectiveness of DRPS. The results indicate that the proposed strategy outperforms other state-of-the-art algorithms in terms of system delay reduction by 39.9%, an increase of 43.3% in the replication numbers, a 27.5% improvement in memory utilization, and a reduction of unreliability rate by 82.0%.**

***Index Terms*—Cloud-Edge-Terminal Orchestrated Computing; Distributed Storage System; Data Replication Placement; Load Balance**

The work of this paper is supported by the National Natural Science Foundation of China under Grant (No.61873309, No.92046024, No.92146002), Shanghai Science and Technology Project under Grant (No.22510761000), and Intel Sponsored Research Agreement under Grant (Intel CG # 89533661). (Corresponding author: Xin Du, Xiaolong Xu.)

P. Chen and X. Long are with the School of Software and Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China. (e-mail: chenpenghehedawang@gmail.com, xlxu@ieee.org).

M. Zheng and Z. Lu are with the School of Computer Science, Fudan University, Shanghai 200433, China. (e-mail: mkzheng23@m.fudan.edu.cn, lzh@fudan.edu.cn).

Z. Lu is also with Shanghai Blockchain Engineering Research Center, Shanghai 200433, China.

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, Lancaster LA1 4WA, UK. (e-mail: m.bilal@ieee.org)

Qiang Duan is with the School of Information Sciences and Technology, Pennsylvania State University, State College, PA 16802 USA. (e-mail: qduan@psu.edu).

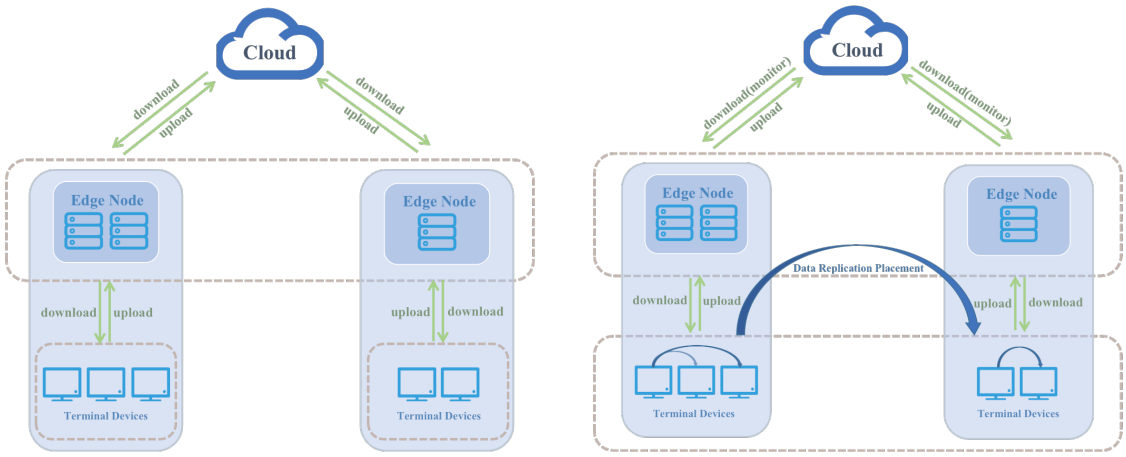
Xin Du is with the School of Software Technology, Zhejiang University, Zhejiang 310027, China, and also with the State Key Lab of Brain-Machine Intelligence, Zhejiang University, Zhejiang 310027, China. (e-mail: jsjduxin@gmail.com).

I. Introduction

WITHE the rapid development of the Internet of Things (IoT) technology, some smart terminal devices with certain computing and storage capabilities have emerged and are widely used in industrial environments [1]. The combination of these terminal devices, such as smart cars, cameras, and sensors, with their associated edge nodes and cloud servers, forms what is known as cloud-edge-terminal orchestrated computing environments [2]. In the environments, to reduce data transmission and latency while enhancing system response time and reliability[1], these hardware devices, which include smart terminals and edge nodes, are typically located at the network edge, allowing for local processing and storage of data. However, due to the limitations of resources and capabilities, data collected and transmitted by terminal devices may experience loss, errors, or corruption, which lowers the reliability and integrity of the data. Therefore, in real environments, a distributed storage system usually be maintained to replicate the data collected and processed by the terminals to improve the accessibility and availability of the data [3], [4]. Additionally, the system can also provide a reliable data foundation for subsequent data analysis and mining. To ensure the performance and reliability of data stored in this system, it is necessary to design a data replication placement strategy for it. Especially when terminal devices as storage nodes are involved in the distributed storage system, although this can significantly enhance resource utilization, managing such an extensive number of IoT devices and data, while ensuring the reliability and performance of applications in this environment, becomes a substantial challenge. To the best of our knowledge, while keeping the storage effectiveness of the system, there is no data replication placement strategy that can fully guarantee the reliability of data storage in cloud-edge-terminal orchestrated computing environments.

As shown in Figure. 1(a), for traditional data replication placement strategy for storage systems in cloud-edge-terminal orchestrated computing environments, when data is generated in terminal devices, the system usually chooses to upload data to a cloud server or edge nodes after data collection for data

[1]In this paper, the term "reliability" is used exclusively to refer to storage reliability.



(a) traditional data replication placement strategy (b) proposed data replication placement strategy

Fig. 1. Illustration of the different data replication placement strategy for distributed storage system in cloud-edge-terminal orchestrated computing environments

persistence and conducts data analysis and access in the cloud or edge [3]–[5]. However, with the continuous improvement of smart terminal device performance, an increasing number of these devices have acquired data storage capabilities. Failing to fully utilize this storage capacity would be a waste of hardware resources [3], [4]. Furthermore, the exponential growth of data generated by smart devices has led to edge nodes, originally intended for data storage, being overwhelmed by massive amounts of data. Hence, some researchers have adopted distributed storage systems by using the cooperation of terminal devices in some industrial environments.

In Figure. 1(b), for the proposed data replication placement strategy in the distributed storage system, when data is generated in terminal devices, the system can choose to store and replicate the data in the terminal devices or upload them to a cloud server or edge nodes. Compared to cloud servers or edge nodes, the number of terminal devices is large and widely distributed, and there are significant differences in resource performance and reliability between different devices [6]–[8]. When designing a data replication placement strategy for the distributed storage system, some challenges are summarized as follows: First, the performance capabilities of terminal devices vary significantly, influencing the load they are capable of managing. Second, the reliability of these terminal devices differs, thereby affecting the level of data security they can ensure. Third, there is a diversity of storage resources that are required for the data to be stored and replicated. Fourth, the required reliability levels for data storage and replication also vary. In addition, load balancing also needs to be considered, and the data replication placement strategy should be used to avoid causing excessive load [9].

In this study, we first attempt to construct the complex data replication placement model and combine a ranks-based replication storage node selection algorithm with a greedy load balance algorithm in the proposed DRPS to solve the data replication placement problem. Compared to other state-of-the-art methods in the distributed storage system, the proposed DPPS not only can satisfy the data storage reliability but also make the data storage more effective. In summary, the main contributions of this paper are as follows:

- We construct a comprehensive data replication placement model for cloud-edge-terminal systems, uniquely considering terminal devices as storage nodes with heterogeneous reliability and capacity. For a distributed storage system in cloud-edge-terminal orchestrated computing environments, not only the storage capacity and reliability of the terminal devices need to be considered, but also the potential impact of the dynamic changes in the terminal devices and networks available for distributed storage on the system's load balancing needs to be taken into account.
- Based on the constructed model, we formulate the data replication placement problem as a multi-objective optimization problem and design a data replication placement strategy named DRPS to achieve it. Especially, DRPS employ a ranks-based replication storage node selection algorithm and a greedy load balancing algorithm, which deploys multiple data replicas on appropriate hardware devices and distributes them to different devices based on the load conditions of these devices to achieve load balancing of the whole system. With the premise of ensuring data reliability, the resources of each terminal are balanced as much as possible to achieve load balancing among devices, improve resource utilization, and reduce system delay.
- We simulate a cloud-edge-terminal orchestrated computing environment and implement a distributed storage system for it. The system employs the proposed DRPS to ensure the reliable storage of data and improve the storage performance of the hardware devices. By designing comprehensive experiments and utilizing datasets obtained from a real-world IoT scenario, we validate that the proposed DRPS achieves superior performance in terms of system delay, data storage reliability, and hardware utilization compared to existing approaches. The implementation codes will be released upon publication of this work.

The remainder of the paper is organized as follows. We first review related work in Section 2. Then, we introduce the implementation of a distributed storage system and the construction of a data replication placement model in Section 3. Based on the model, in Section 4, we design the DRPS which combines a ranks-based replication storage node selection algorithm and a greedy load balancing algorithm. The experimental results for performance evaluation are reported in Section 5. Section 6 concludes the paper with a brief discussion of future work.

II. Related Work

There has been some prior work on related topics about cloud-edge-terminal orchestrated computing environments, which include distributed storage systems and data replication placement.

Distributed storage system. With the emergence of edge computing, some researchers began to evaluate the performance of traditional cloud-distributed storage systems (including Rados [10], Cassandra [11], and IPFS [12]) in edge computing environments. A lot of storage systems were developed and deployed on edge nodes based on their characteristics, and the read-write performance and network communication latency between different edge nodes were tested. However, these studies were limited to deploying their system on the edge node. They did not consider the differences between edge computing and traditional cloud computing or the interaction between the cloud and edge nodes. Recently, some authors made modifications to IPFS [12] to reduce the amount of data transferred between different edge nodes to adapt to network conditions between these nodes, but the study was still limited to the consideration of edge nodes. Gupta et al. designed the distributed storage system named FogStore [13], which is an edge key-value storage system (Key-Value Store) developed based on Cassandra [11]. In FogStore, the geographical relationship between different edge nodes and the reliability requirements of different data were taken into account to design corresponding data replication placement strategies. Furthermore, Monga et al. [14] designed the Elfstore, a distributed storage system that combines peer-to-peer (p2p) and Hadoop Distributed File System (HDFS) architectures to effectively address the replica placement challenge. Elfstore utilizes highly reliable fog nodes to manage and monitor edge resources. Similarly, by using cloud services benchmark to assess their proposed solution, Mayer et al. [15] also proposed an architecture for managing data replication within a fog infrastructure, leveraging existing distributed data storage systems as a foundation. However, the focus of these studies was on the properties of terminal devices, and they ignored the heterogeneity of different hardware devices. Moreover, Huang et al. [16] proposed an edge collaborative Internet of Things (IoT) management system that addresses the challenges of high data storage network costs and weak range search capabilities in cloud-edge hybrid environments. Liu et al. [17] proposed offline community discovery and online community adjustment schemes to adaptively solve the replication placement problem in geo-distributed cloud storage systems. For a distributed storage system, a replication placement strategy named TS-REPLICA [18] was proposed based on the entropy weight TOPSIS (a technique for order preference by similarity to the ideal solution) method, which first reflects the performance and defines the load of nodes and then calculated the average comprehensive load score of the entire hardware devices.

Data replication placement. To meet the reliability and security requirements of data storage, the design and implementation of a data replication placement strategy are indispensable in distributed storage systems. Confais et al. [19] proposed a data replication management system that utilizes the Domain Name Service (DNS) protocol and Content Delivery Network (CDN) protocol. This management system creates a network topology-based tree structure to store data locations in fog architecture. Karatas and Korpeoglu [20] presented a heuristic approach for classifying Internet of Things (IoT) data and determining the type of data required by IoT applications (latency-sensitive or computation-sensitive). They also proposed a hierarchical data placement architecture in cloud-edge hybrid environments. In their approach, data replication is selected and utilized based on the highest centrality score, and replication is performed on independent edge nodes in two partitions. Huang et al. [16] proposed a protocol-based strategy to solve the data replication placement problem and minimize overall latency. It provides reasonable solutions in polynomial time by using different heuristic rules to prune infeasible solutions and reduce the search space. Li et al. [21] established a dynamic strategy for data placement, considering the serviceability of each node, estimating the likelihood of data generation, and dynamically selecting edge nodes for placement. Saranya et al. [22] proposed a method for random data replication in cloud-edge-terminal orchestrated computing environments. They conducted tests and validations across various hardware device configurations to verify the effectiveness of data replication placement strategies. As a result, the network latency and bandwidth utilization of distributed storage systems were reduced. Jaber et al. [23] provided a meta-heuristic-based method using the non-dominated sorting genetic algorithm aimed at minimizing network latency and bandwidth. The effectiveness of their technique was compared with other alternatives, demonstrating its efficiency in addressing the proposed solution.

