Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9529526, 14 pages
http://dx.doi.org/10.1155/2016/9529526

Research Article

Multiobjective Reliable Cloud Storage with Its Particle Swarm Optimization Algorithm

Xiyang Liu,[1] Lei Fan,[1] Liming Wang,[1] and Sha Meng[2]

[1]*Institute of Software Engineering, School of Software, Xidian University, Xi'an 710071, China*
[2]*School of Computer Science and Technology, Xidian University, Xi'an, China*

Correspondence should be addressed to Lei Fan; lfan@mail.xidian.edu.cn

Received 26 May 2016; Accepted 9 November 2016

Academic Editor: Marco Mussetta

Information abounds in all fields of the real life, which is often recorded as digital data in computer systems and treated as a kind of increasingly important resource. Its increasing volume growth causes great difficulties in both storage and analysis. The massive data storage in cloud environments has significant impacts on the quality of service (QoS) of the systems, which is becoming an increasingly challenging problem. In this paper, we propose a multiobjective optimization model for the reliable data storage in clouds through considering both cost and reliability of the storage service simultaneously. In the proposed model, the total cost is analyzed to be composed of storage space occupation cost, data migration cost, and communication cost. According to the analysis of the storage process, the transmission reliability, equipment stability, and software reliability are taken into account in the storage reliability evaluation. To solve the proposed multiobjective model, a Constrained Multiobjective Particle Swarm Optimization (CMPSO) algorithm is designed. At last, experiments are designed to validate the proposed model and its solution PSO algorithm. In the experiments, the proposed model is tested in cooperation with 3 storage strategies. Experimental results show that the proposed model is positive and effective. The experimental results also demonstrate that the proposed model can perform much better in alliance with proper file splitting methods.

1. Introduction

The rapidly increasing information resources, which are often recorded in form of data and stored in computer systems, are a kind of new resources and becoming more and more important nowadays. The rapid development of Internet technologies and mobile communication equipment allows more and more people to have opportunity to access various Internet networks, such as Twitters, Amazon, and Taobao sites. While visiting these websites, massive accession information which is very important to the service providers to make business decisions will be produced. The massive accession information is often collected by the service providers and recorded as data resources stored in cloud systems. The storage of such massive data is of great significance for the quality of service of computing systems and poses many challenges to storage service providers. Recently, cloud storage is getting more and more competitive and popular, which is becoming a trend of future storage technique. Various cloud storage products are springing up in the high-tech market, such as Cloud Drive of Amazon and Live Mesh of Microsoft.

The main goal of cloud storage is to provide users with low-cost, infinitely extensible, and high-reliability storage platform, which can also support various data storage types. A cloud storage system also has abilities to satisfy the different requirements of different users. With the development of the era of service economy, customers no longer seek storage products or services with cheap price and high quality but are more concerned about quality of service, which makes the data storage providers concentrate on promoting both of the quality of service and storage efficiency simultaneously. Therefore, optimizing the data storage process is of great importance to the quality of service of the cloud storage systems. However, with the development of cloud storage techniques, various data processing techniques and applications need to be deployed in combination with cloud storage, so that different aspects should be considered while

optimizing the storage process. Users usually select suitable storage service according to the issues of storage cost and reliability. The service providers also do their best to enhance product competitiveness by saving the storage costs and promoting the storage reliability.

Nowadays, the quality of service of cloud storage is being influenced by the following 7 categories of risks or challenges [1]: privileged access, auditing, storage location, data isolation, data disaster recovery, survey, and long-term viability. Among these risks, data disaster recovery is a very significant issue for both cloud storage providers and users. Users choose cloud storage providers to store their sensitive and significant business data [2–4]. Once their data were lost, it will not only lead to fetal disasters to the companies but also bring legal disputes between users and storage service providers. For instance, in a banking system, the clients' data is of great importance to both banks and clients. If lost or unable to be recovered, it will lead to immeasurable losses for both banks and their clients [5, 6]. Hence, it should be one of the business objectives for cloud storage providers to ensure the reliability and recovery for the stored data.

In cloud storage systems, the datacenters are the vital components used to house computer systems and associated components, such as servers, telecommunications, and storage system [7]. Compared with personal computers, the main advantages of the datacenter are over its continuously powerful operation, storage, and network communication, faster crash recovery, and broader convenient expanse space [8]. In almost all the computer systems, data resources are stored on the devices called hard disks, which have direct impacts on the quality of service of the storage process. The storage capacity of a cloud storage system is significantly limited by the volume of the hard disks in its datacenters. In a datacenter, various hard disks with different properties are deployment for certain purposes on storage or analysis of the data files, such as SSD (Solid State Disk), HDD (Hard Disk Drive), and HHD (Hybrid Hard Disk). In datacenters, the mixture of HDDs and SSDs is becoming more and more popular in enterprise-class disk arrays. Practically, datacenters are often located at different places (cities) and connected with each other via Internet or high-performance routers. The geographical distance is another important factor which will influence the data storage process. The cloud storage is getting more and more popular, so more and more individuals and enterprises decide to migrate their business or the data management works to datacenters aiming to cut operating costs. As is well known, the value of data lies not only in its contents but also in the potential business benefits for the business development. Thus, the quality of service of a cloud storage system is conditioned by many different factors.

From the user's point of view, different data files submitted for possible storage often need different storage service accordingly [9]. To meet such demand, the storage service providers have to develop diverse storage products and invest large quantities of human inputs and various storage devices. Hence, an important issue for the providers to consider is how to save the service cost and improve the storage reliability. Many operators have employed different technologies to realize the purpose of reducing the cost and improving the reliability [10–12]. During the storage of the data files, the cost and reliability are influenced by many factors, such as communication and transmission. In this paper, these factors will be quantified to evaluate the cost and reliability of the storage service. Through minimizing the storage cost and maximizing the storage reliability simultaneously, a multiobjective optimization model for the massive data storage in cloud environment is proposed. Various solutions that have reference value for decision-makers will be obtained by solving this multiobjective model, so that more reasonable decisions might be made. Then, a particle swarm optimization (PSO) algorithm is designed to solve the proposed model.

The remainder of this paper is organized as follows: Section 2 briefly reviews the state of the art of the research on the cloud storage optimization. Then, the multiobjective optimization model for reliable cloud storage is proposed through analyzing the storage processes in Section 3. Section 4 briefly introduces the PSO algorithm and a tailor-made multiobjective PSO algorithm is designed to solve the proposed model. Experiments and results analysis are made in Section 5. At last, conclusions of this paper are drawn in Section 6.

2. Related Work

In recent years, with the increasing usage of the cloud storage service, more and more efforts have been made to improve the quality of service via managing resources in clouds. Since the resources in the cloud environment are provided on a pay-as-you-go basis, high efficiency utilization of the resources is necessary. Efficient resource management in cloud datacenters has attracted more and more attentions with different objectives, such as reducing cloud computational cost or energy budget, improving the quality of service of the cloud systems, and optimizing the data storage. Various techniques have been developed to achieve these objectives.

Virtualization technique is a kind of efficient resource management techniques employed in datacenters to reduce the computational cost or energy budget. Virtual machines play significant roles in the cloud computing systems, which can be migrated between different systems or infrastructures. Through migrating, these virtual machines can efficiently manage the resources in the cloud environment. The recent research on virtual machine migration problems mainly focuses on maximizing the resources utilization or minimizing the migration cost. Zhang et al. [13] proposed a multiobjective optimization model to maximize the resource utilization and minimize the migration cost simultaneously, in which the factors of storage capacity, bandwidth, etc. are considered. Ahmad et al. [14] summarized the virtual machine migration schemes in aspects of the network bandwidth optimizations, server consolidation frameworks, DVFS energy optimization, and storage optimization for cloud datacenters in the Internet links. Dong et al. [15] proposed a virtual machine scheduling strategy aiming at reducing the number of the active virtual machines and network devices, in which the resources in IaaS cloud, such as the storage space, bandwidth, and network connectivity, are considered.

