TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/10 pp13-23
Volume 19, Number 1, February 2014

DR-Cloud: Multi-Cloud Based Disaster Recovery Service

Yu Gu, Dongsheng Wang*, and Chuanyi Liu

Abstract: With the rapid popularity of cloud computing paradigm, disaster recovery using cloud resources becomes an attractive approach. This paper presents a practical multi-cloud based disaster recovery service model: DR-Cloud. With DR-Cloud, resources of multiple cloud service providers can be utilized cooperatively by the disaster recovery service provider. A simple and unified interface is exposed to the customers of DR-Cloud to adapt the heterogeneity of cloud service providers involved in the disaster recovery service, and the internal processes between clouds are invisible to the customers. DR-Cloud proposes multiple optimization scheduling strategies to balance the disaster recovery objectives, such as high data reliability, low backup cost, and short recovery time, which are also transparent to the customers. Different data scheduling strategies based on DR-Cloud are suitable for different kinds of data disaster recovery scenarios. Experimental results show that the DR-Cloud model can cooperate with cloud service providers with various parameters effectively, while its data scheduling strategies can achieve their optimization objectives efficiently and are widely applicable.

Key words: multi-cloud; disaster recovery; DR-Cloud

1 Introduction

More and more services rely on IT systems at present, and some of them, such as financial service and health care service, are critical to their customers and society. Even a very short period of downtime or very small amount of data loss could result in huge economic losses or social problems. Therefore, many important business and public services use disaster recovery mechanism to protect critical data and minimize the downtime caused by catastrophic system faults. Different kinds of technologies are adopted in disaster recovery systems[1], such as periodically backup or continuous synchronization of data and preparing standby system in geographically separated places.

However, building and running a private data center for the purpose of data disaster recovery can be quite costly and time-consuming. Research of Greenberg et al.[2] has shown that the cost of a data center includes purchasing servers and infrastructure, maintaining facilities, and employing human resource. And there is no difference in cost no matter whether the service is standby or in use. So, if a service provider chooses to build disaster recovery data center on his own, it requires huge investment and leads to tremendous waste brought by idling resources due to the infrequent yet urgent needs of disaster recovery.

During the past decade, cloud computing has rising as a new service paradigm, which becomes more and more popular. A huge number of services are built on cloud platform now. Essentially, these services utilize resources of cloud platform with a pay-as-

• Yu Gu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: gy98@mails.tsinghua.edu.cn.
• Dongsheng Wang is with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China. E-mail: wds@tsinghua.edu.cn.
• Chuanyi Liu is with the Software School, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: liuchuanyi@gmail.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2013-12-03; accepted: 2013-12-04

you-go pricing model[3]. Wood et al.[4] had proved that the cost of using existing cloud resources to perform data disaster recovery, also called disaster recovery as a cloud service, is far less than the cost of building and maintaining disaster recovery data center on one's own. The on-demand nature of cloud computing vastly reduces the cost of disaster recovery, whose peak resource demands are much higher than average demands.

However, data disaster recovery is a kind of service with highest data reliability requirements. How to perform data disaster recovery service using cloud computing paradigm to maximizing the data reliability while reducing cost is still a challenge. No difference with other computer systems, cloud computing system can also encounter certain risks such as software bug, hardware fault, network intrusion, human-caused damage, natural disasters, etc. All of these risks may lead to cloud service interruption, even loss of data in some cases. To ensure high data reliability, cloud service providers have deployed many data protection strategies. For example, popular distributed storage systems used in cloud platforms currently, such as Amazon S3[5], Google GFS[6], and Apache HDFS[7], have adopted 3-replicas data redundant mechanism by default. However, with an entire data center failure, data may still be lost. To avoid this problem, some cloud service providers use geographical data dispersion to protect the most critical data, while data centers in different locations owned by one cloud service provider mostly use similar software stack, infrastructures purchased in bulk, operation mechanism, and management team. There are still risks of multiple data center failures in a same period due to some common causes across data centers. Also, the number of data centers owned by one cloud service provider is limited. When some of them become unreachable, the surviving data centers may not be applicable to some customers due to the long distance in geography and network, especially in the emergency of restoring data from disaster. Thus, no matter how many preventive measures have been taken, the possibility of data reliability disruption in a cloud cannot be ignored. According to some reports[8,9] in the public, even the most advanced cloud services have encountered several times of wide area outages and resulted in many public services down.

Therefore, we think the best solution of disaster recovery service is to utilize multiple data centers from different cloud service providers. In this pattern, one cloud service provider who provides data disaster recovery service to the public can lease resources from other cloud service providers in the pay-as-you-go mode. Thus, data disaster recovery is no longer limited to the data centers of one cloud service provider. A disaster recovery service provider can selectively backup data to its own data centers or other cloud service provider's data centers. By using an appropriate backup data scheduling strategy, the disaster recovery service provider can achieve better effect, which is to minimize the cost while promoting the service quality.

In this paper, we propose a practical system model for multi-cloud based disaster recovery service, which we call DR-Cloud (DR for disaster recovery). DR-Cloud utilizes resources of multiple cloud service providers cooperatively. Customers only need to deal with the DR-Cloud, using very common and unified service interface.