Furthermore, Shakarami et al. [27] conducted a comprehensive survey and categorization of the most advanced data replication schemes in various existing cloud computing solutions to define current schemes on the topic and proposed open questions. The proposed classification includes three main categories: data deduplication schemes, data auditing schemes, and data processing schemes. Chrysostomos et al. [24] proposed a data placement strategy based on user mobility for mobile application scenarios under a cloud-edge collaborative architecture, considering the trade-off between latency and data migration. They used a causality-aware method to classify users into three mobility categories: static, local, or mobile and then utilized this information to optimize the proposed data placement strategy. Masoumeh et al. [28] addressed the issue of ”over-provisioning” when resources in fog computing

Ref	Evaluation Tool	Technique	Performance Metrics	Dataset
Huang [16]	iFogSim	Greedy algorithm	Overall and execution time	Simulation
Li [21]	Apache JMeter	Fast NSGA-II algorithm	SRT,SSU,ART,RRT,TDW	Simulation
Saranya [22]	Simulation	Random algorithm	Bandwidth savings	Simulation
Jaber [23]	iFogSim	ARIMA method	Data latency/availability/access cost	Simulation
Chrysostomos [24]	Simulation	DL network	QoS, transfer latency and migration costs	Simulation
Konstantinos [25]	Python Simulation	Heuristic approach	Data cost, latency and availability,	Real-world data
Dias [26]	Thyme GardenBed	Mecerra algorithm	Data availability and overhead	Simulation
Ours	Python Simulation	DRPS algorithm	System delay/memory utilization/replication/reliability	Simulation

TABLE I
A CONCLUSION OF SOME RELATED WORKS IN TERMS OF EVALUATION TOOLS, UTILIZED TECHNIQUES, PERFORMANCE METRICS, AND DATASETS.

architectures exceed needs and "under-provisioning" when fog resources are insufficient, by proposing an effective deep learning-based resource auto-scaling mechanism to manage the amount of resources needed to handle dynamic workloads in fog environments. Esmaeil et al. [29] systematically categorized the work on data replica placement in hybrid cloud and edge scenarios, dividing the main methods applied by researchers into four types: framework-based, graph-based, heuristic-based, and meta-heuristic-based algorithms, and discussed their advantages and disadvantages. Konstantinos et al. [25] focused on optimizing data integrity, lifespan, security, and cost while utilizing erasure coding to perform resource allocation, proposing a comprehensive mixed integer linear programming strategy for storage resource orchestration to effectively balance performance and execution time. Mohammad et al. [30] concluded a review paper to provide a taxonomy of social-aware edge caching approaches consisting of game theory-based, machine learning-based, model-based, and heuristic-based approaches. Dias et al. [26] proposed a data replica ranking algorithm tailored to the characteristics of the mobile edge computing environment and formulated a data replica placement strategy based on this algorithm, improving the efficiency of finding replicas and reducing storage costs. Sarwar et al. [31] focused on the importance of data replica privacy for data protection, reliability, and authentication. Based on the privacy level defined by data owners and the service capability of edge nodes, they proposed a data replica placement algorithm that not only protected the privacy of data replicas but also effectively reduced the storage cost of data replicas. Afonso et al. [32] paid attention to the trade-off between consistency and access efficiency when storing data replicas at edge nodes, proposing a data replica storage service based on a causality tracking mechanism, which improved the system's access speed, throughput, and scalability.

In addition, Shao et al. [33] devised an aware collaborative system for data replication placement in cloud-edge hybrid environments to minimize data access costs and ensure data dependability. This study also suggested a deadline-driven scheduling strategy to optimize the burden on the edge and fog infrastructure. Their method estimates the number of data replicas dynamically and selects the optimal storage device based on data block prevalence. Furthermore, Fahs and Pierre [34] proposed Proximity, a straightforward approach that allows system administrators to manage the trade-off between reducing user-to-replica latency and evenly balancing the load among replicas. The findings indicate that their technique effectively reduces the average user-to-replica latency while enabling system administrators to control the level of load balancing. Inspired by this study, load balancing was also considered an important metric of the data replication placement strategy designed, which can maximize the utilization of hardware resources used for storage in the distributed system. To ensure a fair load distribution over a fixed-size replication placement, Aral et al. [35] maintain the tail latency within pre-defined bounds and keep their system's load balancing. Moreover, Klervie et al. [36] introduced the concept of a spare edge device to handle sudden load variations in time and space without having to continuously over-provision. Zhang et al. [37] proposed a mobility-aware service provisioning approach based on multiple digital twin replica placements to enhance service continuity in edge computing environments. Zheng et al. [38] proposed Lion, a transaction processing protocol that minimizes distributed transactions by adaptively provisioning replicas using workload prediction and replica remastering, outperforming existing methods in throughput and scalability. Compared with Lion [38], which focuses on reducing distributed transactions via master replica adjustments, our work targets reliability-aware replica placement in heterogeneous cloud-edge-terminal environments. Furthermore, unlike the DT-oriented placement approach by Zhang et al. [37], which emphasizes delay-sensitive mobility support, our DRPS strategy incorporates reliability, delay, and load balancing into a unified model, offering more practical benefits in distributed storage systems. To provide a clearer depiction of the current state of research, Table I presents a summary encompassing evaluation tools, employed techniques, performance metrics, and datasets. Overall, the majority of current research endeavors are still reliant on simulated evaluation tools and datasets, with a lack of uniformity observed in the techniques employed and evaluation metrics utilized.

In summary, the data replication placement strategies in distributed storage systems have evolved from solely focusing on reducing system delay to gradually considering load balancing among devices. Initially, random multi-replica strategies were commonly used in distributed storage systems, which often neglected the need for load balancing. However, with the emergence of edge computing architectures, researchers began employing load-aware and context-based load-balancing algorithms to achieve device equilibrium within the system. Moreover, researchers discovered a trade-off between load balancing among devices and the system delay required for replica backups, leading to the utilization of evolutionary algorithms to search for optimal solutions. Before the inclusion of terminal devices in the storage system, researchers

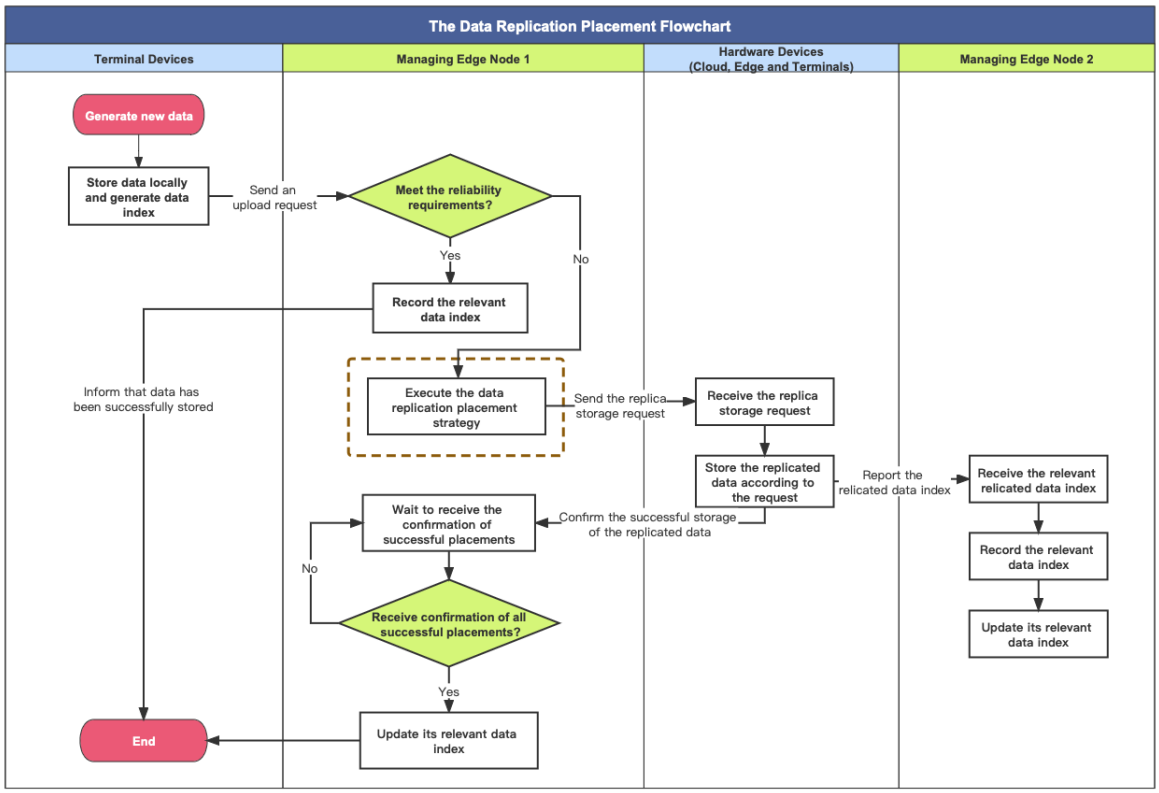


Fig. 2. The flowchart of data replication placement for implemented distributed storage system in cloud-edge-terminal orchestrated computing environments.

primarily emphasized system delay and load balancing during replication. However, because the terminal devices become the storage nodes in cloud-edge-terminal orchestrated computing environments, the reliability of data must be further ensured. Our approach incorporates device reliability as a constraint in the data placement strategy, considering both system delay and load balancing while ensuring data reliability.

III. System and Model

In this section, we first describe the implementation of a distributed storage system in cloud-edge-terminal orchestrated computing environments. Then, we provide a specific example to analyze the motivation of this study. Subsequently, we give fundamental definitions of all components based on the system and construct a data replication placement model. Lastly, according to the constructed model, we formulate the problem as a multi-objective optimization problem.

A. System Overview

In the cloud-edge-terminal orchestrated computing environments, there are a cloud datacenter (DC_c), multiple edge nodes (DC_e), and a mass of terminals ($td_{i,j}$). Edge nodes can be divided into managing edge nodes (ed_i^1) and regular edge nodes (ed_i^0). Managing edge nodes provides gateway services for terminal devices in the distributed storage system and registers and manages terminal devices so the terminal devices can be grouped according to the managing edge nodes to which they belong. Regular edge nodes only are used to store data in the system. Considering the characteristics of terminal devices, which primarily collect various types of unstructured data such as videos, images, and documents, our system employs distributed storage based on a block-based data model. The storage unit is a data block marked by a block ID (bid), and each block consists of two parts: data payload and metadata. Additionally, data blocks have a minimum reliability requirement, indicating the minimum storage reliability that the storage service needs to guarantee. The metadata contains essential information such as block ID, timestamp, and checksum. Furthermore, to support diverse IoT applications and terminal devices, the metadata part of the data block also includes custom attributes. For example, the device that collected the data block, the data type of the block, and the geographical location information of the data collection. The metadata is used for data indexing and location, so the responsible edge node needs to maintain the metadata of data blocks in its managed terminal devices to support data positioning within the system. Moreover, to facilitate cross-regional data positioning, the responsible edge node also needs to maintain a global data index service and write data indexes to the appropriate locations. In the system, we implemented the data manipulation interface as follows:

- PutBlock(bid, reliability, metadata[], data): The operation creates a new data block with a designated identifier, referred to as ”bid.” The parameter ”reliability” indicates the minimum reliability requirement for the data block.

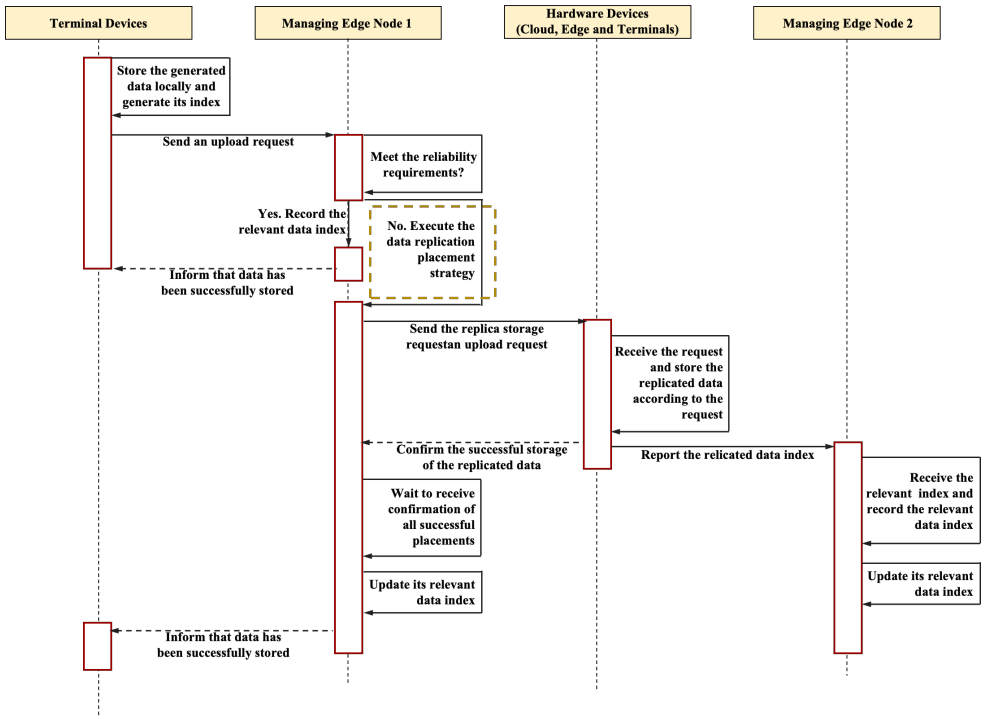


Fig. 3. The sequence diagram of data replication placement for implemented distributed storage system in cloud-edge-terminal orchestrated computing environments.