Cloud storage optimization [16–22] is another issue to improve the quality of service of the cloud computing systems. In [16], optimizations are proposed to reduce the volume of data to be transferred per data access for the respects of privacy and security, which did not consider the storage cost. Guo and Fang [17] proposed an optimization model with the objective of minimizing the total electricity cost of multiple datacenters for cloud service providers, in which the available energy storage capabilities in datacenters were considered to be further used to reduce the electricity. Then, an online algorithm based on Lyapunov optimization technique was developed for this model. However, solving this single objective optimization model, only one solution can be obtained, which cannot help service providers develop customized products. Mao et al. [18] proposed a SAR model to improve the read performance of the cloud storage system. In this model, the high efficiency of the random read performance of SSDs was utilized to improve the read performance of the cloud system converting the read requests on HDDs to the read requests on the SSDs. Liu et al. [19] proposed a security-aware intermediate data placement strategy for data intensive scientific workflows in cloud environment with the aim of storing large volumes of the intermediate datasets cost-effectively, in which the security factor of the datasets is considered. In [20], the efforts on improving the energy efficiency of data accesses and storage were made. Aiming to improve the storage utilization and workflow scheduling performance in the cloud, Wang et al. [21] proposed a user-level file system and a scheduling algorithm for scientific workflow computation in the cloud, in which the storage utilization was improved by using the workflow-aware file system and the scheduler to control the number of concurrent workflow instances at runtime. Jarrah et al. [22] proposed a hierarchical optimization model for energy data flow in smart grid power systems aiming to minimize daily electricity cost through maximizing the used percentage of renewable energy.

However, most of the existing works focus on either the cost (or energy) or the storage safety by scheduling the data storage tasks or the storage devices. Few work considers both the storage cost and reliability simultaneously. The storage reliability is of great significance to the quality of service of the cloud storage systems.

3. Multiobjective Optimization Model for Massive Data Storage

As a kind of open service IT systems, cloud storage systems are usually with the structure as Figure 1. Choosing an open cloud environment to store data files, users often take into account not only the price of data storage services but also the reliability when they send their data files to the cloud computing systems [23–25]. The users' privacy or profit is often involved in their data files to be stored. The users might probably select different storage services for different purposes according to their demands. Therefore, the storage service providers should provide plentiful storage service to satisfy different users' requirements. When pricing the storage service, the providers should have knowledge of the cost and reliability of the certain storage service. Or, given the price of a certain storage service, the providers need to do their best to reduce the storage cost and keep the storage reliability. Aiming to solve this problem, we proposed a new multiobjective optimization model for massive data storage in cloud environment considering storage cost and reliability simultaneously.

3.1. Assumptions. Each server in datacenters of a cloud storage system can be treated as a node, and the linkage between servers can be represented by edges. Therefore, we can have a complete graph to represent the networks of the servers. The data files with different security level have different storage cost. Except for the costs of encryption processing and special device usage, another issue is that data files with high security level may have more backups around different locations. Hence, the monitor system needs to communicate with the servers located in different places and transfer these backups. These processes will not only cost much more but also make the storage unreliable.

Let $S = \{s_1, s_2, \ldots, s_N\}$ be the servers in the datacenters of a cloud storage system. These servers might be arranged in different places. The distance between these servers is denoted by the matrix $(d(s_i, s_j))_{N\times N}$, in which, $\forall i \in \{1, 2, \ldots, N\}$, $d(s_i, s_i) = 0$ and, $\forall i, j \in \{1, 2, \ldots, N\}$ and $i \neq j$, $d(s_i, s_j) \geq 0$. The data files submitted by users are denoted as $F = (f_1, f_2, \ldots, f_M)$, in which f_i denotes an indivisible data file. Each data file has its own security level, denoted as $L(f_i)$, $i = 1{\sim}M$. For convenience, other parameters are listed in Nomenclature.

3.2. Total Service Cost for Massive Data Storage. In each server, the data files with different security levels usually have different storage price denoted as $C_s(s_i, L(f_j))$, which is in proportion to the security level. The number of backups for data file f_i with different security levels is often set to be proportional to its security level. Hence, the higher the security level a data file has, the higher the storage cost it will need. The total service cost for storing a set of data files F is divided into three parts in this paper: storage cost, migration cost, and communication cost.

3.2.1. Storage Cost. The storage cost of data file f_i per unit time can be treated as rental cost for data store space, which is the product of the volume of the data files, the price of the related storage service in the target server, and the storage time. Therefore, the storage cost of data file f_i can be evaluated through the following formula:

$$\zeta_s(f_i) = \sum_{j=1}^{N} C_s\left(s_j, L(f_i)\right) X\left(f_i, s_j\right). \tag{1}$$

Thus, the storage cost of storing the data file set F in time t is the sum of the storage cost of each data file f_i in the cloud storage system, which can be formulated as follows:

$$\begin{aligned}\zeta_s(F, t) &= \sum_{i=1}^{M} \zeta_s(f_i) \cdot t \\ &= \sum_{i=1}^{M} \sum_{j=1}^{N} C_s\left(s_j, L(f_i)\right) X\left(f_i, s_j\right) \cdot t.\end{aligned} \tag{2}$$

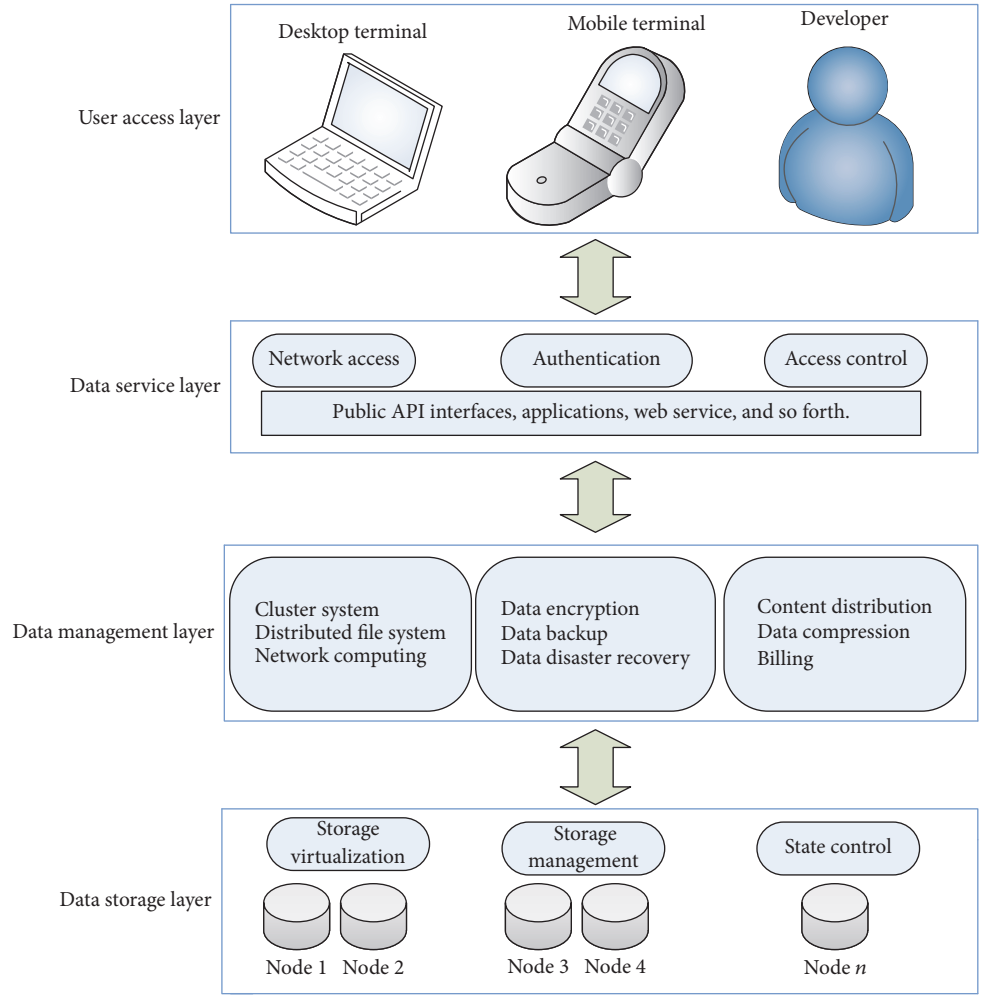

Figure 1: Structure of cloud storage systems.

