Sun DW, Chang GR, Gao S *et al.* Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(2): 256–272 Mar. 2012. DOI 10.1007/s11390-012-1221-4

Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments

Da-Wei Sun[1] (孙大为), *Student Member*, *CCF*, *ACM*, Gui-Ran Chang[2] (常桂然), Shang Gao[3] (高　尚), Li-Zhong Jin[1] (靳立忠), and Xing-Wei Wang[1,*] (王兴伟), *Senior Member*, *CCF*, *ACM*

[1] *School of Information Science and Engineering, Northeastern University, Shenyang 110819, China*

[2] *Computing Center, Northeastern University, Shenyang 110819, China*

[3] *School of Engineering and Information Technology, Deakin University, Geelong, Victoria 3217, Australia*

E-mail: sundaweicn@163.com; chang@neu.edu.cn; shang.gao@deakin.edu.au; jinlizhongneusoft@yahoo.com.cn; wangxw@mail.neu.edu.cn

Received June 17, 2011; revised January 29, 2012.

Abstract Failures are normal rather than exceptional in the cloud computing environments. To improve system availability, replicating the popular data to multiple suitable locations is an advisable choice, as users can access the data from a nearby site. This is, however, not the case for replicas which must have a fixed number of copies on several locations. How to decide a reasonable number and right locations for replicas has become a challenge in the cloud computing. In this paper, a dynamic data replication strategy is put forward with a brief survey of replication strategy suitable for distributed computing environments. It includes: 1) analyzing and modeling the relationship between system availability and the number of replicas; 2) evaluating and identifying the popular data and triggering a replication operation when the popularity data passes a dynamic threshold; 3) calculating a suitable number of copies to meet a reasonable system byte effective rate requirement and placing replicas among data nodes in a balanced way; 4) designing the dynamic data replication algorithm in a cloud. Experimental results demonstrate the efficiency and effectiveness of the improved system brought by the proposed strategy in a cloud.

Keywords system availability, replication perspective, high fault tolerance, temporal locality, cloud computing

1 Introduction

1.1 Background and Motivation

Cloud computing (CC), the long-held dream of "computing as a utility", has opened up the new era of future computing, transformed a large part of IT industry, and reshaped the purchase and use of IT software and hardware[1-4]. Cloud computing is a large-scale distributed computing paradigm driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, highly available, and configurable and reconfigurable computing resources (e.g., networks, servers, storage, applications, data) can be rapidly provisioned and released with minimal management effort in the data centers. Services are delivered on demand to external customers over high-speed Internet with the "X as a service (XaaS)" computing architecture, which is broken down into three segments: "applications", "platforms", and "infrastructure". Its aims[3-4] are to provide users with more flexible services in a transparent manner and with ever cheaper and more powerful processors. Similarly, IT companies with innovative ideas for new application services are no longer required to make large capital outlays in the hardware and software infrastructures. By using cloud computing platforms, they can register necessary services from the Internet and are free from the trivial task of setting up basic hardware and software infrastructures, which allows them to focus on the core aspects of their business. In computational view, cloud computing is a network of data centers and is described as a powerful, low-cost, and energy-efficient approach to future computing. The data centers form what we call clouds. From a sociological standpoint on the other hand, in

Regular Paper

Supported by the National Natural Science Foundation of China under Grant Nos. 61070162, 71071028 and 70931001, the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant Nos. 20110042110024 and 20100042110025, the Fundamental Research Funds for the Central Universities of China under Grant Nos. N100604012, N090504003 and N090504006.

*Corresponding Author

the cloud, applications are accessible anywhere, anytime, and storage becomes infinite for all intents and purposes. And the users can access the powerful applications, platforms, and services delivered over Internet.

The difference between cloud and other large-scale distributed computing platforms can be summarized as follows[3-4].

1) *On-Demand Self-Service.* A user can unilaterally provision computing capabilities, such as network storage, as needed automatically without requiring human interaction with each service's provider.

2) *Broad Network Access.* Capabilities are available over the network and accessed through standard mechanisms that promote the use by heterogeneous thin or thick client platforms.

3) *Resource Pooling.* The computing resources are pooled to serve multiple users by using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

4) *Rapid Elasticity.* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the users, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

5) *Measured Services.* Cloud systems automatically control and optimize resource usage by leveraging a metering capability at some level of abstraction appropriate to the type of the service. Resource usage can be monitored, controlled, and reported to provide transparency for both the provider and consumers of the utilized service.

In a word, clouds can be a highly available, much cheaper, and much more elastic, reliable, and scalable computing environment compared to supercomputers, grids and other large-scale distributed computing environments. Clouds promise "scale by credit card". Moreover, clouds also promise "bags of tasks"[5]. More detailed discussion can be found in [3-9].

High availability, high fault tolerance and high efficiency access to cloud data centers where failures are normal rather than exceptional are significant issues, due to the large-scale data support. Data replication allows reducing user waiting time, speeding up data access and increasing data availability by providing the user with different replicas of the same service, all of them with a coherent state. Replication is a frequently used technique in the cloud, such as GFS (Google file system)[10], HDFS (Hadoop Distributed File System)[11]. However, cloud data centers have grown rapidly in both size and number, and the dynamically-scalable and totally virtualized resources are provided as a service over the Internet[12]. In most of the real cloud, data replication is achieved through data resource pool, the number of data replicas is statically set based on history experience and is usually less than 3. This strategy works well at most time, but it will fail at inclement times. And it is not necessary to create replica for all data files, especially for those non-popular data files. In order to meet the high availability, high fault tolerance and high efficiency requirement, it is necessary to dynamically adjust the popular data files, the number of data replicas and the sites to place the new replicas according to the current cloud environments.

In order to achieve the dynamic data replication, there are three important problems that must be solved. 1) Which data should be replicated and when to replicate in the cloud systems to meet the users' requirements on waiting time reduction and data access speeding up are important issues for further research, as the wrongly selected and too early replicated data will not reduce the waiting time or speed up data access. 2) How many suitable new replicas should be created in the cloud to meet a reasonable system availability requirement is another important issue to be thoroughly investigated. With the number of new replicas increasing, the system maintenance cost will significantly increase, and too many replicas may not increase availability, but bring unnecessary spending instead. 3) Where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirements is also an important issue to be explored in detail. By keeping all replicas active, the replicas may improve system task successful execution rate and bandwidth consumption if the replicas and requests are reasonably distributed. However, appropriate replica placement in ultra-large-scale, dynamically scalable and totally virtualized data centers is much more complicated.

Our work is originally motivated by the fact that a more recently accessed data will be accessed again in the near future according to the current data access pattern, which is called temporal locality[13-14]. With the fact of temporal locality, a popular data is determined by analyzing the users' access to the data. When the popularity of the data passes a dynamic threshold, the replication operation will be triggered. The number of replicas will be determined based on the system availability and failure probability. New replica will be created on near-by locations for users who generate the most requests for the data.

1.2 Contributions

In this paper, a mathematical model is formulated to describe the relationship between the system availability and number of replicas, while the size, access

time and failure probability of each data file are taken into consideration. A popular data file is identified by analyzing the access histories and setting different weights for different accessed data. Basically, the more recently accessed data is more pertinent to the analysis, and is thus set a big weight. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. Replicas are placed among data nodes in a balanced way, taking into account the number of access of all users.

Based on the above facts and considerations, a novel dynamic data replication strategy named D2RS (Dynamic Data Replication Strategy) is proposed. It allows for increasing the data availability and minimizing cloud system bandwidth consumption. For evaluation purposes, the D2RS is implemented using the CloudSim toolkit and the experimental results demonstrate that the proposed strategy increases system availability, improves system task successful execution rate, and reduces bandwidth consumption in the cloud.

Our contributions can be summarized as follows. 1) A mathematical model is formulated to describe the relationship between the system availability and the number of replicas, which is missing in most existing research. 2) The popular data file is identified, when the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. 3) Replicas are placed among data nodes in a balanced way. 4) A dynamic data replication algorithm named D2RS is proposed, implemented on a simulation toolkit and evaluated. It is proved that this algorithm is able to increase the system availability and reduce the bandwidth consumption in the cloud.

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2,the related work on data storage and data replicas of cloud computing systems is analyzed. Section 3 presents a system model, a series of availability definitions, and a mathematical analysis to describe the relationship between the system availability and the number of replicas. Section 4 describes the dynamic data replication strategy, including the replication decision, the number of replicas, the replica placement and the detailed design of the D2RS algorithm. Section 5 addresses the simulation environment, parameter setup and performance evaluation of the proposed dynamic data replication strategy. Finally, conclusions and future work are given in Section 6.

2 Related Work

In this section, two broad categories of related work are presented: cloud data storage and cloud data replication.

2.1 Cloud Data Storage

Many large institutions have set up data centers and cloud computing platforms, such as Google, Amazon, IBM. Compared with traditional large scale storage systems, the clouds which are sensitive to workloads and user behaviors focus on providing and publishing storage service on Internet[15-17]. The key components of the cloud are distributed file systems, such as GFS, HDFS.

In a GFS cluster, there are three components, multiple clients, a single master server, and multiple chunk servers, as shown in Fig.1. Files are stripped into one or many fixed size chunks, and these chunks are stored in the data centers, which are managed by the chunk servers. The master server maintains all the metadata of the file system, including the namespace, the access control information, the mapping from files to chunks, and the current locations of chunks. Clients interact with the master for metadata operations, but all data bearing communication goes directly to the chunk servers[10].

