Accepted Manuscript

Cost-effective replication management and scheduling in edge computing

Yanling Shao, Chunlin Li, Zhao Fu, Jia Leyue, Luo Youlong

PII: S1084-8045(19)30001-3

DOI: https://doi.org/10.1016/j.jnca.2019.01.001

Reference: YJNCA 2275

To appear in: *Journal of Network and Computer Applications*

Received Date: 6 May 2018

Revised Date: 25 December 2018

Accepted Date: 7 January 2019

Please cite this article as: Shao, Y., Li, C., Fu, Z., Leyue, J., Youlong, L., Cost-effective replication management and scheduling in edge computing, *Journal of Network and Computer Applications* (2019), doi: https://doi.org/10.1016/j.jnca.2019.01.001.

Cost-effective Replication Management and Scheduling in Edge Computing

Yanling Shao[1,2], Chunlin Li[1,3 *], Zhao Fu [3] , Jia Leyue [3] , Luo youlong[1]

[1] Department of Computer Science, Wuhan University of Technology, Wuhan 430063, P.R.China

[2] College of Computer and Information Engineering, Nanyang Institute of Technology, Nanyang, 473000, P.R.China

[3] State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System, Baotou City, Inner Mongolia ,014030, P.R.China

* Corresponding author:chunlin74@tom.com

Abstract

The high volumes of data are continuously generated from Internet of Things (IoT) sensors in an industrial landscape. Especially, the data-intensive workflows from IoT systems require to be processed in a real-time, reliable and low-cost way. Edge computing can provide a low-latency and cost-effective computing paradigm to deploy workflows. Therefore, data replication management and scheduling for delay-sensitive workflows in edge computing have become challenge research issues. In this work, first, we propose a replication management system which includes dynamic replication creator, a specialized cost-effective scheduler for data placement, a system watcher and some data security tools for collaborative edge and cloud computing systems. And then, considering task dependency, data reliability and sharing, the data scheduling for the workflows is modeled as an integer programming problem. And we present the faster meta-heuristic algorithm to solve it. The experimental results show that our algorithms can achieve much better system performance than comparative traditional strategies, and they can create a suitable number of data copies and search the higher quality replica placement solution while reducing the total data access costs under the deadline constraint.

Keywords: replica creation, data scheduling, replication management, edge computing

1. Introduction

According to HUAWEI [1], more than 100 billion terminals are interconnected by the Internet in 2025. And user intelligent terminals such as robots, cameras, and wearable devices, are becoming more and more popular. Consequently, there are huge volumes of data are continuously generated from the Internet of Things (IoT) sensor networks. It is more critical and challenging that data management [2] for real-time processing in various industrial workflow applications, such as VR/AR, face recognition, vehicle-to-vehicle communication, and online gaming.

Edge computing [3-4] brings cloud services to the edge of the network, in close proximity to IoT devices, so it has many advantages, such as low communication overheads and faster response time as a local computation. Nevertheless, the processing capability of edge computing is limited. Therefore, the collaboration between edge and cloud computing [5-6] shares their strengths, such as infinite shared storage and computing resources from the Cloud, low-latency data preprocessing of edge computing. The edge and cloud computing paradigms can provide a real-time and cost-effective promising way to deploy workflow applications. However, due to different access cost in edge and cloud systems, it also makes deeper complex to replication management and scheduling.

In order to achieve higher data reliability, data replication is commonly used in distributed file systems. For example, GFS [7] and HDFS [8] generally use static replication by creating 2-4 copies for each dataset. In addition, data replication is an effective technique to reduce network consumption and the total access time. Although data replication has advantages, it also has notably storage costs especially in edge clusters with limited resource constraint. Since the collaborative edge and cloud is a highly dynamic environment, maintaining an optimal data replica distribution implies that the cloud service must be able to modify the number of data objects. The effective replica creation strategies are required to realize the goals of data reliability and system performance without consuming undue amounts of storage and bandwidth resources.

Some researches on data creation and scheduling strategy [9-18] have been proposed to manage data replica and reduce data transfer costs in distribute computing systems. However, very few of studies consider sensitive-delay workflows in a collaborative edge and cloud computing environment. Unlike previous techniques, we study data replica creation and scheduling problem for workflows in collaborative edge-cloud environments. In this work, a novel dynamic data replication management framework is designed in a collaborative edge and cloud computing system. Considering the popularity of data blocks and node load, we propose the dynamic replica creation methods. And then we formulate data replica placement problem for workflows and propose the data scheduling algorithm based on ITÖ algorithm to solve it. This study owns three-fold specific contributions as follows.

1) A data replication management system is designed for processing data-intensive and low-latency workflow applications in collaborative edge-cloud computing.

2) Considering the popularity of data blocks and node workload, we propose the dynamic replica creation methods. And we model the data scheduling for workflows as an integer linear programming problem, then the faster meta-heuristic algorithm is proposed to minimize the total data access costs while meeting deadline constraint.

3) The extensive experiments are conducted to compare our proposed DRSA algorithm with GA, PSO in edge-cloud computing. These results show that our data replication management system can be used to transfer data of workflows between the heterogeneous edge and cloud systems automatically, and our cost-effective data creation and scheduling algorithm can achieve much better system performance than these comparative traditional strategies.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents a dynamic replication management system and section 4 gives the replica creation and scheduling problem statement. Section 5 describes the data replica scheduling algorithm. In section 6 the algorithm is evaluated by extensive experiments. Section 7 makes the conclusion.

2. Related Work

Data replication and scheduling are becoming popular technologies to enhance data availability and fault tolerance in distributed systems. They have been widely used to address the issues of reducing data access time and bandwidth consumption. The related work is summarized in two aspects: **data replica creation** and **data scheduling strategies.**

Data replica creation. Azari et al. [9] presented Popular Groups of Files Replication (PGFR) algorithm to build a connectivity graph to recognize a group of dependent files in each data grid site and replicate the most popular groups of files to increase the local availability. Considering relationships among the files, Khanli et al. [10] proposed Predictive Hierarchical Fast Spread (PHFS) to predict future file requests and replicated the files in advance. Abdurrab et al. [11] proposed the FIRE strategy by analyzing the file access history of each grid site to discover the correlation among local jobs and files.

The cloud systems are heterogeneous in nature as different the data centers have different costs [20-22,40-43]. Gill and Singh [12] used knapsacks to optimize the cost of data replication and copy the copies from higher-cost data centers to lower-cost data centers. Matri et al. [13] introduced a set of algorithms and principles that were well suited for loosely coupled DHT and gossip storage systems. It allowed dynamic copying of popular data objects as close to the end user as possible to ensure the lowest possible delay. Assuming that the nodes were arranged in a ring and the data blocks were transmitted based on the unidirectional ring structure, Higai et al. [14] proposed optimization and heuristic two replication reconstruction schemes to minimize the data transmission costs of the data nodes.

Data scheduling strategies. In collaborative computing environments, data is no more locally accessible and it should be remotely stored and retrieved. Efficient and reliable data scheduling strategies in distributed computing environments have become a major concern for researchers.

To enhance data locality in the Hadoop cluster, Naik et al. [15] presented a data scheduler which allocated input datasets to the servers based on their processing capacity. Li et al. [16] combined the optimal placement of data blocks and the scheduling of tasks to reduce response time and improve user experience in edge computing. Tak et al. [17] secured an efficient data orchestration environment, which was suitable for KSTAR data flow. Cui et al. [18] proposed a genetic algorithm based data replica placement for workflows in Cloud. Shorfuzzaman et al. [19] used data access information for popular files and presented a distributed QoS-aware replica scheduling algorithm based on a fully dynamic programming method. Li et al. [20] adopted Particle Swarm Optimization (PSO) for developing a new multi-datacenter cost-effective data placement strategy. Ramaswamy et al. [21] estimated the utility of the data copy and quantified the costs and benefits of given data-item at given locations and designed a utility-based data placement scheme for cooperative edge cache networks. Shrivastava et al. [22] proposed an improved opportunistic data scheduling algorithm for VANET to improve energy efficiency as well as the throughput.

Different from the aforementioned work, our study presents a replica management system in edge computing. Considering data popularity, node workload and inter-dependencies between tasks, we study jointly replica creation and data scheduling to reduce data access cost while meeting delay-sensitive of applications and employ ITÖ algorithm [23-24] to search for optimal placement of data replicas of IoT systems.

3. Data Replication Management System

3.1 Replication Management Subsystem

We design a layered replication management subsystem [25-26] for reliable and efficient data scheduling in the collaborative edge and cloud computing system shown in Fig.1. And the framework includes two parts, the left side of the figure is the workflow business process, and the right side is the system's entire resource configuration. The three layers are composed of the user equipment, edge MDCs (Micro Data Centers) and Cloud data center. The edge MDCs are deployed in a base station and interconnected through the Internet. In addition to considering the performance of intra-cloud networks, the collaborative edge MDCs focuses on the performance of low-latency networks between the user equipment and edge servers with one-hop distance [27-29].

In the framework, we regard both of these computational and data scheduling processes as jobs. The data scheduling jobs are described in a different way than computational jobs with the specification language, so *Jobs Scheduler* can differentiate these two types of jobs. This work only focuses on data management, which consists of data processing components, such as *Jobs Agent*, *Data job scheduler*.

These jobs and their origin input datasets are sent by *jobs agent*, which deployed in the local front-end server. The functionality of *jobs agent* is to communicate with local *data job scheduler* and users. According to the information collected from the *computational scheduler* and from the resource agent, the *Data Job Scheduler* completely directs the creation, placement, movement and removement of data replicas. And these datasets are placed at the appropriate edge MDCs or remote cloud data center by data scheduling policy. *Data Job Scheduler* includes four parts: *Watcher*, *Replica creator*, *Replica Placement Optimizer and Data Removement.*

A *Watcher* keeps track of a range of data information for use in subsequent processes, which includes data access patterns by users, data location and data access frequency, etc. Once the data popularity exceeds the threshold set in the configure file, the *Replica creator* is triggered. *Replica Placement Optimizer* schedules data replicas to appropriate data servers and makes decisions about data movement between edge MDCs. In order to make enough storage space before data movements, some data must be removed. By reference to the literature [30], we adopt dataset cleanup algorithm plugged in *data removement* com-

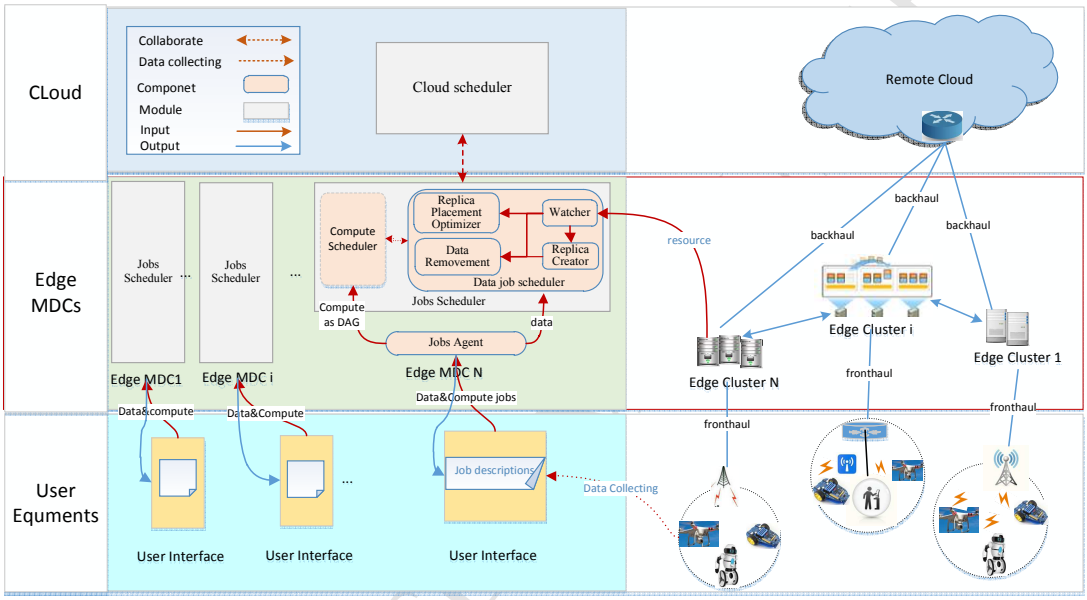


Fig.1. a layered replication management system in the collaborative edge and cloud computing.

ponent by dynamically remove the data files that are no longer required, and the *data removement* component is triggered periodically.