2 Related Work

Cumulus[10] can back up a file system to cloud storage, using a least-common-denominator cloud interface, thus support many kinds of cloud services. It uses only one cloud to maintain one backup, and focuses on the mechanism in local file system, not the cloud platform.

Wood et al.[4] proposed a new cloud service model, i.e., disaster recovery as a cloud service, which leverages the virtual platforms in cloud computing to provide data disaster recovery service. They created a disaster recovery cloud model for web site applications which illustrated that data backup built on top of cloud resources can greatly reduce the cost of data disaster recovery for corporations. However, they didn't study on how to further improve the service quality using multiple clouds.

Some researchers focus on how to backup data in cloud computing environment, such as Bajpai et al.[11] and Zhang and Zhang[12]. CABdedupe[13] employs the deduplication technology to remove redundant data from transmission during backup and recovery, which reduces time and network traffic consumption. Li et al.[14] proposed a data replication strategy which uses an incremental replication method to reduce network and storage cost.

R-ADMAD[15] exploits ECC codes to encode data chunks, then distributes them among storage nodes

within a data center, and uses a distributed dynamic restoring process to handle data recovery. It greatly reduces storage occupancy than replication-based schemes, while still ensuring comparable high data reliability.

PipeCloud[16] is a pipelined synchronous replication-based disaster recovery system, designed to increase throughput and reduce response time while providing zero data loss consistency guarantees. But it uses only one cloud to store data replicas.

Nguyen et al.[17] proposed a differentiated replication strategy that can handle customers' different requirements. The strategy provides data reliability assurance for different service types, and differentiated backup schemes. It also enhances the utilization of cloud resources.

Researches on leveraging resources from multiple cloud providers for data replicas have also been conducted. Cachin et al.[18] analyzed the data integrity and data security in the intercloud mode. They adopted fault-tolerant protocol and secure access control protocol to ensure data integrity and confidentiality in multiple cloud platforms. Metastorage[19] focused on the data consistency and communication latency of data replicas among multiple cloud providers. They established a data access priority queue to reduce the communication delay between clouds, and conjectured that the smaller file size, the better assurance of data consistency.

None of the above approaches has proposed a practical disaster recovery service model that can utilize multiple cloud resources and optimize both backup cost and recovery time while ensuring high data reliability.

3 System Model of DR-Cloud

3.1 Design philosophy

Based on the above analysis, we can see the multi-cloud mode is a suitable approach to the data disaster recovery service. But it also brings in some new challenges to design a practical system model, which are summarized as follows:

• The model should be able to utilize a variety of cloud platforms. Currently different cloud service providers such as Amazon, Google, and Microsoft offer different types of infrastructures, service interfaces, storage & network resources, and charging models[20]. By supporting various kinds of cloud platforms as many as possible, the disaster recovery service provider can leverage more resources and accommodate more customers in wide geographic area.

• To reduce service barriers, the model must be transparent to end users. That is, there should be no difference in usage between this mode and ordinary one-cloud mode, from the users' perspective. Customers should have no need to consider internal details between multiple clouds. It is the service provider's responsibility to negotiate with other cloud service providers on technical and financial parameters and perform the data disaster recovery tasks.

• The model should provide enough space for optimization to achieve enough advantage. In this case, that means it can effectively select the resources from multiple cloud providers to store backup data such that the cost of data disaster recovery stays as low as possible, and the recovery time remains as short as possible when disaster actually occurs, while ensuring high data reliability.

To meet all these challenges, we design a multi-cloud based data disaster recovery service model, named DR-Cloud, from the perspective of the service provider.

By using a least-common-denominator cloud storage interface like Cumulus[10], i.e., get and put of complete files, DR-Cloud allows a service provider to leverage resources from other cloud service providers to build up data disaster recovery service to satisfy more customers' requirements, enhance its market reputation, and increase its corporate revenue. To ensure high data reliability, DR-Cloud adopts 3-replicas data redundancy mechanism. When conducting data backup, the disaster recovery service provider will intelligently pick and choose resources from multiple cloud service providers including itself, based on a certain scheduling strategy. The decision is made according to the storage cost, network communication cost, and data recovery speed, such that it can meet two essential optimization goals: reducing data backup cost and shortening data recovery time as much as possible. All the internal procedures are transparent to customers. In this mode, the customers of DR-Cloud can be individuals or corporations, also some other cloud service providers.

3.2 DR-Cloud architecture

The overall architecture of DR-Cloud is shown in Figs. 1 and 2, which represent the data backup and recovery models respectively.

In general, DR-Cloud is comprised of data disaster recovery service customers and multiple cloud service providers.