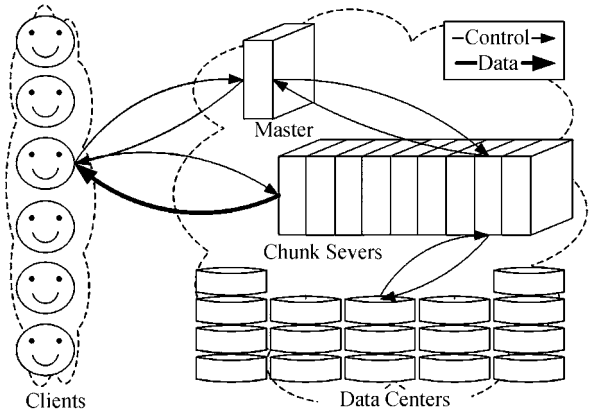


Fig.1. GFS architecture.

In a multi-cluster system, each cluster is a complete GFS cluster and with its own master, and each master maintains the metadata of its own file system. Different masters can share the metadata by the namespace, which describes how the log data is partitioned across multiple clusters[18]. Compared with a single cluster, in a multi-cluster system, the performance of the cloud system and the size of the cloud data storage can be improved significantly.

The mechanism of HDFS is similar to that of GFS, but it is light-weighted and open-source. More detailed discussion can be found in [11].

2.2 Cloud Data Replication

Data replication, a well-known technique from distributed systems, is the main mechanism used in the cloud for reducing user waiting time, increasing data availability and minimizing cloud system bandwidth consumption by offering the user different replicas with a coherent state of the same service[19]. With the advancement and development of various technologies, data replication and replica management in distributed systems have been studied in many works, which are referenced and adopted in cloud data replication. Data replication algorithms can be classified into two groups: static replication[10-11,20] and dynamic replication algorithms[13,15,21-22]. In a static replication model, the replication strategy is predetermined and well defined. On the other hand, dynamic replication automatically creates and deletes replicas according to changing access patterns. Static and dynamic replication algorithms can be further classified into groups as distributed[10-11,15,21] and centralized algorithms[13,20,22].

In [10], a static distributed cloud data replication algorithm is proposed. In the GFS, a single master considers three factors when making decisions on data chunk replications: 1) to place the new replicas on chunk servers with below-average disk space utilization; 2) to limit the number of "recent" creations on each chunk server; 3) to spread replicas of a chunk across racks. A data chuck is replicated when the number of replicas falls below a limit specified by the users. Similarly, in [11], an application can specify the number of replicas for each file, and the block size and replication factor are configurable per file.

In [20], a p-median static centralized data replication algorithm is proposed. The p-median model finds p replica placements sites that minimize the request-weighted total distance between the requesting sites and the replication sites holding the copies assigned.

In [15], a dynamic distributed cloud data replication algorithm CDRM is proposed. The CDRM is designed on the HDFS platform, the data replica placement is based on the capacity and location according to workload changing and node capacity, and the lower bound of the number of replicas is dynamically determined according to the availability requirement. In [21], six different data replication algorithms, Caching-PP, Cascading-PP, Fast Spread-PP, Cascading-Enhanced, and Fast Spread-Enhanced are proposed. All the six algorithms are dynamic replication algorithms and implemented in a distributed fashion.

In [22], a dynamic centralized data replication algorithm MinDmr is proposed. MinDmr treats hot and cold data differently and uses a weighting factor for the replication. And then MinDmr is developed into four prediction-based replica schemes. Similarly, in [13], an algorithm LALW is proposed. LALW selects a popular file for replication and calculates a suitable number of copies and grid sites for replication.

In the cloud, to reduce the access time, the data storage unit is a block. If a data file is too large, it will be stripped into many blocks. However, the data access unit usually is data file. The differences between the mentioned replication algorithms and our proposed strategy lie in the following aspects. 1) A mathematical model is formulated to describe the relationship between the system availability and the number of replicas. 2) The popular data is identified according to the temporal locality. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. 3) Replicas are placed among data nodes in a balanced way.

3 System Model and Problem Statement

In this section, a system model, a series of availability definitions and a mathematical analysis to describe the relationship between system availability and the number of replicas are presented in detail.

3.1 System Model

The multi-tier hierarchical cloud system architecture[16-17,22-25] supports an efficient method for sharing data and computational and other resources, as shown in Fig.2. It typically consists of different tiers of data centers with different regions and sizes. The super data centers in tier 0 will handle the data analysis in the intra domain and exchange data information

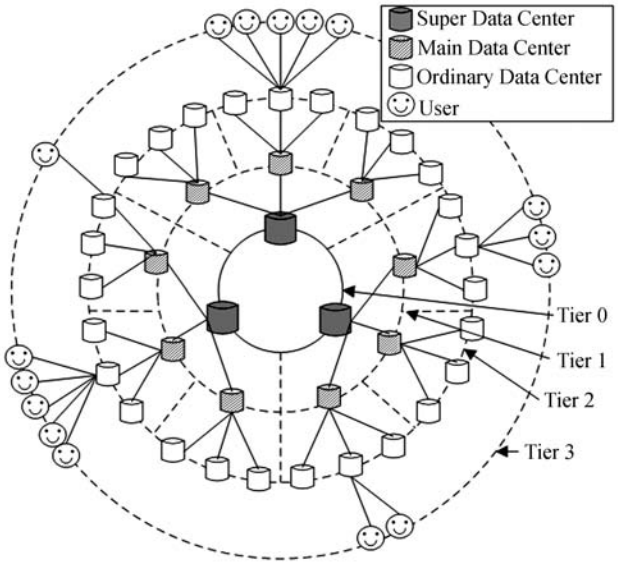


Fig.2. Multi-tier hierarchical cloud system topology.

among the inter domains. The main data centers are in tier 1, ordinary data centers are in tier 2, and users are in tier 3. The architecture minimizes the data access time and network load by creating and spreading replicas from the super data centers to main data centers, or to ordinary data centers. The super data centers periodically collect and broadcast the global information.

A cloud data service system typically consists of the scheduling broker, the replica broker and data centers, as shown in Fig.3. The scheduling broker is the central managing broker. The replica managers hold the general information about the replica locations in data centers. The specific features of cloud data servers can be described as follows.

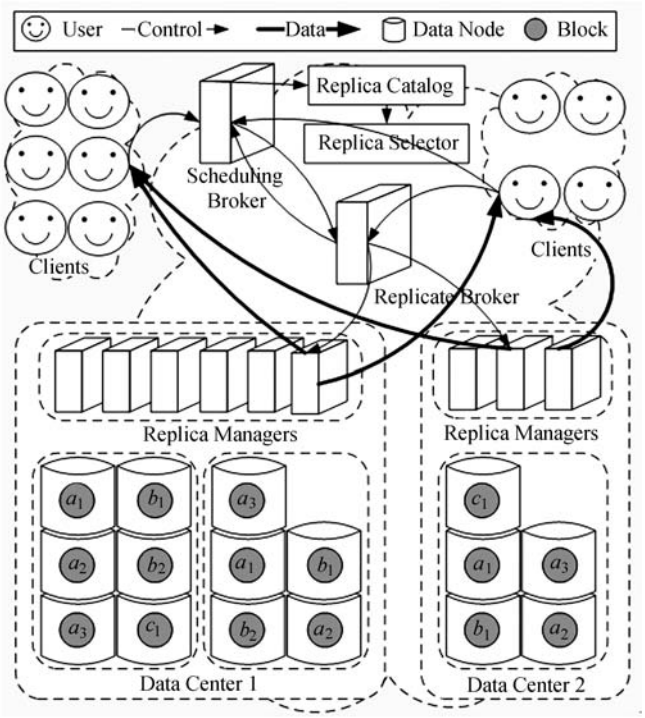


Fig.3. Cloud data server architecture.

Let $U = \{u_1, u_2, \ldots, u_m\}$ be a user set composed of m users, $TS = \{TS_1, TS_2, \ldots, TS_m\}$ be a set of tasks of the user set U, and $TS_j = \{ts_{j_1}, ts_{j_2}, \ldots, ts_{j_{m_j}}\}$ be a sub-set of tasks of the j-th user u_j, where m_j is the number of sub-tasks, and ts_k is the k-th task submitted to the scheduling broker through a user interface and independent of the other users. The replica broker schedules them to the appropriate cloud data server sites. If u_0 has two tasks, then $TS_0 = \{ts_{0_1}, ts_{0_2}\}$, and $m_0 = 2$. A task ts_k is characterized by a 4-tuple $ts_k = (tid_k, tr_k, td_k, tfn_k)$, where tid_k, tr_k, td_k and tfn_k are the task identification, task generation rate, task deadline time and the number of required files of task ts_k, respectively. For simplicity, we assume that the tasks are non-preemptable and non-interruptible[16,23], which mean that a task cannot be broken into smaller sub-tasks and it has to be executed as a whole on a single processor with given resources. In addition, as soon as a task starts its execution on a processor, it cannot be interrupted and it occupies the processor until its execution completes successfully or a failure occurs.

Let $DC = \{dnd_1, dnd_2, \ldots, dnd_n\}$ be a data center composed of n data nodes, which are running virtual machines on physical machines. A data node dnd_k is characterized by a 5-tuple $dnd_k = (dnd_k, dr_k, dst_k, df_k, dbw_k)$, where did_k, dr_k, dst_k, df_k and dbw_k are the data node identification, request arrival rate, average service time, failure probability and network bandwidth of data node dnd_k, respectively.

In order to guarantee the service performance of the data center DC, the task generation rate tr_k of user set U, the request arrival rate dr_k and failure probability df_k of DC should meet (1).

$$\sum_{j=0}^{jk} tr_j \leqslant \sum_{i=0}^{n} dr_i \times (1 - df_i), \quad jk = \sum_{j=0}^{m} j_{m_k}, \qquad (1)$$

where tr_j is the task generation rate of task j, dr_i is the request arrival rate of task j, df_k is the failure probability of task j, j_{m_k} is the number of tasks of user j.