3.2.2. Migration Cost. As is well known, users can access and upload their data files via Internet to a cloud system almost everywhere. The data files are often uploaded by the users to the servers located at the management layer of a cloud system to decide which target storage servers will be chosen to store these data files. Therefore, some data files might be migrated to other servers located at different places. The migration cost will be required, which is basically proportional to the migration distance and the data volume. Suppose that the data files are uploaded at server s_b, the migration cost of a data block f_k from s_b to s_j can be calculated as follows:

$$\zeta_{\text{mig}}\left(f_k, s_b, s_j\right) = C_m X\left(f_k, s_b\right) d\left(s_b, s_j\right), \tag{3}$$

in which C_m is the migration price. Thus, the migration cost of the data files F during the storage process can be calculated as follows:

$$\begin{aligned}\zeta_m(F) &= \sum_{k=1}^{M}\sum_{j=1}^{N}\zeta_{\text{mig}}\left(f_k, s_b, s_j\right) \\ &= \sum_{k=1}^{M}\sum_{j=1}^{N} C_m X\left(f_k, s_j\right) d\left(s_b, s_j\right).\end{aligned} \tag{4}$$

3.2.3. Communication Cost. During the whole process of migrating the data files to the cloud system, the storage management system should collect the storage information of the candidate servers to make sure whether the certain data file can be stored, verify the integrity of the migrated data files, and monitor the storage process. Therefore, the communication is throughout the whole storage process. The required communication cost is proportional to the distance between the server s_m where the management system located and the target server s_i candidated to store the data file f_k. The communication cost of storing data file f_k can be formulated as follows:

$$\zeta_{\text{com}}\left(f_k\right) = \sum_{i=1}^{N}\left[X\left(f_k, s_i\right)\omega\left(s_i, s_m\right) d\left(s_i, s_m\right)\right]. \tag{5}$$

Therefore, the total communication cost of the set of data files F submitted for storage is calculated as the following formula:

$$\begin{aligned}\zeta_c(F) &= \sum_{k=1}^{M}\zeta_{\text{com}}\left(f_k\right) \\ &= \sum_{k=1}^{M}\sum_{i=1}^{N}\left[X\left(f_k, s_i\right)\omega\left(s_i, s_m\right) d\left(s_i, s_m\right)\right].\end{aligned} \tag{6}$$

3.2.4. Total Cost of Storage Service. The dataset F is supposed to be submitted via terminal server s_b. Let $\zeta(F, S)$ be the total cost of storing the dataset F in the cloud storage system, which is considered to be the sum of storage cost, communication cost, and migration cost. The formula is as follows:

$$\begin{aligned}\zeta(F,S,t) &= \zeta_s(F,t) + \zeta_m(F) + \zeta_c(F) \\ &= \sum_{j=1}^{M}\sum_{i=1}^{N} C\left(s_i, L\left(f_j\right)\right) X\left(f_j, s_i\right) \cdot t \\ &\quad + \sum_{k=1}^{M}\sum_{j=1}^{N} C_m X\left(f_k, s_j\right) d\left(s_b, s_j\right) \\ &\quad + \sum_{k=1}^{M}\sum_{i=1}^{N} \left[X\left(f_k, s_i\right) \omega\left(s_i, s_m\right) d\left(s_i, s_m\right)\right].\end{aligned} \tag{7}$$

3.3. Storage Reliability. The reliability of a storage system is one of the most important measures of the service of quality and the calculation of the reliability is so difficult that very few attention has been paid. During the data storage process, the reliability is influenced by many factors (e.g., data transmission rate, equipment stability, and service system stability). However, it is difficult to determine the quantitative criteria to evaluate these impactors. The evaluation of these factors is essentially predictions of these factors according to their historic behavior. In this paper, the data transmission reliability, equipment stability, and software reliability are considered in the evaluation of the storage reliability.

3.3.1. Transmission Reliability. This factor is used to estimate the stability of the networks in practice. To store a set of date files into a cloud storage system, the data transmission in the network is necessary and very important for the distributed storage. The transmission reliability is of crucial importance for successful data storage.

In networks, the information and data with large datagrams are broken up into several small packets which will be transferred along with different network paths in one or several networks and reassembled at the target end. However, when one or more packets of data traveling across a computer network fail to reach their destination, some packets may be lost. Packet loss is one of the common faults in the network and typically caused by a number of factors that can corrupt or lose packets in transit, such as network congestion, radio signals that are too weak due to distance or multipath fading (in radio transmission), faulty networking hardware, or faulty network drivers. What is more, packets are sometimes intentionally dropped by normal routing routines (such as Dynamic Source Routing in ad hoc networks [26]) and through network dissuasion technique for operational management purposes [27]. Packet loss can reduce throughput for the senders, whether unintentionally due to network malfunction or intentionally as a means to balance available bandwidth between multiple senders when a given router or network link reaches near its maximum capacity [28].

This kind of network fault is measured as a percentage of packets lost with respect to packets sent. Therefore, the packet loss rate is an important criterion to measure the stability of networks.

$$\begin{aligned}\text{Packet\_loss\_rate} = &\frac{\text{input\_packets} - \text{output\_packets}}{\text{input\_packets}} \\ &\times 100\%.\end{aligned} \tag{8}$$

When reliable delivery is necessary, packet loss increases latency due to additional time needed for retransmission. Assuming no retransmission, packets experiencing the worst delays might be preferentially dropped (depending on the queuing discipline used) resulting in lower latency overall at the price of data loss. The acceptable amount of packet loss depends on the type of data being sent [29]. Therefore, the packet loss rate is used to evaluate the network reliability. However, packet loss often happens randomly. It is difficult to determine the exact rate for current usage. In this paper, the average packet loss rate in a period of time is used to measure the reliability of the candidate path. The packet loss rate can be easily obtained via log files. Thus, the transmission reliability between servers s_i and s_j can be measured as follows:

$$R_T\left(s_i, s_j\right) = \frac{1}{t}\sum_{t} \text{Packet\_loss\_rate}\left(s_i, s_j\right). \tag{9}$$

3.3.2. Equipment Stability. A datacenter is an integrated system composed of equipment systems and software systems. The stability of the equipment in the cloud storage system is to measure the reliability of the physical devices while storing data files. There are many devices in a datacenter, which includes infrastructure (electric generators, uninterruptible power system, air conditioners, etc.), IT equipment (servers, switchers, etc.), and networks. All the computing and storage services are implemented upon these devices and the fundamental premise of the successful service implementation is the trouble-free operation of the critical equipment, which is measured by mean time between failures (MTBF).

The operating ratio is used to measure the equipment stability in this paper. Given a device, its operating ratio during interval Δt can be calculated as the following formula:

$$\text{Operating\_ratio} = \frac{\text{MTBF}}{\Delta t} \times 100\%, \tag{10}$$

where MTBF denotes mean time between failures during the interval Δt. The operating ratio of the storage process is evaluated to be the minimal ratio among all involved devices:

$$R_E\left(s_i, s_j\right) = \frac{\min_{\text{device}(s_i \to s_j)} \{\text{MTBF}\}}{\Delta t} \times 100\%, \tag{11}$$

where device($s_i \to s_j$) denotes the devices involved in the path from s_i to s_j. The number of MTBF of each device can be found in the operating log files.