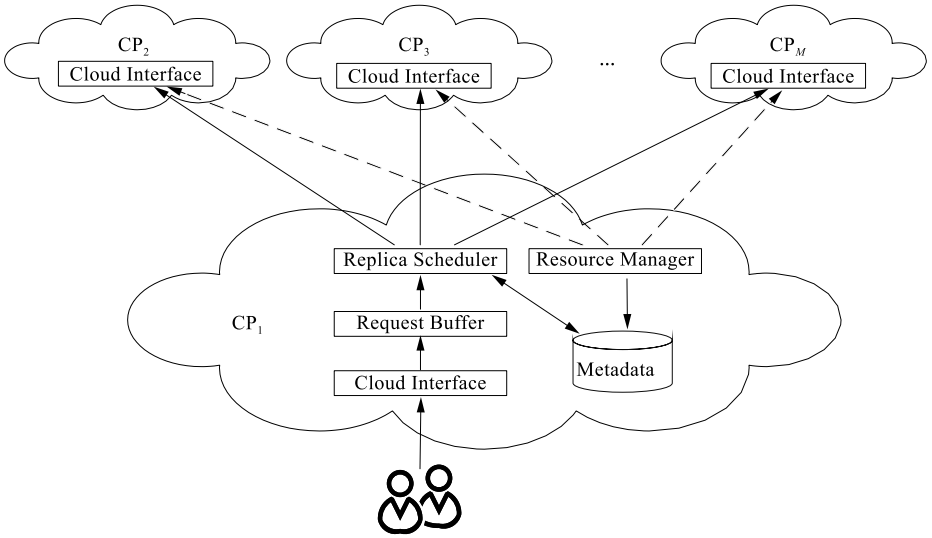

Fig. 1 Data backup model.

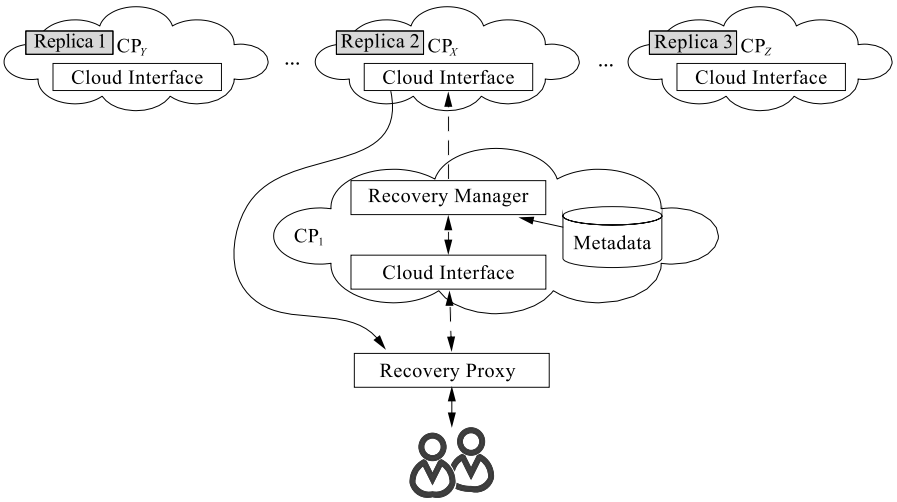

Fig. 2 Data recovery model.

In Fig. 1, The CP_1 represents the data disaster recovery service provider. All customers are the users of CP_1, which can be individuals, enterprises or even other cloud service providers. They have appropriate accounts and privileges of CP_1. And CP_2-CP_M are other cloud service providers that provide common cloud resources to CP_1. Cloud Interface of each CP is the least-common-denominator cloud storage interface, which receives/sends data from/to users. Request Buffer of CP_1 holds data backup requests arriving in one time period, which are waiting to be scheduled. Replica Scheduler reads requests from Request Buffer, makes three replicas each, then dispatches them to those CPs. Resource Manager is responsible for monitoring the changes of resource usages of all CPs. And Metadata is a database containing information about replicas' locations and CPs' resource usages.

In Fig. 2, CP_X, CP_Y, and CP_Z represent the cloud service providers who contain replicas of a data recovery request. Recovery Manager of CP_1 receives and checks the recovery request, then selects an appropriate CP who contains at least one replica of that request. And Recovery Proxy is an agent installed to the customer side, which in charge of restoring data from CPs.

3.3 Data backup procedure

The execution flow of the data backup procedure in DR-Cloud is shown in Fig. 1, also described as follows:

(1) Customers send their data backup requests, which include data to be stored and store duration parameters, to CP_1 through CP_1's Cloud Interface.

(2) CP_1 receives those requests and keeps them in Request Buffer.

(3) CP_1 checks customers' accounts and privileges and refuses illegal requests.

(4) Replica Scheduler triggers its scheduling strategy once per unit time or when there are certain number of requests in Request Buffer.

(5) Scheduling strategy of Replica Scheduler reads all requests' data size and store duration parameters from Request Buffer, makes three replicas for each data, and intelligently determines storage location of each replica. Then it generates an overall data backup scheme.

(6) According that scheme, Replica Scheduler sends all replicas to their destinations respectively, which may be CP_1's own data centers or other CPs' data centers.

(7) Other CPs record CP_1's occupancy of their resources, which are used to charge CP_1 later.

(8) When finishing transmission of data replicas in the scheme, CP_1 commits all data replicas' location information to Metadata database.

(9) CP_1 deletes all fulfilled requests from Request Buffer, and records those requests' information for charging customers later.