Let $F = \{f_1, f_2, \ldots, f_l\}$ be a data file set of a data center DC. $B = \{B_1, B_2, \ldots, B_l\}$ be a set of blocks in the data center DC, and $B_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_{n_i}}\}$ be the i-th sub-set of blocks belonging to the i-th data file f_i, which is stripped into n_i fixed blocks according to its length. As shown in Fig.3, if data file A is stripped into 3 blocks, then $A = \{a_1, a_2, a_3\}$, and $n_i = 3$. An block b_k is characterized by a 5-tuple $b_k = (bid_k, bp_k, bs_k, bn_k, bt_k)$, where bid_k, bp_k, bs_k, bn_k and bt_k are the block identification, number of requests, block size, the number of replicas and the last access time of block b_k, respectively.

When user u_j requests a block b_k from a data node dnd_i with bandwidth performance guarantee, bandwidth bs_k/dst_i should be assigned to this session. The total bandwidth used to support different requests from use set U should be no more than dbw_i, as shown by (2).

$$\sum_{i=0}^{s_i} \frac{bs_k}{dst_i} \leqslant dbw_i, \qquad (2)$$

where s_i is the maximum number of network sessions of data node dnd_i that can serve concurrently, bs_k is the block size of block b_k, dst_i is the average service time of data node dnd_i, dbw_i is the network bandwidth of data node dnd_i.

3.2 System Byte Effective Rate

One of the most important objectives of cloud is to

provide the highest availability by placing all replicas of blocks of data files in a load balanced way on different data nodes of data centers, which is similar to that for grid environments[25-27].

Definition 1 (Availability). *It is the ability of a system to limit, control, and provide proper service under given constraints, defined as the "readiness for correct service" of a system*[28-29]*. The lifetime of a cloud system can be divided into a set of "up states" and a set of "down states". And the availability can be categorized as instantaneous availability, steady-state availability and inherent availability.*

The instantaneous availability $a_i(t)$ of a system is defined as the probability that a system is in an "up state" at time t under the constraint that it is correct at time $t = 0$ (i.e., $a_i(0) = 1$), shown by (3).

$$a_i(t) = \begin{cases} r(t), & \text{if no repair operations,} \\ p\{s_t = p_i(up) | s_0 = p_i(init)\}, & \text{otherwise,} \end{cases} \tag{3}$$

where $r(t)$ is the reliability at time t of the system. If no repair operations, the instantaneous availability $a_i(t)$ equals to the system reliability $r(t)$. s_t refers to the state at time t, $p_i(up)$ is a predicate that specifies the states where the system is operational, doing something useful. $p_i(init)$ specifies the initial states, and $p_i(init) = true$.

The steady-state availability $a_s(t)$ of a system is defined as the probability that a system is in an "up state" for "sufficiently long time" after the system starts and is examined at an arbitrary point of time. It is the limit value of $a_i(t)$ as t approaches infinity, as given by (4).

$$a_s(t) = \lim_{t\to\infty} a_i(t) = \lim_{t\to\infty} \frac{\int_0^t a_i(k)\mathrm{d}k}{t}. \tag{4}$$

The inherent availability $a(t)$ of a system is the expected value of the percentage of the time interval during which the system performs its required function, as defined by (5).

$$a(t) = \frac{MTBF}{MTBF + MTTR}, \tag{5}$$

where $MTBF$ is the mean time between faults, and $MTTR$ is the mean time to repair.

Definition 2 (Block Availability). *Block availability is the ability of a data block to limit, control, and provide proper service under given constraints. The block availability of a block b_k is denoted as BA_k. $P(BA_k)$ is the probability of block b_k in an available state. $P(\overline{BA_k})$ is the probability of block b_k in an unavailable state, and $P(\overline{BA_k}) = 1 - P(BA_k)$.*

The number of replicas of block b_k is bn_k. It is obvious that block b_k is considered unavailable only if all the replicas of block b_k are not available. So the availability and unavailability of block b_k is given in Theorem 1.

Theorem 1. *If the number of replicas of block b_k is bn_k, the available and unavailable probability of each replica of block b_k are $p(ba_k)$ and $p(\overline{ba_k}) = 1 - p(ba_k)$, respectively, then,*

$$P(BA_k) = 1 - (1 - p(ba_k))^{bn_k}, \tag{6}$$

and

$$P(\overline{BA_k}) = (1 - p(ba_k))^{bn_k}. \tag{7}$$

Proof. The available and unavailable probability of each replica of block b_k are $p(ba_k)$ and $p(\overline{ba_k})$, and the available and unavailable probability of block b_k are $P(BA_k)$ and $P(\overline{BA_k})$. As there are bn_k replicas of block b_k, block b_k is unavailable if and only if all the bn_k replicas of block b_k are unavailable. Therefore,

$$P(\overline{BA_k}) = p(\overline{ba_{k_1}}, \overline{ba_{k_2}}, \cdots, \overline{ba_{k_{bn_k}}}).$$

All the bn_k replicas are distributed in different data nodes, and all the bn_k replicas are independent of each other, thus,

$$\begin{aligned} P(\overline{BA_k}) &= p(\overline{ba_{k_1}}) \times p(\overline{ba_{k_2}}) \times \cdots \times p(\overline{ba_{k_{bn_k}}}) \\ &= \prod_{i=1}^{bn_k} p(\overline{ba_{k_i}}). \end{aligned}$$

Then,

$$\begin{aligned} P(\overline{BA_k}) &= \prod_{i=1}^{bn_k} p(\overline{ba_{k_i}}) = \prod_{i=1}^{bn_k} (1 - p(ba_{k_i})) \\ &= \prod_{i=1}^{bn_k} (1 - p(ba_k)) = (1 - p(ba_k))^{bn_k}. \end{aligned}$$

As $P(\overline{BA_k}) = 1 - P(BA_k)$, we obtain

$$P(BA_k) = 1 - P(\overline{BA_k}) = 1 - (1 - p(ba_k))^{bn_k}. \quad \square$$

As shown in Fig.3, if the number of replicas of block a_1 is 3, that is $bn_k = 3$, and the available probability of each replica is 0.98, i.e., $p(ba_{a_1}) = 0.98$, then the available probability of block a_1 is $P(BA_{a_1}) = 1 - (1 - p(ba_{a_1}))^{bn_k} = 1 - (1 - 0.98)^3 = 0.999\,92$, the unavailable probability of block a_1 is $P(\overline{BA_{a_1}}) = (1 - p(ba_{a_1}))^{bn_k} = (1 - 0.98)^3 = 0.000\,008$.

Definition 3 (File Availability). *File availability is the ability of a data file to limit, control, and provide proper service under given constraints. The file availability of a data file f_i is denoted as FA_i. $P(FA_i)$ is the probability of data file f_i in an available state. $P(\overline{FA_i})$ is the probability of data file f_i in an unavailable state, and $P(\overline{FA_i}) = 1 - P(FA_i)$.*

If the data file f_i is stripped into n_i fixed blocks denoted by $B_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_{n_i}}\}$, which are distributed

on different data nodes. $N_i = \{bn_{i_1}, bn_{i_2}, \ldots, bn_{i_{n_i}}\}$ is the set of the numbers of replicas of the blocks of B_i. The availability and unavailability of data file f_i is given in Theorem 2.

Theorem 2. *If the data file f_i is stripped into n_i blocks, there are bn_i replicas of each block in data file f_i, and all blocks at the same site will have the same available probability as all blocks are stored in data nodes with the same configuration in cloud data centers, the available probability of each replica is $p(ba_i)$ in data file f_i, then,*

$$P(FA_i) = \left(1 - (1 - p(ba_i))^{bn_i}\right)^{n_i}, \tag{8}$$

and

$$P(\overline{FA_i}) = \left(1 - (1 - p(ba_i))^{bn_i}\right)^{n_i}. \tag{9}$$

Proof. As the data file f_i is stripped into n_i blocks, the data file f_i is available if and only if each block in $B_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_{n_i}}\}$ are available, and all blocks are independent of each other. Therefore,

$$\begin{aligned} P(FA_i) &= P(BA_{i_1}, BA_{i_2}, \ldots, BA_{i_{n_i}}) \\ &= P(BA_{i_1}) \times P(BA_{i_2}) \times \cdots \times P(BA_{i_{n_i}}) \\ &= \prod_{j=1}^{n_i} P(BA_{i_j}) \\ &= \prod_{j=1}^{n_i} \left(1 - \prod_{k=1}^{n_{i_j}} \left(1 - p(ba_{i_j})\right)\right). \end{aligned}$$

All blocks are stored in data nodes with the same configuration in cloud data centers, without loss of generality. So we set the probability of all blocks of data file f_i to the same, that is,

$$p(ba_{i_1}) = p(ba_{i_2}) = \cdots = p(ba_{i_{n_i}}) = p(ba_i).$$

As the set of numbers $N_i = \{bn_{i_1}, bn_{i_2}, \ldots, bn_{i_{n_i}}\}$ of replicas of all blocks of data file f_i are the same, that is,

$$bn_{i_1} = bn_{i_2} = \cdots = bn_{i_{n_i}} = bn_i.$$

We obtain

$$\begin{aligned} P(FA_i) &= \prod_{j=1}^{n_i} \left(1 - \prod_{k=1}^{bn_{i_j}} (1 - p(ba_{i_j}))\right) \\ &= \prod_{j=1}^{n_i} \left(1 - (1 - p(ba_i))^{bn_{i_j}}\right) \\ &= \left(1 - (1 - p(ba_i))^{bn_i}\right)^{n_i}. \end{aligned}$$

As $P(\overline{FA_i}) = 1 - P(FA_i)$, we obtain

$$P(\overline{FA_i}) = 1 - P(FA_i) = 1 - \left(1 - (1 - p(ba_i))^{bn_i}\right)^{n_i}.$$

□

As shown in Fig.3, if the data file a is stripped into 3 blocks, that is $n_i = 3$, the number of replicas of data file a is also 3, that is $bn_k = 3$, and the available probability of each replica is 0.98, i.e., $p(ba_a) = 0.98$, then the availability and unavailability of data file f_i are $P(FA_i) = \left(1 - (1 - 0.98)^3\right)^3 = 0.999\,976$ and $P(\overline{FA_i}) = 1 - \left(1 - (0.98)^3\right)^3 = 0.000\,024$, respectively.