The system achieves the desired reliability by replicating the data block multiple times and placing its replicas on different devices.

- UpdateBlock(bid, data, vData): The ”UpdateBlock” operation is used to update all replicas of the data block identified as ”bid.” It involves modifying the content of the data block across its replicas. The operation utilizes an optimistic locking mechanism, where the parameter ”vData” represents the version number of the old data, enabling concurrency control. By comparing the version number of the old data with the current version, conflicts can be detected, and appropriate synchronization or conflict resolution mechanisms can be applied to ensure data consistency.
- UpdateMeta(bid, metadata[], vMeta): The ”UpdateMeta” operation is used to update the metadata of the data block identified as ”bid.” It involves modifying the metadata associated with the data block. Similar to the ”UpdateBlock” operation, the ”UpdateMeta” operation also utilizes an optimistic locking mechanism and concurrency control. The parameter ”vMeta” represents the version number of the old metadata, enabling conflict detection and resolution.
- GetBlock(bid): The operation is used to download the data block identified as ”bid”.
- GetMeta(bid): The operation is used to request the metadata of the data block identified as ”bid.”

The data replication placement flowchart and sequence diagram of our proposed distributed storage system under cloud-edge-terminal orchestrated computing environments are shown in Figure 2 and Figure 3. Upon collecting and generating new data in an intelligent terminal device, it is initially stored locally. Subsequently, the terminal device sends the storage information to the managing edge node to which it belongs, and the edge nodes responsible for managing it determine if the data meets the reliability requirements. When an edge node responsible for managing a specific region receives an upload request from a smart terminal device if the data meets the reliability requirements, the edge node records the relevant data index and informs the system that the data has been successfully stored. However, for the data storage that does not meet the reliability requirements, the edge node responsible for regional management executes the proposed data replication placement strategy (which is described in Section 4) to determine an appropriate solution. In real-world system implementations, although terminal devices offer substantial potential for data storage due to their vast numbers, the heterogeneity of these devices also leads to more severe reliability issues during the data storage process. The purpose of data replication, therefore, is to ensure the integrity and availability of the data. Then, the edge node responsible for management communicates the selected replication strategy to the terminal device that needs to store the replicas. The terminal devices that need to place their data replication send the requests to the designated hardware devices based on the received solution. The respective hardware devices store the replicated data according to the request, report the stored data index to the corresponding managing edge node, and confirm the successful storage of the replicated data to the system. Upon receiving confirmation of all successful placements, the managing edge node updates its data index and informs the system that the entire data replication process has been

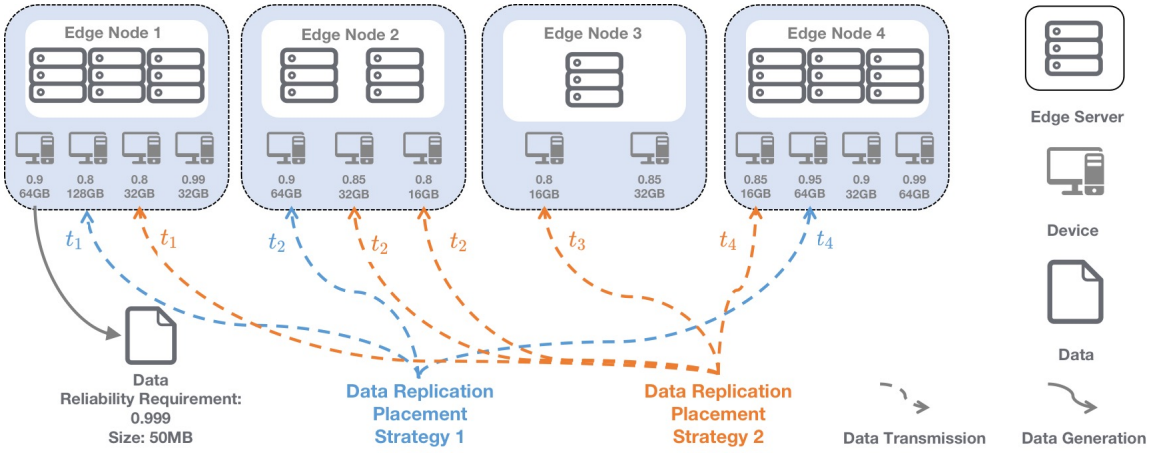


Fig. 4. Sample of data replication placement by two strategies.

TABLE II
THE RELIABILITY AND STORAGE CAPACITY OF TERMINAL DEVICES IN THE SYSTEM

Devices	td_{11}	td_{12}	td_{13}	td_{14}	td_{21}	td_{22}	td_{23}	td_{31}	td_{32}	td_{41}	td_{42}	td_{43}	td_{44}
Reliability	0.8	0.9	0.8	0.99	0.9	0.85	0.8	0.8	0.85	0.85	0.95	0.9	0.99
Storage	64	128	32	32	64	32	16	16	32	16	64	32	64

completed.

B. Motivation

To analyze the data replication placement problem in the distributed storage system, Figure 4 shows two data replication placement strategies for a real-world case study example in cloud-edge-terminal orchestrated computing environments. There are 4 managing edge nodes and 13 terminals, which have their attributes of reliability and storage capacity. Based on the constructed model, we denote the four managing edge nodes as $\{ed_1^1, ed_2^1, ed_3^1, ed_4^1\}$, $ed_1^1 = \{td_{11}, td_{12}, td_{13}, td_{14}\}$, $ed_2^1 = \{td_{21}, td_{22}, td_{23}\}$, $ed_3^1 = \{td_{31}, td_{32}\}$, and $ed_4^1 = \{td_{41}, td_{42}, td_{43}, td_{44}\}$. In this example, to explain the effectiveness of the data replication placement strategy more simply, the replicas are only considered in the terminal devices, and the system time is only calculated by the data transfer time. Table II lists the respective attributes (i.e., the reliability of the devices, storage capacity of the devices) of these terminal devices in the scenario. When a terminal device td_{11} generates a data block, for which the reliability requirement is 0.999, and the size of it is 50 MB. The data can be denoted as $bid_{case} =< 50MB, 0.999 >$. If the data needs to be replicated in different terminals, the system will select some devices to minimize the system time and satisfy the reliability requirement. It is well known that the data transfer time will be affected by the data placement location, the data placed in the same region with the generation device is lower than others. Furthermore, the different distances can lead to different data transfer times. We assume the bandwidth between different terminal devices covered in the same managing edge node is 200 M/s, the bandwidth between terminal devices and their' managing edge node is 100 M/s, the bandwidth between different managing edge nodes is 50 M/s, and the bandwidth between a managing edge node with the cloud datacenters is 20M/s. In this case, the data replication placement strategy 1 adopts our proposed method. While meeting the reliability requirements, data replication placement strategy 1 requires replicating the data 3 times, whereas strategy 2 requires 5 replications to meet the same requirements. The data transfer time of strategy 1 is 3.25 s, and the result of strategy 2 is 4.75 s. Expectedly, different data replication placement strategies will significantly affect the system time and efficiency of the storage process. In these terms, the data replication placement strategy 1 is superior to strategy 2.

In summary, the presented case clearly demonstrates that data replication placement strategies significantly affect system performance, resource utilization, and data reliability in cloud-edge-terminal orchestrated computing environments. Traditional random or static placement methods fail to adequately address the heterogeneity of terminal devices, load balancing, and varying reliability requirements, often resulting in resource waste, increased system delay, and excessive replication. Therefore, there is an urgent need for an intelligent replication placement strategy that incorporates device performance levels, data replication, memory utilization, and reliability to optimize overall system efficiency, which forms the core motivation of this work.

C. Model Construction

In our model, the hardware devices include a cloud data center, edge nodes, and terminal devices. The DC_c is denoted as the cloud datacenter and DC_e is represented as the set of n edge nodes contained in the distributed system:

$$DC_e = \{ed_1, ed_2, ed_3, ..., ed_n\} \tag{1}$$

For an edge node ed_i, we can denote it as $ed_i =< s_i, c_i, r_i, type_i >$, which has a pre-defined reliability attribute r_i, a current available storage capacity attribute s_i, a current

available calculation capacity attribute c_i. In addition, the attribute $type_i$ is used to flag whether the edge node is a managing edge node or a regular edge node. When $type_i$=1, the edge node is denoted as a managing edge node; when $type_i$=0, the edge node is a regular data storage node. For simplicity, we describe the managing edge node as ed_i^1 in this paper. Besides, in the system, cloud datacenter DC_c at the long-distance end, which generally has unlimited storage resources and intolerable latency for delay-sensitive applications. Hence, it only stores a huge amount of data that none of the edge nodes can handle. The reliability attribute of a device can be obtained through performance testing, manufacturer data, and other channels. The reliability and storage capacity of different hardware devices are heterogeneous, which is one of the characteristics of cloud-edge-terminal orchestrated computing environments. To demonstrate the reliability of a hardware device more conveniently, the pre-defined reliability attribute is set as $[0, 1]$, which is a larger value indicating that the reliability of the device is higher. Each managing edge node corresponds to a set of k terminal devices that it manages:

$$ed_i^1 = \{td_{i1}, td_{i2}, td_{i3}, ..., td_{ik}\} \tag{2}$$

where td_{ij} represents the j-th terminal device managed by edge node ed_i^1. Similarly, for each terminal device td_{ij}, it also can be described as $td_{ij} =< s_{ij}, c_{ij}, r_{ij} >$. To calculate system delay in the distributed storage system, we also represent the bandwidth across different hardware devices as b_{ij}, which is the value of the bandwidth, and the device i is not device j.

In the system, the data that is generated through the terminal devices and needs to be stored and replicated is called a data block. For a data block bid_i, it also includes three corresponding attributes: the required data reliability r_{bid_i}, the data block size s_{bid_i} and computing capacity to process the data block storage task c_{bid_i}. To achieve the minimum data reliability requirement for a data block, it is necessary to select several terminal devices as replication placement nodes from the terminal device layer and the edge node to which they belong. Let E_m denote a set of m_1 terminal devices and m_2 edge nodes that serve as data replication placement nodes for the distributed storage system. These m_1 terminal devices may belong to different managing edge nodes for management. A data block needs to be replicated to different hardware devices, and hardware devices also store a lot of data blocks, it is a many-to-many map. Hence, we denoted the bid_i, E_m as follows:

$$bid_i =< s_{bid_i}, c_{bid_i}, r_{bid_i} > \tag{3}$$

$$E_m = \{td_1, td_2, ..., td_{m_1}\} \cup \{ed_1, ed_2, ..., ed_{m_2}\} \cup DC_c \tag{4}$$

While guaranteeing the system delay is as small as possible, the final selected data replication placement is related to the data reliability requirements, the reliability, the storage capacity, and the calculation capacity of hardware devices in the system. This means that the actual number of replicas and storage space occupied by each data block may vary. The hardware devices used to place replicas may need to place several replicas, which should ensure the storage reliability of all replicas. In other words, a device must satisfy all data blocks whose replicas need to be placed in it. Based on this, when hardware devices are selected to place multiple data replicas in the system, the selected m devices must meet the reliability, storage, and calculation capacity requirements, that is:

$$s_i \geq s_{bib}, \quad \forall i \in [1, m] \cap \forall\ bib \tag{5}$$

$$c_i \geq c_{bib}, \quad \forall i \in [1, m] \cap \forall\ bib \tag{6}$$

$$1 - r_{bid_i} \geq \prod_{i=1}^{m}(1 - r_i), \quad \forall (td_i \cup ed_i \cup DC_c) \in E_m \tag{7}$$

where $\forall\ bib$ represents a set of data blocks stored in a hardware device i. For a given data block, excessive data replication would result in the wastage of system storage resources, while insufficient replication would fail to meet the reliability requirements of this data. Hence, we formulate the data replication placement problem as a multi-objective optimization problem and propose DRPS to achieve it.