3.3.3. Software Reliability. Software reliability is more difficult to be ensured than hardware reliability in computing service

field, which will impact the reliability of the whole system. In cloud storage systems, the service providers usually pay more attention to the speed of the data processing, the correctness of the results, and the amiability of the interface of the software but less attention to the reliability. However, reliability problems emerging during the employment of the software often increase the difficulties and effort of the software maintenance. Serious reliability problems might cause the system paralysis.

Software reliability is defined to be the probability of failure-free operation for a software system in a given environment [30] or the ability of a system or component to perform its required functions under stated conditions for a specified period of time [31], which is of great importance for the safety-critical systems and influenced by both the software itself and software operating environment. Before or during the storage process, the data files are usually processed using different software for various purposes (e.g., file encryption, file splitting, and merging). A reliable software should have abilities to run without fault and output correct results. Hence, the software reliability is an important factor that should be considered when evaluating the data storage reliability. There are various metrics to specify the software reliability, such as failure rate, mean time to failure (MTTF), and reliability function [32]. In this paper, the software reliability is evaluated to be the mission success rate, which can be calculated to the ratio of allocating tasks to the successfully completed tasks. For software, the numbers of allocating tasks and successfully completed tasks can be obtained via its log files.

In this paper, the software reliability is to measure the behavior of software in recent time interval Δt. Let the number of the allocating tasks be n_a and the successfully completed tasks be n_s. Hence, the software reliability R_S will be calculated as

$$R_S = \frac{n_s}{n_a} \times 100\%, \tag{12}$$

where n_a and n_s can be obtained from the log files of the software or the operation system. What is more, the failure rate R_f can be easily formulated as follows:

$$R_f = \frac{n_a - n_s}{n_a} = 1 - R_s. \tag{13}$$

3.3.4. Service Reliability of Cloud Storage. Storage reliability is a significant metric to measure the quality of service of a storage strategy. After being submitted to the storage system, both clients and service providers expect that the data files can be stored with high reliability. The evaluation of the reliability of storage strategies is essential before making reasonable decisions.

For each inseparable data file f_i, its storage reliability is influenced by not only the transmission reliability, software reliability, and equipment stability involved in the uploading server s_b to the destination server s_j but also the physical storage mediums in the destination server, which can be formulated as follows:

$$\xi_r\left(f_i, s_j, s_b\right) = \sum_{j=1}^{N} R_t\left(s_j, s_b\right) \times R_E\left(s_j, s_b\right) \times R_S\left(s_j, s_b\right) \times P\left(Y_{ij}\right). \tag{14}$$

The reliability of a storage strategy for dataset F is calculated to be the product of the reliability of each inseparable data file as follows:

$$\xi\left(F, S\right) = \prod_{i=1}^{M} \xi_r\left(f_i, s_j, s_b\right) = \prod_{i=1}^{M} \left(\sum_{j=1}^{N} R_t\left(s_j, s_b\right) \times R_E\left(s_j, s_b\right) \times R_S\left(s_j, s_b\right) \times P\left(Y_{ij}\right) \right). \tag{15}$$

3.4. Multiobjective Reliable Storage Model. As a matter of fact, the storage strategies are often limited by various real restrictions or limitations, such as storage capacity, off-site backup, impacts of storage devices, storage security, and data integrity. These constraints should be considered inevitably during the determination of storage strategies.

The constraints can be stated as follows:

(1) During storage, the original data files and the backup files are treated to be the same. In a feasible storage strategy, each inseparable data file (including the backup file) should be assigned to only one server.

(2) For an inseparable data file f_i, the number of its backups should not be less than the number $L(f_i)$ for the backup with the same security level. This constraint can be formulated as $\sum_{j=1}^{N} \text{sign}(Y_{ij}) \geq L(f_i)$.

(3) The data file f_i and its backups should be assigned to be stored at different servers in different places between each other. This constraint can also be reflected via the inequality $\sum_{j=1}^{N} \text{sign}(Y_{ij}) \geq L(f_i)$.

(4) If data file f_i was assigned to be stored in server s_j, the available storage space in s_j should be enough to store the assigned data file with the corresponding reliability. This constraint can be expressed as $L(f_i) \leq R(Y_{ij})$.

Based on the above analysis, the multiobjective optimization model for massive data storage can be formulated as follows:

$$
\begin{aligned}
\min \quad & \zeta(F,S,t) = \zeta_s(F,t) + \zeta_m(F) + \zeta_c(F) \\
& = \sum_{i=1}^{M}\sum_{j=1}^{N} C\left(s_j, L\left(f_i\right)\right) X\left(f_i, s_j\right) \cdot t + \sum_{k=1}^{M}\sum_{j=1}^{N} C_m X\left(f_k, s_j\right) d\left(s_u, s_j\right) + \sum_{k=1}^{M}\sum_{i=1}^{N} \left[X\left(f_k, s_i\right) \omega\left(s_i, s_m\right) d\left(s_i, s_m\right)\right] \\
\max \quad & \xi(F,S) = \prod_{i=1}^{M} \xi_r\left(f_i, s_j, s_b\right) = \prod_{i=1}^{M} \left(\sum_{j=1}^{N} R_t\left(s_j, s_b\right) \times R_E\left(s_j, s_b\right) \times R_S\left(s_j, s_b\right) \times P\left(Y_{ij}\right) \right) \\
\text{s.t.} \quad & L\left(f_i\right) \le R\left(Y_{ij}\right), \quad i = 1, 2, \ldots, M, \\
& f_i \le A_{Y_{ij}}\left(s_j\right), \quad i = 1, 2, \ldots, M, \\
& \sum_{j=1}^{N} \operatorname{sign}\left(Y_{ij}\right) \ge L\left(f_i\right), \quad i = 1, 2, \ldots, M, \\
& R\left(s_i\right) \in [0, 1), \quad i = 1, 2, \ldots, M, \\
& P(h) \in [0, 1), \quad h = 1, 2, \ldots, K.
\end{aligned}
\tag{16}
$$

The proposed multiobjective optimization model for reliable cloud storage is a constrained multiobjective combination optimization problem, which is too difficult to be solved by traditional optimization algorithms. To solve this problem, one has to design a specific optimization algorithm using some particular but effective optimization technique. In this paper, we design a multiobjective particle swarm optimization algorithm to solve this proposed model.

4. Multiobjective Particle Swarm Optimization

As is well known, it is difficult to solve a constrained multiobjective problem using traditional optimization algorithms. Hence, researchers or engineers often turn to intelligent approaches to solve such multiobjective optimization problems. The mathematical model proposed in this paper for massive data storage in cloud environment is an obviously constrained multiobjective optimization problem.

Swarm or population intelligence inspired optimization techniques are becoming more and more popular. They simulate the evolution process of the nature, or the social behavior of insects swarms, flocks of birds, or schools of fish. The advantages of these optimization approaches over traditional techniques are their robustness and flexibility. These properties make swarm intelligence a successful design paradigm for algorithms that deal with increasingly complex problems. Thus, we employ a swarm intelligence optimization algorithm to solve the above model in this paper.

4.1. Brief Introduction of PSO. Particle swarm optimization (PSO), derived from the research of birds predation [33–36], is essentially an evolutionary computation technique, which optimizes the problems by iteratively improving the candidate solutions dubbed particles according to one or several given quality measurements. When solving optimization problems, PSO moves the particles around in the decision space by utilizing simple mathematical formulae over the position and velocity of the particles. The movement of each particle is under direct control of its local best known position and the global best position found so far by the population [37]. It is expected to move the particles toward the best solutions. Algorithm 1 represents a simple PSO algorithm [38].