(10) CP_1 sends acknowledges to customers to return corresponding data handles, which can be used to restore certain data from CP_1.

From the above description, the backup data is sent to CP_1 first, then replicated to 3 copies and dispatched by CP_1 to final destinations. Actually, there is another flow to be chosen: only backup data's parameters are sent to CP_1 first, then CP_1 generates the backup scheme and returns it to customers, customers send their data to corresponding CPs according the scheme finally. In this flow, a customer needs to send 3 copies of his data to multiple CPs. It could reduce throughput and increase response time of the customer's application, and consume more bandwidth of the customer. In addition, it would be more complicated to maintain consistency between the actual data locations and CP_1's Metadata database, due to data transmission failure. So we don't think that it's a good design choice.

From customers' perspective, the data backup procedure in DR-Cloud is very similar to the common data store procedure in one-cloud mode. Customers have no need to know the details of the multi-cloud processes, also have no need to record actual data locations. This reduces the service barriers of the DR-Cloud.

3.4 Data recovery procedure

The execution flow of the data recovery procedure in DR-Cloud is shown in Fig. 2, also described as follows:

(1) A customer sends a data recovery request to CP_1 through CP_1's Cloud Interface, with the corresponding data handle got from CP_1 before.

(2) CP_1 receives the request, checks the customer's account and privilege, and refuses the request if it's illegal.

(3) Recovery Manager of CP_1 looks up the data handle in Metadata database to find backup locations of that data.

(4) CP_1 compares the 3 replicas' locations of that data, and chooses the fastest location, i.e., the one with the highest recovery bandwidth. Supposing the location is CP_X.

(5) Recovery Manager informs CP_X to set up a temporary access authorization, which can only be used to read that replica, and can only be used by the Recovery Proxy of that customer.

(6) CP_X establishes the authorization, then notifies CP_1.

(7) CP_1 sends the data location and authorization information to the Recovery Proxy of that customer.

(8) The Recovery Proxy pulls data from CP_X, then notifies CP_1.

(9) CP_1 notifies CP_X to destroy that temporary access authorization and records the recovery request's information for charging the customer later.

(10) CP_X records network traffic consumption of the recovery request, which is used to charge CP_1 later.

In particular, if CP_X is CP_1 itself, then no temporary access authorization needs to be set up by CP_1, the Recovery Proxy will use the customer's account of CP_1 directly.

From the above description, the data is restored from CP_X directly, not retransmitted by CP_1, which reduces recovery time when disaster occurs. We think this is a better choose because of the extreme importance of recovery time. As we know, even a very short system down time could cause huge economic loss, thus a disaster recovery service should attempt to shorten the recovery time as much as possible.

The temporary access authorization mechanism used by this procedure is very important to ensure data security in DR-Cloud. Although CP_1 has permanent accounts of other CPs, these accounts cannot be lent to CP_1's customers, which will lead to security risks. In

addition, the interface that Recovery Proxy exposes to customers is the same as the common cloud storage getting data interface. By using Recovery Proxy to deal with CPs, the internal processes of restoring data from multiple clouds are transparent to customers. It's just like the data restore procedure in one-cloud mode.

4 Scheduling Strategy in DR-Cloud

Inside the architecture of DR-Cloud, the scheduling strategy of CP_1's Replica Scheduler is the most important, which determines the effect of DR-Cloud. The responsibility of scheduling strategy in DR-Cloud is to reduce data backup cost and shorten data recovery time as much as possible.

We think that the scheduling strategy should be a one-time scheduling pattern. That is, one data replica will stay in one CP until expired, and won't be transmitted to another CP during its lifetime, unless the CP it belongs to is permanently down. When some better storage resources have been released, relocating a replica to there will consume additional network traffic. In fact, later backup requests can also utilize those resources effectively, and the gain of relocating is very trivial compared to the network resource waste.

In DR-Cloud, all CPs have their storage resource limitations for data disaster recovery service of CP_1, which may not be the limitations of their total storage resources for all kind of services. As a result, backup requests will compete with each other for advantaged storage sources. Because of the diversity of requests' parameters, i.e., data size and store duration, it is a typical multi-objective combinatorial optimization problem to make excellent scheduling decision.

So we set the optimization objective of DR-Cloud's scheduling strategy to be a combination of two factors: data backup cost and recovery time. It should be emphasized that the data backup cost is the cost spent by CP_1, not the customer. The data backup fee which the customer pays CP_1 is a simple function of data size and store duration, offered by CP_1 based on market factors. Thus, the lower the backup cost of CP_1 is achieved, the more profit CP_1 gains.

To further discuss the scheduling problem, we established a formal description of these factors in the following.

Assuming there are M cloud service providers: CP_1, CP_2, $\cdots$, CP_M. The i-th CP's price of storage space, price of network traffic, data recovery bandwidth, and storage resource limitation are SP_i (dollar/(GB·h)), TP_i (dollar/GB), BW_i (Gbit/s), and SL_i (GB), respectively. And we use L_i (s) to denote the recovery transmission start-up delay of CP_i, which is spent in setting up temporary access authorization and etc.