3.2 Data Security in the Data Replica Management System

Collaborative edge and cloud computing are surrounded by security issues，such as securing data. In our work, we focus on the semi-honest threat model [31]. That is, the edge-cloud platform is no longer honest, but may try to capture the user's private information. In the framework, we adopt a user data encryption model as Mylar [32] to solve the issue, which is transparent to service providers. The data is uploaded to the server after being encrypted by the client, and the decryption key is controlled by the user, and the service provider can only see the ciphertext, and the data control right is completely in the hands of the user. Privacy and security are protected by encrypting the data that the application sends to the server, thus the system can protect against malicious server administrators. Our data security mechanism consists of four components: the app plugin, user library, and server library and identity service as Fig.2 shows.

The app extension component is responsible for verifying that the digital signature of user data loaded from a server, and ensuring that has been tampered with. User library encrypts or decrypts the data which been sent to and from the server. The server library component makes search computation on

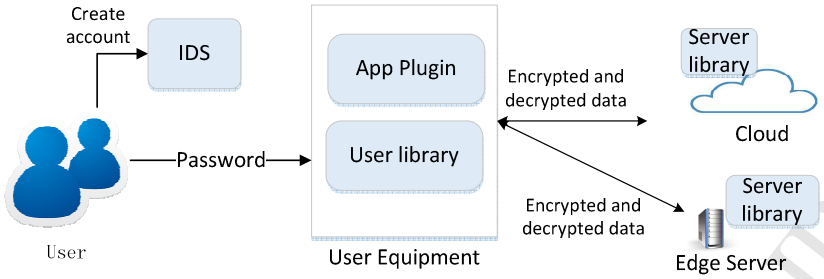


Fig.2. data security in a data replica management system.

encrypted data at the edge or cloud server. And Identity services verify that the public key belongs to a specific user.

3.3 A Use Case: Smart Industrial Control

There are some leading use cases using collaborative edge and cloud environments, such as smart industrial control, as Fig.3 shows. It can effectively reduce round-trip delay to support for low-latency control by deploying edge servers on IoT gateways. In addition, it can handle industrial data analytics by coordinating the use of ubiquitous cloud computing resources. We will illustrate our data replica placement study based on smart industrial control as the followings.

When the *startService* of a smart industrial control system starts to run, the component *Watcher* periodically records and updates system metrics of the collaborative edge and cloud environment. Industrial data analytic workflows are submitted randomly to the system by users continuously. The data generated from IoT sensors (such as equipment sensors, production lines sensors, etc.) are collected to a front agent by *User Interface* and then distribute to the jobs scheduler of the resource management system.

In particular, the workflow execution follows the control flow of tasks and data, and intermediate datasets are generated during the execution process. Therefore, the replica placement strategy has a great effect on the data access cost. In this paper, the proposed method dynamic created the appropriate replicated copies and searches for the optimal data placement solution to minimize data access cost while meeting user's low latency requirement. In order to realize factory sensor data processing quickly and

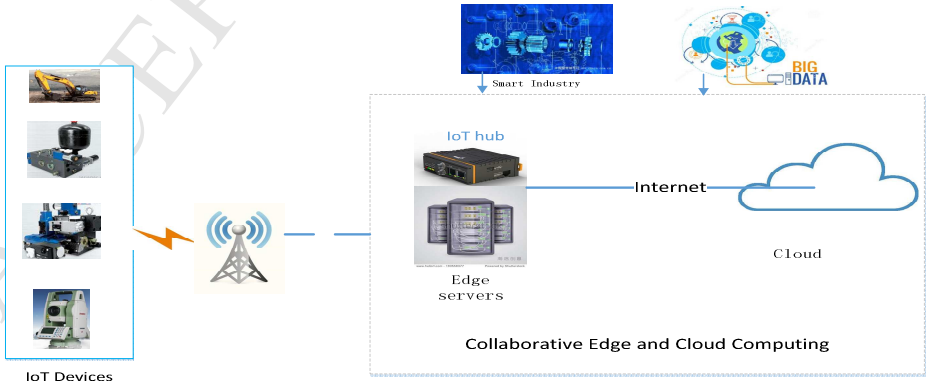


Fig.3. a use case: smart industrial control in collaborative edge and cloud environments.

real-time decision making, the *Data job scheduler* places these data replicas to the suitable computing nodes according to our proposed data scheduling optimal policy.

4. Replica Creation and Scheduling Problem

In an IoT-based system, there are huge volumes of incoming data from user devices. The workflow applications will consume these high volumes of datasets and generate new datasets. All these shared IoT data should be processed in the reliable and low-cost method. Furthermore, data replicas are created and distributed on multiple nodes of the network. The authorized users can submit jobs to the edge computing system. And the edge cluster responds to the request immediately if these datasets have been deployed locally. So the replica creation and data scheduling problem can be described as below.

For the edge servers with constrained capacity and the cloud datacenter with scalable resources, and the collaborative user groups, the workflow jobs are submitted by authorized users to the collaborative edge and cloud computing system and the shared data are submitted to our replication management subsystem. And the workflow jobs consume these huge volumes of the dataset and generate the intermediate datasets. On one hand, the generated intermediate datasets will be sent as input datasets to other tasks of the workflows following the rules of the control flows. On the other hand, the intermediate datasets and final results will be returned to the cooperative user groups. The goal of this study is to dynamically create a suitable number of data replicas and search optimal data placement solution to minimize the total data access cost while ensuring data reliability and meeting deadline constraint for delay-sensitive applications.

TABLE 1 NOTATIONS

Notations	Descriptions
$MDCC$	the edge and cloud computing environment
dn_k	the kth data node in $MDCC$
e_{ij}	a link between data nodes dn_i and dn_j.
IW	a workflow
WT	the set of tasks
WD	the collection of the dataset in WT
wd_{ij}	the j-th replica of the data wd_i
R	the number of a data copies
P (*)	the reliability of the data objects
pop_i	the i-th data file's popularity
D_H ,D_M ,D_L	three level subsets of data popularity
Lev_H ,Lev_M Lev_L	three level subsets of data popularity
g_h	the hth collaboration group
DSC	the total data access cost
C_{tc}	the data transmission cost
C_{sc}	the data storage cost
C_{mc}	the multicast intermediate dataset cost
WT_{IW}	the executive time of Workflow IW
$delay_k$	the unit cost of a replica stored on data node k.
x_{ijk}	the binary decision variable whether the data replica wd_{ij} is scheduled to server k
y_{ab}	the network distance from dn_a to dn_b.

4.1 Preliminaries

In this section, we provide some fundamental background used in the data replica management sys-

tem of edge and cloud environments. The main notations used in this study are listed in Table 1.

The workflows can be described as a Directed Acyclic Graph (DAG), which include a number of tasks connected through the dependency relationship of the task flow. Let each workflow IW be an ordered ternary set $IW=<WT, WC, WD>$, where $WT=\{wt_1, \dots, wt_m\}$ is the task set, and adjacency matrix WC represents the relationship of tasks of the workflow. WD indicates the collection of dataset in WC. And a task without immediate predecessors is known as an entry-task and a task without immediate successors as an exit-task.

For a data set WD in IW, it involves two types of data: an input dataset WD_{in} and an output dataset WD_{out}. $WD_{in}=\{wd_1,wd_2,...,wd_a\}$is an input dataset set, $wd_i \in WD_{in} =< size, R, wt_p, wt_{succ}, location, owner >$, the definition includes the *size* of the dataset, the number of copies R, preprocessing tasks wt_p, succeeding tasks wt_{succ} ,the deployed *location*, and the data *owner*. Here, $wd_i^{'} \in WD_{out}$ is as an intermediate dataset. A collaboration group g_k is a set of users who can share their outputs with each other.

The collaborative edge MDCs and Cloud system $MDCC$ may offer a set of servers with different access costs (*unit: time*) to execute workflow applications. The physical machine $dn_i \in MDCC$ are interconnected by Internet, E is the set of network links among these MDCs and/or between an MDC and a remote cloud C, U is as a set of front-end servers. And the identifier e_i is a link in E between data node i and j. We denote $B(e_{ij})$ as the bandwidth capacity of link e_{ij}.

4.2 Problem Formulation

4.2.1 Data replica creation formulation

In this paper, based on the data popularity and service performance of a data node, we propose a dynamic adjustment method for the replica number. And the method includes two-step process: 1) initialization of replicas number; and 2) The optimal number of replicas.

1) Initialization number of replicas

It is assumed that a data replica wd_{ij} of the data block wd_i is placed on a node n_j .Let P (wd_i=1) denote the reliability of the data block wd_i, Let $P(n_j=1)$ be the online probability of a node n_i, then $P(wd_i = 1) = 1 - P(n_1 = 0) * P(n_2 = 0) * ... * P(n_\gamma = 0) = 1 - \prod_{j=1}^{\gamma} (1\text{-}P(n_j = 1)) \ \forall i$. Let us define P ($WD=1$) to be the reliability of the data file WD, and P ($WD=1$)=P ($wd_1=1$) $* P$ ($wd_2=1$) $* ... * P$ ($wd_n=1$). To simplify the calculation, it is assumed that each node has the same online probability q. So the definition of P ($WD=1$) is shown in Eqn. (1).

$$P(WD=1) = \prod_{i=1}^{n} (1\text{-}\prod_{j=1}^{\gamma} (1\text{-}P(n_j = 1))) = (1\text{-}(1\text{-}q)^{\gamma})^{n} \tag{1}$$

When the data file is uploaded to the edge cluster, its importance parameter configured α is set by the user as the value of the range [0, 1]. The larger the value, the more important the data file. Thus, the original number of data copies γ can be calculated by the formula $P(WD) = (1\text{-}(1\text{-}q)^{\gamma})^{n} \geq \alpha$, and we can get γ as Eqn. (2) shows.

$$\gamma \geq \log_{1-q} (1\text{-}\sqrt[n]{\alpha}) \tag{2}$$

The value γ is the minimum value $(\log_{1-q} (1\text{-}\sqrt[n]{\alpha}))$ to ensure the reliability of the file system.

2) The optimal number of replicas

In a collaborative edge and cloud environment, there are a huge number of data nodes with a significant difference in performance. Once more hot data stored in the data nodes with poor performance, “hotspots” issues will arise. And the overall performance of the system will be degraded. In this

study, we propose a dynamic creation method for the number of replicas based on the data popularity and service performance of a data node, which can reduce the average access time while keeping lower storage cost. The detailed process of the method is as follows.

S1: Calculation of data popularity based on the access frequency of a data file

Set a five-tuple (pop_0,k_0,v_1,v_2,v_3) describe the data file's popularity. Different from the traditional calculation method, we count the number of data visits on three different periods T_1, T_2, T_3, and pop_0 represents data file's popularity on the last T_1 period, k_0 indicates the size of the data file in the last T_1 period, v_1,v_2,v_3 indicate the visits of a data file in the last statistical period T_1, T_2, T_3, respectively. Let f_1, f_2, f_3 represent the access frequency of a data file in these periods T_1, *T2, and* T_3. So $f_i = (V - v_i)/\Delta t_i \; i = 1,2,3$. Here, V represents the total number of times that a data file was accessed. Δt_i represents the difference between the current time and the corresponding time T_i. We design the parameter $\mu = \Delta t_1 / T_1$ to adjust the impact of burst access on the popularity of data blocks. When the value of the parameter is bigger, it indicates that the frequency of burst accesses is relatively large. The data popularity is denoted based on Eqn. (3) as following.

$$pop_i = (1-\mu)\, pop_0 + \mu(f_1 + (V_2 \times \frac{f_2}{f_1} + V_2 \times \frac{f_3}{f_1}) / T_1) \tag{3}$$

Then the average data popularity $pop_{avg} = \sum_{i=1}^{n} pop_i / total_f$, where n represents the number of data replicas, $total_f$ represents the total number of data files on the clusters.