Because the system level data availability is more important than the single file availability in the cloud, higher system data availability will result in more benefits than higher file availability, especially when some data files are popular while other data files are not.

Definition 4 (System Byte Effective Rate). *System byte effective rate is the rate of the number of bytes potentially available and the total number of bytes requested by all tasks in a system. The system byte effective rate of a system is denoted as $R(SBER)$.*

Given the fact that a particular data file access operation will only request one file, any two file request will access different replicas and be independent of each other. Each user from the user set U will request one or more data files at one data file request. The system byte effective rate and system byte non-effective rate of a system is given in Theorem 3.

Theorem 3. *If $F = \{f_1, f_2, \ldots, f_{f_n}\}$ is the data file set of a data center DC and is composed of fn data files, $B = \{B_1, B_2, \ldots, B_{fn}\}$ is a set of blocks of data center DC, $B_i = \{b_{i_1}, b_{i_2}, \ldots, b_{i_{n_i}}\}$ is the i-th sub-set of blocks belonging to the i-th data file f_i, which consists of n_i blocks, the number of accesses of all users to the i-th data file f_i is an_i, the number of replicas of each block in data file is bn_i, and the size of block b_k is bs_k, then,*

$$R(SBER) = \frac{\sum_{i=1}^{fn} \left(an_i \times \left(\sum_{h=1}^{n_i} bs_h\right) \times \left(1 - (1 - p(ba_i))^{bn_i}\right)^{n_i}\right)}{\sum_{i=1}^{fn} \left(an_i \times \sum_{h=1}^{n_i} bs_h\right)}. \tag{10}$$

Proof. The data file set $F = \{f_1, f_2, \ldots, f_{fn}\}$ of a data center DC is composed of fn data files, and system byte effective rate $R(SBER)$ of a system is the rate of the number of bytes potentially available and the total number of bytes requested by all tasks. Therefore,

$$R(SBER) = \frac{\sum_{i=1}^{fn} (P(FA_i) \times FS_i)}{\sum_{i=1}^{fn} FS_i},$$

where FS_i is the total number of bytes requested by all

tasks of the i-th data file f_i.

If the number of accesses of the total users of the i-th data file f_i is an_i, and the size of data file f_i is fs_i, then, FS_i is

$$FS_i = \sum_{k=1}^{an_i} fs_k.$$

If the i-th data file f_i is stripped into n_i blocks, and the size of block b_k is bs_k, then,

$$fs_i = \sum_{h=1}^{n_i} bs_h.$$

We have

$$FS_i = \sum_{k=1}^{an_i} fs_k = \sum_{k=1}^{an_i} \sum_{h=1}^{n_i} bs_h,$$

then,

$$R(SBER) = \frac{\sum_{i=1}^{fn} (P(FA_i) \times FS_i)}{\sum_{i=1}^{fn} FS_i}$$

$$= \frac{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \left(P(FA_k) \times \sum_{h=1}^{n_i} bs_h \right)}{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \sum_{h=1}^{n_i} bs_h}$$

$$= \frac{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \left(\left(\sum_{h=1}^{n_i} bs_h \right) \times \left(1 - (1 - p(ba_i))^{bn_i} \right)^{n_i} \right)}{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \sum_{h=1}^{n_i} bs_h}.$$

Without loss of generality, assume the available probabilities $p(ba_k)$ of all replicas of block b_k are the same in all an_i access cycles. Then the following is obtained,

$$R(SBER)$$

$$= \frac{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \left(\left(\sum_{h=1}^{n_i} bs_h \right) \times \left(1 - (1 - p(ba_i))^{bn_i} \right)^{n_i} \right)}{\sum_{i=1}^{fn} \sum_{k=1}^{an_i} \sum_{h=1}^{n_i} bs_h}$$

$$= \frac{\sum_{i=1}^{fn} \left(an_i \times \left(\sum_{h=1}^{n_i} bs_h \right) \times \left(1 - (1 - p(ba_i))^{bn_i} \right)^{n_i} \right)}{\sum_{i=1}^{fn} \left(an_i \times \sum_{h=1}^{n_i} bs_h \right)}. \quad \square$$

For a scenario given in Fig.4, assume there are 3 data files a, b, and c in data center DC, and $l = 3$. Table 1 shows the detailed parameter setup for the example in Fig.4.

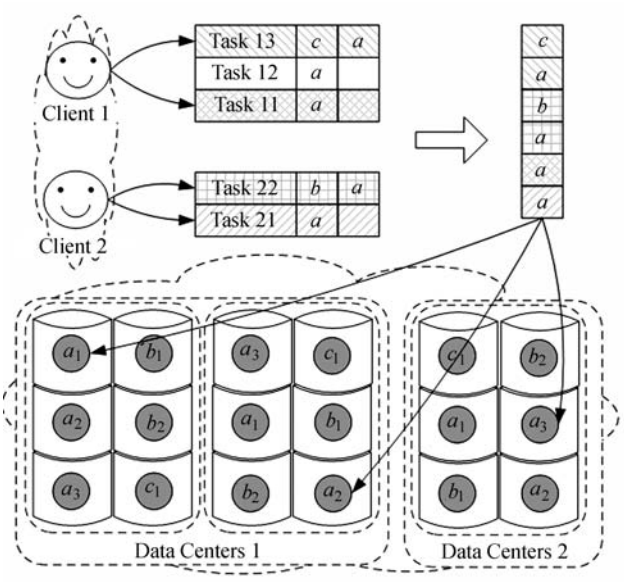


Fig.4. Cloud data server instance.

Table 1. Parameter Setup of $R(SBER)$

Parameter	Data File					
	a			b		c
	a_1	a_2	a_3	b_1	b_2	c_1
n_i	3.00	3.00	3.00	2.00	2.00	1.00
an_i	4.00	4.00	4.00	1.00	1.00	1.00
bn_i	3.00	3.00	3.00	3.00	3.00	3.00
bs_i	64.00	64.00	53.00	64.00	58.00	39.00
$p(ba_i)$	0.89	0.89	0.89	0.55	0.55	0.62

$R(SBER) = ((4 \times (64 + 64 + 53) \times 0.999976 + 1 \times (64 + 58) \times 0.709867 + 1 \times 39 \times 0.945128)/(4 \times (64 + 64 + 53) + 1 \times (64 + 58) + 1 \times 39) = 0.957567.$

4 Dynamic Data Replication Strategy

The dynamic data replication strategy D2RS (Dynamic Data Replication Strategy) has three important phases: 1) which data file should be replicated and when to replicate in the cloud system to meet users' requirements such as waiting time reduction and data access speeding up; 2) how many suitable new replicas should be created in the cloud system to meet a given availability requirement; 3) where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirements.

4.1 Decide Which and When to Replicate

Given the fact that a more recently accessed data file

might be accessed again in the near future according to the current status of data access pattern, which is called temporal locality, a popular data file is determined by analyzing the access to the data from users. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered[13,22,25,30-31].

Definition 5 (Time-Based Forgetting Function). *A time-based forgetting function ω is defined over the domain Time, with values within the interval $[0,1]$. It is used to calculate the popularity degree PD_{b_k} of a block b_k at the present time t_p according to the access frequency at the start time t_s, as shown by* (11),

$$\omega(t_p,t_s)=a^{-(\Delta t)^k}=a^{-(t_p-t_s)^k},\quad a>1,\ k\in\{1,2,\ldots\},\tag{11}$$

where $\Delta t=t_p-t_s$, as usual, parameter a is assigned as e, as shown by

$$\omega(t_p,t_s)=e^{-(\Delta t)^k}=e^{-(t_p-t_s)^k},\quad k\in\{1,2,\ldots\}.\tag{12}$$

The value of k determines the rate of decay of the popularity degree with time Δt, and is assigned by the block b_k based on its perception about the change. Fig.5 shows the nature of the change of $\omega(t_p,t_s)$ with different values of k. If $\Delta t=0$, then $\omega(t_p,t_s)=e^{-0}=1$. If $\Delta t\to+\infty$, then $\omega(t_p,t_s)=\lim_{\Delta t\to+\infty}e^{-(\Delta t)^k}=0$. This corroborates the fact that the time-based forgetting weight is asymptotic to zero at infinite time.

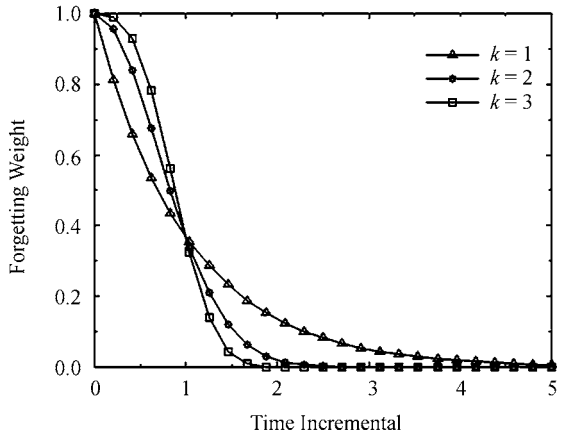


Fig.5. Time-based forgetting function.