In a certain scheduling period, there are N data backup tasks, i.e., valid requests, to be processed by CP_1: $T_1, T_2, \cdots, T_N$. The j-th task's data size and store duration are S_j (GB), and D_j (h), respectively.

The cost of data backup includes data storage cost and network communication cost. So the total backup cost of all tasks in this scheduling period, which is represented by COST, can be formulated as follows:

$$\mathrm{COST}=\sum_{j=1}^{N}\sum_{i=1}^{M}(S_j\times D_j\times R_{ij}\times \mathrm{SP}_i+S_j\times E_{ij}\times \mathrm{TP}_i).$$

In this formula, R_{ij} denotes the count of data replicas stored in CP_i for task T_j. While E_{ij} is a Boolean variable. It should be 1 when CP_i contains task T_j's data replica, i.e., when R_{ij} is equal to or greater than 1, otherwise should be 0.

Obviously, for $\forall j \in[1, N]$, we have

$$\sum_{i=1}^{M} R_{ij} = 3.$$

In DR-Cloud, we try to store replicas of one task in multiple CPs, so $\forall i \in[1, M]$ and $\forall j \in[1, N]$, we have

$$0 \leqslant R_{ij} \leqslant 2.$$

The recovery time, which is represented by Recovery Time Objective (RTO), is the length of time between when a customer sends the recovery request to CP_1 and the customer gets all the data from a certain CP. The main parts of it include recovery transmission start-up delay and transmission time. Other parts such as connecting and lookup time are very little constants. So we define RTO as follows:

$$\mathrm{RTO} = \sum_{j=1}^{N} \underset{i=1}{\overset{M}{\mathrm{MIN}}}(S_j \times E_{ij}/\mathrm{BW}_i + L_i \times E_{ij}).$$

And we also have a formula representing storage resource limitations. That is, for $\forall i \in[1, M]$, we have

$$\sum_{\text{begin}}^{\text{now}}\sum_{j=1}^{N}(S_j \times R_{ij}) \leqslant \mathrm{SL}_i.$$

Now we have formulated optimization objectives, i.e., COST and RTO, and restrictions. Thus we can discuss scheduling strategy quantitatively.

In DR-Cloud, we propose five different strategies in the following to be compared. (In order to simplify the

description, we use CPE to denote a CP which has enough storage space left to contain a certain replica at a certain scheduling time.)

(1) Random strategy: For every new task in Request Buffer, send 3 replicas one by one to CPEs randomly selected.

(2) COST preferred strategy: For every new task in Request Buffer, send 3 replicas one by one to the CPE which has the lowest backup cost. If the first and second replicas reside in a same CPE, send the third replica to another CPE with lowest backup cost if possible.

(3) RTO preferred strategy: For every new task in Request Buffer, send the first replica to the CPE which has the shortest recovery time, then send the second and third replicas to other CPEs randomly selected if possible.

(4) COST/RTO tradeoff strategy: For every new task in Request Buffer, if CP_1 has enough storage space to contain 2 replicas of the task, then store the first and second replicas in CP_1 and send the third replica to the CPE which has the shortest recovery time; if CP_1 can only contain 1 replica of the task, then store the first replica in CP_1, then send the second replica to the CPE which has the shortest recovery time and send the third replica to the CPE which has the lowest backup cost; if CP_1 can't contain any replica of the task, then send the first replica to the CPE which has the shortest recovery time and send the second and the third replicas one by one to the CPE which has the lowest backup cost.

(5) Dynamic optimization strategy: For all replicas created by CP_1 during a scheduling period, use multi-objective combinatorial optimization algorithm to generate a suboptimal backup scheme considering all of them, then dispatch those replicas following that scheme. Without loss of generality, we use the MOPSO[21] algorithm, i.e., multi-objective particle swarm optimization algorithm, to compare with other strategies in this paper.

As described above, the first three strategies are very simple and intuitional. We propose the first strategy for comparison with other strategies as a straw man. The second and third strategies tend to optimize COST or RTO separately. And both of them try to ensure the 3 replicas of each task reside in at least 2 CPs if possible.

The fourth strategy is a greedy algorithm, which tries to optimize both COST and RTO. The design start points of this strategy include: (1) CP_1 resource preferred: From the perspective of CP_1, the storage cost of itself is calculated using cost price, while storage cost of other CPs is calculated using market price. So storage cost of CP_1 should be the lowest. In addition, there is no network traffic fee when storing data to CP_1 itself, and the recovery start-up delay of CP_1 is also the shortest due to no need to set up temporary access authorization; (2) COST/RTO tradeoff: The COST is the sum of all three replicas of each task while the RTO is the shortest of all replicas. So it is enough to ensure the shortest RTO by sending only one replica to the CPE with the shortest recovery time, then other replicas can be stored in the CPE with the lowest backup cost to ensure COST as low as possible.

The fifth strategy can use various kinds of algorithms to generate backup scheme. While it's complicated because it has two optimization objectives, i.e., COST and RTO, with restriction of storage resource limitation. We didn't simplify the problem by combining two objectives to one, for example using a formula like "COST+$a\times$RTO" as the only optimization objective. We think this method is too simple and not appropriate. So we investigated several multi-objective optimization algorithms and picked up the MOPSO algorithm for its good global convergence and fast convergence rate. We describe the implementation details of it in the experiment section below.