The data popularity can be divided into three subsets: D_H ,D_M ,D_L, $D = D_H \cup D_M \cup D_L$,which stand for the hot, normal and unpopular set of data blocks. The rule of its level of data popularity for a data block is defined as follows. The data Files are sorted based on the value pop_i in descending order. Here, we set the average the popularity rate threshold β for measuring the popularity of data files. For any data file in the distributed system, the rule of its level of data popularity is defined as Eqn.(4).

$$wd_i \in \begin{cases} D_H & pop_i > pop_{avg}(1+\beta) \\ D_M & pop_{avg}(1-\beta) \le pop_i \le pop_{avg}(1+\beta) \\ D_L & pop_i < pop_{avg}(1-\beta) \end{cases} \tag{4}$$

S2: Workload Evaluation of Data Node Based on Service Performance

In this study, we consider that the number of data replicas is not only related to the access number of the data file but also related to the workload of the data node where the replica is located. Furthermore, the workload of the data node is depended on the service performance of the data node and storage utilization. Considering these factors such as CPU's utilization L_{cpu} , memory's utilization L_{mem} , and network's utilization L_{net} , the service performance N_{per} of data node is as $N_{per} = K_{cpu} \times L_{cpu} + K_{mem} \times L_{mem} + K_{net} \times L_{net}$ K_{cpu} K_{mem} and K_{net} is denoted the proportion of CPU, memory and network width, and $K_{cpu} + K_{mem} + K_{net} = 1$. Let se_i to represent the service workload of the i-th data node, $se_i = f_i / N_{per}$. Storage service load ds_i reflects disk utilization of the data node, $ds_i = used_i / tt_i$, where $used_i$ to denote the size of disk used, tt_i to describe the i-th data node's the whole space. Then the i-th data node's load can be defined as the following.

$$ld_i = \begin{cases} ds_i & (ds_i > 0.9) \\ ds_i \times sp_i & (\mathrm{ds}_i \le 0.9) \end{cases} \tag{5}$$

To determine the total load of the node, we use the average load as an assessment of the total node load as l_{avg}. Based on the workload of data nodes, the data nodes are divided into three categories: heavy,

moderate and light. For any node $node_i$, the classification rule of data nodes is as the follows:

$$node_i \in \begin{cases} Lev_H & ld_i > l_{avg}(1+\delta) \\ Lev_M & l_{avg}(1-\delta) \le ld_i \le l_{avg}(1+\delta) \\ Lev_L & ld_i < l_{avg}(1-\delta) \end{cases} \tag{6}$$

where Lev_H denotes heavy workload of data nodes, Lev_M denotes moderate workload of data nodes, Lev_L denotes light workload of data nodes. The identifier δ is as a threshold to measure data node's workload.

S3: Dynamic data replica creation based on data popularity and workload

Assuming that the total number of all data replicas is *M*, the data replicas are stored on the data nodes. In the current period, the corresponding workload of each data node is {ld_1, ld_2, ..., ld_p }. Considering data popularity and workload of a data node, the process of dynamic adjustment for copies is as the follows.

Condition1: If the popularity of a data object d_i is L_H and the workload level of the data node is Lev_H , the replica number m_i of data object d_i is as shown in formula (7):

$$m_i = \eta_1 \frac{M * pop_i}{pop_{avg}(1+\alpha)} + \eta_2 \frac{\sum_{i=1}^{M} ld_i}{l_{avg}(1+\delta)} \tag{7}$$

Condition2: If the popularity of the data object d_i is L_L and the workload level of the data node is Lev_L , the replica number of the data chunk m_i is as shown in formula (8):

$$m_i = \eta_1 \frac{M * pop_i}{pop_{avg}(1-\alpha)} + \eta_2 \frac{\sum_{i=1}^{M} ld_i}{l_{avg}(1-\delta)} \tag{8}$$

where, η_1 and η_2 are the influence factors of data popularity and data node workload, $\eta_1+\eta_2=1$ .Users can set different values according to their own preferences. In other cases, the replica number of the data chunk is still kept to γ .

4.2.2 Data replica scheduling model

Before the workflow execution begins to run, the original input datasets are allocated to the edge MDCs and the Cloud. Because of the higher cost latency of moving large-scale data to where the required tasks are, the primary goal of this work is to reduce the total data access cost between data nodes during the execution of the workflow. In a workflow management system, the data replica placement solution includes a dataset-node map and task-dataset map. For $\forall wd_{ij} \in WD$, it exists only one node to be stored. A dataset-node map $wd_{ij} \rightarrow dn_k$ means that a replica wd_{ij} is stored in a data node dn_k. This study focuses on the dataset-node map and data replica scheduling strategy [25], that is how to placement data replicas to the suitable data nodes and cooperate with workflow computational scheduling to reduce data access cost as well as meet deadline constraint.

Let *X* as the replica placement matrix, the element x_{ijk} is to 1 if the *j*th replica of data *i* should be placed at the *k*th data node in collaborative edge and cloud environments; otherwise, the x_{ijk} is to 0. The total data access costs *DAC* are accumulated by data storage cost C_{sc}, data transmission cost C_{tc} and multicast intermediate dataset cost C_{mc} [18, 19], that is $DAC=C_{sc}+C_{tc}+C_{mc}$. So the data scheduling problem can be formulated as (9-14).

$$\min(DAC(IW, MDCC)) \tag{9}$$

Subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{wd_i.R}(x_{ijk}\times wd_{ij}.size)\le c_k, \forall k, k=1,2,...,p \tag{10}$$

$$\sum_{k=1}^{p}\sum_{j=1}^{wd_i.R}x_{ijk}\le wd_i.\mathrm{R}, \quad \forall i, i=1,2,...,n \tag{11}$$

$$WT_{IW}\le DL \tag{12}$$

$$\sum_{i=user.location}\ \sum_{j=wd_k.location} wd_k.size \le B(e_{ij}), \forall a, \forall k \tag{13}$$

$$x_{ijk}\in\{0,1\}\ \ \forall i, j, k \tag{14}$$

where n is the number of datasets, p is the number of servers and c_k is the maximum available capacity of server k.

The aim of our work is to search for the optimal data scheduling solution for the workflow management in a collaborative edge-cloud environment under the objective function (9) with all resource and the deadline constraints (10-14). These constraints (10-11) define that the capacity of data node and replication factor should not exceed the set value. In constraint (12), DL is the time constraint for low-latency workflow execution. The binary decision variable x_{ijk} indicates whether the replica is scheduled to a data node, so 0-1 integer programming optimization method can be better suited to the problem.

The total data access costs are accumulated by data storage cost, data transmission cost and multicast intermediate dataset cost. We can give a brief discussion as the followings.

If a replica is stored in its local edge MDC, a task accesses it with a unit delay cost as tr_l. When it has to be placed at another edge MDC, the unit delay is to be tr_n, If that particular data replica is to be placed to the cloud data center, the unit delay is tr_c, $tr_l \le tr_n \le tr_c$. Let $delay_k$ be the unit delay cost of a data replica stored on the specified data node. Thus, the data storage cost C_{sc} for the workflow IW in $MDCC$ is described as Eqn. (15)

$$C_{sc}(IW, MDCC)=\sum_{i=1}^{n}\sum_{j=1}^{wd_i.\gamma}\Big(\sum_{dn_k\in MDCC}x_{ijk}\times delay_k\times wd_i.size\Big) \tag{15}$$

The data transmission cost is mainly influenced by the network distance and the network bandwidth between the data nodes where the task is executed and the location of the data replica. Obviously, before transferring data, it just needs to check whether the storage space of the target node meets the requirements. If there is not enough disk space, our data placement strategy will adjust the appropriate location to minimize cost under deadline constraints. And we defined the check variable *IMove*, if there are some enough disk for storing the data before moving to the data node, *IMove=1*, else *IMove=0*. Therefore, according to the data placement map, the total data transmission cost of the workflow IW in the collaborative edge and cloud environment $MDCC$ is calculated as below:

$$C_{tc}(IW, MDCC)=\sum_{j=1}^{m}\sum_{\substack{wd_i\in wt_j.WD_{in}\\ wt_j\in IW}} wd_i.size / B_{ab}\times y_{ab}\ \ \forall dn_a, dn_b\in MDCC \tag{16}$$

where m is the number of the tasks of the workflow IW, B_{ab} is the bandwidth and y_{ab} is the network dis-

tance from dn_a to dn_b.

The members in a cooperation group can share intermediate datasets with each other. And the output datasets are returned to related group g_h. And the multicast intermediate datasets cost of d_i during the execution of t_j is as the following Eqn. (17).

$$C_{mc}(IW, MDCC) = \sum_{j=1}^{m} \sum_{wd_i^{'} \in wt_j.out\_data} wd_i^{'}.size / B_{ih} \times y_{ih} + \sum_{user_a \in g_h} wd_i^{'}.size / B_{ah} \times y_{ah} \quad (17)$$

where, in (17) the first item denotes the output cost of the intermediate result from the source to the first hop router, and the second item denotes the output cost of the intermediate result from the first hop router to the users in the group g_h.

In addition, let us define WT_{IW} as the execution time of the Workflow *IW*, which is equal to the completion time of the final task wt_{exit}. So, let $EFT(wt_{exit})$ denote the finish time of last completed task. According to these references [33-34], thus, the execution time PT_G is described as the following

$$WT_{IW} = EFT(wt_{exit}) = EST(wt_{exit}) + \omega(wt_{exit})/\chi_k \quad (18)$$

where $\omega(wt_{exit})$ is weight to be assigned to task $wt_{exit,,}$ χ_k represents computing speed of the sever *k*.

4.3 The proof on the NP-Complete problem

In this section, we give the proof that *the data replica placement (DRP) problem is NP-Complete*.

Proof: 1) The DRP problem belongs to the class NP.

From the object function (9), we can learn that is the problem can be formalized as a decision problem. The variable x_{ijk} is to describe the placement of data replica wd_{ij}. And combined with the constraints (10)-(14), given a data placement scheme and a target cost, we can verify that placement results with minimal data access costs can be completed in polynomial time. According to the definition of NP, the *DRP* problem belongs to the class NP.

2) The DRP problem can be reduced to 0-1 integer linear programming (ILP).

The 0-1 ILP has been proved to be an NP-complete problem in the literature [35]. To prove that *DRP* is NP-Complete, we show that $DRP \leq_p$ integer linear programming (ILP) and an instance of the *DRP* problem can be reduced in polynomial time to any instance of 0-1 ILP.

The reduction procedure starts with an instance of 0-1 ILP. Set $X=\{x_1,x_2,...,x_n\}$ be the set of items each is associated with profit p_j and weight ω_j, where $j=\{1,2,...,n\}$, and set c_n as capacity, $x_{k=}\{0,1\}$. The *DRP* instance is satisfiable if and only if the 0-1 ILP instance is satisfiable. The satisfiability of the problem means that the decision-making problem is a "yes" answer. The instance of the *DRP* problem is constructed as follows:

For each placement decision-making variable x_{ijk} of *Y*, the instance of the *DRP* problem has an attribute A_j. Further, the cost $cost_j$ of an item x_{ijk} corresponds to the size of the attribute A_j. The cost of item x_{ijk} corresponds to the data access cost of all data objects *D* as given in the minimization problem of DRP. The data access cost can be computed as Eqn. (9), where x_{ijk}= 1 or 0 for A_j in the *DRP* problem under capacity, communication/bandwidth constraints. This instance of *DRP* problem can easily be reduced from the instance of 0-1 ILP in polynomial time. Therefore, from 1) and 2), we can prove that the DRP problem is NP-complete.

Because the DRP problem with large scale is proved to be an NP-Complete problem, the exact algorithm cannot solve the problem in reasonable CPU time. So the problem can be solved by heuristic or meta-heuristic algorithm. Thus, we adopt the ITÖ algorithm to solve it.