D. Problem Formulation

To fully utilize the storage capacity in a cloud-edge-terminal system for data replication, any data block bib_i generated on the terminal device td_i may be transmitted to various types of replication locations, including terminals in the same cluster (local terminals), terminal in other clusters (remote terminals), and regular edge nodes, which lead to different transmission patterns and delay. Let E_i be the set of all selected replication locations for bib_i, then $E_i = E_i^a \cup E_i^b \cup E_i^c$, where E_i^a, E_i^b, and E_i^c are the sets of selected local terminals, edge nodes, and remote terminals respectively. We first analyze the transmission delay for data replication in the following three cases and then derive the delay for the general case as a combination of these cases.

Case 1: All replication locations are local terminals, i.e., $E_i = E_i^a$. In this case, a copy of bib_i is first transmitted from its source td_i to the managing edge node and then forwarded by the edge node to all the selected terminal devices for replication. The transmission link from the source device td_i to the edge node (referred to as the uplink) is a point-to-point connection. If more than one device is selected (i.e., $|E_i^a| \geq 2$), data transfer from the edge node to all the destination devices $td_j \in E_i^a$ will be multicast through multiple downlinks one for each $td_j \in E_i^a$; therefore, the downlink delay will be determined by the delay of the slowest downlink. So, the total latency of data transmission for replication is

$$T_{trans} = \frac{s(bib_i)}{v_{i,e}} + \max_{j \in E_i^a} \frac{s(bib_i)}{v_{e,j}}, \tag{8}$$

where $s(bib_i)$ is the size of data black bib_i, $v_{i,e}$ is the transmission capacity of the uplink, and $v_{e,j}$ is the transmission capacity of the downlink from the edge node to terminal td_j. According to [39]–[41], the transmission capacity of a wireless communication channel between devices i and j can be calculated as

$$v_{ij} = b_{ij} * log_2(1 + \frac{p_{ij} g_{ij}}{\mu^2 + I_{ij}}), \tag{9}$$

where $b_{i,j}$ is the available spectrum bandwidth for the channel, p_{ij} is the transmission power allocated to the devices, g_{ij} is the channel power gain, μ^2 is the ambient noise power, and I_i is the power of inter-cell interference experienced by device i.

Case 2: All replication locations are edge nodes, i.e., $E_i = E_i^b$. In this case, the data block bib_i will be transmitted from td_i to its managing edge node and then forwarded by the managing edge node to other edge nodes selected for replication through the edge network. Similarly, when $|E_i^b| \geq 2$ the data forwarding will be multicast using multiple paths in the edge network, one to each destination; therefore, transfer delay is determined by the path with the longest latency. Then the total transmission delay for data replication can be determined as

$$T_{trans} = \frac{s(bib_i)}{v_{i,e}} + \max_{j \in E_i^b} \frac{s(bib_i)}{v_{e,j}}. \tag{10}$$

where $v_{e,j}$ is the bandwidth of the network route from the managing edge node to the replication location $ed_j \in E_i^b$.

Case 3: All replication locations are remote terminals, i.e., $E_i = E_i^c$. In this case, the data block bib_i will be transmitted via an uplink to its managing edge node, forwarded to the managing edge nodes of the selected remote terminals, and then delivered to the destined remote terminals through downlinks. Therefore, the end-to-end transmission delay for data replication, in this case, will be

$$T_{trans} = \frac{s(bib_i)}{v_{i,e}} + \frac{s(bib_i)}{v_{e,e}} + \max_{j \in E_i^c} \frac{s(bib_i)}{v_{e,j}}. \tag{11}$$

where $v_{e,e}$ is the bandwidth of the network route from the managing edge node of the source terminal td_i to the managing edge node of the destination terminal td_j.

In order to analyze the transmission delay for the general case where all three types of locations (local terminals, edge nodes, and remote terminals) are selected for replicating the data block bib_i, we define the following indicator variable L_e

$$L_e = \begin{cases} 0 & \text{if } E_i = E_i^a \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

Similarly, we define another indicator variable L_d

$$L_d = \begin{cases} 0 & \text{if } E_i = E_i^b \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

Then, the transmission delay for data replication in general can be presented as

$$T_{trans} = \frac{s(bib_i)}{v_{i,e}} + L_e \max_{j \in E_i^b} \frac{s(bib_i)}{v_{e,e}} + L_d \max_{j \in E_i^a \cup E_i^c} \frac{s(bib_i)}{v_{e,j}}. \tag{14}$$

In addition, inspired by [41], [42], we denote the computing resource capacity (CPU frequency) that is used to replicate the data as f_{bib_i}. The data replication execution time T_{exe} can calculate as follows:

$$T_{exe} = \sum_{i=1}^{|DC_e|} \sum_{j \neq i}^{|DC_e|} \frac{s_{bib_i}}{\theta_i * f_{bib_i}} \tag{15}$$

The proportionality factor of computing is denoted as θ_i, which can represent the resource capacity allocated by the device i to deal with the data replication task.

The total system time can be calculated as:

$$T_{sys} = T_{trans} + T_{exe} \tag{16}$$

Hardware resource utilization refers to the ratio between the hardware resources currently used by each device in the system and the maximum storage resources it can accommodate. It reflects the utilization of hardware resources in the distributed storage system and serves as an indicator of whether the system is load-balanced. The load balance of a distributed storage system depends on the hardware device with the worst storage capacity, calculation capacity, or network state. When available resources of one of the hardware devices in the system are exhausted, the efficiency of data replication tends to decrease quickly [43], [44]. The load of a device can be calculated as follows:

$$\begin{aligned} dev.load_i = & a(\frac{dev.c_i}{dev.c_{i,max}}) + b(\frac{dev.s_i}{dev.s_{i,max}}) \\ & + c(\frac{dev.b_i}{dev.b_{i,max}}) \end{aligned} \tag{17}$$

$$max(dev.load_i) \quad i \in \{1, 2, .., m\} \tag{18}$$

where $dev.c_i$ and $dev.c_{i,max}$ denote the available calculation resource and the maximum calculation resource of the hardware device i, respectively. Similarly, the $dev.s_i$ and $dev.s_{i,max}$ are the available and maximum storage resources of the hardware device. And, the $dev.b_i$ and $dev.b_{i,max}$ are the available and maximum bandwidth resources of the hardware device. For the load of each hardware device, a, b, and c are the weights of the calculation resource usage, storage resource usage, and bandwidth resource usage, respectively. To fairly measure resources across different dimensions, parameters a, b, and c must be normalized so that $a + b + c = 1$. This prevents any single resource from being magnified or ignored.

In the cloud-edge-terminal orchestrated computing environment, intelligent service requests from terminal devices are often highly sensitive to latency. Hence, data replication for distributed storage systems is also required to minimize system time. To ensure satisfactory performance, our study aims to minimize the system time of data replication requests through effective data placement and allocation of hardware resources. In addition, considering the data placement strategies with different business data reliability requirements and different terminal device storage reliability, the number of required data replicas is definitely less than the traditional fixed multi-replica redundancy scheme, which can more efficiently utilize the storage resources of the different hardware devices. To determine the corresponding E_m for any data blocks, this paper formulated the selection of hardware devices used to store the

data replicas as a multi-objective optimization problem. The problem can be formally expressed as:

$$\begin{cases} \min\{T_{sys},\ max(dev.load_i)\} \\ \textbf{s.t.} \quad s_i \geq s_{bib}, \quad \forall i \in [1,m] \cap \forall\ bib \\ \qquad c_i \geq c_{bib}, \quad \forall i \in [1,m] \cap \forall\ bib \\ \quad 1 - r_{bid_i} \geq \prod_{i=1}^{m}(1-r_i),\ \forall (td_i \cup ed_i \cup DC_c) \in E_m \end{cases} \tag{19}$$

IV. DRPS

A data replication placement model has been constructed based on real-life parameters, which involve multiple different hardware devices. In this section, a novel data replication placement strategy (DRPS) based on the model is described, which provides the algorithm for finding a better data replication placement map to minimize system delay and satisfy the reliability requirement of all data in the system. In addition, a greedy load balancing algorithm also be used in the DRPS, which balances the calculation, storage, and bandwidth resources of each hardware device as much as possible to achieve load balancing among all hardware devices in the distributed storage system.

A. Ranks-based replication storage node selection algorithm

The terminal device td_{ij} stores a data block bid_i by initiating a written request to the managing edge node ed_i^1 using the PutBlock(bid, reliability, metadata[], data) operation. The request parameters specify the minimum storage reliability requirement r_{bid_i} and the data block size s_{bid_i}. In addition to storing the data block bid_i locally, the managing edge node ed_i^1 responsible for that region selects several additional data replication placement nodes to ensure the reliability of the data.

All terminal devices within the system are potential candidates for data replication replacement nodes, posing a challenge due to the large number of terminal devices and their respective management by different edge nodes. It is impractical to maintain the currently available storage capacity and device reliability of all terminal devices on every managing edge node. To address this challenge, we have designed a ranks-based replication placement node selection algorithm for the storage performance and reliability of terminal devices. This algorithm involves categorizing devices into different ranks, which helps in making decisions to select data replication placement nodes.

Terminal devices report their storage performance information, including available storage capacity and device reliability, to their corresponding managing edge nodes through periodic heartbeat mechanisms. The managing edge nodes maintain the storage attributes of all the terminal devices they oversee and use this information to calculate the minimum, median, and maximum values of storage capacity and reliability for the managed devices, denoted as $(s_i^{min}, s_i^{med}, s_i^{max})$ and $(r_i^{min}, r_i^{med}, r_i^{max})$ respectively. Using s_i^{med} as a threshold, the managed terminal devices are categorized as low-capacity devices (LS) or high-capacity devices (HS). Similarly, using r_i^{med} as a threshold, the edge devices are classified as low-reliability devices (LR) or high-reliability devices (HR). By combining both attributes, the terminal devices can be divided into four categories: low-capacity low-reliability (LSLR), low-capacity high-reliability (LSHR), high-capacity low-reliability (HSLR), and high-capacity high-reliability (HSHR). We denote the four categories as follows:

$$LSLR_i = \{td_{ij} | s_{i,j} \in [s_i^{min}, s_i^{med}]\ \ \&\ \ r_{i,j} \in [r_i^{min}, r_i^{med}]\} \tag{20}$$

$$LSHR_i = \{td_{ij} | s_{i,j} \in [s_i^{min}, s_i^{med}]\ \ \&\ \ r_{i,j} \in [r_i^{med}, r_i^{max}]\} \tag{21}$$

$$HSLR_i = \{td_{ij} | s_{i,j} \in [s_i^{med}, s_i^{max}]\ \ \&\ \ r_{i,j} \in [r_i^{min}, r_i^{med}]\} \tag{22}$$

$$HSHR_i = \{td_{ij} | s_{i,j} \in [s_i^{med}, s_i^{max}]\ \ \&\ \ r_{i,j} \in [r_i^{med}, r_i^{max}]\} \tag{23}$$

The managing edge nodes maintain information about the performance levels of devices in other managing edge nodes based on a ranks-based replication storage node selection algorithm. This information includes the maximum, median, and minimum reliability values of the terminal devices under each managing edge node, the maximum, median, and minimum available capacity, and the number of devices in each performance level category. Let U denote the collection of brief information on all n managing edge nodes in the system, where u_i represents the device performance level information for the edge node ed_i with the corresponding index i, and $\{cnt_i^{LSLR}, cnt_i^{LSHR}, cnt_i^{HSLR}, cnt_i^{HSHR}\}$ are denoted as the devices number of different ranks in the managing edge node.

To prioritize the selection of devices with high remaining storage capacity for data storage, the storage capacity that can be provided by high-capacity devices in each managing edge node can be estimated using the grading indicators. Let sc_i^{HSLR} and sc_i^{HSHR} respectively denote the storage capacity that can be provided by high-capacity low-reliability terminal devices and high-capacity high-reliability terminal devices in managing edge node i. This can be estimated as:

$$sc_i^{HSLR} = s_i^{med} * cnt_i^{HSLR} \tag{24}$$

$$sc_i^{HSHR} = s_i^{med} * cnt_i^{HSHR} \tag{25}$$

Compute the median values of sc^{HSLR} and sc^{HSHR} for all n managing edge nodes, denoted as $medsc^{HSLR}$ and $medsc^{HSHR}$, respectively. Based on these median values, edge nodes are added to the high-capacity high-reliability edge node candidate set U^{HSLR} and the high-capacity low-reliability candidate set U^{HSHR} as follows:

$$U^{HSLR} = \{u_i | sc_i^{HSLR} \geq medsc^{HSLR}\} \tag{26}$$

$$U^{HSHR} = \{u_i | sc_i^{HSHR} \geq medsc^{HSHR}\} \tag{27}$$

The storage nodes will be selected from these two candidate sets. It is worth noting that the elements in U^{HSLR} and U^{HSHR} correspond to managing edge nodes that are abundant in high-capacity low-reliability or high-capacity high-reliability terminals, respectively. These two sets are not entirely exclusive, i.e., a managing edge node may belong to both sets at the same time.