5 Experiment Design and Results Analysis

5.1 Experiment parameters setting

We have performed simulation experiments on DR-Cloud model. Our experiment platform consists of four modules: (1) experiment parameters generating module which generates parameters of multi-cloud based data disaster recovery scenarios; (2) information collecting module which is responsible for receiving backup task parameters and monitoring resource changes of CPs; (3) replica scheduling module that simulates the Replica Scheduler of CP_1 and generates backup schemes based on different strategies; and (4) result recording module that writes down each backup scheme and calculates the total COST and RTO throughout the whole simulation period.

There are 8 CPs in our experiment. Table 1 shows the parameters of each CP. Without special selection, these parameters have practical representativeness except smaller storage resource limitations for raising resource competition earlier.

As we mentioned above, CP_1 should have the lowest Storage Price (SP) and start-up delay (L), and network

Table 1 Parameter setting of CPs.

CP No.	SL	SP	TP	BW	L
	TB	10^{-4}dollar/(GB·h)	dollar/GB	Gbit/s	s
1	2.0	0.5	0	0.3	1
2	2.0	2.0	0.010	0.3	10
3	1.5	1.0	0.020	0.3	10
4	1.5	2.5	0.015	0.3	10
5	1.0	1.5	0.020	0.1	10
6	1.0	1.5	0.020	0.2	10
7	0.5	2.5	0.025	0.5	10
8	0.5	1.0	0.010	0.2	10

Traffic Price (TP) of CP_1 should be 0.

During our experiment, we simulated a time duration of 10 days, using 500 and 1000 tasks. And parameters for those backup tasks were generated following rules below:

(1) Arriving time: randomly and uniformly distributed in 10 days.

(2) Data size (GB): uniformly distributed random integer from 1 to 10.

(3) Store duration (h): uniformly distributed random integer from 1 to 200.

We generate a set of tasks first, then trigger the replica scheduling module every 0.5 h. If there are more than 10 tasks waiting to be processed, or it has been for 2 h from last scheduling, the module will execute scheduling algorithm once. All replicas' durations are also checked every 0.5 h to update the storage resource usages of CPs. When simulation time reaches 10 days, we record the total COST and RTO of all the tasks.

Especially, we describe the implementation detail of MOPSO algorithm of the fifth strategy as follows.

The particle space's dimension is defined to be $3\times N$, where N is the count of tasks need to be scheduled together. Thus a particle can represent a backup scheme of 3 replicas of these tasks, that is, the $[3\times(j-1)]$-th, $[3\times(j-1)+1]$-th, and $[3\times(j-1)+2]$-th dimensional position of a particle represent the locations of the 3 replicas of j-th task respectively. A position $\in [(i-1)/M, i/M)$ means the corresponding replica's location is the i-th CP, where M is the count of CPs.

For example, if the fourth dimensional position of a particle is 0.63 in our experiment, it means the second task's second replica resides in CP_6.

In our simulation, the initial velocity of each particle is set to 0, the mutation probability is set to 0.5, the inertia weight of velocity is set to 0.4, and the two acceleration constants are both set to 1. The population size and the count of iterations are both 100, and the initial positions of each particle is set to random numbers between 0 and 1.

As we know, the result of MOPSO algorithm is a set, which may contain multiple particles. We randomly select one particle as the final optimized result, without manual selection.

5.2 Experiment result and analysis

Like other multi-objective combinatorial optimization algorithms, running process of the fifth strategy using MOPSO algorithm is random, and the results of multiple running are probably different. The first and third strategies are also random. Therefore, to validate their convergence, we designed a comparison test.

We generated a set of 500 tasks first, and scheduled them 10 times against each strategy. Then we generated another set of 1000 tasks and did it again.

Figure 3 shows the results of the comparison tests, while (a) used 500 tasks and (b) used 1000 tasks. We can see the first, third, and fifth strategies have random character. Their result positions of each run are very close even hard to distinguish, so all of them have very good convergence. And obviously, the other two

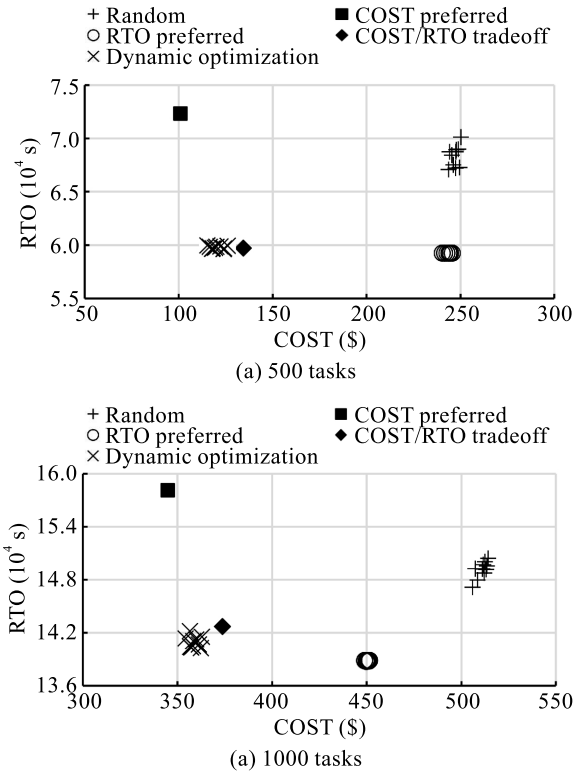

Fig. 3 COST & RTO, each strategy scheduled 10 times using same task set.

strategies are stabilized.