5. Data Replica Creation and Scheduling Algorithms

5.1 Data Replica Creation Algorithm

Different from traditional threshold-triggered copy creation strategies, this study, consider the status of both data block popularity and node loads to dynamically create copies of a data block. Furthermore, in order to avoid the impact of burst access to the data block, we use overall of the data block popularity

ALGORITHM 1: Dynamic Replica Creation Algorithm (DRCA)

Input: *MDCC, WD,*; M_{ini}: the set on data block copy's number before modifying;
Output: M_{ac}, the set on data block copy's number modified.

1:initialize *MDCC*, *WD*, M_{ini};
2: for data b_i in *WD* **do**
3: Statistics(b_i ,v_1,v_2,v_3);// Statistics on access frequency of data block b_i
4: *Reckonpop(* b_i,pop_0, δ *)by Eqn.(3) ;*
5: *Classifyhot(*b_i, *pop*,L_H,L_M,L_L*)by Eqn.(4)*
6: **for** *data node* dn_j *in MDCC do ;*
7: **if** b_i *is located in* dn_j*;*
8: **then** *Reckonload*(dn_j,ld_j) by *Eqn.(5)*;
9: *Classifyload(* dn_j, ld_{avg}, Lev_H, Lev_M, Lev_L*)* by *Eqn.(6);*
10: **endif***;*
11: **if** $b_i \in D_H$ **and** $dn_i \in Lev_H$
12: $M_{ac} \leftarrow Comput1$ by *Eqn.(7)*
13: **else if** $b_i \in D_L$ **and** $dn_i \in Lev_L$
14: $M_{ac} \leftarrow Comput2$ by *Eqn.(8)*
15: $M_{ac} \leftarrow M_{ini}$
16: **end for***;*
17:end for
18:output M_{ac}

and the access frequency of three historical periods to predict the required copies of a data block in the next cycle. The proposed dynamic replica creation algorithm (*DRCA*) is described in Algorithm 1.

5.2 Data Replica Scheduling Algorithm

We present the *D*ata *R*eplica *S*cheduling *A*lgorithm (DRSA) based on ITÖ algorithm [23-24] designed by professor Dong. The new algorithm is proposed to abstract and simulate particle motion and their interactions based on Ito's stochastic process. In view of the two characteristics of the Ito stochastic process, namely random volatility, and drift, the random volatility represents the local performance of the particle, while the drift represents the macroscopic trend of the particle. The corresponding volatility operator and drift operator are designed in ITÖ algorithm. The volatility operator is used to control the exploration characteristics of the particle, which can make the particle locally perturbation in the neighborhood. The drift operator is used to control the moving direction of the particle so that the particle moves macroscopically toward the attractor.

We encode a replica placement solution into binary serialization [18] and calculate its fitness by Eqn.(16). All data replica placement solutions in the solution space can be encoded using a fixed length. According to [23-24], the update of the new data scheduling location is shown in Eqn. (19).

$$X_{id}^{'} = X_{id} + \varphi_{id}(r,T)(p_{pd} - X_{id}) + \varphi'_{id}(r,T)(X_{id} - X_{kd}) \tag{19}$$

where p_{pd} as an attractor,which is the global optimal solution, $\varphi_{id}(r,T)(p_{pd}-X_{id})$ as drift term, $\varphi'_{id}(r,T)(X_{id}-X_{kd})$ as volatility term. And $\varphi_{id}(r,T)$ as drift operator, $\varphi'_{id}(r,T)$ as volatility operator.

These two operators depend on particle radius and annealing temperature. In the process of finding the optimal copy placement position, we balance the volatility operation and drift operation by adjusting the different values of $\varphi_{id}(r,T)$ and $\varphi'_{id}(r,T)$. Thereby, the probability of drifting to the global best solution is guaranteed, and the diversity of solution is guaranteed, so as to avoid falling into a local optimum.

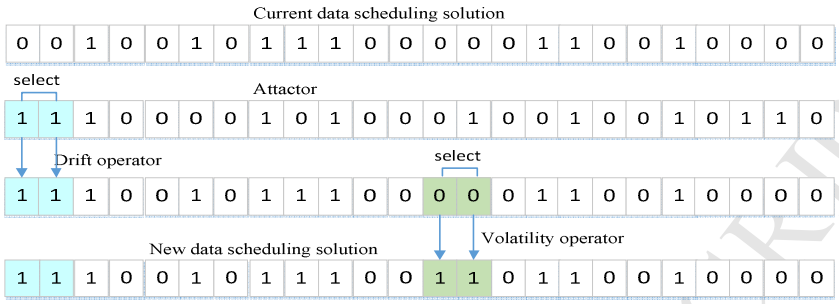


Fig.4. the operations of the DRSA algorithm on the data scheduling solution.

Figure 4 shows the operation of the DRSA algorithm on the data scheduling solution. Furthermore, we propose DRSA for data replica placement as Algorithm 2 in detail.

ALGORITHM 2: Data Replica Scheduling Algorithm (DRSA)

Input: *MDCC,* $WT=\{wt_1,wt_2,...,wt_m\}$, $WD=\{wd_1,wd_2,...wd_n\}$, $g_k=\{user_1,user_2,...,user_u\},1\le k\le K$
Output: ***X,*** replica placement solution set

1: initialize T_0 , α , r , β $maxGen$, $curGen$, M , MAX_NO_UPDATE , $Deadline$;
2: **while** (a new request arrives)
3: read the data set of the workflow
4: produce M different random valid particles' solution by *probability generating model*
5: **while**((*curGen++* ≤ *maxGen*)|| *noUp*<*MAX_NO_UPDATE*||WT_{IW}<=*DL*))
6: Sort all particles with fitness in ascending order;
7: *curBestSolution* ← *particles[0].solution*, *Solution* ← *curBestSolution*
8: **if(**Constraints in Eqn.(10)-(14) are satisfied) Calculate data access cost by *Eqn.(9)*
9: *Solution.fitness=Total data access cost*
10: *curBestSolution.fitness=Solution*
11: **if**(*Solution.fitness* <*curBestSolution.fitness*)
12: *noUp* ← 0,*olution* ← *curBestsolution* ;
13: **else** *noUp++;*
14: **end if**
15: update all particles activity ability by (19);
16: **for** *j*=1 **to** *M* **do**// for each particle
17: make drift and volatility operators
18: *newSolution* ←construct a feasible solution by a probability generating model;
19: **if (***newSolution.fitness*: < *particles*[*k*].*fitness)* *particles*[*j*] ← *newSolution*;
20: **else if (**rand() < 0.7) *particles*[*j*] ← *newSolution*;
21: **end if**
22: **end for**
23: **end while**
24:**end while**
25: **return** *Solution.X* ;

The first line initializes the population size, termination conditions, and algorithm parameters. Steps 2-23 are the procedure of dealing with the arriving request. Steps 3-4 read workflow data and generate the initial population. Steps 5-22 are the iterative process of the algorithm, and the particles will perform drift and volatility operators to find the optimal solution. Step 6 classifies the particles according to their fitness. The purpose of steps 7-14 is to find the current optimal solution and the number of unchanged times in the process of updating the optimal solution. Step 15 is to update the particle radius, ambient temperature, drift and volatility strength of the algorithm. Steps 16-22 make drift and volatility operators for all the particles to build a new solution. Once meeting the stop condition, if the new position is better than the current optimal particle position, replace the particle. In this way, the optimal data replica placement solution is achieved.

In addition, we make further analysis of the time complexity of the DRSA algorithm. It is related to the size of the population, the iteration stop condition. The number of particles is the same as the size of the problem instance. The shutdown condition is that the continuous 20-generation algorithm stops without improvement. For each iteration, the length of particle's solution is related to the gene length of the solution, which is related to the size of the task of each workflow, the number of data copies, and the number of data nodes. Therefore, the DRSA algorithm runs in polynomial time.

5.3 Convergence Analysis of *DRSA* Algorithm

According to [23-24], the almost sure convergence of our algorithm is to be proved in this section. Each particle moves following by its own motion law. So we can analyze each particle independently and then combine them into the performance of the whole algorithm. We can show the analysis and give the proof as followings.

Given the population in the DRSA algorithm $X = (x_1, x_2, ..., x_N)$, where $\forall i, x_i \in X$ the particle in the population and N is the population size. The maximum fitness value of the population at the t-th iteration is $f(X^{(t)}) = \max(f(X_i^{(t)})), X_i^{(t)} \in X^{(n)}$, and the average fitness value of the population at the t-th iteration is $\overline{f}^{(t)} = (1/n)\sum_{i=1}^{n} f(x_i^{(t)}), x_i^{(t)} \in X^{(t)}$,and set global optimal solution as $G = \{x \mid \forall y \in X, f(y) \le f(x)\}$.The t-th generation of the DRSA based on ITÖ algorithm almost sure converges to the global optimal solution, if $P(\lim_{t\to\infty} X^{(t)} \subseteq G) = 1$, short as $X^{(t)} \xrightarrow{s.a.s} G$.

From the flow of DRSA algorithm, the t-generation population of the algorithm depends only on the t-1th generation population, it has no relationship with other populations in the past, and the transition probabilities of each generation are different from each other, so the behavior of DRSA algorithm is a non-homogeneous Markov chain. There are five steps in the transition from $X^{(t)}$ to $X^{(t+1)}$ as the following.

$$X^{(t)} \xrightarrow{attractors} U^{(t)} \xrightarrow{drift} Y^{(t)} \xrightarrow{select} Z^{(t)} \xrightarrow{volatility} W^{(t)} \xrightarrow{select} X^{(t+1)}$$

Next, we analyze the transition probability of each step.

1) The first step is to select an attractor. An attractor is a particle that is attractive to the particles. If the optimal solution in the population is taken as the attractor, the probability of selecting the attractor in the population according to the reference [23] can be described as:

$$P_A^{(t)}(U^{(t)} = u \mid X^{(t)} = x) = \prod_k P_A^{(t)}(x(k), u(k))$$

2) The second step is the drift operator. The drift operator is a random operator from X^2 to X. According to the reference [24], the drift probability of the population is as below.

$$P_M^{(t)}(Y^{(t)} = y \mid U(t) = u) = \prod_k P_M^{(t)}(u(k) \times a(u(k)), y(k))$$

The function of a drift operator is to speed up the improvement of the particle. The operator does not have global reachability and has little significance for the convergence of the algorithm.

3) The selection operators in steps 3 and 5 play the same role, that is, only new particles are improved, they are selected, and the method of selection is independent of the iterative process. The transition probability of step 3 is as below.

$$P_S(\mathrm{Z}^{(t)} = z \mid Y^{(t)} = y) = \prod_k P_s^{(t)}(y(k), z(k))$$

The transition probability of step 5 is as below.

$$P_S(\mathrm{X}^{(t+1)} = x \mid \mathrm{W}^{(t)} = w) = \prod_k P_s(w(k), x(k))$$

4) The fourth step is a volatility operator. From the definition of the operator, if the intensity of the drift is not zero, then given any state, the operator can reach. According to the reference [X], the minimum transition probability is as the followings.

$$P_W^t(i,\ j) = \prod_k \gamma_{\min} / (n_k - 1) = e$$

The transition probability of the entire population is:

$$P_W^t(W^{(t)} = w \mid Z^{(t)} = z) = \prod_k P_w^t(z(k), w(k)) \geq e^N$$

Therefore, the transition probability of the non-homogeneous Markov chain $\left\{X^t, t = 1, 2, ...\right\}$ is:

$$\begin{aligned} P(X^{(t+1)} = x' \mid X^{(t)} = x) = \sum_u (P_A^{(t)}(U^t = u \mid X^{(t)} = x) \times \\ (\sum_y (P_M^t(Y^t = y \mid U^{(t)} = u) \times \\ \sum_z (P_S(Z^{(t)} = z \mid Y^{(t)} = y) \times \\ \sum_W (P_w^t(W^t = w \mid Z^{(t)} = z) \times \\ P_s(X^{(t+1)} = x' \mid W^t = w)))))) \end{aligned}$$

Given the vector $F(x) \equiv (f(x(1)), f(x(2)), ..., f(x(N)))^T$, if $F(x) \leq F(x')$ then $P(X^{(t+1)} = x') \mid X^{(t)} = x) \geq e^N$, else $P(X^{(t+1)} = x') \mid X^{(t)} = x) = 0$.

