Ensuring Cloud data reliability with minimum replication by proactive replica checking

Wenhao Li, Yun Yang (IEEE Senior Member), Dong Yuan (IEEE Member)

Abstract — Data reliability and storage costs are two primary concerns for current Cloud storage systems. To ensure data reliability, the widely used multi-replica (typically three) replication strategy in current Clouds incurs a huge extra storage consumption, resulting in a huge storage cost for data-intensive applications in the Cloud in particular. In order to reduce the Cloud storage consumption while meeting the data reliability requirement, in this paper we present a cost-effective data reliability management mechanism named PRCR based on a generalized data reliability model. By using a proactive replica checking approach, while the running overhead for PRCR is negligible, PRCR ensures reliability of the massive Cloud data with the minimum replication, which can also serve as a cost effectiveness benchmark for replication based approaches. Our simulation indicates that, compared with the conventional 3-replica strategy, PRCR can reduce from one-third to two-thirds of the Cloud storage space consumption, hence significantly lowering the storage cost in a Cloud.

Index Terms — minimum data replication, proactive replica checking, data reliability, cost-effective storage, Cloud computing

1 Introduction

THE size of Cloud storage is expanding at a dramatic speed. It is estimated that by 2015 the data stored in the Cloud will reach 0.8 ZB (i.e., $0.8*10^{21}$ Bytes or 800,000,000 TB), while even more data is "touched" by the Cloud within the data lifecycle [8]. Meanwhile, with the development of the Cloud computing paradigm, Cloud-based applications have put forward a higher demand for Cloud storage. While the requirement of data reliability should be met in the first place, data in the Cloud needs to be stored in a highly cost-effective manner.

Reliability is defined in standard TL9000 [4] as "the ability of an item to perform a required function under stated conditions for a stated time period". For data reliability specifically, it can be understood as "the probability of the data surviving in the system for a given period of time" [12]. In the area of distributed data storage, because of the inevitable occurrence of disk failures, data reliability has become one of the most important metrics of the storage system, indicating the ability to keep data consistent. In modern Clouds, data replication is the most commonly applied approach for providing data reliability assurance, which creates and stores multiple replicas of the data to reduce the probability of data loss. For example, storage systems such as Amazon S3 [1], Google File System [10] and Hadoop Distributed File System [5] all adopt similar data replication strategies that we call the conventional 3-replica strategy, in which three replicas, i.e., three data copies including the original data, are stored for all data. However, due to the accelerating growth of Cloud data, current replication-based data reliability management has become a bottleneck for the development of Cloud data storage, because these data replication strategies are consuming too much extra storage space, thus incurring a huge storage cost.

In this paper, our research focuses on minimizing the Cloud storage consumption by minimizing data replication while meeting the data reliability requirement. This paper presents three major contributions.

Firstly, through analysis of existing studies, a generalized data reliability model for multiple replicas is proposed, in which the data reliability with variable disk failure rates is well investigated. Compared with much research that assumes a constant disk failure rate [2], [12], [20], our generalized data reliability model is able to better describe data reliability over a wide range of disk failure rate patterns, hence data reliability management over both virtual and physical disks is feasible.

Secondly, to minimize replicas, and therefore the storage consumption in the Cloud, a cost-effective data reliability management mechanism named PRCR is presented. This mechanism is designed to be implemented by the Cloud storage providers in order to increase the profit and/or the competitiveness by cost saving, as well as serve as, a benchmark for storage consumption of different approaches. PRCR could also potentially benefit Cloud storage users with cheaper storage services without jeopardizing data reliability. By applying PRCR, the Cloud data can be stored with the minimum replication while meeting the reliability requirement of Cloud data. PRCR has the following features:

1. It is able to ensure the data reliability of storage devices with variable disk failure rates.
2. It is able to manage large amounts of data in the Cloud with a negligible running cost.
3. It provides data reliability management in a highly

• *W. Li is with the School of Computer Science and Technology, Shandong University, China, Y. Yang is with the School of Computer Science and Technology, Anhui University, China and the School of Software and Electrical Engineering, Swinburne University of Technology, Australia, and D. Yuan is with the School of Electrical and Information Engineering, University of Sydney, Australia. E-mails: ayumi_5420467@hotmail.com, yyang@swin.edu.au, yd0116@gmail.com*

• xxxx-xxxx/0x/$xx.00 © 200x IEEE

cost-effective way. By applying PRCR, a wide range of data reliability assurance can be provided with the minimum number of replicas, which is no more than two.

Thirdly, as a direct consequence of PRCR, the minimum replication benchmark with any data reliability requirement can be provided. This benchmark can be used for evaluating various replication-based data storage approaches. By comparing the minimum replication benchmark with the replication level of a data storage approach, the cost effectiveness of the data storage approach can be clearly demonstrated.