After receiving a PutBlock data write request, managing edge nodes will select data replication storage devices from the two candidate sets based on the storage device statistics of

Algorithm 1 Ranks-based replication storage node selection algorithm

Input: : Data block bib, reliability r_{bib}, terminal device td_{ij}, the high-capacity low-reliability hardware device candidate set U^{HSLR} and the high-capacity high-reliability hardware device candidate set U^{HSHR}

Output: : Replication storage locations: S

1: $S \leftarrow td_{ij}$ (Initialize the data replication storage queue S, which contains the device td_{ij}).
2: $\hat{r} \leftarrow 1 - r_{ij}$ (Initialize reliability of the data replication storage queue S, which contains the evaluation score of device td_{ij}).
3: **if** $r_{ij} \geq r_i^{med}$ **then**
4: $U \leftarrow U^{HSLR}$
5: **else**
6: $U \leftarrow U^{HSHR}$
7: **end if**
8: **while** $\hat{r} \geq 1 - r_{ij}$ **do**
9: $ed_k^1 \in U$ (the managing edge node is selected randomly)
10: Gets the terminal devices td_{k1} of the corresponding type from ed_k^1
11: $S \leftarrow S \cup td_{k1}$
12: $\hat{r} \leftarrow \hat{r} * (1 - r_{k1})$
13: **if** $U = U^{HSHR}$ **then**
14: $U \leftarrow U^{HSLR}$
15: **else**
16: $U \leftarrow U^{HSHR}$
17: **end if**
18: **end while**
19: **Output** S

each managing edge node, as shown in Algorithm 1. Specifically, the edge terminal device td_{ij} first sends a PutBlock operation to its managing edge node ed_i^1, hoping to write a data block bid_i into the system with a storage reliability requirement of r_{bid_i}. Let S represent the data replication storage location that makes up the ranks-based replication storage node selection algorithm. Since td_{ij} stores a replication of the data, S is initialized as td_{ij}, as shown in lines 1-2 of Algorithm 1. The candidate sets are initialized in lines 3-7. If td_{ij} is a high-reliability device, the next device to be selected should be a low-reliability device; if td_{ij} is a low-reliability device, the next device to be selected should be a high-reliability device. Then, ed_i^1 randomly selects a managing edge node ed_k^1 and requests the corresponding category of terminal device as a storage node. ed_k^1 returns the terminal device td_{k1}, which is randomly added to the data replication placement locations, resulting in $S = \{td_{ij}, td_{k1}\}$. If S meets the storage reliability requirement r_{bid_i}, that is, $1-r_{bid_i}(1-r_{ij})*(1-r_{k1})$, the algorithm is completed. Otherwise, ed_i^1 will select a managing edge node from the other candidate set and request the corresponding type of terminal devices as a data replication storage node. The process is repeated, using terminal devices from the two candidate sets alternately as replication storage nodes, until the storage reliability of the data replication meets the requirement, as shown in lines 8-18 of Algorithm 1. When selecting terminal devices with different reliabilities as storage nodes, high-capacity and high-reliability terminals (HSHR or HSLR) are given priority to balance the remaining storage capacity of each hardware device as much as possible. It is noteworthy that the runtime of this algorithm increases linearly with the size of the input data. Its time complexity is O(n).

Algorithm 2 Data Replication Placement Strategy

Input: : All hardware devices, data blocks, and the currently available load of each hardware device are $dev.load_i = \{dev.c_i, dev.s_i, dev.b_i\}$

Output: : E (data replication placement map)

1: Initial E by performing ranks-based replication storage node selection algorithm (i.e., Algorithm 1)
2: $G \leftarrow sort(dev.load_i)$ (Sort all the hardware devices in the system).
3: **for** dev_i **in** E **do**
4: find dev_{i-1} in G
5: **if** $r_{dev_{i-1}} \geq r_{dev_i}$ **then**
6: Calculate the $T_{sys_{i-1}}$ and T_{sys_i} by Equation (14),
7: **if** $T_{sys_{i-1}} \leq T_{sys_i}$ **then**
8: dev_i = dev_{i-1}
9: Update the E
10: **end if**
11: Keep dev_i in E and Update the E
12: **end if**
13: Keep dev_i in E and Update the E
14: **end for**
15: **Output** E

B. Data Replication Placement Strategy

When it comes to data replication placement strategies, there are two main objectives to consider: ensuring data reliability and maximizing load balancing within the system. Algorithm 1 effectively guarantees the reliability requirements for all data stored in the system. By combining a greedy load balancing algorithm, Algorithm 2 is an extension of Algorithm 1. This enhanced approach not only maximizes resource utilization within the system but also effectively reduces system delay during the storage process. The time complexity of Algorithm 2 is O(n^2).

In Algorithm 2, we first initiated the data replication map by performing Algorithm 1, and the result can satisfy the data reliability. However, the result does not consider the load balancing. In this study, we consider the system's load by taking into account the available computing capacity, storage capacity, and network resources among devices. In the storage node queue G, the hardware devices available for storage are sorted in ascending order based on their load capacities. Among these devices, we identify the placement nodes that already meet the reliability requirements. For each of these nodes, we search for a slightly smaller load node. If replacing the original node with this node fails to meet the reliability requirements, the replacement cannot be performed. However, if the replacement maintains data reliability, we then calculate

the system delay for both placement options. If the latency of the replaced node is not greater than the latency of the original placement, we replace the node with the device for data replication placement. Through this iterative greedy approach, we have achieved a data replication placement strategy that satisfies both data reliability and minimizes system delay and hardware load performance.

V. Performance Evaluation

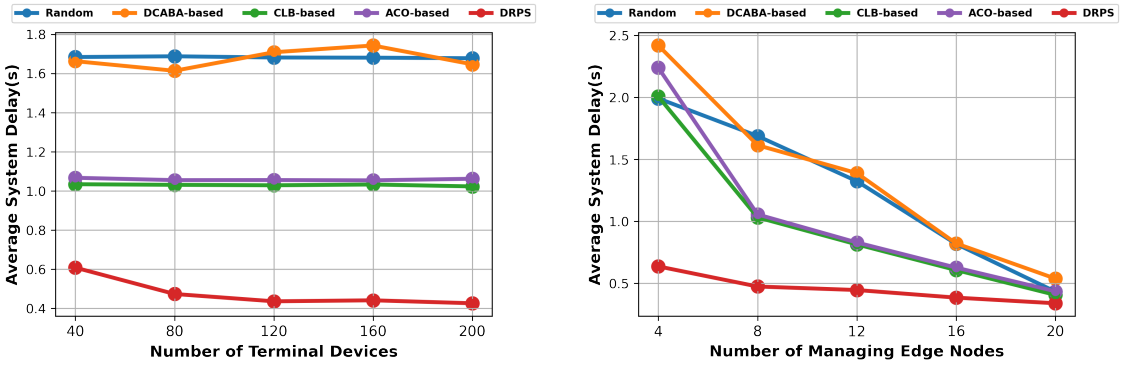
In this section, we design comprehensive simulation experiments to evaluate the effectiveness of the proposed data replication placement strategy and discuss the impact factors in our constructed model. Comparing the results with those from other strategies and also considering the different scenarios of these factors, the advantages of our proposed strategy in this context, as well as the impact factors, are evaluated. The impact of important parameters on the performance of the proposed algorithm, including the number of edge nodes, the number of terminal devices, the computing and storage resource capacity of edge nodes and terminal devices, and the average hardware utilization of each hardware device in the system, are mainly investigated. We run the simulation experiments on Python 3.5 hosted by a PC with an Intel i7 3.5 GHz CPU and 16 GB RAM.

A. Experimental Setup

In the experiment, we consider a cloud-edge-terminal orchestrated computing environment consisting of 100 edge nodes, of which 20 nodes are set as managing edge nodes. Furthermore, in the environment, there is also one cloud data center and 4000 terminal devices. To better describe the impacts of the number of these hardware devices, the number of the managing edge node varies between 4 and 20 with an increment of 4, and the average number of the terminal devices under each managing edge node is set from 40 to 200 with an increment of 40. According to the real configurations of reliability and storage capacity for terminal devices, the settings for our terminal devices in the distributed storage system are as follows: 1) Reliability: The reliability is set to follow a normal distribution with a mean of 0.9 and a standard deviation of 0.04, denoted as $\mathbf{N}(0.9, 0.04^2)$. This means that the reliability is distributed around a mean of 0.9 with a standard deviation of 0.04. 2) Storage Capacity: The available storage capacity of these terminal devices is uniformly distributed between 8GB and 64GB. 3) Computing Capacity: Following [41], [45], the computing resource capacity used for storage and replication is set as $5 \sim 10$ GHz. 4) Network setting: According to the [42], we model the channel power gain between the terminal devices under the same managing edge node as $140.7 + 36.7log_{10}(d_{ij}) + \mu$, where we represent the transmission distance of terminals i and j as d_{ij}, which is randomly assigned within the range of 0.02 km to 0.4 km. The parameter is μ denotes the log-normal shadowing standard deviation, following a normal distribution $\mathcal{N}(0, 8\ \text{dB})$. The cable transmission latency is also randomly selected within the range of 20 ms to 200 ms, while the channel bandwidth is fixed at 2 MHz [40]. The data blocks that require replication are generated by the terminal devices in the system and have storage-related loads (including the memory capacity they require for storage and computational capabilities) as well as reliability requirements. In our experiment, the reliability requirements for the data blocks to be replicated are categorized into five levels: $\{0.9, 0.93, 0.95, 0.99, 0.999\}$. The memory loads required by the data blocks are divided into three types: $\{10\text{MB}, 30\text{MB}, 50\text{MB}\}$, with their proportions among all generated data blocks being $\{50\%, 30\%, 20\%\}$ respectively. Regarding computational capabilities, during the storage and replication process, the hardware devices only need to handle task requests. We randomly set the computational capability requirement within the range of 200 to 400 Megacycles. In addition, we set the bandwidth between different edge nodes as 100 M/s$\sim$200 M/s. Unless otherwise specified, the default setting for the experimental parameters will be used. Each data point in the experiments is obtained through 100 independent runs.

To evaluate the effectiveness of the proposed DRPS in this paper, we compared DRPS with the following four methods:

- **Random Data Replication Placement:** This strategy is commonly used in traditional file systems like HDFS for data replication and placement. For the common HDFS, each data block that needs to be stored is replicated into three copies. These three replicas are randomly selected among the storage nodes in the system for storage. In our work, the number of replicas depends on the reliability requirements of data blocks. The strategy needs to satisfy the requirement, which is the same as other strategies.
- **DCABA-based Data Replication Placement:** The DCABA algorithm is a load-balancing algorithm based on task clustering, which achieves load balancing by assigning tasks to nodes with similar load states. Specifically, the data replication placement strategy based on the DCABA algorithm divides the hardware devices in the system into multiple groups and maintains the average load value for each group. When a new task request is received, this strategy selects the group with the minimum load and randomly places the replica on a device within that group.
- **CLB-based Data Replication Placement:** The CLB algorithm is a content-based load-balancing algorithm that selects appropriate nodes for data replication placement based on the characteristics of the data block. Specifically, the data replication placement strategy based on CLB identifies a few managing edge nodes closest to the data block based on its characteristics and access frequency. The data is then allocated to the hardware devices under the jurisdiction of those nodes.
- **ACO-based Data Replication Placement:** The ACO (Ant Colony Optimization) algorithm possesses characteristics of distributed computing, positive feedback of information, and heuristic search, making it a heuristic global optimization algorithm within the domain of evolutionary algorithms. The ACO-based data replication strategy dynamically selects suitable hardware devices for data storage based on the load status of the hardware



(a) (b)

Fig. 5. Comparison of Average System Delay with Different Data Replication Placement Strategies

devices in the system and the requirements of the placement tasks. It achieves an optimized selection of hardware devices for storage through the updating of pheromone values.