For a single particle $\forall x \in X$,

$$E(f(X^{(t+1)}) \mid X^{(t)} = x) = \sum_{x'} P(X^{(t+1)} = x' \mid X^{(t)} = x) \times f(x') \geq f(x) \times \sum_x P(X^{(t)} = x) = f(x)$$

Since each particle has such a feature, we have the following conclusions:

$$E(\overline{f}(X^{(t+1)}) \mid X^{(t)}) \geq \overline{f}(X^{(t)})$$

Based on the above analysis, we can get the following conclusions.

Lemma 1. The stochastic process formed during the Ito iteration process is a non-negative bounded sub-martingale and it has a limit θ.

Lemma 2. For the 2n+1 segment directed graph model constructed for combinatorial optimization

problems, each particle in the population $\forall x \in \Omega_k, k < M$ has the following conclusions, $\lim_{t\to\infty} P(X^{(t)} \in \Omega_k \mid X^{(0)} = x) = 0$ and $\lim_{t\to\infty} P(X^{(t)} \notin \Omega_k \mid X^{(0)} = x) = 1$.

Proof: According to transition probability of a particle of DRSA, due to $\forall x \in \Omega_k$, once it is transferred out, it must move in the direction of the direction, and will never return Ω_k. So we can get the result as the following. Here $\gamma_{\min}, \gamma_{\max}$ describe the maximum and minimum volatility intensity of the particles, respectively, and $\gamma_{\max}$ =1-$\gamma_{\min}$.

$$P(X^{(t)} \in \Omega_k \mid X^{(0)} = x) = \prod_{j=1}^{t} P(X^{(t)} = x \mid X^{(0)} = x) \le \prod_{j=1}^{t} (1-\gamma_{\min})^n = \prod_{j=1}^{t} \gamma_{\max}^n$$

Due to $\gamma_{\max} < 1,$ so we can get $\lim_{t\to\infty} P(X^{(t)} \in \Omega_k \mid X^{(0)} = x) = 0,$ and $\lim_{t\to\infty} P(X^{(t)} \notin \Omega_k \mid X^{(0)} = x) = 1$.

Lemma 3. The limit of the stochastic process formed $\{\overline{f}(X^{(t)}), t = 1, 2, ...\}$ during the ITÖ process iteration is θ, and $\theta = f_M(a.s)$.

Proof: Since $\lim_{t\to\infty} P(\overline{f}(X^{(t)})) = \theta$, for each particle, there is $\lim_{t\to\infty} P(f(X^{(t)})) = \theta$. So we can focus on to analyze the limits of a particle.

If $\theta \neq f_M$, it can say $\forall x \in \Omega_\pi$, $\lim_{t\to\infty} P(X^{(t)} \in \Omega_k \mid X^{(0)} = x) = 1$, this contradicts the conclusion of Lemma 2. So we must get the result $\theta = f_M(a.s)$.

Obviously, our proposed DRSA algorithm is based on swarm intelligence algorithm, which is appreciated by biological behavior so also known as the biomimetic optimization algorithm. However, these swarm intelligence algorithms are all directional and iterative methods based on probabilistic search. Therefore, they have poor real-time performance. In order to improve our DRSA algorithm performance to meet the low latency of the workflows, on one hand, we adopt a smaller radius of the particle to strengthen the local search ability to quickly obtain the optimal solution. On the other hand, we adopt the current global best solution in the current iteration as a final choice once the current execution time of a workflow job is equal to or greater than the deadline.

6. Performance Evaluation

In this section, our data replica creation and scheduling strategy for the workflows are validated in a simulated edge and cloud environment.

6.1 Experimental Environment

(1) Experiment Settings

Our experimental environment is the Eclipse Java EE IDE on a PC with an Intel(R) Core(TM) i5-5300 CPU @2.30 GHz and 8.0 GB of RAM. Each simulation runs for 10 minutes and repeats 25 times. We show the average statistics result and the confidence interval for a confidence level of 95% in the Student's t-distribution. The deadline [33, 36] for a workflow is *k* times of the average optimal running time, and the range for *k* is [3, 20], and it can get from the specific configuration file.

We use the information gathered from actual executions of workflows (such as Montage, CyberShake, Epigenomics, LIGO inspiral analysis and SIPHT [37-38]) to generate realistic, synthetic workflows. The structures and characteristics of these five workflows are shown in Fig.5 and Table 2. These workflows are developed by Pegasus to construct workflows in abstract terms for different domains like bioinformatics and astronomy. The types of the dataset used are images data from Montage, earthquake hazards data from CyberShake, genome sequence data from Epigenomics, gravitational waveforms data from

LIGO and bioinformatics from SIPHT. We select the small-size of each workflow with about [3,16] random number of tasks. According to [18], the size of each replica varies from 17GB to 26GB, and the storage of each edge computing node is 1T. Our simulator can automatically locate the necessary input data and computational resources necessary for workflow execution.

These jobs are coming randomly followed a Poisson distribution, the key request rate parameter *r* represents the average number of jobs randomly arrival within one minute intervals and set *r* as 3 requests per minute.

TABLE 2 THE CHARACTERISTICS OF EACH DATASET

Name	Workflow Type	Characteristics
Montage	the images on science-grade mosaics of the sky	I/O-bound
Epigeomcs	the DNA sequence genomic data	CPU-bound
SIPHT	the bioinformatics data in bacteria	small-scale tasks
CyberShake	the earthquake hazards data	data-intensive
LIGO	the gravitational waveforms data	large memory requirements

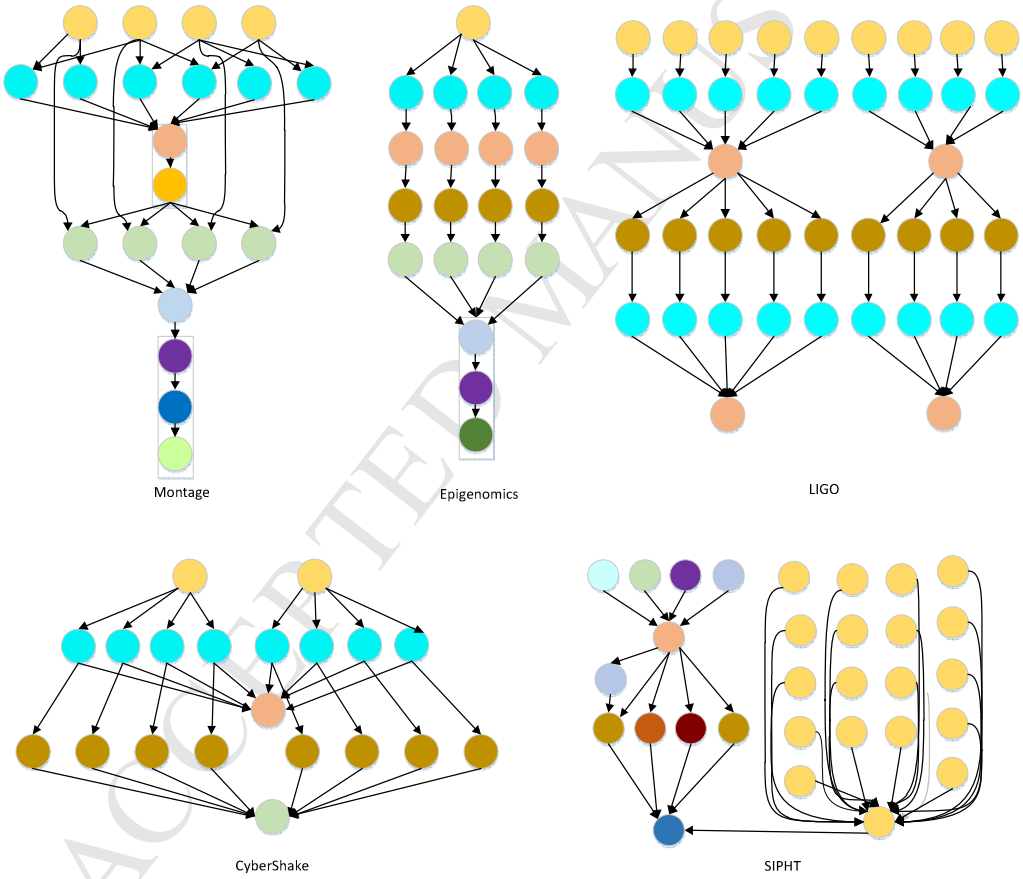


Fig. 5. the structure of five realistic scientific workflows [37-38].

6.2 Experiment Result

6.2.1 The validity evaluation of our dynamic replica creation strategy

The part includes two sets of simulations: 1) the effectiveness evaluation of the replica creation

method on the data blocks' popularity and 2) the relationship between the number of data replicas and file reliability.

(1) The effectiveness of the replica creation on the data blocks' popularity

We use the data file with the size of 1024M, which is divided into four data blocks, which described as *A*, *B*, *C*, and *D*. In the initial stage, the average data access frequency is set to 100 times per hour. And we set T_1, T_2, T_3 as 2, 1 and 0.5 hours, respectively. The importance parameter of a data file is set as 0.8. According to Eqn. (2), we can calculate the minimum γ of data block copies to ensure the file reliability and get the value as 2.73. Because the number of the block copies is an integer, so γ is rounded up and we get the value as γ=3. We submit the read-only access requests to data blocks *A*, *B*, *C* and *D*. In this simulation, assume that the access to the data block *A* tends to progressive increase, the access to the data block B keeps stable, the access to the data block *C* tends to progressive decrease, and the access to the data block *D* tends to burst.

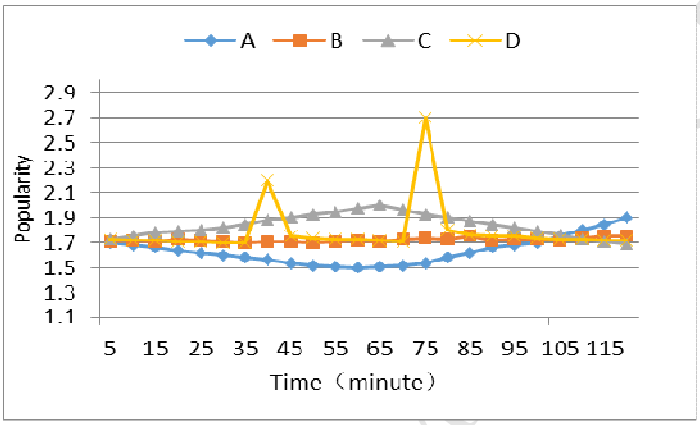


Fig. 6. the popularity trend of the data blocks at different access frequencies.

The result of the test is shown as Fig.6. On the premise of the same average access frequency, the final popularity of data block *A* is higher than that of data block *B*. The reason is that the access frequency to data block *A* is continuously increased while the data block *B* keeps stable access frequency. And as the access frequency to *C* is constantly descending, its final popularity is lower than *B*. So we can know that the calculation method of the data block popularity in this paper is an access frequency-sensitive algorithm. At the same time, the data block *D* has a sharp increase of more than 10 times the access frequency during the statistical period, but its popularity change range is very low, and can quickly back to a lower popularity value. The results show that the calculation method of the data block popularity in this paper can not only reflect the access rules of data blocks but also solve the problem of sudden access to data blocks.

(2) The relationship between the number of data replicas and file reliability

According to Eqn. (1), we know that the number of file partitions, the number of data replicas and the online rate of nodes all affect the availability of files. In the test, let the file partitions as 3, the online rate of the nodes is set to 0.1, 0.3, 0.5, and 0.7, respectively. The result of the test is shown as Fig. 7. In the case of the online rate of nodes as 0.5, we set the number of file partitions as 3, 5, 7, and 9, respectively. The result of the test is shown in Fig. 8.