Similar to the genetic algorithm (GA), PSO algorithms need very little additional mathematical information of the problems to be solved, start the search process from randomly generated initial population, use fitness mechanisms to evaluate the particles, do stochastic search according to evaluation value of the particle, and so on. PSO and GAs do not use the gradient information of the problem being optimized, so that they can solve a wider range of optimization problems than the classic gradient-based optimization methods.

Unlike GAs, the PSO algorithm has different information sharing mechanism. GAs share information among the chromosomes, which makes the whole population approach optimal regions uniformly. But in PSO algorithms, there are no crossover and mutation operators, and the information is provided to other particles via the current global best particles. Therefore, such one-way information sharing makes the update processes follow the current best solutions, which might help the swarm converge to the optimal solutions fast. Compared to GAs, PSO algorithms have fewer parameters and are easier to be implemented. PSO has found a wide range of applications in function optimization, neural network, fuzzy systems, and many other fields [39–41]. An extensive survey of PSO applications has been made by Poli and Engelbrecht [42, 43]. In [36, 41], several variations of PSO algorithms for multiobjective optimization problems are introduced and their convergence is also analyzed. Also, the convergence of PSO algorithm had been studied empirically in [43, 44]. Thus, according to the guidance and analysis in

(1) Randomly generate an initial swarm
(2) **while** termination criterion is not satisfied **do**
(3) **for** each particle i **do**
(4) **if** the particle i is better than *pbest* **then**
(5) $pbest_i \leftarrow$ position of i
(6) **end if**
(7) $gbest = \min(p_{\text{neighbours}})$
(8) Update velocity of particle i
(9) Update position of particle i
(10) **end for**
(11) **end while**
(12) Return the results

Algorithm 1: The PSO algorithm.

[36, 41], we construct a simple PSO algorithm to solve the proposed storage model.

4.2. PSO for the Proposed Model

4.2.1. Encoding. One of the most important issues in designing a PSO algorithm to solve the proposed model is to determine reasonable encoding of the solutions. The particles in the swarm of the PSO algorithm is a candidate storage strategy. In this paper, we use a matrix $G = (g_{ij})_{M\times N}$ to express a possible storage strategy, in which

$$g_{ij} = \begin{cases} 0, & f_i \text{ is not stored in } s_j, \\ h, & f_i \text{ is stored in the } h\text{th HDD of } s_j, \end{cases} \tag{17}$$

where $h = 1, 2, \ldots, K$.

However, using matrices to express the particles in the swarm often makes the evolution process difficult to implement. As is well known, the matrix G can be converted to a MN-vector $x = (x_1, x_2, \ldots, x_{M\cdot N})$, in which $x_{(i-1)\cdot N+j} = g_{ij}$. Therefore, we use the MN-dimensional vector to take the place of the matrix for coding the candidate storage strategies in the PSO algorithm. Another issue is to decode the results obtained by the PSO algorithm to be readable storage strategies. For $\forall k \in \{1, 2, \ldots, MN\}$, x_k can be decoded as follows:

$$g_{ij} = x_k, \tag{18}$$

where $i = \lceil k/N \rceil$ and $j = k - (\lceil k/N \rceil - 1)N$. Here, $\lceil \alpha \rceil$ is the function that its value is the smallest integer larger than or equal to α.

4.2.2. Evolution and Selection. The evolution process in the PSO algorithm is essentially to update the individuals in population. This process is controlled by the velocity of each particle in PSO, which is updated iteratively by its coordinates in the search space associated with the best solution it has achieved so far and the best position found by its neighbors. Let *pbest* be the best position of a particle, *lbest* the best position of the neighbors of the particle, and *gbest* the best position of all particles in the population. At each generation, the particle swarm optimization changes the velocity of each particle toward its *pbest* and *gbest* locations. The acceleration toward *pbest* and *gbest* locations is generated randomly and separately.

Let x be the position of current particle in the designed PSO algorithm. Let *pbest* be the best position that x has found, *gbest* the global best position of the population, and v the velocity. The particle x in the tth generation is updated as follows [33]:

$$\begin{aligned} v(t+1) = {} & \omega v(t) + c_1 r_1(t)\left(pbest(t) - x(t)\right) \\ & + c_2 r_2(t)\left(gbest_i(t) - x(t)\right), \end{aligned} \tag{19}$$

$$x(t+1) = x(t) + v(t+1), \tag{20}$$

where the parameters $r_1(t)$ and $r_2(t)$ are two separate random functions according to the generation t returning a random value uniformly generated in $[0, 1]$, respectively. $c_1, c_2 \in [0, 2]$ are the acceleration coefficients. ω is a coefficient inertia weight set to be time-varying and gradually decreasing typically from 0.9 to 0.4 [45].

It has been studied in many literatures [46, 47] that elitism has advantages in achieving better convergence in MOEAs. Hence, during the evolution process of the designed PSO algorithm, a tournament based selection strategy is employed to select particle to be updated. So as to ensure the convergence of the designed PSO algorithm, we employ the elitism strategy to select the next population from the offspring and parents and follow the guidance in [36, 41, 43].

4.2.3. Constraint Handling. When solving the constrained multiobjective model proposed in this paper, infeasible solutions will be produced inevitably. Infeasible solutions have so significant impacts on the performance of the solution PSO algorithm that they often result in failure of finding feasible solution of the optimization problem. It is necessary to handle the constraints reasonably to improve the performance of the solution PSO algorithm. Therefore, the constraint handling method in [46] is employed in the solution of PSO algorithm in this paper.

Suppose A and B are two given storage strategies. A is called to be better than B, if and only if one of the following conditions is satisfied:

(i) Both A and B are feasible, and the strategy A is with better objective value.

(ii) Both A and B are infeasible, and the strategy A is with smaller constraint violation value.

(iii) A is feasible and B is infeasible.

According to these comparison rules and Pareto domination, the individuals in the population can be sorted.

4.2.4. Designed PSO Algorithm. According to the analysis on the proposed multiobjective reliable cloud storage model and the above operators, we design a constrained multiobjective PSO algorithm (CMPSO) as Algorithm 2.

(1) Randomly generate an initial swarm S_0
(2) Initialize the parameters, *pbest* and *gbest* for each particle in S_0
(3) archive $\leftarrow \emptyset$, $t \leftarrow 0$
(4) **while** termination criterion is not satisfied **do**
(5) $t \leftarrow t + 1$
(6) $O_t \leftarrow \emptyset$
(7) archive = non_dominate(S_{t-1})
(8) **for** each particle x_i in S_{t-1} **do**
(9) **for** each particle y_j in archive **do**
(10) **if** y_j dominates $pbest_i$ **then**
(11) $pbest_i \leftarrow y_j$
(12) **end if**
(13) $gbest = \min(p_{neighbours})$
(14) Update velocity of x_i using Eq. (19)
(15) Update position of x_i using Eq. (20)
(16) Put the new position of x_i in O_t
(17) **end for**
(18) **end for**
(19) Select S_t from $S_{t-1} \cup O_t$
(20) **end while**
(21) **return** the nondominated solutions

Algorithm 2: The designed CMPSO algorithm.

5. Experiments

5.1. Experimental Settings

5.1.1. Experiment Design. The experiments are designed based on the storage prices of two cloud companies, which have different storage prices for 3 different places of company A. The detailed price can be found in Tables 1–3. When storing high security level data files, special storage devices or related encryption algorithm will be needed to process these data files. Thus, it will cost much CPU time and high level storage disk usage. It is assumed that there are 4 types of storage disks in each datacenter and the storage price is as listed in Tables 1 and 4. Table 1 lists the storage price of normal disks and Table 4 the storage price of high level disks.