Definition 6 (Popularity Degree). *The popularity degree of a block b_k is defined as the access frequency based on time factor. During the period from the start time t_s to the present time t_p, the popularity degree pd_k of a block b_k can be calculated by* (13).

$$pd_k=\sum_{t_i=t_s}^{t_p}(an_k(t_i,t_{i+1})\times\omega(t_i,t_p)),\tag{13}$$

where $an_k(t_i,t_{i+1})$ is the number of accesses during the time interval t_i to t_{i+1}.

Definition 7 (Replica Factor). *The replica factor is defined as the ratio of the popularity degree and the total number of bytes of data file f_i requested by all tasks under given constraints. It is used to determine whether the data file f_i should be replicated, denoted as RF_i in* (14).

$$RF_i=\frac{PD_i}{RN_i\times FS_i},\tag{14}$$

where PD_i, RN_i, FS_i are the popularity degree, number of replicas and file size of data file f_i in million bytes, respectively.

Theorem 4. *If the data file f_i is stripped into n_i blocks, bn_i, bs_i, and $an_k(t_i,t_{i+1})$ are the number of replicas, block size and number of accesses in the time interval t_i to t_{i+1} of each block in the data file f_i, respectively, then,*

$$RF_i=\frac{\sum_{t_i=t_s}^{t_p}(an_k(t_i,t_{i+1})\times\omega(t_i,t_p))}{bn_i\times\sum_{j=1}^{n_i}bs_j}.\tag{15}$$

Proof. As the data file f_i is stripped into n_i blocks, and $B_i=\{b_{i_1},b_{i_2},\ldots,b_{i_{n_i}}\}$, if the block $b_k\in f_i$, once one user accesses block b_k, he will access all the blocks of the file f_i, then the popularity degree pd_k of a block b_k is equal to the popularity degree PD_i of the file f_i. At the same time, assume the set of numbers $N_i=\{bn_{i_1},bn_{i_2},\ldots,bn_{i_{n_i}}\}$ of replicas of all blocks in the data file f_i be the same, denoted as bn_i. Therefore,

$$RF_i=\frac{PD_i}{RN_i\times FS_i}=\frac{pd_k}{\sum_{l=1}^{n_i}\sum_{h=1}^{bn_i}bs_{l_h}}=\frac{\sum_{t_i=t_s}^{t_p}(an_k(t_i,t_{i+1})\times\omega(t_i,t_p))}{bn_i\times\sum_{j=1}^{n_i}bs_j}.$$

□

According to Definition 7, we can prove that the system replica factor RF_{sys} can be calculated by (16).

$$RF_{sys}=\frac{\sum_{h=1}^{l}\Big(\sum_{t_i=t_s}^{t_p}(an_h(t_i,t_{i+1})\times\omega(t_i,t_p))\Big)}{\sum_{i=1}^{l}\Big(bn_i\times\sum_{j=1}^{n_i}bs_j\Big)}.\tag{16}$$

In each time interval T, the replication operation of the data file f_i will be triggered if the condition shown

in (17) is met.

$$RF_i > \min\Big((1+\alpha)\times RF_{sys}, \max\big(\underset{k\in[1,2,\ldots,l]}{\forall} RF_k\big)\Big), \quad \alpha\in[0,1], \tag{17}$$

where α is the adjustable parameter according to different system performance. The better the requested system performance, the greater α can be selected.

If $\alpha = 0$, then all files whose replica factors are greater than $(1+\alpha)\times RF_{sys}$ will be replicated.

If $(1+\alpha)\times RF_{sys} = \max\big(\underset{k\in[1,2,\ldots,l]}{\forall} RF_k\big)$, then the only one file with the maximum replica factor will be replicated.

If $(1+\alpha)\times RF_{sys} > \max\big(\underset{k\in[1,2,\ldots,l]}{\forall} RF_k\big)$, then no file will be replicated.

For the scenario given in Fig.4, assume that data file a is stripped into 3 blocks a_1, a_2 and a_3, the number of replicas of each block is 3, the block sizes of the blocks are 64 MB, 64 MB and 53 MB. Table 2 shows the detailed parameter setup used within the time interval t_0 to t_5.

$$\begin{aligned} RF_a =& (1.388794\text{e}{-}011\times 96 + 1.125352\text{e}{-}007\times 89 + \\ & 1.234098\text{e}{-}004\times 902 + 1.831564\text{e}{-}002\times 883 + \\ & 3.678794\text{e}{-}001\times 1087 + \\ & 1.000000\text{e}{-}000\times 1279)/(3\times(64+64+53)) \\ =& 3.121\,858. \end{aligned}$$

Table 2. Parameter for PD about File a

k	t_s	t_p	Δt	$\omega(t_p,t_s)$	$an_k(t_i,t_{i+1})$
2	0.0	5.0	5.0	1.388794e−011	96
2	1.0	5.0	4.0	1.125352e−007	89
2	2.0	5.0	3.0	1.234098e−004	902
2	3.0	5.0	2.0	1.831564e−002	883
2	4.0	5.0	1.0	3.678794e−001	1 087
2	5.0	5.0	0.0	1.000000e−000	1 279

The data file b is stripped into 2 blocks b_1 and b_2, as shown in Fig.4, the number of replicas of each block is 3, the block sizes of the blocks are 64 MB and 58 MB. Table 3 shows the detailed parameter setup used within time interval t_1 to t_5.

$$\begin{aligned} RF_b =& (1.125352\text{e}{-}007\times 523 + \\ & 1.234098\text{e}{-}004\times 678 + \\ & 1.831564\text{e}{-}002\times 2365 + \\ & 3.678794\text{e}{-}001\times 1987 + \\ & 1.000000\text{e}{-}000\times 3645)/(3\times(64+58)) \\ =& 9.342\,559. \end{aligned}$$

Table 3. Parameter for PD about File b

k	t_s	t_p	Δt	$\omega(t_p,t_s)$	$an_k(t_i,t_{i+1})$
2	1.0	5.0	4.0	1.125352e−007	523
2	2.0	5.0	3.0	1.234098e−004	678
2	3.0	5.0	2.0	1.831564e−002	2 365
2	4.0	5.0	1.0	3.678794e−001	1 987
2	5.0	5.0	0.0	1.000000e−000	2 645

Assume that data file c be stripped into block c_1, as shown in Fig.4, the number of replicas of block c_1 is 3, the block size of block c_1 is 39 MB. Table 4 shows the detailed parameter setup used within time interval t_3 to t_5.

$$\begin{aligned} RF_c =& (1.831564\text{e}{-}002\times 1035 + \\ & 3.678794\text{e}{-}001\times 1256 + \\ & 1.000000\text{e}{-}000\times 898)/(3\times 39) = 11.786\,438. \end{aligned}$$

$$\begin{aligned} RF_{sys} =& (1.388794\text{e}-011\times 96 + \\ & 1.125352\text{e}{-}007\times 89 + \\ & 1.234098\text{e}{-}004\times 902 + \\ & 1.831564\text{e}{-}002\times 883 + \\ & 3.678794\text{e}{-}001\times 1087 + \\ & 1.000000\text{e}{-}000\times 1279 + \\ & 1.125352\text{e}{-}007\times 523 + \\ & 1.234098\text{e}{-}004\times 678 + \\ & 1.831564\text{e}{-}002\times 2365 + \\ & 3.678794\text{e}{-}001\times 1987 + \\ & 1.000000\text{e}{-}000\times 3645 + \\ & 1.831564\text{e}{-}002\times 1035 + \\ & 3.678794\text{e}{-}001\times 1256 + \\ & 1.000000\text{e}{-}000\times 898)/(3\times(64+64+53)+ \\ & 3\times(64+58)+3\times 39) = 6.329\,005. \end{aligned}$$

Table 4. Parameter for PD about File c

k	t_s	t_p	Δt	$\omega(t_p,t_s)$	$an_k(t_i,t_{i+1})$
2	3.0	5.0	2.0	1.831564e−002	1 035
2	4.0	5.0	1.0	3.678794e−001	1 256
2	5.0	5.0	0.0	1.000000e−000	898

If $\alpha = 0.2$, $(1+\alpha)\times RF_{sys} = (1+0.2)\times 6.329\,005 = 7.594\,806$, the files b and c will be replicated.

If $\alpha = 0.5$, $(1+\alpha)\times RF_{sys} = (1+0.5)\times 6.329\,005 = 9.493\,507$, the file c will be replicated.

If $\alpha = 0.9$, $(1+\alpha)\times RF_{sys} = (1+0.9)\times 6.329\,005 = 12.025\,109$, no file will be replicated.

4.2 Determine the Number of New Replicas

To meet the system byte effective rate requirement, new replicas should be created[31-33]. With a reasonable increase of file availability, the number of new replicas that need to be created can be calculated according to

(18), which determines the new replicas on the basis of old file availability $P_{old}(FA_i)$ of data file f_i and the replica factor based adjustable parameter β.

$$P_{new}(FA_i) = P_{old}(FA_i) + \beta \times (1 - P_{old}(FA_i)), \quad \beta \in [0, 1], \tag{18}$$

where $P_{new}(FA_i)$ and $P_{old}(FA_i)$ are the new file availability and the old file availability of data file f_i, respectively. β is the replica factor based adjustable parameter. It can be calculated according to (19).

$$\beta = \frac{RF_i}{\sum_{k=1}^{num_s} RF_k}, \tag{19}$$

where num_s is the number of files selected to be replicated.