According to Fig. 7 and Fig.8, when the number of data block copies is lower (<=5), an increase in the number of copies will lead to a rapid increase in file reliability. In addition, the greater the online rate of a node and the smaller the number of partitions of a file, the faster the file reliability converges to 1.

However, from Fig.8, we can see when the number of data copies is equal to 7, the reliability of the file is already close to 1. And the reliability of the file does not change when the number of copies is increased again. Therefore, when the number of data block copies is dynamically created, a suitable number of data block copies should be created as far as possible to meet the needs of users of saving storage resources and reduce user overhead.

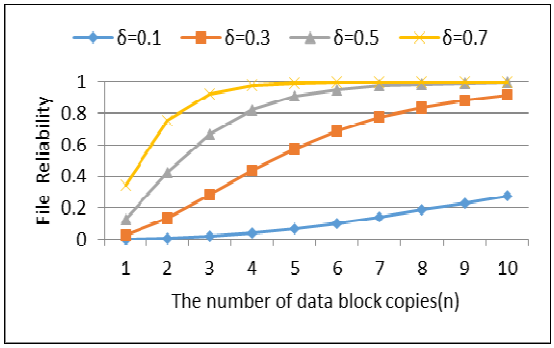


Fig.7. the relationship between file reliability and the number of data copies with different node online rates.

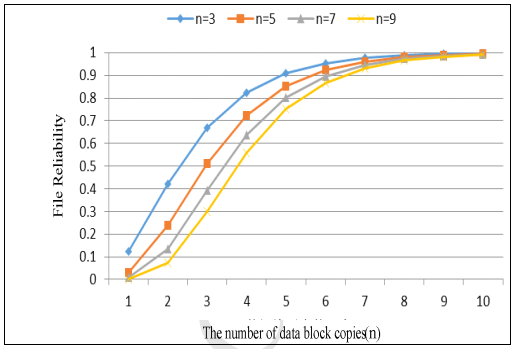


Fig.8. the relationship between file reliability and the number of data copies with different file partitions (n).

6.2.2 The effectiveness of the data scheduling algorithm

In this section, to evaluate the effectiveness of the data scheduling algorithm, we perform extensive experiments and compare the performance of DRSA with CUI's [18] strategy based on Genetic Algorithm and PSO in [34]. We compare our DRSA strategy with GA, PSO in three aspects: 1) the different number of edge MDCs, and 2) the different size of the datasets and 3) the different scale of workloads of in the edge and cloud system.

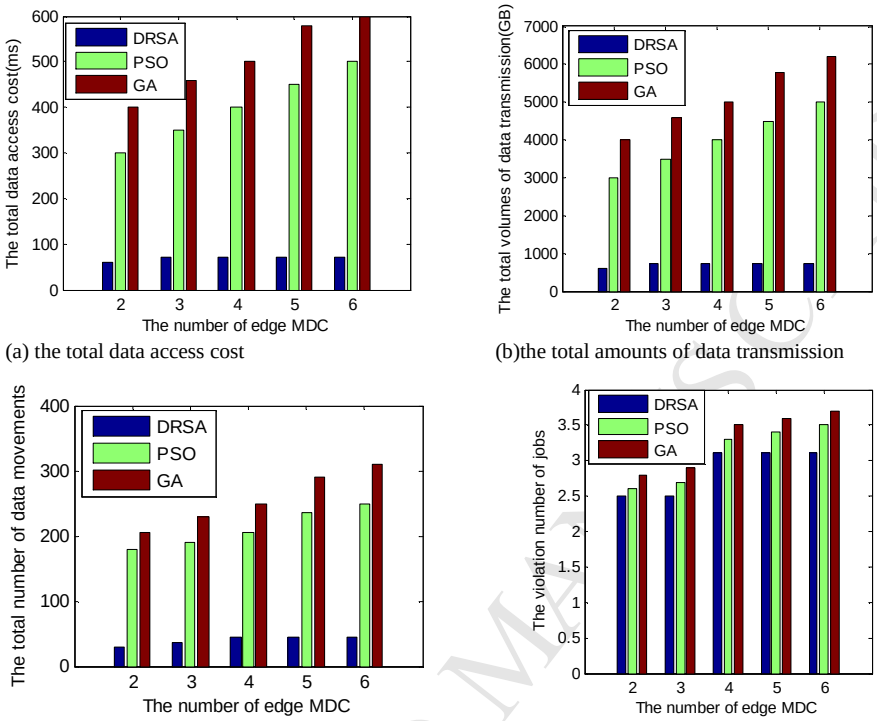
Refers to these study [18-19], we validate the effectiveness of the four metrics in the experiments: C_{access}, the total data access cost; T_{all}, the amounts of data transfer; N_{tran}, data movements times, $N_{t>deadline}$, The number of violations of the workflow that exceeded the due date. The data storage cost parameters (unit:ms) such as tr_l, tr_n, and tr_c, refer to these works in [39], the corresponding values are 1,6, and 63ms. And according to [24], the most parameters of meta-heuristics ITÖ are set as: the size of the population $M=30$,particle movement ability $\alpha=1.4$, annealing rate $\beta=0.9$, the initial temperature T_0=1000, The maximum times of no updating Max_NO_UPDATE=20 and the maximum iteration numbers=100.

(1) The different number of edge MDCs

According to references [8] and [18], the number of data replica is set to fixed value 3 and the number of the user in every group changes from 10 to 25 randomly. We process the workflow requests with the arriving rate 3 reqs/min. The number of origin datasets is set to 90. In this set of evaluations, we set different number of edge clusters for performance comparison analysis. And the number of edge clusters is set to 2, 3, 4, 5, and 6, respectively. Each cluster consists of three edge compute nodes. The number of front-end servers is the same as the number of edge clusters. We use mean, confidence interval and the standard deviation of the simulation results as Table 3 and 6.

Figure 9 plots the comparison curves for the four strategies on the total data access cost, the total amounts of data transmission and the total number of data movements under the different number of edge clusters. As is shown in Fig.9, the results increased with the number of edge cluster increasing in all three strategies. With the increasing of the number of edge clusters, the performance on four metrics of our ITÖ strategy is much better than Genetic and PSO. In the current round of experiments, the per-

formance difference between GA and PSO is not obvious, and PSO is slightly better than GA. The reason is that the PSO algorithm has better than GA in real-time performance, calculation accuracy and convergence speed in the case of small-scale problems. This study has designed the suitable drift and volatility operators of ITÖ algorithm to make it suitable for data replica placement problem for the workflow via the online model. Therefore, it can be seen clearly that our proposed DRSA strategy significantly reduces the data access costs compared to another two strategies.



(a) the total data access cost

(b)the total amounts of data transmission

(c) the data movement times

(d) the violation number

Fig.9.The four metrics: C_{access}, T_{all}, N_{tran} and $N_{t>deadline}$ with a different number of edge MDCs.

(2) The different size of the datasets

Each application selected is with the different size of data workload in these simulations. That is, the application's workload is changed by setting a different number of datasets. And the number of each group of datasets is set to 70, 90, 110, 130 and 150 for each test. And of course, the number of tasks of the application is increased with increased workloads. The fixed number of edge computing nodes is set to 3 in edge clusters. We use mean, confidence interval and the standard deviation of the simulation results as Table 4 and 7.

We can see from the Fig.10, as the dataset workload increases, the results show that the total data access cost under DRSA is significantly lower than GA and PSO effectively, too. As the size of the workflow workload grows, the total access cost of the random strategy increases much faster. And GA and PSO strategies show increases steadily. And in the next two tests, data movement times and the total amounts of data transmission have the same tendency with the total data access cost. Compared with GA, PSO, our DRSA strategy clearly has a greater advantage. As the amount of dataset increases, the scale of the problem becomes larger. When the number of datasets is 130, the performances on the four metrics of GA are better than PSO. This is because as the increase in the scale of the problem, the PSO algorithm is easy to fall into a local optimum.

TABLE 3. THE TOTAL DATA ACCESS COST MEAN, CONFIDENCE INTERVAL(CI) AND STANDARD DEVIATION(SD) UNDER DIFFERENT EDGE CLUSTER

edge cluster	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
2	60.4	[58.99,61.81]	3.42	399.4	[396.57,402.15]	6.77	299.84	[298.17,301.51]	4.03	4180.52	[4154.11,4206.92]	63.97
3	69.44	[68.49,70.39]	2.29	461.56	[459.53,463.59]	4.93	351	[346.53,355.47]	10.83	4592.04	[4563.70,4620.37]	68.65
4	71.08	[70.05,72.11]	2.5	501.24	[498.97,503.51]	5.51	400.16	[397.52,402.80]	6.39	5437.84	[5410.11,5465.56]	67.16
5	72.04	[70.91,73.17]	2.73	583.12	[579.56,586.68]	8.62	450.08	[445.05,455.11]	12.19	6012.16	[5986.73,6037.58]	61.59
6	75.04	[73.98,76.10]	2.56	621.2	[618.25,624.15]	7.15	500.32	[496.82,503.82]	8.49	6508.84	[6479.72,6537.95]	70.53

TABLE 4. THE TOTAL DATA ACCESS COST MEAN, CONFIDENCE INTERVAL (CI) AND STANDARD DEVIATION (SD) UNDER DIFFERENT DATASET

dataset	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
70	60.4	[58.95,61.76]	3.4	409.3	[409.09,413.47]	10.1	299.5	[297.53,301.43]	4.7	4160.5	[4136.43,4184.61]	58.4
90	69.6	[68.59,70.52]	2.4	459.2	[439.65,478.75]	47.4	326.2	[322.84,329.64]	8.2	4592.0	[4563.70,4620.37]	68.7
110	91.4	[90.22,92.5]	2.8	601.6	[596.84,606.28]	11.4	400.2	[396.48,403.80]	8.9	5228.6	[5198.55,5258.72]	72.9
130	92.5	[91.27,93.68]	2.9	652.6	[645.55,659.73]	17.2	749.7	[744.83,754.61]	11.8	5589.0	[5553.10,5624.81]	86.9
150	120.0	[116.85,123.15]	7.6	1415.8	[1406.88,1424.72]	21.6	1706.4	[1697.53.1715.19]	21.9	14940.0	[14756.60,15123.48]	444.4

TABLE 5. THE TOTAL DATA ACCESS COST MEAN, CONFIDENCE INTERVAL (CI) AND STANDARD DEVIATION (SD) UNDER DIFFERENT ARRIVING RATE OF THE WORKFLOWS

Lamda	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
1.0	0.5	[0,1.47]	2.4	160.2	[152.75,167.65]	18.0	122.6	[115.13,129.98]	18.0	1619.8	[1596.8,1642.8]	55.8
3.0	70.8	[67.89,73.78]	7.1	459.2	[439.65,478.75]	47.4	217.6	[209.95,225.33]	18.6	4592.0	[4563.7,4620.4]	68.7
5.0	74.4	[70.75,78.13]	9.0	785.0	[758.62,811.46]	64.0	693.3	[679.59,706.97]	33.2	7795.1	[7737.4,7852.7]	139.6
7.0	136.8	[70.91,73.2]	18.0	1277.7	[1225.05,1330.39]	127.6	1534.2	[1507.2,1561.2]	65.4	12565.6	[12484.9,12646.3]	195.5
9.0	163.0	[152.2,173.8]	26.2	1688.2	[1614.66,1761.8]	178.2	1988.8	[1950.7,20270]	92.3	16871.8	[16727.8,17015.9]	348.9