In the experiments, the total size of the set of data files is set to be 120 TB, which is composed of 10 subfiles with different size and different security level. To store these data files, we conduct three storage strategies: the integral storage strategy, separate storage strategy, and splittable storage strategy. For the integral storage strategy, all the data files in a storage task are treated as an integral file to be stored. For the separate storage strategy, each subfile can be stored separately in which the subfile cannot be split. In the splittable storage strategy, the data files can be partitioned into several smaller files using the file splitters when needed. These three storage strategies are used to confirm the effectiveness of the proposed model and further investigate the influences of different storage strategies on the storage cost and reliability.

5.1.2. Parameter Settings of the Designed PSO Algorithm. According to the model in this paper, the other parameters in the PSO algorithm are adopted as follows:

(i) Population size: $N = 200$

(ii) Acceleration coefficients: $c_1 = c_2 = 2.0$ and $\omega = 0.9 - t/2\text{MAX}_{\text{gen}}$, where t is current generation and MAX_{gen} is the maximum generation

(iii) Selection strategy: the next population is selected using elitism strategy according to the fast nondominated sorting approach [46]

(iv) Stopping criterion: the algorithm will be stopped when the evolution generation reaches 2000 or the solutions found by the algorithm do not change in 30 continuous generations.

5.2. Experimental Results. In the experiment of the integral storage, the PSO algorithm is employed to solve the proposed multiobjective model, in which all the test data files are treated as an integral file. The feasible storage solutions found by the PSO algorithm are as shown in Figure 2 and Table 5. Under this storage strategy, all data files were stored in the same disk of a server. The total cost has direct relationship with the storage reliability and the data security level. In order to satisfy the constraint $L(f_i) \leq R(Y_{ij})$, all the data files should be stored in the HD which can fit the highest security level among all the data files. It can be seen vividly from Figure 2 that the total cost of storing the test data files is increasing rapidly with the storage reliability. Practically, the higher the storage reliability is, the more it will cost. The storage cost is proportional to the highest level of the data files, which has also direct impacts on the storage reliability. It is because the suitable target servers should be explored when storing these files into certain disk and the transformation process of these data files will significantly reduce the storage reliability. What is more, if the suitable target server cannot be found, this storage strategy will not be able to be carried out. Therefore,

Table 1: Storage price of cloud company A.

Name	CPU (s)	Memory (GB)	Price ($/hour)		
			Place 1	Place 2	Place 3
m1.small	1	1.7	0.085	0.095	0.095
m1.large	4	7.5	0.34	0.38	0.38
m1.xlarge	8	15	0.68	0.76	0.76
c1.medium	5	1.7	0.17	0.19	0.19
c1.xlarge	20	7	0.68	0.76	0.76
m2.xlarge	6.5	7	0.50	0.57	0.57
m2.2xlarge	13	34.2	1.00	1.14	1.14
m2.4xlarge	26	68.4	2.00	2.28	2.28

Table 2: Storage price of cloud company B.

Name	CPU (s)	Memory (GB)	Price ($/hour)
			Place 3
X-small	0.5	0.5	0.095
Small	1	1	0.19
Medium	2	2	0.38
Large	4	4	0.76
X-large	8	8	1.52
XX-large	16	16	3.04

Table 3: Transmission price.

Cloud service providers	Price ($/GB)	
	Data incoming	Data outgoing
Company A (all)	0.10	0.15
Company B	0.00	0.29

Table 4: Price of high level storage disks.

Type	Volume (GB)	Price ($/month)	Throughput (MB/second)
P10	128	27.19	100
P20	512	100	150
P30	1024	186.45	200

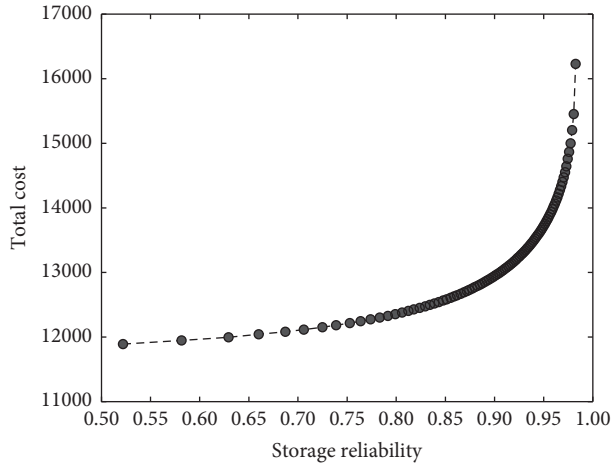

Figure 2: Integral storage of data files.

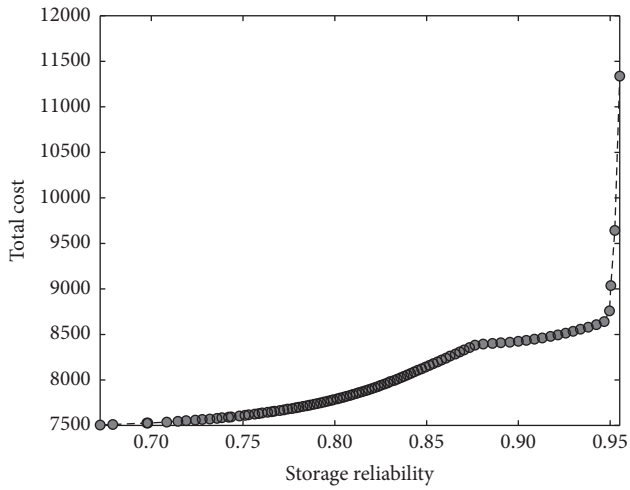

Figure 3: Separable storage strategy with indivisible subfiles.

it is unpractical and unreasonable to store all the data files as an integral file.

Another storage strategy is that each subfile is treated to be inseparable and can be stored independently. The data files submitted by the client might be stored in different servers or places. Hence, data files with different security level can be handled specifically, so that the cost can be saved. Also, it is easy to find feasible storage solutions with higher utilization rates of storage spaces. The experimental results are as shown in Figure 3 and Table 5. Using the proposed storage model to store these test data files, one can find feasible storage solutions easily. From Figure 3, it can be seen that when the storage reliability is less than 0.95, the total cost grows smoothly. When the storage reliability is more than 0.95, the total cost will increase rapidly. Compared with the integral storage strategy, it can be found from Table 5 that the cost of the separable storage strategy with each subfile to be inseparable is much lower than that of the integral storage strategy. These results indicate that the separable storage strategy is much practical and the proposed multiobjective model is correct and effective.

As there are many file splitting techniques, each data file can be partitioned into smaller data files by using these

Table 5: Storage cost of data files with different storage strategies.

Storage reliability	Integral storage ($)	Separable storage with divisible subfiles ($)	Separable storage with indivisible subfiles ($)
0.50–0.70	$1.21e+4$	$7.54e+3$	$6.93e+3$
0.70–0.80	$1.23e+4$	$7.71e+3$	$7.08e+3$
0.80–0.85	$1.25e+4$	$7.99e+3$	$7.26e+3$
0.85–0.90	$1.27e+4$	$8.39e+3$	$7.49e+3$
0.90–0.95	$1.35e+4$	$8.58e+3$	$7.71e+3$
0.95–1.00	$1.46e+4$	$1.06e+4$	$8.69e+3$

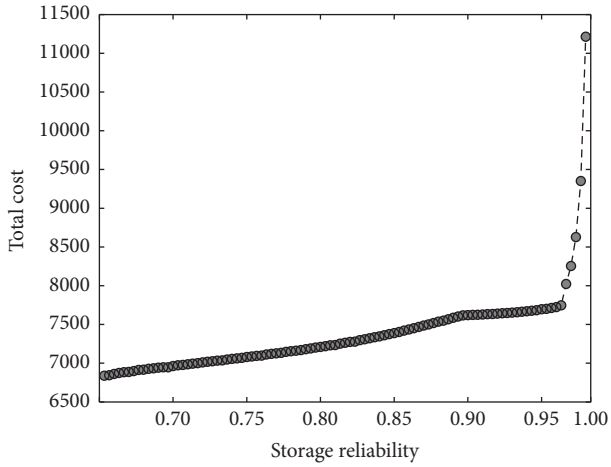

Figure 4: Separable storage strategy with divisible subfiles.