B. Experimental Results and Evaluation

The performance of the proposed DRPS is first compared to the other related methods in terms of the average system delay, the number of replications of hardware devices, and the memory resource utilization while processing a batch of IoT-generating data. Meanwhile, we consider and analyze the effect of experimental parameters, including the number of managing edge nodes and terminal devices.

1) Comparison of Average System Delay: The average system delay refers to the system delay required for data storage and replication when all the data blocks generated in the distributed storage system meet the reliability requirements. A smaller average system delay implies faster data storage and replication, which is of significant importance for time-sensitive applications. As shown in Figure 5, regardless of the variations in managing edge nodes and terminal devices within the system, our proposed DRPS (Data Replication Placement Strategy) consistently achieves the lowest average system delay, with a 39.9% reduction compared to the baselines, proving the superiority of DRPS. In Figure 5(a), compared to Random and DCABA-based strategies, the CLB-based and ACO-based strategies are superior as they also take into account the data transmission delay mentioned in formula(14). However, by using Algorithm 1, the DRPS achieves a 39.9% reduction in the average system delay. On the other hand, the DCABA-based strategy exhibits the highest average system delay, surpassing even the random strategy. This is because the DCABA-based strategy primarily only focuses on load conditions without considering the time taken for data transmission due to spatial distance and strategy execution time.

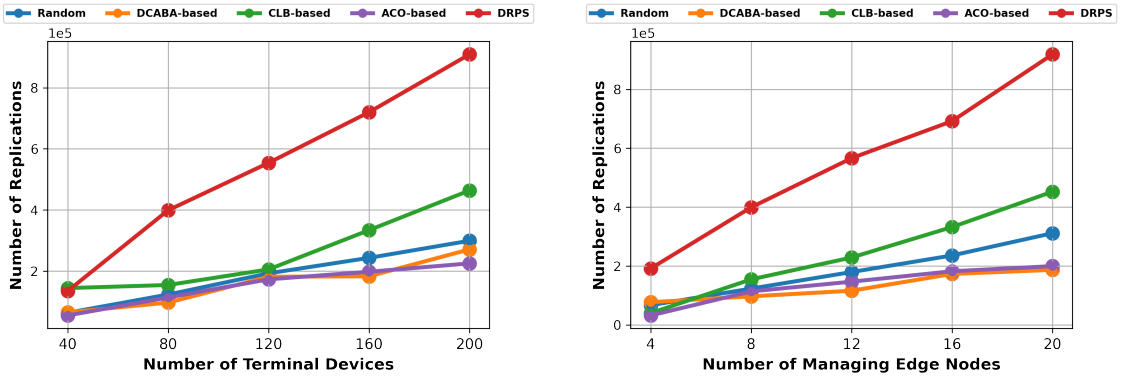
In Figure 5(b), as the number of managing edge nodes in the system increases, the average system delay of all five data replication placement strategies decreases. However, among all the strategies, the proposed DRPS strategy remains the most stable. The reason for the decreasing average system delay in all strategies is that as the number of managing edge nodes increases, there are more available hardware devices for storing replicas, allowing for faster storage completion for each data block. In addition, another reason for the decreasing average system delay as the number of managing edge nodes increases is the locations of these nodes and terminal devices. The locations of the managing edge nodes and terminal devices were determined based on the Shanghai Telecom base stations and the data accessed from these base stations. Therefore, as the number of edge nodes increases, resulting in a higher density of their actual positions, the average spatial distance between the nodes naturally decreases. As a result, the average system delay also decreases.

TABLE III
COMPARISON OF OTHER METHODS AND DRPS BY USING ANALYSIS OF VARIANCE (ANOVA) AND TUKEY-HSD

	Average System Delay	
	Tukey's P_1	Tukey's P_2
Random v.s. DRPS	¡ 0.00001	0.055057
ACO-based v.s. DRPS	¡ 0.00001	0.037527
CLB-based v.s. DRPS	¡ 0.00001	0.073492
DCABA-based v.s. DRPS	¡ 0.00001	0.0011195

* The P_1(ANOVA) ¡ 0.00001 and P_2(ANOVA) = 0.00284.

Figure 5(a) shows experimental results comparing the average system delay across different methods as the number of terminal devices varies. Meanwhile, Figure 5(b) shows the results comparing the average system delay across different methods as the number of managing nodes varies. To better statistically analyze the results presented in Figure 5(a) and Figure 5(b), following [46], [47], we conduct a two-way analysis of variance (ANOVA) and Tukey multiple comparisons to distinguish our DRPS from other methods. The values obtained from the analysis of the results in Figure 5(a) are denoted as P_1, while those derived from Figure 5(b) are represented as P_2. As shown in Table III, the results indicate that both P_1(ANOVA) and P_2(ANOVA) values are less than 0.05, and Tukey's P_1 and Tukey's P_2 remain comparatively small. This demonstrates a significant distinction between our method and others, further validating the effectiveness of the results presented in Figure 5.



(a) (b)

Fig. 6. Comparison of Replication Numbers with Different Data Replication Placement Strategies

TABLE IV
COMPARISON OF OTHER METHODS AND DRPS BY USING ANALYSIS OF VARIANCE (ANOVA) AND TUKEY-HSD

	Number of Replications	
	Tukey's P_1	Tukey's P_2
Random v.s. DRPS	0.000233	0.000109
ACO-based v.s. DRPS	0.000091	0.000026
CLB-based v.s. DRPS	0.002538	0.000689
DCABA-based v.s. DRPS	0.0001102	0.000022

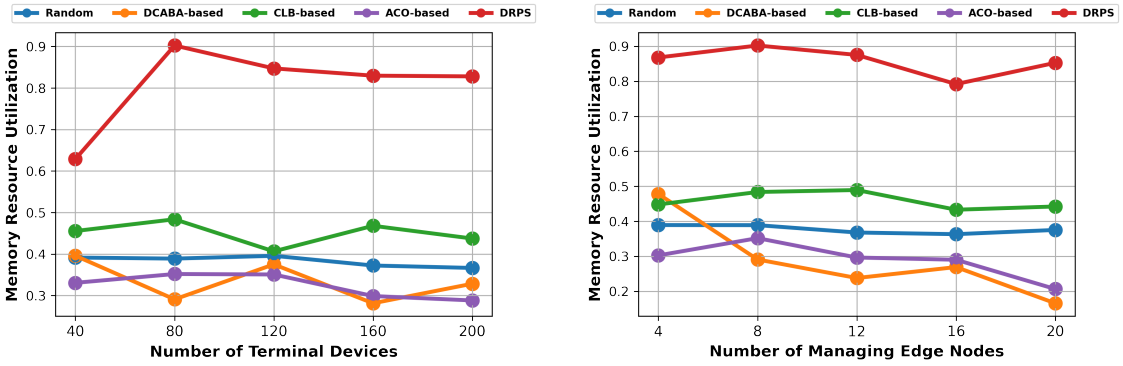
* The P_1(ANOVA) = 0.000043 and P_2(ANOVA) = 0.000011.

2) Comparison of The Number of Replications: The number of replications refers to the number of successful data replication placement operations that can be completed by the system from the start of storing replicas until the available resources of one of these hardware devices are exhausted. A higher number of replications indicates that the data replication placement strategy achieves better load balancing. This enables more convenient addition of hardware devices or an increased quantity of data blocks to the system, meeting the growing demands for data storage and replication without compromising service quality, such as system delay, and ensuring effective data storage and replication. As shown in Figure 6, as the number of managing edge nodes or terminal devices increases, all methods exhibit an increase in the number of replications. This is because the addition of hardware devices translates to an increase in available resources, leading to better load distribution among the devices. Furthermore, it is evident that the proposed strategy demonstrates a significantly faster growth in the number of replicas stored as the device resources increase compared to other methods. By employing Algorithm 2, the proposed DRPS effectively increases replicas by 43.3%. This result highlights the superior load-balancing capability of our proposed strategy.

When comparing our proposed strategy to the best-performing CLB-based strategy among others, it is observed that as the average number of terminal devices managed by each edge node increases from 40 to 200, our strategy achieves an additional increase of 774,660 replicas, which is 58.7% more in growth rate compared to the CLB-based strategy. Similarly, when the number of edge nodes increases from 4 to 20, our proposed strategy can store 918,613 replicas, whereas the best-performing CLB-based strategy in other methods can store only 452,104 replicas. Both in terms of the growth in the number of replicas and the growth rate, our proposed strategy outperforms the CLB-based strategy by more than 43.3%. In Table IV, similarly to Table III, we denote the analysis of the results in Figure 6(a) as P_1, while representing the analysis of the results derived from Figure 6(b) as P_2. The results indicate that both P_1(ANOVA) and P_2(ANOVA) values are less than 0.05, and Tukey's P_1 and Tukey's P_2 remain comparatively small. This demonstrates a significant distinction between our method and others, further validating the effectiveness of the results presented in Figure 6.

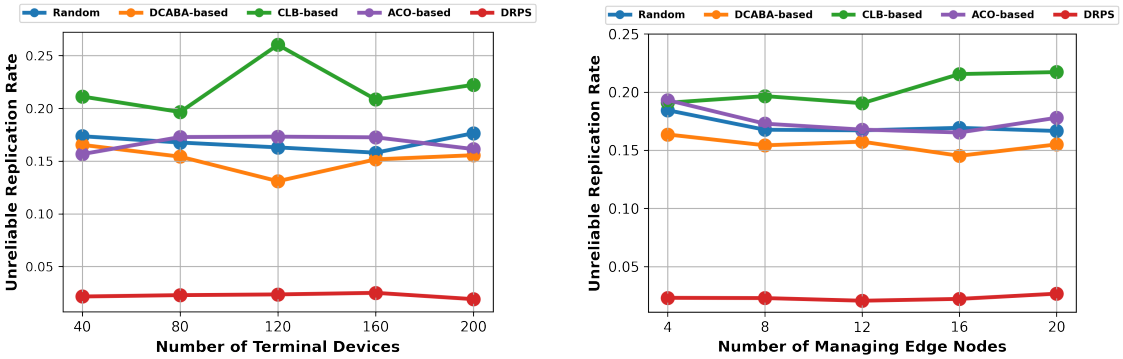
For each hardware device and generated data block, the strategy we have designed takes into account detailed considerations of storage resources and reliability. We address the overall load balancing of the system, allowing for a more precise selection of suitable hardware devices for data storage and replication while avoiding situations where a single device becomes overloaded and exhausts its resources. Furthermore, we have implemented a ranks-based replication storage node selection algorithm, categorizing devices into different levels of reliability. This allows for more accurate control over the replication and storage of data with different reliability requirements on each hardware device in the system. It helps reduce storage resource wastage to a certain extent and enhances the overall load-balancing capability of the system.

3) Comparison of Memory Resource Utilization: Memory resource utilization refers to the overall utilization of system resources when a hardware device's resources of these hardware devices are exhausted. The specific calculation determines the ratio between the currently available resources and the maximum available resources in the system. It is an important metric that reflects the effectiveness of the data replication placement strategy in achieving load balancing. Based on the formula (17), if there is a high utilization rate of memory resources, it implies that more storage nodes in the system can handle increased loads, leading to improved load balancing within the system. According to the results depicted



(a) (b)

Fig. 7. Comparison of Memory Resource Utilization with Different Data Replication Placement Strategies



(a) (b)

Fig. 8. Comparison of Unreliability Rate with Different Data Replication Placement Strategies

in Figure 7, our proposed strategy consistently outperforms other solutions in terms of resource utilization, regardless of variations in the number of managing edge nodes and terminal devices. For the statistical analysis of Figure 7(a) and 7(b), Table V show that P_1(ANOVA) and P_2(ANOVA) values are less than 0.00001, and Tukey's P_1 and Tukey's P_2 values are also less than 0.00001.

The outcome can be attributed to the comprehensive considerations embedded in our strategy. By resource consumption grading and reliability grading for devices and data blocks, our data replication placement strategy imposes more constraints during the selection of storage devices. As a result, it enhances the efficiency of resource utilization, ensuring better utilization of storage resources. Comparing the best-performing CLB-based strategy, the proposed DRPS increases 27.5% $\sim$ 52.0% in terms of memory resource utilization, demonstrating the superiority of our approach.