TABLE 6. THE VIOLATION NUMBER MEAN, CONFIDENCE INTERVAL (CI) AND STANDARD DEVIATION (SD) UNDER DIFFERENT EDGE CLUSTER

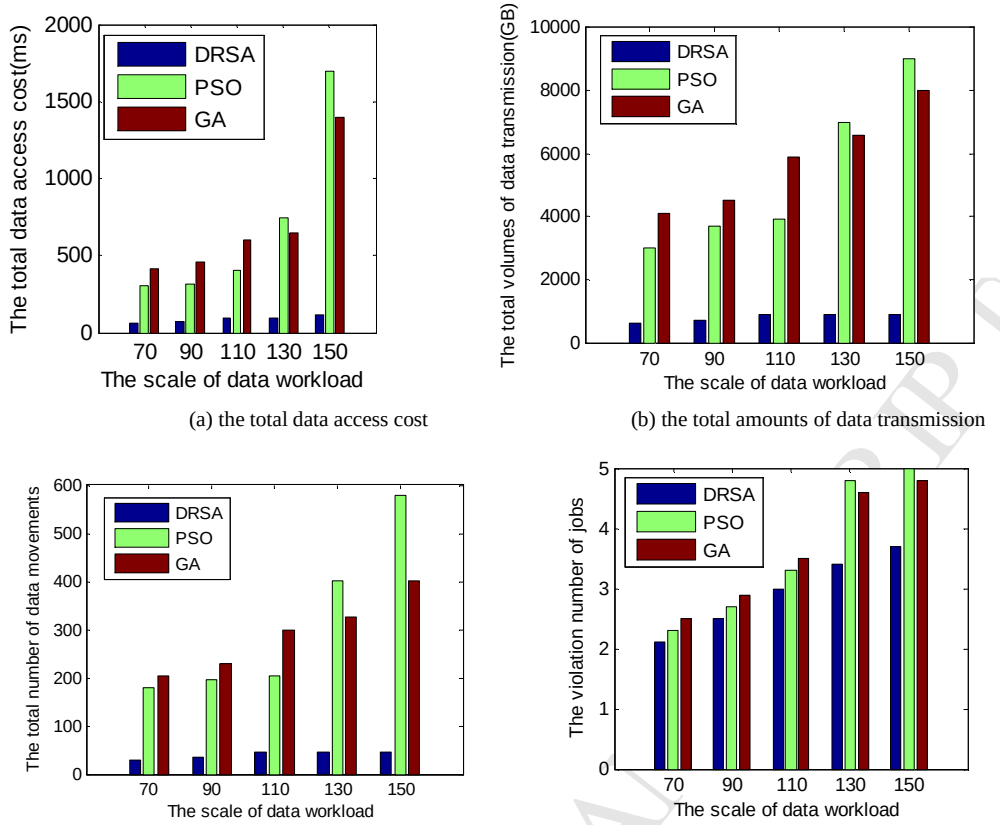
Edge cluster	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
2.0	2.4	[1.98,2.86]	1.1	3.0	[2.53,3.37]	1.0	2.6	[2.23,2.94]	0.9	5.0	[4.33,5.66]	1.6
3.0	2.4	[2.01,2.75]	0.9	3.1	[2.65,3.50]	1.0	2.7	[2.36,3.05]	0.8	5.1	[4.48,5.77]	1.6
4.0	3.0	[2.51,3.65]	1.4	3.5	[3.06,4.01]	1.2	3.3	[2.82,3.76]	1.1	6.2	[5.59,6.83]	1.5
5.0	3.1	[2.77,3.32]	0.7	3.7	[3.24,4.09]	1.0	3.4	[2.92,3.83]	1.1	6.3	[5.67,6.99]	1.6
6.0	3.1	[2.65,3.36]	0.9	3.7	[3.33,4.17]	1.0	3.5	[3.06,4.02]	1.2	7.0	[6.26,7.82]	1.9

TABLE 7. THE VIOLATION NUMBER MEAN, CONFIDENCE INTERVAL (CI) AND STANDARD DEVIATION (SD) UNDER DIFFERENT DATASET

Dataset	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
70	2.04	[1.68,2.41]	0.89	2.46	[2.03,2.89]	1.04	2.33	[1.98,2.68]	0.85	4.25	[3.77,4.73]	1.16
90	2.46	[2.10,2.81]	0.87	2.96	[2.54,3.38]	1.01	2.67	[2.34,3.76]	0.79	5.12	[4.49,5.76]	1.53
110	3.08	[2.51,3.65]	1.38	3.54	[3.06,4.02]	1.15	3.33	[2.83,3.83]	1.21	5.75	[5.09,6.40]	1.58
130	3.41	[2.93,3.91]	1.19	4.46	[3.82,5.11]	1.55	4.79	[4.13,5.44]	1.58	6.5	[5.89,7.11]	1.47
150	3.71	[3.10,4.30]	1.46	4.83	[4.21,5.44]	1.49	5.04	[4.25,5.83]	1.9	8	[7.34,8.65]	1.58

TABLE 8. THE VIOLATION NUMBER MEAN, CONFIDENCE INTERVAL (CI) AND STANDARD DEVIATION (SD) UNDER THE DIFFERENT ARRIVING RATE OF THE WORKFLOWS

Lamda	DRSA			GA			PSO			Random		
	mean	CI	SD	mean	CI	SD	mean	CI	SD	mean	CI	SD
1.0	0.1	[0,0.20]	0.3	0.8	[0.48,1.18]	0.9	0.5	[0,1.46]	0.9	2.1	[1.72,2.52]	1.0
3.0	2.4	[2.01,2.75]	0.9	2.9	[2.49,3.34]	1.0	2.7	[2.36,3.05]	0.8	5.1	[4.48,5.77]	1.6
5.0	4.5	[4.03,4.89]	1.0	6.2	[5.39,6.94]	1.9	5.8	[4.98,6.61]	2.0	8.2	[7.49,8.83]	1.6
7.0	6.5	[5.48,7.44]	2.4	8.0	[7.22,8.70]	1.8	8.1	[6.80,8.79]	2.2	12.0	[10.78,13.13]	2.8
9.0	7.1	[6.02,8.22]	2.7	8.1	[6.91,9.34]	2.9	8.5	[7.25,8.87]	1.9	15.0	[12.90,17.09]	5.1



(a) the total data access cost

(b) the total amounts of data transmission

(c) the data movement times

(d) the violation number of jobs

Fig.10. The four metrics: C_{access}, T_{all}, N_{tran} and $N_{t>deadline}$ with a different number of the dataset.

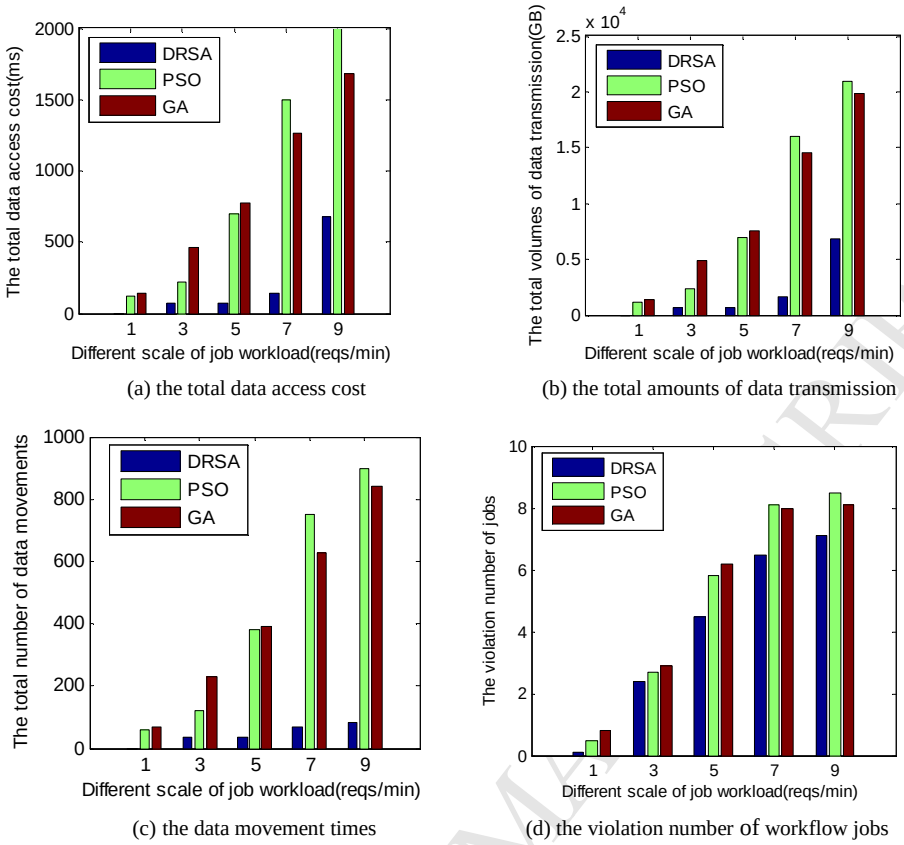
(3) The different scale of workloads of in the edge and cloud system.

We test how well DRSA, GA, PSO when the scales of the workflow workload are increased. The arriving rate of jobs is set 1 request per minute, 3, 5, 7 and 9, respectively. The corresponding number of data objects is 30, 90, 150, 210 and 270. The number of data replicas is set to 3. The number of the edge clusters is 3 and there are three edge computing nodes in each edge cluster. We use mean, confidence interval and the standard deviation of the simulation results as Table 5 and 8.

As the scale of task and workload is bigger, the data access costs under the three strategies tend to increase. Compared with genetic and PSO strategies, our DRSA strategy clearly has a greater advantage, too. Especially when the size of workflows is larger, our proposed strategy saves 10 times more data access cost than GA strategy. As the Fig.11 shows, our strategy has better performance than GA and PSO in terms of total data access cost, the total data transfer cost, the total number of data movements and the violation number of workflows which exceed the deadline. When the workload scare is greater than or equal to 7, the performance of the GA becomes better than the PSO.

From the above simulations, we can see that the results show that our data replica scheduling strategy for workflow jobs based on ITÖ algorithm outperforms Genetic algorithm and PSO in the complex collaborative distributed edge and cloud computing environment. For the small scale of the problem, the PSO data scheduling optimization ability is slightly better than GA. When the scale of the problem is relatively large, it is worse than GA. The reason is that for data scheduling problems, the particle radius and drift and volatility operators have been redesigned by mathematics formulas and our DRSA algorithm has to been proven almost inevitable strong convergence. Thus, the convergence speed and global

optimization ability are significantly enhanced much than general GA and PSO. Our algorithm also improves the performance in the real-time aspect for low-latency IOT applications.



(a) the total data access cost

(b) the total amounts of data transmission

(c) the data movement times

(d) the violation number of workflow jobs

Fig.11. The four metrics: C_{access}, T_{all}, N_{tran} and $N_{t>deadline}$ with different scales of the workflow workload.

6.3 Discussion

It is very important to evaluate a data replica placement scheduler algorithm very well before it is deployed in a real edge and cloud cluster. Unfortunately, evaluating in a real edge and cloud cluster is always time and cost consuming. Hence, in this study, we use a simulator to predict how well our proposed algorithm for some specific workload would be quite useful. Of course, there are some ways to simulate on a little bit larger scale using public cloud resource, such as Azure cloud service. It can be designed as the followings.

As Fig.12 shows, it describes clusters and communication resource of the whole system. The collaborative edge and cloud cluster includes three-tier resource, and we rent Ali public cloud service to form a cloud cluster, which consists of eleven virtual machines.

The three edge clusters are composed of five edge heterogeneous servers and the user equipment is composed of three smartphones. The mobile user terminal adopts Android 7.0 system, and is equipped with JDK 1.8.0_05, SDK Android 4.0, and installs the development plug-in ADK 22 to realize mobile terminal application development. The edge server cluster and cloud server use Ubuntu 14.04 operating system with OpenStack++ resource management framework, Docker CE v17.03 engine and Kubernetes 1.8.4 open source platform for lightweight resource management. The Internet of Things terminal uses TinyOS operating system.