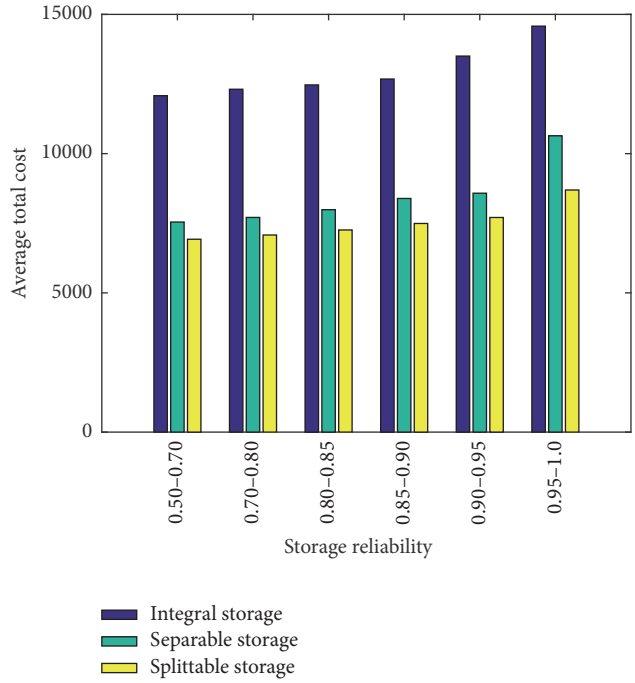

Figure 5: Comparisons of the average costs of the storage schemes obtained by the storage strategies.

techniques while being stored. In the further experiment of the separable storage strategy, each data file is treated to be divisible. When storing these files, each data file will be partitioned according to the available storage space in servers. While storing, each subfile will be partitioned into several parts with the same security level when needed. Therefore, the data file can be divided for any usable space during the storing process, which can make the solution algorithm find feasible solution much easier. This way can also improve the space utilization rates effectively. The experimental results can be found in Table 5 and Figure 4. Comparing the results shown in Figures 3 and 4, one can find vividly that, with the storage reliability growing, the total cost in Figure 4 grows more slowly. Also, in this experiment, the PSO algorithm can find much more storage solutions. As shown in Figure 4, the solutions obtained by the PSO algorithm are uniformly distributed, which indicates that many more varied storage schemes can be obtained with high probability coupled with the splittable storage strategy. From Table 5, it can be seen that the storage strategy with divisible subfiles can reduce the storage cost efficiently.

Comparing the results listed in Table 5, we can find that, to achieve given storage reliability, splittable storage strategy costs the least among the 3 storage strategies conducted in the experiments. Figure 5 intuitively shows the differences among these three storage strategies. Furthermore, the results of these experiments can demonstrate the validity and correctness of the proposed multiobjective data storage model, which can also indicate that the PSO algorithm can solve the proposed multiobjective model effectively.

6. Conclusions

Data has become a kind of important resources, which has increasingly significant effects on the Internet companies and telecommunication enterprises. The volume of the data resources is growing so rapidly that their storage is challenging the quality of service of the storage systems significantly. Based on the analysis of data storage in the cloud environment, we built a multiobjective optimization model considering storage cost and reliability simultaneously. To store a set of data files, the total cost is composed of storage cost, communication cost, and migration cost. The device stability and transmission reliability are considered in the evaluation of storage reliability. To solve the proposed multiobjective optimization model and find feasible storage solutions, a PSO algorithm is tailor-made. Different experiments are made to validate the correctness of the proposed

model. In the experiments, the data files to be stored are handled in different ways in advance. Experimental results confirm that the proposed model is positive and effective for cloud storage.

Furthermore, an obvious phenomenon that emerged during the experiments is that how to preprocess the data files has significant effect on the experimental results or storage strategies obtained by the PSO algorithm. If the data files submitted by the clients are treated as an integral file, all the data files need to be stored in a server which has enough available storage space with the reliability meeting the demand of the highest security level among the data files. Hence, it costs much in searching the servers with sufficient available storage space of certain reliability. However, when the available storage fragments in the cloud storage systems were too small and scattered, it might lead to failure in finding a suitable fragment to store all these data files. These data files preprocessing strategy will lead to low utilization ratio of the storage space. The other two preprocessing strategies can effectively raise the storage space utilization, because each data file can find suitable storage fragment easily. Therefore, there will be more possible storage schemes and the best scheme can be found by using the model and PSO algorithm proposed in this paper.

The main goal of the model proposed in this paper is to find optimal storage scheme for storing the submitted data files. From the experiments, it can be seen vividly that the cost and reliability of the storage scheme for the submitted data files strongly depend on the preprocessing strategies of the data files, which also impacts the utilization efficiency of the storage space. The proposed model in this paper should be coupled with reasonable data file preprocessing method, so that it can improve the quality of service and the resource utilization efficiency of the cloud storage system effectively.

Nomenclature

N: Number of severs
M: Number of files to be stored
f_i: ith file to be stored
$L(f_i)$: Security level of f_i
$A_i(s_j)$: Available storage space of jth HD in s_i
$P(h)$: Storage reliability of hth HD
$\omega(s_i, s_j)$: Communication price between s_i and s_j
sign$(\cdot)$: Sign function, where sign$(y) = \{1,\ y > 0;\ 0,\ y \le 0\}$
$\zeta_s(f_i)$: Storage cost of f_i
$\zeta_{\text{mig}}(f_k, s_u, s_j)$: Migration cost of f_k from s_u to s_j
$\zeta_{\text{com}}(f_k)$: Communication cost of storing f_k
$\zeta(F, S, t)$: Total cost of storage service cost for F
$R_E(s_i, s_j)$: Equipment stability of link $s_i \leftrightarrow s_j$
$\xi(f_i, s_j, s_b)$: Storage reliability of f_i
s_i: ith server
d_{ij}: Distance between s_i and s_j
F: Set of files to be stored
Y_{ij}: f_j is stored in Y_{ij}th HD of s_i
K: Number of the types of HDs in a datacenter
$R(s_i)$: Reliability of s_i
TR(s_i, s_j): Transmission reliability between s_i and s_j
G: Candidate storage strategy
$X(f_i, s_j)$: Data volume of f_i stored in s_j
$\zeta_s(F, t)$: Storage cost of storing F
$\zeta_m(F)$: Migration cost in storing F
$\zeta_c(F)$: Total communication cost of storing F
$R_T(s_i, s_j)$: Transmission reliability of link $s_i \leftrightarrow s_j$
$R_S(s_i)$: Reliability of software in s_i
$\xi(F, S)$: Reliability of storing F.

Disclosure

This paper is an extended version of our conference paper [48]. The extension includes the following: (1) we summarize the related work on multiobjective reliable storage; (2) we update the proposed multiobjective reliable storage optimization model with more precise objective function; (3) we analyze the evaluation of storage reliability in detail; (4) a tailor-made PSO algorithm is designed in this paper; (5) we conduct new experiment with new storage strategy to evaluate the correctness and effectiveness of our model.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61502371 and 61472311), the Fundamental Research Funds for the Central Universities (BDZ011401, JB151005, and JB161003), the Novel Technology Research of Universities Cooperation Project sponsored by the State Key Laboratory of Satellite Navigation System and Equipment Technology (KX152600027), Fundamental Research Project sponsored by State Administration of Science, Technology and Industry for National Defense (JCKY2016110C006), and the Basic Research Program of Sichuan Province of China (2017JY0209).