Theorem 5. *If the data file f_i is stripped into n_i blocks, $P_{old}(FA_i)$ is the old file availability of data file f_i, RF_i is the replica factor of data file f_i, $bn_i(old)$ is the old replica number of data file f_i, num_s is the number of files selected to be replicated, the number of new replicas $bn_i(inc)$ to be created is,*

$$bn_i(inc) = \left\lfloor \frac{\ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{num_s} RF_k} \times (1 - P_{old}(FA_i))\right)^{\frac{1}{n_i}}\right)}{\ln(1 - p(ba_i))} \right\rfloor - bn_i(old). \tag{20}$$

Proof. As $P_{new}(FA_i)$ and $P_{old}(FA_i)$ are the new file availability and old file availability of data file f_i, respectively, and $P_{new}(FA_i) = \left(1 - \left(1 - p(ba_i)\right)^{bn_i(new)}\right)^{n_i}$, according to (18) and (19), we obtain,

$$\left(1 - (1 - p(ba_i))^{bn_i(new)}\right)^{n_i} = P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{num_s} RF_k} \times (1 - P_{old}(FA_i)),$$

and,

$$bn_i(new) = \left\lfloor \frac{\ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{num_s} RF_k} \times (1 - P_{old}(FA_i))\right)^{\frac{1}{n_i}}\right)}{\ln(1 - p(ba_i))} \right\rfloor.$$

If the old number of replicas is $bn_i(old)$, the number of new replicas $bn_i(inc)$ to be created is,

$$bn_i(inc) = \lfloor bn_i(new) - bn_i(old) \rfloor = \left\lfloor \frac{\ln\left(1 - \left(P_{old}(FA_i) + \frac{RF_i}{\sum_{k=1}^{num_s} RF_k} \times (1 - P_{old}(FA_i))\right)^{\frac{1}{n_i}}\right)}{\ln(1 - p(ba_i))} \right\rfloor - bn_i(old).$$

□

For the scenario given in Subsection 4.1, assume $\alpha = 0.2$ and the files b and c will be replicated, Table 5 shows the detailed parameters used.

Table 5. Parameter for bn_i

Parameter	Data File	
	b	c
$P_{old}(FA_i)$	0.826 054	0.978 048
$p(ba_i)$	0.46	0.62
RF_i	9.342 559	11.786 438
n_i	2	1

$bn_b(inc) = \ln(1 - (0.709\,867 + (1 - 0.709\,867) \times (9.342\,559/(9.342\,559 + 11.786\,438)))^{1/2})/(\ln(1 - 0.46)) - 3 = 1.010\,306$, so the number of new replicas of data file b to be created is 1;

$bn_c(inc) = \ln(1 - (0.945\,128 + (1 - 0.945\,128) \times (11.786\,438/(9.342\,559 + 11.786\,438)))^{1/1})/(\ln(1 - 0.62)) - 3 = 0.843\,406$, so the number of new replicas of data file c to be created is 0.

4.3 Placement of New Replicas

To meet the system task successful execution rate and bandwidth consumption requirement, different tiers of data centers which have the selected replica data file f_i will decide the replica placement and the placement of new replicas to be created according to the access information of directly connected data centers. The number of new replicas created at the directly connected data center dc_k is calculated according to (21), based on the total number of new replicas $bn_i(inc)$ and

the replica factor $RF_i(dc_k)$.

$$bn_i(dc_k) = \begin{cases} bn_i(inc) - \sum\limits_{\neg \max(RF_i(dc_k))} \left\lfloor \dfrac{RF_i(dc_k)}{RF_i} \times bn_i(inc) \right\rfloor, \\ \qquad \text{if } RF_i(dc_k) \text{ is } \max(RF_i(dc_k)), \\ \left\lfloor \dfrac{RF_i(dc_k)}{RF_i} \times bn_i(inc) \right\rfloor, \quad \text{otherwise}, \end{cases} \tag{21}$$

where $bn_i(dc_k)$ is the number of new replicas to be created at the directly connected data center dc_k, $RF_i(dc_k)$ is the replica factor of data file f_i of the data center dc_k directly connected, and RF_i is the replica factor of the data file f_i.

For the scenario given in Fig.6, assume the data file d has 2 replicas at a main data center and an ordinary data center, respectively, and 3 replicas need to be created. The detailed replica factor parameter setup is shown in Fig.6.

$$\begin{aligned} bn_d(dc_1) &= (3.256\,398/(3.256\,398 + 0.125\,999\,8 + \\ &\quad 5.960\,163)) \times 3 = 1; \\ bn_d(dc_2) &= 0; \\ bn_d(dc_3) &= 0; \\ bn_d(dc_4) &= (0.125\,999\,8/(3.256\,398 + 0.125\,999\,8 + \\ &\quad 5.960\,163)) \times 3 = 0; \\ bn_d(dc_5) &= 0; \\ bn_d(dc_6) &= (5.960\,163/(3.256\,398 + 0.125\,999\,8 + \\ &\quad 5.960\,163)) \times 3 = 2. \end{aligned}$$

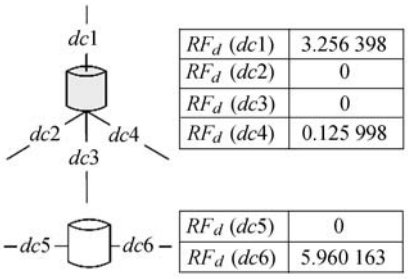


Fig.6. Replica placement instance.

The main data center will create 1 replica at the link $dc1$ directly connected to the data center, and the ordinary data center will create 2 replicas to the link $dc6$ directly connected to the data center.

4.4 D2RS Algorithm

According to the above analysis, the replication decision is based on the theory of temporal locality. A popular data file is determined by the analysis of the access information to the data from users. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. The number of replicas depends on the reasonable increase of file availability; the replica placement is determined by the access information of directly connected data centers and is accomplished in a balanced way. The core part of the D2RS algorithm is described as follows.

Algorithm. D2RS Algorithm
Input: the available probability $p(ba_k)$ and unavailable probability $p(\overline{ba_k})$ of all replicas of block b_k of the data file f_i, the number of replicas bn_i, the block size bs_i and the number of accesses $an_k(t_i, t_{i+1})$ within time interval t_i to t_{i+1} for each block of data file f_i.
Output: system byte effective rate $R(SBER)$.
Initialize available and unavailable probability of each replica of block b_k $p(ba_k)$ and $p(\overline{ba_k})$.
for each data file f_i at all data centers DC **do**
    Calculate the popularity degree pd_k of a block b_k of data file f_i by (13).
    Calculate replica factor RF_i of data file f_i by (15).
end for
Calculate replica factor RF_{sys} of the cloud system by (16).
for each data file f_i at all data centers DC **do**
    if $RF_i > \min\left((1+\alpha) \times RF_{sys}, \max\left(\underset{k\in[1,2,\ldots,l]}{\forall} RF_k\right)\right)$ **then**
        The replication operation of the data file f_i will be triggered.
    end if
end for
for each data file f_i at all data centers DC **do**
    Calculate the old file availability $P(FA_i)$ of data file f_i by (8).
end for
for each triggered data file **do**
    Calculate the new file availability $P_{new}(FA_i)$ of data file f_i by (18).
    Calculate the number of new replicas needed $bn_i(inc)$ by (20).
end for
for each triggered data file and $bn_i(inc) > 0$ **do**
    for each directly connected data center dc_k **do**
        Calculate the number of new replicas $bn_i(dc_k)$ to be created at the directly connected data center dc_k by (21).
    end for
    Determine the replica placement according to $bn_i(dc_k)$.
    if the storage space of the target data center DC_{obj} is not enough **then**

28 Quick sort all data file in descending order by replica factor of data file at data center DC_{obj}.
29 Delete the data file with the smallest replica factor.
30 **end if**
31 **end for**
32 Calculate the new system byte effective rate $R(SBER)$ by (10).
33 **Return** $R(SBER)$.

The input of D2RM algorithm are the available probability $p(ba_k)$ and unavailable probability $p(\overline{ba_k})$ of all replicas of block b_k of the data file f_i, the number of replicas bn_i, block size bs_i and number of accesses $an_k(t_i, t_{i+1})$ within time interval t_i to t_{i+1} for each block in the data file f_i, and the output is the system byte effective rate $R(SBER)$. Steps 5∼14 calculate the replica factor and determine which data file should be replicated and when to replicate in a cloud system to meet the users' requirements, such as waiting time reduction and data access speeding up. Steps 15∼21 calculate the number of new replicas $bn_i(new)$ and determine how many suitable new replicas $bn_i(inc)$ should be created in the cloud system to meet a given availability requirement. Steps 22∼31 calculate the number of new replicas $bn_i(dc_k)$ to be created at the directly connected data center dc_k and decide where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirement. Step 32 calculates the new system byte effective rate $R(SBER)$. The time complexity of the D2RS algorithm is $O(L \times N)$, in which L is the number of data files, and N is the maximum number of directly connected data centers.

5 Simulation and Performance Evaluation

In order to evaluate the performance of the proposed D2RS algorithm, simulation environment and parameter setup are discussed in this section, followed by the precise performance evaluation results.

5.1 Simulation Environment and Parameter Setup

CloudSim toolkit[9,34-36] is used as the simulation environment. As a Java based simulation platform, it supports modeling and simulation of large scale cloud computing data centers. Special users and resources can be generated by rewriting the corresponding codes, which aligns well with various users and resources of the cloud system. It is feasible to simulate the proposed D2RS algorithm with CloudSim.

32 data centers are created in the simulation environment. The corresponding topology is shown in Fig.2 and detailed configuration is shown in Table 6. 570 virtual machines are set as the service providers, and the processing elements (PEs) number of each virtual machine is within the range of 2 to 4. Forty different data files are placed in the cloud storage environment, with each size in the range of [0.1, 10] GB. Each file is stored in fixed size ($bs = 0.2$ GB) storage unit called block. Blocks of the same data file are scattered across different virtual machines. At the very beginning, the number of replicas of each data file is 1 and placed randomly. For simplicity, it is assumed that the basis element of data storage is block and the element of replication is one total data file.