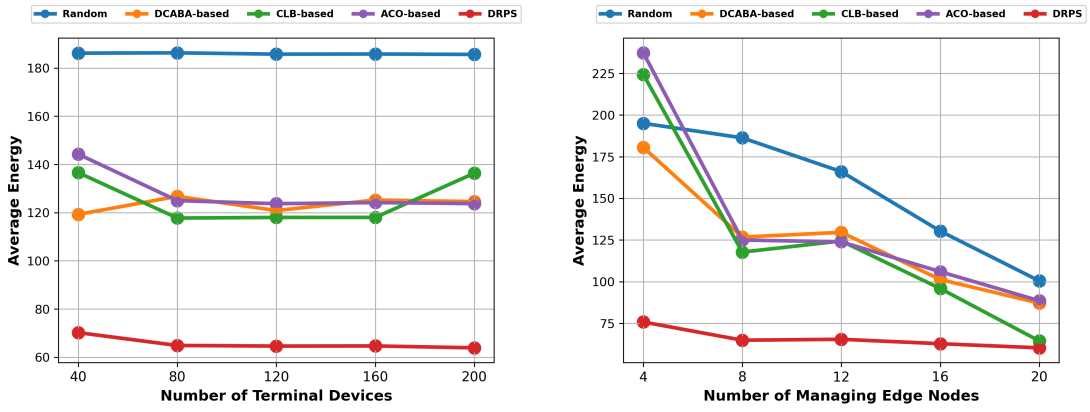
4) Comparison of Unreliability Rate: By fixing the number of data replications, the unreliability of data replication placement strategies in the distributed storage system can be assessed by quantifying the success rate of these strategies in replicating that data. In this study, we only focus on ensuring data storage reliability without imposing constraints on the number of data replications. Instead, in this experiment, we impose a limitation whereby all data is replicated with exactly three replicas, which are generally used in traditional distributed storage systems. Under this constraint, when a hardware device resource is depleted, some data fail to meet the reliability requirements, and the ratio of such data to the total replicated data stored is defined as the unreliability rate of the strategy.

TABLE V
COMPARISON OF OTHER METHODS AND DRPS BY USING ANALYSIS OF VARIANCE (ANOVA) AND TUKEY-HSD

	Memory Utilization	
	Tukey's P_1	Tukey's P_2
Random v.s. DRPS	¡ 0.00001	¡ 0.00001
ACO-based v.s. DRPS	¡ 0.00001	¡ 0.00001
CLB-based v.s. DRPS	¡ 0.00001	¡ 0.00001
DCABA-based v.s. DRPS	¡ 0.00001	¡ 0.00001

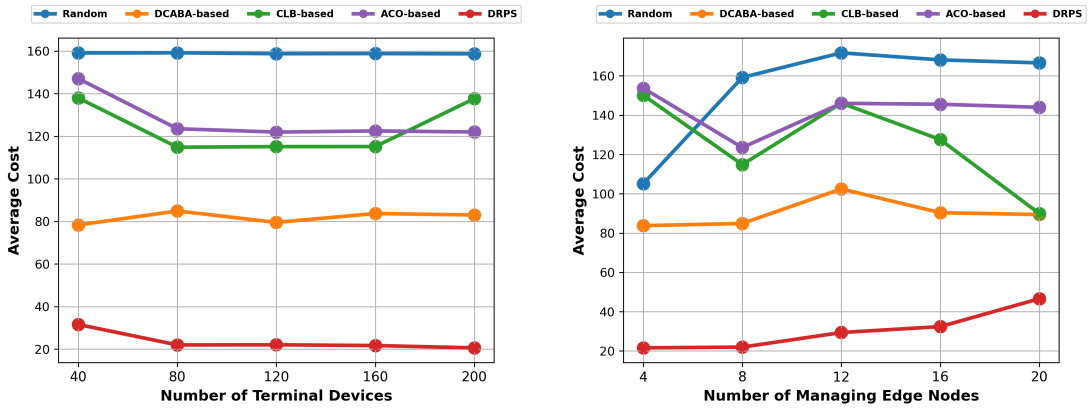
* The P_1(ANOVA) ¡ 0.00001 and P_2(ANOVA) ¡ 0.00001.

Compared with other data replication placement methods, the results indicate that our proposed DRPS exhibits the lowest data unreliability rate. Following the formula (7), it is observed that as the pre-defined reliability attribute of r_i the hardware devices used for storing replicas increases, the unreliability rate correspondingly decreases. As shown in Figure 8, compared to the best strategy in alternative data replication methods, the proposed DRPS reduces the unreliability rate by 82.0%$\sim$87.8% in a cloud-edge-terminal



(a) (b)

Fig. 9. Comparison of Average Energy with Different Data Replication Placement Strategies



(a) (b)

Fig. 10. Comparison of Average Cost with Different Data Replication Placement Strategies

TABLE VI
COMPARISON OF OTHER METHODS AND DRPS BY USING ANALYSIS OF VARIANCE (ANOVA) AND TUKEY-HSD

	Unreliability Rate	
	Tukey's P_1	Tukey's P_2
Random v.s. DRPS	¡ 0.00001	¡ 0.00001
ACO-based v.s. DRPS	¡ 0.00001	¡ 0.00001
CLB-based v.s. DRPS	¡ 0.00001	¡ 0.00001
DCABA-based v.s. DRPS	¡ 0.00001	¡ 0.00001

* The P_1(ANOVA) ¡ 0.00001 and P_2(ANOVA) ¡ 0.00001.

orchestrated environment characterized by varying numbers of edge nodes and terminal devices. This can be attributed to the fact that our proposed data replication strategy leverages more precise data and device information during the node selection process, facilitating the placement of replicated data in a manner that optimally ensures storage location accuracy while minimizing the number of replications. In addition, as shown in Table VI, for Figure 8, P_1(ANOVA) ¡ 0.00001 and Tukey's P_1 ¡ 0.00001 denote that the proposed DRPS exhibits a significant difference compared to other methods when varying the number of terminal devices. Similarly, the results indicated by P_2(ANOVA) ¡ 0.00001, coupled with P_2 ¡ 0.00001, signify that the proposed DRPS exhibits a significant difference compared to other methods when varying the number of managing edge nodes.

5) Comparison of Average Energy and Cost: The average energy and cost pertain to the energy consumption and expenses associated with data transfer required for storing and replicating data in a distributed storage system while meeting reliability requirements for all data blocks. The average energy represents the consumption of data replication, considering both node distance and the number of data replicas. The average cost refers to storage expenses and is proportional to the number of data replicas [48]. Comparative analysis against alternative data replication placement methods reveals that our proposed DRPS strategy exhibits the lowest average energy consumption and cost, proving the superiority of our approach. As depicted in Figure 9 and Figure 10, as the number of terminal devices increases, all strategies maintain stability in average energy consumption for data storage and replication. Moreover, the proposed DRPS consistently outperforms other strategies by approximately 50%. This superiority stems from DRPS's ability to consistently meet reliability storage requirements more efficiently and with lower energy consumption.

Regarding the number of managing edge nodes, an increase in their count leads to a gradual decrease in the average energy consumption across all data replication strategies. However, among these strategies, the proposed DRPS maintains the highest level of stability. This can be attributed to the utilization of more edge nodes for management, which provides additional a priori information and hardware resources for storing replicas in the system. Furthermore, the decrease in average energy with an increasing number of managing edge nodes is influenced by the geographical distribution of these nodes and terminal devices. These locations were determined based on Shanghai Telecom base stations and the data accessed through them. Consequently, as the number of edge nodes rises, resulting in a higher density of their actual positions, the average spatial distance between nodes naturally decreases, leading to a reduction in average energy consumption.

In addition, it is imperative to consider the experimental findings depicted in Figure 10(b). While the average cost of the DRPS proposed in this paper exhibits a significant reduction compared to other strategies, it marginally increases with the growing number of managing edge nodes. These results may be from the fact that the computation of cost is intricately linked to both the number of data replicas to be stored and the volume of information to be stored on managing edge nodes. The escalation in the number of managing edge nodes responsible for storing information is a corresponding increase in the information load associated with data replication within the system. The challenge deserves further attention and solutions in future research work.

6) Comparison of Different Storage Patterns: As mentioned above, an important corollary is that in a cloud-edge-terminal orchestrated computing environment, adopting terminal resource-sharing for data storage and replication not only provides the edge nodes in the system with more resources to handle an increasing number of intelligent tasks but also significantly reduces the system delay required for data replication placement. In the experiment, we differentiated the storage patterns as the traditional data replication placement patterns and the proposed strategy (i.e., terminal devices also as storage nodes). To better compare the difference between placing data replicas through both edge nodes and terminal devices and placing them solely on edge nodes, we first simulated 20 edge nodes and 1000 data blocks that require storage and replication tasks in the distributed storage system. We obtained the system delay required for storage using various data replica placement strategies. Next, we designated 8 out of the simulated 20 edge nodes as managing edge nodes, responsible for managing 640 terminal devices that can share resources for replica placement. As shown in Figure 11, except for the DCABA-based strategy, storing data replications through the joint placement of edge nodes and terminal devices can improve the average system delay during the data storage and replication process. It is worth noting that expanding terminal devices in the system to become storage nodes is done to increase the utilization of available resources. Therefore, as long as the average system delay is acceptable, the results mean that the proposed storage pattern can be applied.

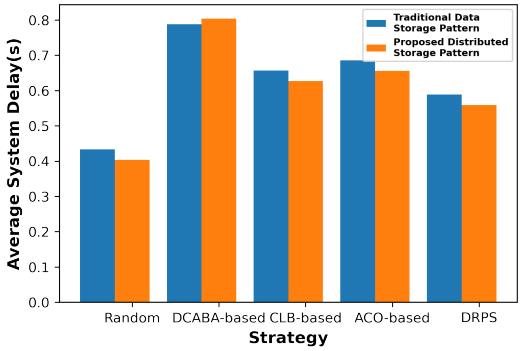


Fig. 11. Comparison Different Storage Patterns With and Without Terminal Devices

TABLE VII
COMPUTATIONAL COMPLEXITY OF DIFFERENT STRATEGIES

Strategy	Node Selection	Load Balancing	Overall Complexity
DRPS	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Random	$\mathcal{O}(1)$	-	$\mathcal{O}(1)$
DCABA	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$
CLB	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
ACO	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \sim n^3)$	$\mathcal{O}(n^3)$

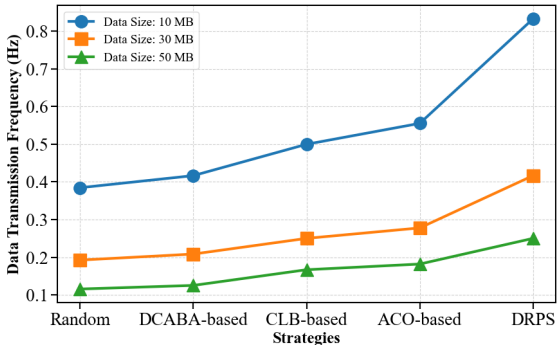


Fig. 12. Comparison of Data Transmission Frequency for Different Strategies

7) Comparison of Overhead: To prove the effectiveness of the DRPS strategy, we analyzed its computational overhead, as shown in Table VII. By combining node selection and load-balancing efficiency, the DRPS strategy maintains moderate computational overhead $\mathcal{O}(n^2)$ compared to other strategies. In terms of computational overhead, the Random, DCABA, and CLB strategies are competitive to some extent. However, the random strategy performs poorly in resource utilization and latency. DCABA fails to consider data transmission distance and latency. CLB has significant deficiencies in system latency, replication, and memory utilization reliability. Therefore, our proposed DPRS strategy has an advantage over other comparison strategies in terms of computational overhead.

Additionally, DRPS offers better data transmission efficiency. As shown in Figure 12, we compare the data transmission frequency of different data blocks under 40 edge nodes and 200 terminal devices. For the same data volume, DRPS achieves better transmission frequency, benefiting from its

advantage in system latency. This demonstrates the superiority of our proposed algorithm in terms of data communication overhead.

Combined with the overhead advantages of data transfer and computation, DRPS significantly improves system performance by reducing system latency by 39.9%, increasing replication efficiency by 43.3%, improving memory utilization by 27.5%, and reducing instability by 82%. These benefits justify the DRPS strategy, ensuring better reliability, balanced resource utilization, and optimized storage performance.

VI. Conclusion and Future Works

In this paper, a data replication placement strategy, named DRPS, is proposed, leveraging the synergy of a ranks-based replication storage node selection algorithm and a greedy load balancing algorithm. This strategy aims to optimize the mapping of replicas and hardware devices, focusing on enhancing system delay and resource utilization within cloud-edge-terminal orchestrated computing environments. Firstly, a data replication placement model was constructed to consider multiple different replication tasks across heterogeneous hardware devices. Furthermore, the data replication placement was formulated as a multi-objective optimization problem and the DRPS was proposed to solve it. The data replication strategy not only properly trades off system delays and resource utilization but also guarantees the reliability of every data block in the distributed storage system. The comprehensive experiments were designed to verify that, in comparison to other state-of-the-art methods, DRPS reduced system delay by 39.9%, increased replication by 43.3%, enhanced memory utilization by 27.5%, and decreased unreliability rate by 82.0%.