References

[1] J. Brodkin, *Gartner: Seven Cloud-computing Security Risks*, Infoworld, 2008.

[2] S. Zhang, S. Zhang, X. Chen, and S. Wu, "Analysis and research of cloud computing system instance," in *Proceedings of the 2nd International Conference on Future Networks (ICFN '10)*, pp. 88–92, IEEE, Hainan, China, 2010.

[3] B. Hayes, "Cloud computing," *Communications of the ACM*, vol. 51, no. 7, pp. 30–34, 2008.

[4] S. Gurumurthi, "Architecting storage for the cloud computing era," *IEEE Micro*, vol. 29, no. 6, pp. 68–71, 2009.

[5] J. Y. Wu, L. D. Ping, and X. P. Ge, "Cloud storageas the infrastructure of cloud computing," in *Proceedings of the International Conference on Intelligent Computing and Cognitive Informatics (ICICCI '10)*, pp. 380–383, 2010.

[6] P. Buxmann, T. Hess, and S. Lehmann, "Software as a service," *Wirtschaftsinformatik*, vol. 50, no. 6, pp. 500–503, 2008.

[7] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, Morgan & Claypool Publishers, 2009.

[8] L. Fan, S. Meng, X. Liu, and Y. Liang, "Improved CTT-SP algorithm with critical path method for massive data storage in scientific workflow systems," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 30, no. 8, Article ID 1659023, 22 pages, 2016.

[9] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[10] S. Aameek, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of the ACM/IEEE Conference on High Performance Computing (SC '08)*, pp. 1–12, Austin, Tex, USA, November 2008.

[11] M. Vukolic and I. Z. Laboratory, "Reliable distributed storage," *Computer*, vol. 42, no. 4, p. 6067, 2009.

[12] Y. Deng, F. Wang, N. Helian, S. Wu, and C. Liao, "Dynamic and scalable storage management architecture for Grid Oriented Storage devices," *Parallel Computing*, vol. 34, no. 1, pp. 17–31, 2008.

[13] X. Y. Zhang, K. Q. Li, and Y. Zhang, "Minimum-cost virtual machine migration strategy in datacenter," *Concurrency and Computation-Practice & Experience*, vol. 27, pp. 5177–5187, 2015.

[14] R. W. Ahmad, A. Gani, S. H. A. Hamid et al., "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[15] J. Dong, H. Wang, Y. Li, and S. Cheng, "Virtual machine scheduling for improving energy efciency in IaaS cloud," *China Communications*, vol. 11, no. 3, Article ID 6825253, pp. 1–12, 2014.

[16] M. Sanchez-Artigas, "Toward efficient data access privacy in the cloud," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 39–45, 2013.

[17] Y. X. Guo and Y. G. Fang, "Electricity cost saving strategy in data centers by using energy storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1149–1160, 2013.

[18] B. Mao, H. Jiang, S. Wu, Y. Fu, and L. Tian, "Read-performance optimization for deduplication-based storage systems in the cloud," *ACM Transactions on Storage (TOS)*, vol. 10, no. 2, pp. 193–206, 2014.

[19] W. Liu, S. Peng, W. Du, W. Wang, and G. S. Zeng, "Security-aware intermediate data placement strategy in scientific cloud workflows," *Knowledge and Information Systems*, vol. 41, no. 2, pp. 423–447, 2014.

[20] M. Y. Guo, "Static and dynamic locality optimizations using integer linear programming," *ACM SIGPLAN Notices*, vol. 49, no. 5, p. 83, 2014.

[21] Y. Wang, P. Lu, and K. B. Kent, "WaFS: a workflow-aware file system for effective storage utilization in the cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2716–2729, 2015.

[22] M. Jarrah, M. Jaradat, Y. Jararweh, M. Al-Ayyoub, and A. Bousselham, "A hierarchical optimization model for energy data flow in smart grid power systems," *Information Systems*, vol. 53, pp. 190–200, 2015.

[23] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proceedings of the 17th International Workshop on Quality of Service (IWQoS '09)*, pp. 1–9, IEEE, July 2009.

[24] M. Alves, C. V. Damasio, W. Nejdl, and D. Olmedilla, "A distributed tabling algorithm for rule based policy systems," in *Proceedings of the 7th IEEE International Workshop on Policies for Distributed Systems and Networks*, pp. 123–132, June 2006.

[25] T. C. Jepsen, "The basics of reliable distributed storage networks," *IT Professional*, vol. 6, no. 3, pp. 18–24, 2004.

[26] C. E. Perkins, *Ad Hoc Networking*, Addison-Wesley, Boston, Mass, USA, 2001.

[27] V. PournaghshbandL, L. Kleinrock, P. Reiher, and A. Afanasyev, "Controlling applications by managing network characteristics," in *Proceedings of the IEEE International Conference on Communications (ICC '12)*, pp. 1085–1090, IEEE, Ottawa, Canada, June 2012.

[28] F. Kurose James and W. Ross Keith, *Computer Networking: A Top-Down Approach*, Pearson Education, 6th edition, 2012.

[29] K. C. Mansfield and J. L. Antonakos, *Computer Networking from LANs to WANs: Hardware, Software, and Security*, Course Technology Cengage Learning, Boston, Mass, USA, 2010.

[30] S. Krishnamurthy and A. P. Mathur, "On the estimation of reliability of a software system using reliabilities of its components," in *Proceedings of the 8th International Symposium on Software Reliability Engineering*, pp. 146–155, Albuquerque, NM, USA, November 1997.

[31] IEEE Std 1633, "IEEE Recommended Practice on Software Reliability," IEEE, 2008.

[32] D. W. Carman, A. A. Dolinsky, M. R. Lyu, and J. S. Yu, "Software reliability engineering study of a large-scale telecommunications software system," in *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pp. 350–359, Toulouse, France, October 1995.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, November 1995.

[34] J. Kennedy, "The particles warm: sociala daptation of knowledge," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 303–308, Indianapolis, Ind, USA, April 1997.

[35] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 69–73, IEEE Press, Anchorage, Alaska, USA, May 1998.

[36] M. Mussetta, P. Pirinoli, S. Selleri, and R. E. Zich, "Meta-PSO for multi-objective EM problems," in *Multi-Objective Swarm Intelligent Systems: Theory & Experiences*, N. Nedjah, L. dos Santos Coelho, and L. de Macedo Mourelle, Eds., vol. 261 of *Studies in Computational Intelligence*, pp. 125–150, Springer, Berlin, Germany, 2010.

[37] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[38] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence*, Natural Computing Series, pp. 43–85, Springer, Berlin, Germany, 2008.

[39] M. Kotinis, "A particle swarm optimizer for constrained multi-objective engineering design problems," *Engineering Optimization*, vol. 42, no. 10, pp. 907–926, 2010.

[40] X. Wang, C. Wang, and M. Yu, "Swarm intelligence algorithm for interconnect model order reduction with sub-block structure preserving," *International Journal of Systems Science*, vol. 47, no. 9, pp. 2178–2192, 2016.

[41] S. Selleri, M. Mussetta, P. Pirinoli, R. E. Zich, and L. Matekovits, "Differentiated meta-PSO methods for array optimization," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 1, pp. 67–75, 2008.

[42] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, Article ID 685175, 10 pages, 2008.

[43] A. P. Engelbrecht, "Particle swarm optimization with crossover: a review and empirical analysis," *Artificial Intelligence Review*, vol. 45, no. 2, pp. 131–165, 2016.

[44] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1945–1960, Washington, DC, USA, July 1999.

[45] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 84–88, IEEE, San Diego, Calif, US, 2000.

[46] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[47] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[48] X. Liu, L. Fan et al., "PSO based multiobjective reliable optimization model for cloud storage," in *Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pp. 2263–2269, Liverpool, UK, October 2015.