One thousand tasks are submitted to the 570 virtual machines, each task is submitted according to Poisson distribution after the previous task and asks for 1 or 2 data files.

5.2 Performance Evaluation

The experiments set up four evaluation parameters: system byte effective rate, number of replicas, response time and successful execution rate.

1) *System Byte Effective Rate.* System byte effective rate reflects the byte availability of cloud systems under given constraints. It can be obtained by (10).

As shown in Fig.7, as the number of replicas increasing, the D2RS algorithm ensures the system byte effective rate at a high level. When the average block availability is more than 0.8, and the number of replicas is 4, the system byte effective rate is close to 1, even when the average block availability is less than 0.2. To keep the system byte effective rate close to 1, the number of replicas should be no more than 30. It can be seen that the D2RS algorithm is able to significantly improve the system byte effective rate.

In Fig.8, while the block availability is fixed at 0.8 and 0.9 and the number of replicas increases from 1 to 5, for the system byte effective rate, the gap between the experimental results of D2RS algorithm and the theoretical analysis results are dramatically narrowed. We assume all tasks can be completed within the deadline

Table 6. Configuration of Data Centers

Data Center ID	Machine Number	PE per Machine	Processing Ability (MIPS)	Bandwidth
Super Data Center_0∼2	20	8∼16	400∼800	10.0 Gbps
Main Data Center_3∼11	10	4∼8	200∼400	1.0 Gbps
Ordinary Data Center_12∼38	5	1∼4	100∼200	0.1 Gbps

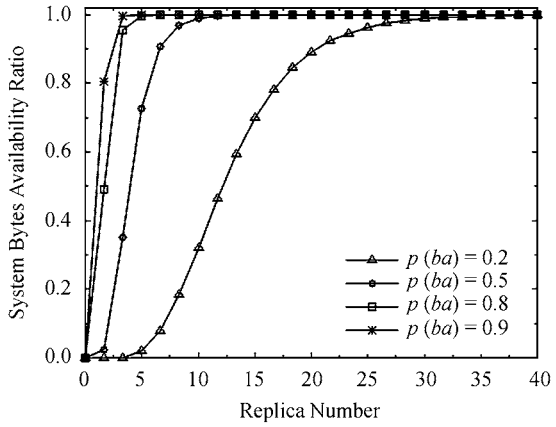


Fig.7. System byte effective rate with different number of replicas.

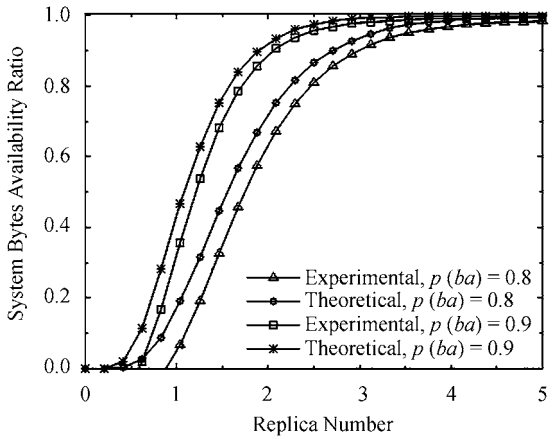


Fig.8. System byte effective rate comparison.

at the first time submission according to theoretical analysis.

2) *Number of Replicas.* To maintain system byte effective rate at a high level, a reasonable number of replicas of data files is needed, which can be obtained by (20). To evaluate the convergence of D2RS algorithm, the block availability is fixed at 0.8, and different values of adjustable parameter α are chosen to compute the needed number of replicas.

As shown in Fig.9, with time elapses, the number of replicas is increasing within a very short period of time. Then this number of replicas is maintained at a relatively stable level, which is determined by the adjustable parameter α. We conclude that the greater the adjustable parameter α, the more replicas are needed to maintain the requested system byte effective rate. This proves the D2RS algorithm has a good convergence rate.

3) *Response Time.* The response time for a data file is the interval between the submission time of the task and return time of the result. The average response

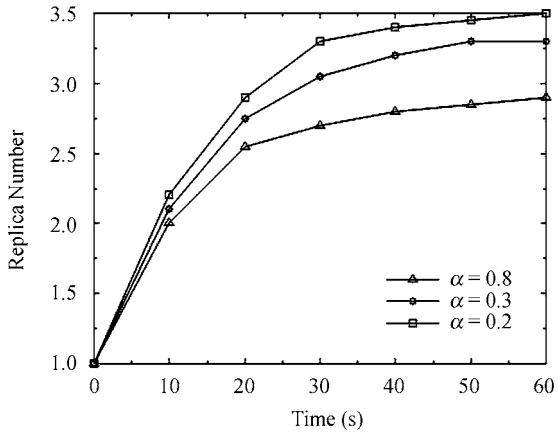


Fig.9. Number of replicas with different parameter α.

time of a system is the mean value of the response time for all data request tasks of the users, which can be obtained by (22).

$$rt_{avg} = \frac{\sum_{j=1}^{m}\sum_{k=1}^{m_j}(ts_{j_k}(rt) - ts_{j_k}(st))}{\sum_{j=1}^{m} m_j}, \qquad (22)$$

where $ts_{j_k}(st)$ and $ts_{j_k}(rt)$ are the submission time and the return time of the result of task k of the user j, respectively, and m_j is the number of the tasks of user j.

When D2RS algorithm is not used, we set the number of replicas of large-scale data file fixed as 2 and the total number of copies of a data file is 3. As shown in Fig.10, with the number of tasks increasing, the response time increases dramatically, especially when the number of tasks is more than 70%. The less the block availability is, the longer the response time will be. A conclusion can be made that the D2RS algorithm improves the response time and maintains the response time at a stable level within a short period of time.

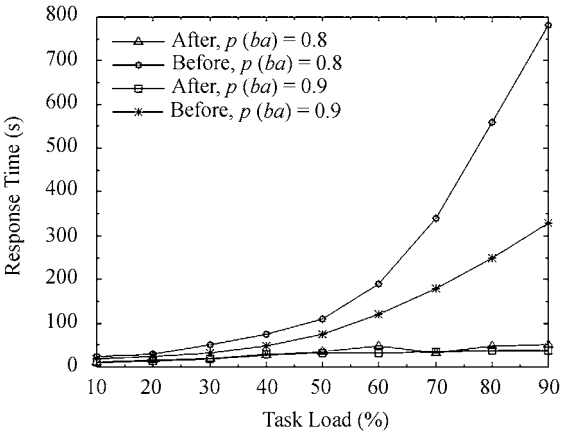


Fig.10. Response time comparison.

4) *Successful Execution Rate.* The successfully executed task is a task which is completely accomplished before its deadline dl at the first submission, where the deadline dl of a task can be obtained by (23).

$$dl_{j_k} = ts_{j_k}(st) + \mu_{j_k} \times rt_{avg} + \frac{len_{j_k}}{pb_{avg}}, \qquad (23)$$

where $ts_{j_k}(st)$ is the submission time of task k of user j, rt_{avg} is the average response time of a system, μ_{j_k} is the network status based adjustable parameter and is usually set to 0.4, len_{j_k} is the length of task k of user j, and pb_{avg} is the average processing ability.

The successful execution rate can be obtained by (24).

$$ser_{avg} = \frac{\sum_{j=1}^{m}\sum_{k=1}^{m_j}\xi_{j_k}}{\sum_{j=1}^{m} m_j},$$
$$\xi_{j_k} = \begin{cases} 1, & \text{if } t_j \text{ is success to } r_k, \\ 0, & \text{otherwise.} \end{cases} \qquad (24)$$

When D2RS algorithm is not used, we set the number of replicas of large-scale data file fixed as 2 and the total number of copies of a data file is 3. As shown in Fig.11, with the number of tasks increasing, the successful execution rate decreases dramatically, especially when the number of tasks is more than 50%. The less the block availability is, the less the successful execution rate will be. When the D2RS algorithm is used in the cloud, the successful execution rate can be maintained at a high level (more than 92%). A conclusion can be made that the D2RS algorithm improves the successful execution rate and maintains the successful execution rate at a high and stable level.

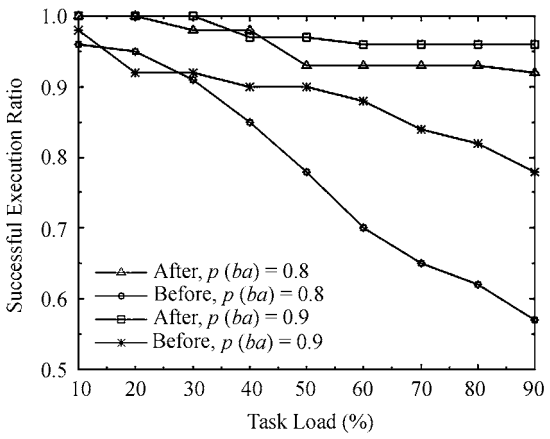


Fig.11. Successful execution rate comparison.

From the above experimental results, the following conclusions can be drawn that 1) the proposed dynamic data replication strategy effectively increases data availability and reduces user waiting time by very small number of replicas; 2) the proposed dynamic data replication strategy improves the system task successful execution rate, minimizes cloud system bandwidth consumption and reduces bandwidth consumption by placing the popular data files closer to the users; 3) the proposed dynamic data replication strategy effectively achieves system load balance by placing the popular data files according to the access history and the theory of temporal locality.