To further compare these 5 strategies, we ran simulation 100 times against each strategy, 50 times using different sets of 500 tasks and the other 50 times using different sets of 1000 tasks. In order to compare COSTs and RTOs at a same chart, we normalized the results, i.e., all COSTs were divided by COST of the third strategy and all RTOs were divided by RTO of the second strategy, then calculated average values of every 50 runs of each strategy.

Figure 4 shows the comparison of these average values of normalized COST and RTO, while (a) used different sets of 500 tasks and (b) used different sets of 1000 tasks.

As shown in Fig. 4, the first strategy is the worst, both COST and RTO are unacceptable. The second strategy has the lowest COST while its RTO is the longest, and the third strategy has the shortest RTO and relatively higher COST than others except the first strategy.

Both of the fourth and fifth strategies can get balance between COST and RTO. They can achieve relatively approximate COST of the second strategy which is COST preferred and RTO of the third strategy which is RTO preferred. In addition, the fifth strategy can generate better results than the fourth strategy.

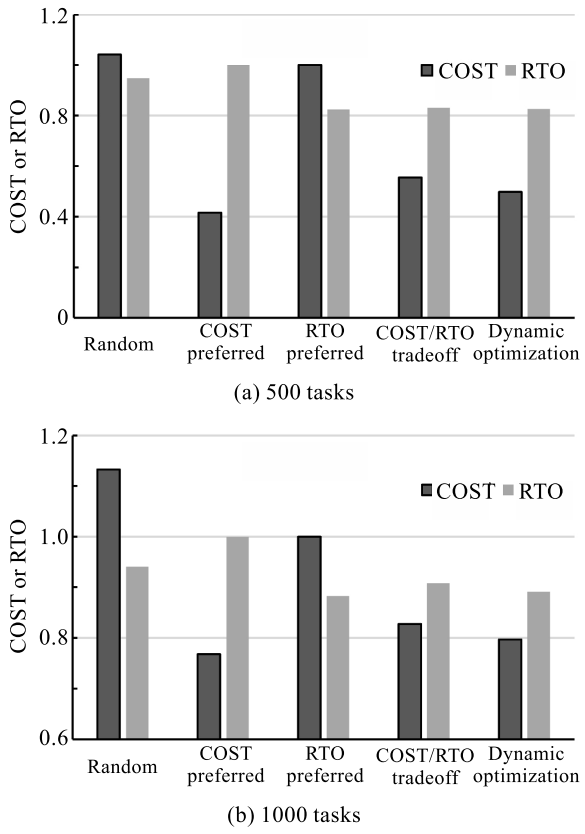

Fig. 4 Averages of normalized COST & RTO, each strategy scheduled 50 times using different task sets.

Figure 5 shows the final usages of all CPs after a certain simulation of the fifth strategy with 1000 tasks. We can see that each CP contains some backup data, with different storage resource utilizations.

So we can conclude that the DR-Cloud system model can cooperate with multiple cloud service providers with various parameters effectively, and different scheduling strategies of DR-Cloud have different characteristics. The first strategy is inappropriate in practical scenarios. And the second and third strategies may be suitable for some special scenarios, which have extreme high demands on one factor of COST or RTO, and have very loose requirements on the other factor. The fourth and fifth strategies both can achieve balanced COST and RTO, so they can be applied to general scenarios. And the fifth strategy has more optimized result than the fourth strategy, despite of its greater computational complexity, which is more suitable for most scenarios except those with a huge number of cloud service providers.

6 Conclusions and Future Work

This paper presents a practical multi-cloud based disaster recovery service model: DR-Cloud. With DR-Cloud, resources of multiple cloud service providers can be utilized cooperatively by the data disaster recovery service provider. And customers only need to deal with that service provider, using very simple and unified service interface, without concerning the internal processes between heterogeneous clouds. DR-Cloud can ensure high data reliability, low backup cost, and short recovery time by using intelligent data scheduling strategies. Different scheduling strategies based on DR-Cloud are proposed, which are suitable for different kinds of data disaster recovery scenarios.

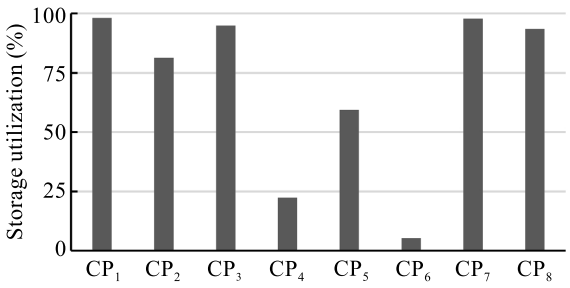

Fig. 5 Storage utilization of CPs after a certain simulation.