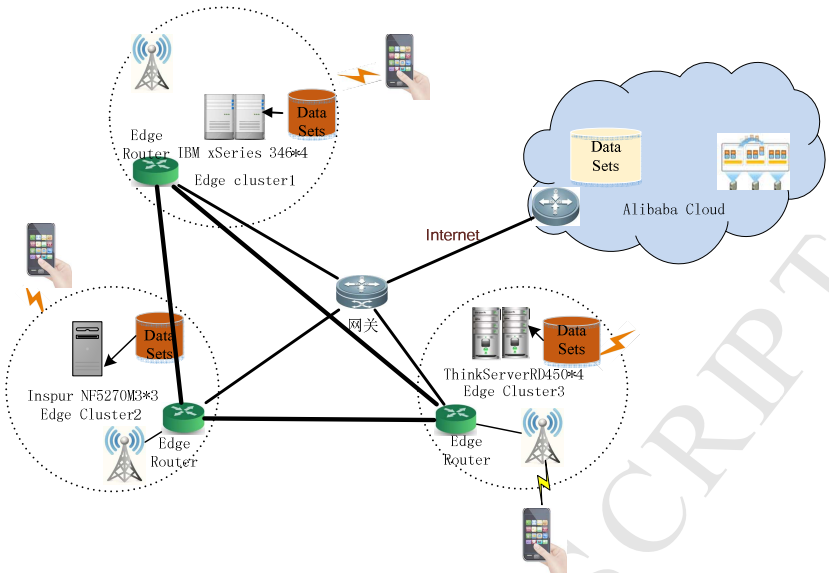


Fig.12. the experimental environment setting using public cloud service.

7. Conclusion and future work

In this paper, we focus on how many data copies are created and how to place them in an edge computing system and studied the replica creation and data scheduling strategies by deadline-driven for dependency workflow jobs. The replica management system is proposed which mainly includes two parts: data replica creator and data job scheduler for workflows. Considering the popularity of data blocks and node load, we propose the dynamic replica creation methods to predict the replica number of a data replica. And then to reduce data access costs while meeting deadline constraint, the data replication scheduling problem has been modeled as an integer programming problem and the faster meta-heuristic DRSA algorithm to solve it. The experimental results indicate that our proposed algorithm outperforms the compared methods such as GA and PSO algorithms in term of total data access cost, the amounts of data transfer, data movement times and the SLA violations.

In the future, some potential issues in the proposed replica management framework are also worthy to study. 1) The data and computational scheduler should be merged into the presented framework, and 2) the data security and privacy be further analyzed.

Acknowledgment

The work was supported by the National Natural Science Foundation (NSF) under grants (No.61672397, No. 61873341, No.61771354), Application Foundation Frontier Project of WuHan (No. 2018010401011290), State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System (GZ2018KF002). Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

[1] HUAWEI. HUAWEI 2015 Global Connectivity Index. http://www.huawei.com/minisite/gci/files/gci_2015_whitepaper_cn.pdf. 2015.4.

[2] Uddin, M. Z. (2019). A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system,Journal of Parallel and Distributed Computing,Vol.123, 46-53,.

[3] Satyanarayanan, M. (2017). The emergence of edge computing. Computer, 50(1), 30-39.
[4] Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016). Edge computing: vision and challenges. IEEE Internet of Things Journal, 3(5), 637-646.
[5] Hossain, S. A., Rahman, M. A., Hossain, M. A. (2018) Edge computing framework for enabling situation awareness in IoT based smart city, journal of Parallel and Distributed Computing, Vol.122, 226-237.
[6] Karim, M. B. A., Ismail, B. I., Tat, W. M., Goortani, E. M., Setapa, S. and Jing, Y. L., et al. (2016). Extending Cloud Resources to the Edge: Possible Scenarios, Challenges, and Experiments. International Conference on Cloud Computing Research and Innovations, Singapore, May 4-5, 2016, pp.78-85, IEEE Computer Society, Washington.
[7] Howard, S. G., Gobioff, H., and Leung, S. (2003). The Google file system. Acm Sigops Operating Systems Review, 37(5), 29-43.
[8] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). IEEE.
[9] Azari, L., Rahmani, A. M., Daniel, H. A., Qader, N. N. (2017). A data replication algorithm for groups of files in data grids. Journal of Parallel and Distributed Computing, 113 (2018), 115-126.
[10] Khanli,L. M., Isazadeh,A.,Shishavan, T.N. (2011). PHFS: A dynamic replication method to decrease access latency in the multi-tier data grid, Future Gener. Comput. Syst.27 (3), 233-244.
[11] Abdurrab, A. R. Xie, T. (2010). FIRE: A file reunion based data replication strategy for data grids, in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid, 215-223.
[12] Gill, N. K., and Singh, S. A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. Future Generation Computer Systems,65, 10-32 (2016)
[13] Matri, P., Costan, A., and Antoniu, G. (2016). Towards efficient Location and Placement of Dynamic Replicas for Geo-Distributed Data Stores. ACM, Workshop on Scientific Cloud Computing, ACM, 3-9.
[14] Higai, A., Takefusa, A., Nakada, H., Oguchi, M. (2014). A Study of Effective Replica Reconstruction Schemes at Node Deletion for HDFS. Ieee/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 512-521.
[15] Naik, N., Negi, Atul., Bapu, T., and Anitha, R.(2019). A data locality based scheduler to enhance MapReduce performance in heterogeneous environments, Future Generation Computer Systems, Vol.90, 423-434.
[16] Li, C., Bai J., and Tang. J. (2019). Joint optimization of data placement and scheduling for improving user experience in edge computing, Journal of Parallel and Distributed Computing, Vol. 125, 93-105.
[17] Tak, T., Hong, J., Park, K., Lee, W., Lee, T., Han, H., Kwon, G., Park, J. (2018).Conceptual design of new data integration and process system for KSTAR data scheduling, Fusion Engineering and Design, 129, 330-333.
[18] Cui, L. Zhang, J., Yue, L., Y. Shi, H. Li, and D. Yuan. (2018).A Genetic Algorithm Based Data Replica Placement Strategy for Scientific Applications in Clouds. IEEE Transactions on Services Computing, 11(4) 727-739.
[19] Shorfuzzaman, M., Graham, P., & Eskicioglu, R. (2011). Distributed Placement of Replicas in Hierarchical Data Grids with User and System QoS Constraints. International Conference on P2p. IEEE.
[20] Li, X. , Wu, Y. , Ma, F. , Zhu, E. , Wang, F. , & Wu, L. , et al. (2014). A new particle swarm optimization-based strategy for cost-effective data placement in scientific cloud workflows. Lecture Notes in Electrical Engineering, 309, 115-120.
[21] Ramaswamy, L. , Iyengar, A. , & Chen, N. J. (2006). Cooperative Data Placement and Replication in Edge Cache Networks. International Conference on Collaborative Computing: Networking. IEEE,1-9.
[22] Shrivastava, A., Bansod, P., Gupta, K., Merchant, S. N. (2018). An improved multicast based energy efficient opportunistic data scheduling algorithm for VANET, International Journal of Electronics and Communications, Vol. 83, Pages 407-415.
[23] Dong, W. Y., Zhang, W. S. and Yu, R. G. (2011). Convergence and runtime analysis of ito algorithm for one class of combinatorial optimization. Chinese Journal of Computers, 34(4), 636-646.
[24] Wang, Y. F., Dong, W. Y., Dong, X.S.(2018) A novel ITÖ Algorithm for influence maximization in the large-scale social networks, Future Generation Computer Systems,Volume 88, 755-763.
[25] Kosar, T. and Livny, Miron. (2005). A framework for reliable and efficient data placement in distributed computing systems. Journal of Parallel & Distributed Computing, 65(10), 1146-1157.
[26] Alamgir Hossain, S. K. , Md., A. R. , & Anwar, H. M. . (2018). Edge computing framework for enabling situation awareness in iot based smart city. Journal of Parallel and Distributed Computing, 122(2018), 226-237.
[27] Persico, V., Marchetta, P., Botta, A., & Pescapè, A. (2015). Measuring network throughput in the cloud: the case of amazon ec2. Computer Networks, 93(P3), 408-422.

[28] Botta, A., Donato, W. D., & Persico, V. (2016). Integration of cloud computing and internet of things. *Future Generation Computer Systems,56* (C), 684-700.

[29] Persico, V., Botta, A., Marchetta, P., Montieri, A., & Pescapé, A. (2017). On the performance of the wide-area networks interconnecting public-cloud datacenters around the globe. Computer Networks, 112, 67-83.

[30] Singh, G., Vahi, K., Ramakrishnan, A., Mehta, G., Deelman, E. and Zhao, H. (2007). Optimizing workflow data footprint. Scientific Programming, 15(7), 249-268.

[31] Golle, P., & Mironov, I. (2001). Uncheatable Distributed Computations. Topics in Cryptology-CT-RSA 2001. vol 2020,425-440,Springer Berlin Heidelberg.

[32] Popa, R. A., Stark, E., Helfer, J., Valdez, S., Zeldovich, N., & Kaashoek, M. F., et al. (2014). Building web applications on top of encrypted data using mylar. ;login:: the magazine of USENIX & SAGE, 39(3), 22-27

[33] Rimal, B. P., & Maier, M. (2017). Workflow scheduling in multi-tenant cloud computing environments. IEEE Transactions on Parallel and Distributed Systems, 28(1), 290-304.

[34] Prasad, R. B., & El-Refaey, M. A. (2010). A Framework of Scientific Workflow Management Systems for Multi-Tenant Cloud Orchestration Environment. IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises. IEEE, 90-93.

[35] Rango, F. D., Santamaria, A. F., Tropea, M., & Marano, S. (2008). Meta-Heuristics Methods for an NP-Complete Networking Problem. IEEE Vehicular Technology Conference. IEEE.

[36] Jiang, H., Haihong, E., & Song, M. (2017). Dynamic scheduling of workflow for makespan and robustness improvement in the iaas cloud. Ieice Transactions on Information & Systems, 100(4), 813-821.

[37] Pegasus. WorkflowGenerator. Pegasus Workflow Management System https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator#space-menu-link-content. Accessed on 2018.4.

[38] Deelman, E. (2013). Characterizing and profiling scientific workflows. Future Generation Computer Systems, 29(3), 682-692.

[39] Zaman, S. and Grosu, D. (2011). A distributed algorithm for the replica placement problem. IEEE Transactions on Parallel & Distributed Systems, 22(9), 1455-1468.

[40] Chunlin Li, Jingpan Bai, JiangHang Tang，Joint optimization of data placement and scheduling for improving user experience in edge computing，Journal of Parallel and Distributed Computing，Volume 125, March 2019, Pages 93-105

[41] Chunlin Li, Zhu Liye, Tang Hengliang, Youlong Luo, Mobile user behavior based topology formation and optimization in ad hoc mobile cloud, Journal of Systems and Software, Elsevier, Volume 148, February 2019, Pages 132-147

[42] Hengliang Tang, Chunlin Li, Jingpan Bai, JiangHang Tang, Youlong Luo, Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment, Computer Communications Volume 134, 15 January 2019, Pages 70-82

[43] Chunlin Li, Jing Zhang, Youlong Luo, Real-Time Scheduling Based on Optimized Topology and Communication Traffic in Distributed Real-Time Computation Platform of Storm, Journal of Network and Computer, Elsevier, Volume 87, 1 June 2017, Pages 100–115

Yanling Shao is currently working toward the Ph.D. degree in the Department of Computer and Science at Wuhan University of Technology. She is an Associate Professor of Computer Science in Nanyang Institute of Technology, Nanyang, China. She received her M.S. degree from Xi'an Technological University, Xi'an, China, in 2010. Her current research interests include big data processing, cloud computing and distributed computing.

Chunlin Li is a Professor of Computer Science in Wuhan University of Technology. She received the ME in Computer Science from Wuhan Transportation University in 2000, and PhD in Computer Software and Theory from Huazhong University of Science and Technology in 2003. Her research interests include cloud computing and distributed computing.

Zhao Fu is an engineer of State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System.

Jia Leyue is an engineer of State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission System.

Youlong Luo is a vice Professor of Management at Wuhan University of Technology. He received his M.S. in Telecommunication and System from Wuhan University of Technology in 2003 and his Ph.D. in Finance from Wuhan University of Technology in 2012. His research interests include cloud computing and electronic commerce.