6 Conclusions and Future Work

High availability, high fault tolerance and high efficiency accesses to Internet based cloud data centers where failures are normal rather than exceptional are significant issues, and are often considered more valuable than high performance. Data replication allows reducing user waiting time and speeding up data access. It increases data availability by providing users with different replicas of the same service, and all of them in a coherent state. This paper presents a novel dynamic data replication strategy. It strives to increase data availability, improve cloud system task successful execution rate and minimize cloud system bandwidth consumption. Our contributions can be summarized as follows. 1) A mathematical model is formulated to describe the relationship between system availability and the number of replicas, which is missing in most existing research. 2) The popular data is identified, and corresponding replication operation will be triggered when the popularity of a data file passes a dynamic threshold. 3) Replicas are placed among data nodes in a balanced way. 4) A dynamic data replication strategy is proposed and evaluated. Experimental results demonstrate the efficiency of the improved system brought by the proposed strategy in a cloud.

There are still some studies to be done in the future. For instance, further reducing the user waiting time, speeding up data access, and further increasing data availability. In addition, the replication strategy will be deployed and tested on a real cloud computing platform. It is also planned to make data replication strategy as a part of cloud computing services to satisfy the special demands of cloud computing, and finally, to develop a complete dynamic data replication framework based on the proposed model.

Acknowledgement The authors gratefully thank Jun-Ling Hu for her help and comments.

References

[1] Foster I, Zhao Y, Raicu I, Lu S Y. Cloud computing and grid computing 360-degree compared. In *Proc. Grid Computing Environments Workshop*, Austin, TX, USA, Nov. 12-16, 2008, pp.1-10.

[2] Buyya R, Yeo C S, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 2009, 25(6): 599-616.

[3] Armbrust M, Fox A, Griffith R, Joseph A D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM*, 2010, 53(4): 50-58.

[4] Mell P, Grance T. The NIST definition of cloud computing. *Communications of the ACM*, 2010, 53(6): 50.

[5] Iosup A, Ostermann S, Yigitbasi N, Prodan R, Fahringer T, Epema D H J. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(6): 931-945.

[6] Han Y B, Sun J Y, Wang G L, Li H F. A cloud-based BPM architecture with user-end distribution of non-compute-intensive activities and sensitive data. *Journal of Computer Science and Technology*, 2010, 25(6): 1157-1167.

[7] Wang H. Privacy-preserving data sharing in cloud computing. *Journal of Computer Science and Technology*, 2010, 25(3): 401-414.

[8] He K Q, Wang J A, Liang P. Semantic interoperability aggregation in service requirements refinement. *Journal of Computer Science and Technology*, 2010, 25(6): 1103-1117.

[9] Xu B M, Zhao C Y, Hu E Z, Hu B. Job scheduling algorithm based on Berger model in cloud environment. *Advances in Engineering Software*, 2011, 42(7): 419-425.

[10] Ghemawat S, Gobioff H, Leung S T. The Google file system. *ACM SIGOPS Operating Systems Review*, 2003, 37(5): 29-43.

[11] Shvachko K, Hairong K, Radia S, Chansler R. The Hadoop distributed file system. In *Proc. the 26th Symposium on Mass Storage Systems and Technologies*, Incline Village, NV, USA, May 3-7, 2010, pp.1-10.

[12] Wang S S, Yan K Q, Wang S C. Achieving efficient agreement within a dual-failure cloud-computing environment. *Expert System with Applications*, 2010, 38(1): 906-915.

[13] Chang R S, Chang H P. A dynamic data replication strategy using access-weights in data grids.*Journal of Supercomputing*, 2008, 45(3): 277-295.

[14] Kim Y H, Jung M J, Lee C H. Energy-aware real-time task scheduling exploiting temporal locality. *IEICE Transactions on Information and Systems*, 2010, 93(5): 1147-1153.

[15] Wei Q, Veeravalli B, Gong B, Zeng L, Feng D. CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster. In *Proc. 2010 IEEE International Conference on Cluster Computing*, Heraklion, Crete, Greece, Sept. 20-24, 2010, pp.188-196.

[16] Bonvin N, Papaioannou T G, Aberer K. A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In *Proc. the 1st ACM Symposium on Cloud Computing*, Indianapolis, IN, USA, June 10-11, 2010, pp.205-216.

[17] Nguyen T, Cutway A, Shi W. Differentiated replication strategy in data centers. In *Proc. the IFIP International Conference on Network and Parallel Computing*, Zhengzhou, China, Sept. 13-15, 2010, pp.277-288.

[18] Mckusick M, Quinlan S. GFS: Evolution on fast-forward. *Communications of the ACM*, 2010, 53(3): 42-47.

[19] Ahmad N, Fauzi A A C, Sidek R M, Zin N M, Beg A H. Lowest data replication storage of binary vote assignment data grid. In *Proc. the 2nd International Conference Networked Digital Technologies*, Prague, Czech Republic, July 7-9, 2010, pp.466-473.

[20] Rahman R M, Barker K, Alhajj R. Replica placement design with static optimality and dynamic maintainability. In *Proc. the 6th IEEE International Symposium on Cluster Computing and the Grid*, Singapore, May 16-19, 2006, pp.434-437.

[21] Dogan A. A study on performance of dynamic file replication algorithms for real-time file access in data grids. *Future Generation Computer Systems*, 2009, 25(8): 829-839.

[22] Lei M, Vrbsky S V, Hong X. An on-line replication strategy to increase availability in data grids. *Future Generation Computer Systems*, 2008, 24(2): 85-98.

[23] Litke A, Skoutas D, Tserpes K, Varvarigou T. Efficient task replication and management for adaptive fault tolerance in mobile grid environments. *Future Generation Computer Systems*, 2007, 23(2): 163-178.

[24] Dobber M, van der Mei R, Koole G. Dynamic load balancing and job replication in a global-scale grid environment: A comparison. *IEEE Transactions on Parallel and Distributed Systems*, 2009, 20(2): 207-218.

[25] Yuan D, Yang Y, Liu X, Chen J. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 2010, 26(8): 1200-1214.

[26] Rood B, Lewis M J. Grid resource availability prediction-based scheduling and task replication. *Journal of Grid Computing*, 2009, 7(4): 479-500.

[27] Latip R, Othman M, Abdullah A, Ibrahim H, Md Sulaiman N. Quorum-based data replication in grid environment. *International Journal of Computational Intelligence Systems*, 2009, 2(4): 386-397.

[28] Avizienis A, Laprie J C, Randell B R, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 2004, 1(1): 11-33.

[29] Al-Kuwaiti M, Kyriakopoulos N, Hussein S. A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability. *IEEE Communications Surveys & Tutorials*, 2009, 11(2): 106-124.

[30] Ray I, Ray I, Chakraborty S. An interoperable context sensitive model of trust. *Journal of Intelligent Information Systems*, 2009, 32(1): 75-104.

[31] Tu M, Li P, Yen I L, Thuraisingham B M, Khan L. Secure data objects replication in data grid. *IEEE Transactions on Dependable and Secure Computing*, 2010, 7(1): 50-64.

[32] Wang J Y, Jea K F. A near-optimal database allocation for reducing the average waiting time in the grid computing environment. *Information Sciences*, 2009, 179(21): 3772-3790.

[33] Jung D, Chin S H, Chung K S, Suh T, Yu H C, Gil J M. An effective job replication technique based on reliability and performance in mobile grids. In*Proc. the 5th International Conference Advances in Grid and Pervasive Computing*, Hualien, Taiwan, China, May 10-13, 2010, pp.47-58.

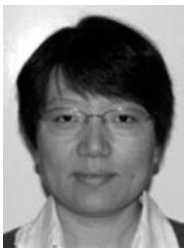
[34] Buyya R, Ranjan R, Calheiros R N. Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *Proc. 2009 International Conference on High Performance Computing & Simulation*, Leipzig, Germany, June 21-24, 2009, pp.1-11.

[35] Belalem G, Tayeb F Z, Zaoui W. Approaches to improve the resources management in the simulator CloudSim. In *Proc. the 1st International Conference Information Computing and Applications*, *Tangshan*, China, Oct. 15-18, 2010, pp.189-196.

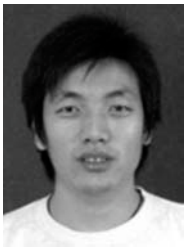
[36] Calheiros R N, Ranjan R, Beloglazov A, De Rose C A F, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software-Practice & Experience*, 2011, 41(1): 23-50.

Da-Wei Sun is a Ph.D. candidate at the School of Information Science and Engineering, Northeastern University, Shenyang, China. He received his M.Sc. degree in computer science from Northeastern University in 2009. His current research interests include cloud computing, trusted computing, green computing, virtualization technology, and information security.

Gui-Ran Chang received his Ph.D. degree in electrical engineering from the University of Tennessee, Knoxville, USA in 1991. He is currently a professor at the Computing Center of Northeastern University, Shenyang, China. His current research interests include computer networks, multimedia technology, and information security.



Shang Gao received her Ph.D. degree in computer application from the Northeastern University, China in 2000. She is currently a lecturer in School of Information Technology, Deakin University, Victoria, Australia. Her current research interests include computer network and applications, hypertext and hypermedia, and distance education.



Li-Zhong Jin is a Ph.D. candidate at the School of Information Science and Engineering, Northeastern University, Shenyang, China. He received his M.Sc. degree in computer science from Northeastern University in 2008. His current research interests include wireless sensor network, trusted computing and information security.

Xing-Wei Wang received his Ph.D. degree in computer science from Northeastern University, China in 1998. He is currently a professor at the School of Information Science and Engineering, Northeastern University, China. His research interests are mainly in routing algorithms and protocols, QoS control schemes, fault-tolerance and survivability models, mobility management mechanisms and resource assignment methods in next-generation Internet (NGI).