Experimental results show that the DR-Cloud model can cooperate with cloud service providers with various parameters effectively, while data scheduling strategies of DR-Cloud can achieve their optimization objectives efficiently, which are widely applicable.

In the future, we will further investigate other multi-objective optimization algorithms to explore better effect. We also plan to build a real DR-Cloud system to demonstrate its effectiveness. And we will study on replacing 3-replicas mechanism of DR-Cloud with ECC codes to reduce backup data size, like R-ADMAD[15] did between multiple data servers, which will improve storage and network resource utilization.

Acknowledgements

This work was supported by the National High-Tech Research and Development (863) Program of China (No. 2012AA012609).

References

[1] C. Warrick and S. John, A disaster recovery solution selection methodology, IBM Corporation, February 2004.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, The cost of a cloud: Research problems in data center networks, *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68-73, 2008.

[3] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, and I. Stoica, Above the clouds: A Berkeley view of cloud computing, Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 2009.

[4] T. Wood, E. Cecchet, K. K. Ramakrishnan, P. Shenoy, J. Van der Merwe, and A. Venkataramani, Disaster recovery as a cloud service: Economic benefits & deployment challenges, in *2nd USENIX Workshop on Hot Topics in Cloud Computing*, Boston, USA, 2010.

[5] Amazon Corporation, Amazon simple storage service (Amazon S3), http://aws.amazon.com/s3, 2008.

[6] 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.

[7] D. Borthakur, The hadoop distributed file system: Architecture and design, http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf, 2007.

[8] J. R. Raphael, The 10 worst cloud outages (and what we can learn from them), http://www.infoworld.com/d/cloud-computing/the-10-worst-cloud-outages-and-what-we-can-learn-them-902, 2011.

[9] J. R. Raphael, The worst cloud outages of 2013 (so far), http://www.infoworld.com/slideshow/107783/the-worst-cloud-outages-of-2013-so-far-221831, 2013.

[10] M. Vrable, S. Savage, and G. M. Voelker, Cumulus: Filesystem backup to the cloud, *ACM Transactions on Storage (TOS)*, vol. 5, no. 4, p. 14, 2009.

[11] A. Bajpai, P. Rana, and S. Maitrey, Remote mirroring: A disaster recovery technique in cloud computing, *International Journal of Advance Research in Science and Engineering*, vol. 2, no. 8, 2013.

[12] J. Zhang and N. Zhang, cloud computing-based data storage and disaster recovery, in *IEEE International Conference on Future Computer Science and Education (ICFCSE)*, 2011, pp. 629-632.

[13] Y. Tan, H. Jiang, D. Feng, L. Tian, and Z. Yan, CABdedupe: A causality-based deduplication performance booster for cloud backup services, in *Parallel & Distributed Processing Symposium (IPDPS)*, 2011, pp. 1266-1277.

[14] W. Li, Y. Yang, and D. Yuan, A novel cost-effective dynamic data replication strategy for reliability in cloud data centers, in *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC),* 2011, pp. 496-502.

[15] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, R-ADMAD: High reliability provision for large-scale de-duplication archival storage systems, in *Proceedings of the 23rd International Conference on Supercomputing*, 2009, pp. 370-379.

[16] T. Wood, H. A. Lagar-Cavilla, K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, PipeCloud: Using causality to overcome speed-of-light delays in cloud-based disaster recovery, in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, p. 17.

[17] T. Nguyen, A. Cutway, and W. Shi, Differentiated replication strategy in data centers, in *Proc. the IFIP International Conference on Network and Parallel Computing*, Guangzhou, China, 2010, pp. 277-288.

[18] C. Cachin, R. Haas, and M. Vukolic, Dependable storage in the Intercloud, *IBM Research*, vol. 3783, pp. 1-6, 2010.

[19] D. Bermbach, M. Klems, S. Tai, and M. Menzel, Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs, in *IEEE International Conference on Cloud Computing (CLOUD)*, 2011, pp. 452-459.

[20] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, Distributed systems meet economics: Pricing in the cloud, in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, Boston, USA, 2010, p. 6.

[21] J. Kennedy and R. Eberhart, Particle swarm optimization, in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942-1948.

Yu Gu is currently a PhD candidate in Department of Computer Science and Technology of Tsinghua University. He got his BEng degree from Tsinghua University in 2002. His research interests include distributed system, mass storage, cloud computing, and data reliability.

Dongsheng Wang received his PhD degree from Harbin Institute of Technology in 1995. He is now a professor of Research Institute of Information Technology in Tsinghua University. He is a senior member of China Computer Federation, member of IEEE. His research interests include computer architecture, high performance computing, storage and file systems, and network security.

Chuanyi Liu is an assistant professor of Beijing University of Posts and Telecommunications. He received his PhD degree in computer science and technology from Tsinghua University in 2009. His research interests include computer architecture, cloud computing & cloud security, storage and file system, information security, and data protection.