In future work, an investigation into the integration of DPRS and federated learning/reinforcement learning is anticipated. This approach will be aimed at managing online-generated data requiring replication, with the objective of further enhancing both system delay and reliability performance. The explorations are expected to achieve the optimization of data replication strategies within dynamic computing environments.

VII. Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

VIII. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Li, W. Liang, J. Li, X. Cheng, D. Yu, A. Y. Zomaya, and S. Guo, "Energy-aware, device-to-device assisted federated learning in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 7, pp. 2138–2154, 2023.

[2] Y. Yang, M. Ma, H. Wu, Q. Yu, P. Zhang, X. You, J. Wu, C. Peng, T.-S. P. Yum, S. Shen *et al.*, "6g network ai architecture for everyone-centric customized services," *arXiv preprint arXiv:2205.09944*, 2022.

[3] X. Du, S. Tang, Z. Lu, J. Wet, K. Gai, and P. C. Hung, "A novel data placement strategy for data-sharing scientific workflows in heterogeneous edge-cloud computing environments," in *2020 IEEE International Conference on Web Services (ICWS)*, 2020, pp. 498–507.

[4] X. Du, S. Tang, Z. Lu, K. Gai, J. Wu, and P. C. Hung, "Scientific workflows in iot environments: a data placement strategy based on heterogeneous edge-cloud computing," *ACM Transactions on Management Information Systems (TMIS)*, vol. 13, no. 4, pp. 1–26, 2022.

[5] R. Besharati, M. H. Rezvani, M. M. Gilanian Sadeghi *et al.*, "An auction-based bid prediction mechanism for fog-cloud offloading using q-learning," *Complexity*, vol. 2023, 2023.

[6] D. Prerna, R. Tekchandani, and N. Kumar, "Device-to-device content caching techniques in 5g: A taxonomy, solutions, and challenges," *Computer Communications*, vol. 153, pp. 48–84, 2020.

[7] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On the fundamental limits of device-to-device private caching under uncoded cache placement and user collusion," *IEEE Transactions on Information Theory*, vol. 68, no. 9, pp. 5701–5729, 2022.

[8] M. F. Mohamed, M. Dahshan, K. Li, A. Salah *et al.*, "Virtual machine replica placement using a multiobjective genetic algorithm," *International Journal of Intelligent Systems*, vol. 2002, 2023.

[9] S. Tang, X. Du, Z. Lu, K. Gai, J. Wu, P. C. Hung, and K.-K. R. Choo, "Coordinate-based efficient indexing mechanism for intelligent iot systems in heterogeneous edge computing," *Journal of Parallel and Distributed Computing*, vol. 166, pp. 45–56, 2022.

[10] W. Lv, Y. Lu, Y. Zhang, P. Duan, and J. Shu, "Infinifs: An efficient metadata service for large-scale distributed filesystems," in *20th USENIX Conference on File and Storage Technologies (FAST 22)*, 2022, pp. 313–328.

[11] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS operating systems review*, vol. 44, no. 2, pp. 35–40, 2010.

[12] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[13] H. Gupta and U. Ramachandran, "Fogstore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access," in *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*, 2018, pp. 148–159.

[14] S. K. Monga, S. K. Ramachandra, and Y. Simmhan, "Elfstore: A resilient data storage service for federated edge and fog resources," in *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 2019, pp. 336–345.

[15] R. Mayer, H. Gupta, E. Saurez, and U. Ramachandran, "Fogstore: Toward a distributed data store for fog computing," in *2017 IEEE Fog World Congress (FWC)*. IEEE, 2017, pp. 1–6.

[16] T. Huang, W. Lin, Y. Li, L. He, and S. Peng, "A latency-aware multiple data replicas placement strategy for fog computing," *Journal of Signal Processing Systems*, vol. 91, pp. 1191–1204, 2019.

[17] K. Liu, J. Peng, J. Wang, W. Liu, Z. Huang, and J. Pan, "Scalable and adaptive data replica placement for geo-distributed cloud storages," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1575–1587, 2020.

[18] J. Liu, M. Xie, S. Chen, G. Xu, T. Wu, and W. Li, "Ts-replica: A novel replica placement algorithm based on the entropy weight topsis method in spark for multimedia data analysis," *Information Sciences*, 2023.

[19] B. Confais, B. Parrein, and A. Lebre, "A tree-based approach to locate object replicas in a fog storage infrastructure," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[20] F. Karatas and I. Korpeoglu, "Fog-based data distribution service (f-dad) for internet of things (iot) applications," *Future Generation Computer Systems*, vol. 93, pp. 156–169, 2019.

[21] C. Li, M. Song, M. Zhang, and Y. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 107–128, 2020.

[22] N. Saranya, K. Geetha, and C. Rajan, "Data replication in mobile edge computing systems to reduce latency in internet of things," *Wireless Personal Communications*, vol. 112, pp. 2643–2662, 2020.

[23] J. Taghizadeh, M. Ghobaei-Arani, and A. Shahidinejad, "A metaheuristic-based data replica placement approach for data-intensive iot applications in the fog computing environment," *Software: Practice and Experience*, vol. 52, no. 2, pp. 482–505, 2022.

[24] C. Symvoulidis, A. Kiourtis, G. Marinos, J.-D. Totow Tom-Ata, G. Manias, A. Mavrogiorgou, and D. Kyriazis, "A user mobility-based data placement strategy in a hybrid cloud/edge environment using a causal-aware deep learning network," *IEEE Transactions on Computers*, vol. 72, no. 12, pp. 3603–3616, 2023.

[25] K. Kontodimas, P. Soumplis, A. Kretsis, P. Kokkinos, M. Fehér, D. E. Lucani, and E. Varvarigos, "Secure distributed storage orchestration on heterogeneous cloud-edge infrastructures," *IEEE Transactions on Cloud Computing*, vol. 11, no. 4, pp. 3407–3425, 2023.
[26] J. Dias, J. A. Silva, and H. Paulino, "Adaptive replica selection in mobile edge environments," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2021, pp. 243–263.
[27] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami, "Data replication schemes in cloud computing: a survey," *Cluster Computing*, vol. 24, pp. 2545–2579, 2021.
[28] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "A cost-efficient auto-scaling mechanism for iot applications in fog computing environment: a deep learning-based approach," *Cluster Computing*, vol. 24, no. 4, pp. 3277–3292, 2021.
[29] E. Torabi, M. Ghobaei-Arani, and A. Shahidinejad, "Data replica placement approaches in fog computing: a review," *Cluster Computing*, vol. 25, no. 5, pp. 3561–3589, 2022.
[30] M. Reiss-Mirzaei, M. Ghobaei-Arani, and L. Esmaeili, "A review on the edge caching mechanisms in the mobile edge computing: A social-aware perspective," *Internet of Things*, p. 100690, 2023.
[31] K. Sarwar, S. Yongchareon, J. Yu, and S. ur Rehman, "Efficient privacy-preserving data replication in fog-enabled iot," *Future Generation Computer Systems*, vol. 128, pp. 538–551, 2022.
[32] N. Afonso, M. Bravo, and L. Rodrigues, "Combining high throughput and low migration latency for consistent data storage on the edge," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020, pp. 1–11.
[33] Y. Shao, C. Li, and H. Tang, "A data replica placement strategy for iot workflows in collaborative edge and cloud environments," *Computer Networks*, vol. 148, pp. 46–59, 2019.
[34] A. J. Fahs and G. Pierre, "Proximity-aware traffic routing in distributed fog computing platforms," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2019, pp. 478–487.
[35] ——, "Tail-latency-aware fog application replica placement," in *Service-Oriented Computing: 18th International Conference, ICSOC 2020, Dubai, United Arab Emirates, December 14–17, 2020, Proceedings 18*. Springer, 2020, pp. 508–524.
[36] K. Toczé, A. J. Fahs, G. Pierre, and S. Nadjm-Tehrani, "Violinn: Proximity-aware edge placementwith dynamic and elastic resource provisioning," *ACM Transactions on Internet of Things*, vol. 4, no. 1, pp. 1–31, 2023.
[37] Y. Zhang, W. Liang, Z. Xu, and X. Jia, "Mobility-aware service provisioning in edge computing via digital twin replica placements," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 295–11 311, 2024.
[38] Q. Zheng, Z. Zhao, W. Lu, C. Yao, Y. Chen, A. Pan, and X. Du, "Lion: Minimizing distributed transactions through adaptive replica provision," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 2012–2025.
[39] X. Li, Z. Zhou, Q. He, Z. Shi, W. Gaaloul, and S. Yangui, "Rescheduling iot services in edge networks," *IEEE Transactions on Network and Service Management*, vol. 2023, 2023.
[40] Y. Qu, H. Dai, F. Wu, D. Lu, C. Dong, S. Tang, and G. Chen, "Robust offloading scheduling for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2581–2595, 2022.
[41] H. Liu, X. Long, Z. Li, S. Long, R. Ran, and H.-M. Wang, "Joint optimization of request assignment and computing resource allocation in multi-access edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1254–1267, 2023.
[42] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.
[43] L. Zhang, Y. Deng, W. Zhu, J. Zhou, and F. Wang, "Skewly replicating hot data to construct a power-efficient storage cluster," *Journal of Network and Computer Applications*, vol. 50, pp. 168–179, 2015.
[44] J. Li, Y. Deng, Y. Zhou, Z. Wu, S. Pang, and G. Min, "Tadrp: Towards thermal-aware data replica placement in data-intensive data centers," *IEEE Transactions on Network and Service Management*, 2023.
[45] Y. Ma, W. Liang, M. Huang, W. Xu, and S. Guo, "Virtual network function service provisioning in mec via trading off the usages between computing and communication resources," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2949–2963, 2022.
[46] L. Cheung, P. C. Cheung, and V. E. Ooi, "Antioxidant activity and total phenolics of edible mushroom extracts," *Food chemistry*, vol. 81, no. 2, pp. 249–255, 2003.
[47] P. Glynn and L. D'croz, "Experimental evidence for high temperature stress as the cause of el niño-coincident coral mortality," *Coral reefs*, vol. 8, pp. 181–191, 1990.
[48] M. Séguéla, R. Mokadem, and J.-M. Pierson, "Energy and expenditure aware data replication strategy," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. IEEE, 2021, pp. 421–426.

Peng Chen received the Ph.D. degree in computer science and technology from Fudan University, China, in 2024. He is currently a lecturer at the School of Software, Nanjing University of Information Science and Technology. His research interests include trustworthy AI, edge computing, federated learning, and fintech.

Mengke Zheng is a M.D. student at School of Computer Science, Fudan University. His research interests are distributed computing and edge computing.

Xin Du is an Assistant Professor at School of Software Technology, Zhejiang University, China. He received a Ph.D. computer science degree from Fudan University in 2024. His research interests include distributed system, brain-inspired computing, and service computing.

Muhammad Bilal received the Ph.D. degree in information and communication network engineering from the School of Electronics and Telecommunications Research Institute (ETRI), Korea University of Science and Technology, Daejeon, South Korea, in 2017. In 2018, he joined Hankuk University of Foreign Studies, South Korea, where he is currently working as an Associate Professor with the Division of Computer and Electronic Systems Engineering. Since 2023, he has been a Senior Lecturer (Associate Professor) with the School of Computing and Communications, Lancaster University, United Kingdom. His research interests include the design and analysis of network protocols, network architecture, network security, the IoT, named data networking, blockchain, cryptology, and future Internet.

Zhihui Lu is a Professor at School of Computer Science, Fudan University. He received a Ph.D. computer science degree from Fudan University in 2004, and he is a member of the IEEE and China Computer Federation's Service Computing specialized committee. His research interests are cloud computing and service computing technology, big data architecture, edge computing, and IoT distributed systems.

Qiang Duan is currently a Professor of Information Sciences and Technology with Pennsylvania State University Abington College. His current research interests include network-edge-cloud convergence, cognitive and autonomous networking, and ubiquitous intelligence in the future Internet. He has published more than 100 research papers in these areas and served on the editorial boards of various research journals in the field of networking and distributed computing.

Xiaolong Xu received the Ph.D. degree in computer science and technology from Nanjing University, China, in 2016. From April 2017 to May 2018, he was a Research Scholar with Michigan State University, USA. He is currently a Professor with the School of Software, Nanjing University of Information Science and Technology. His research interests include edge computing, the Internet of Things (IoT), cloud computing, and big data.