This paper is one of a series of articles that towards reducing data storage consumption in the Cloud [17], [16], [23]. Prior to this paper, our preliminary version of this research was published in [16]. In that paper, we only investigated a special case of the data reliability model and PRCR by assuming that the disk failure rate is a constant. Compared to [16], this paper has substantially extended the data reliability model, PRCR mechanism and evaluation to demonstrate that the data reliability management with a variable disk failure rate is feasible and explicitly presents the minimum replication benchmark. In addition, existing research on how the variable disk failure rate affects distributed replicas is very limited. This paper is one of the few works investigating the data replication techniques with a variable disk failure rate.

The rest of the paper is organized as follows. The related work on data reliability, data replication and cost-effective data storage is addressed in Section 2. The problem analysis is stated in Section 3. The generalized data reliability model is proposed in Section 4. The idea of PRCR and its high level design are presented in Section 5. The detailed design for PRCR, including the working process and optimization algorithms, is described in Section 6. The evaluation of PRCR is discussed in Section 7. Conclusions and future work are summarized in Section 8.

2 Related Work

Data reliability has always been a key issue in the field of distributed data storage. Considering factors that are due to the storage system itself, permanent disk failure caused by non-human factors is considered to be the major reason for data loss. The reliability of disks has been investigated for decades in both academia and industry [7], [20], [22]. Most of these studies have assumed an exponential data reliability model, in which the failure rate of each disk is a constant. For example, recent studies that analyze data reliability with Markov chain models assume that the failure rates of all disks in the storage system are the same [13], [20]. However, a constant disk failure rate cannot explain all of the phenomena happening in reality. It has been very well known that the failure rate of disk drives follows what is often called a "bathtub" curve, where disk failure rate is higher in the disk's early life, drops during the first year, remains relatively constant for the remainder of the disk's useful lifespan and rises again at the end of the disk's lifetime [11], [22]. Some other studies have also obtained results that contradict the constant disk failure rate model. For example, [7] shows that the disk failure probability of populations of disks generally do not follow an exponential distribution. For fixing this inconsistency, the International Disk Drive Equipment and Materials Association (IDEMA) proposed a compromised presentation for disk failure rates that uses discrete disk failure rates [14], where the lifespan of each disk are divided into different life stages with different failure rates. Such model has been demonstrated to be feasible in [22], and a nine-month investigation conducted by Google also obtained results very consistent to this model [19]. In this paper, we describe the disk failure rate pattern in the IDEMA style, which divides the lifespan of disks into discrete life stages with discrete disk failure rates, and we also conduct our research based on the disk failure rates provided by IDEMA standards and Google's nine-month disk failure trend study.

Apart from research on disk reliability, many efforts for ensuring data reliability have also been made in the software aspect. In [15], analytical data reliability model and data replica schemes are proposed for minimizing the data missing rate of the storage system. In [10], a data partitioning approach is implemented to improve the reliability, availability and data access performance in the storage system, in which the original data is stored in the form of data chunks of the same size. However, instead of taking the variable disk failure rate patterns of storage devices into consideration, these studies also consider the failure rates of the storage devices as a simple constant value.

Among all the existing approaches for supporting data reliability, data replication has been considered as a dominant approach in current distributed storage systems. Some existing works on large-scale distributed storage systems have been proposed such as [3], [9]. Specifically, in Cloud computing, data replication technologies have been widely adopted in current commercial Cloud systems. Some typical examples include Amazon Simple Storage Service (Amazon S3) [1], Google File System (GFS) [10], Hadoop distributed file system (HDFS) [5]. Although data replication has been widely used, there is a side effect that it would consume considerable extra storage resources and incur significant additional cost. To address this issue, Amazon S3 published its Reduced Redundancy Storage (RRS) solution to reduce the storage consumption [1]. However, such cost reduction is realized by sacrificing data reliability. By using RRS, only a lower level of data reliability can be offered. In addition to RRS, some of our previous works made contributions in reducing storage consumption for replication-based Cloud storage. For

example, in [17], we proposed a cost-effective dynamic data replication strategy for data reliability in Cloud data centers, in which an incremental replication method is applied to reduce the average replica number while meeting the data reliability requirement. However, for long-term storage or storage with a very high reliability requirement, this strategy could generate even more than three replicas for the data, so that its ability to reduce storage consumption is limited.

Besides direct data replication, another type of data storage approach that leverages erasure coding techniques has also been studied [13], [21]. In [13], novel LRC codes are applied to Windows Azure Storage service as an alternative approach to replication that stores non-changeable data blocks of over 1 GB. In [21], an erasure coded storage system named Hitchhiker was proposed and implemented in HDFS. Unlike direct data replication approaches for storage, erasure coding approaches divide data into several data blocks and store them with additional erasure coding blocks. By using erasure coding approaches, data reliability can be ensured at a quite high level with very low data redundancy. However, the major disadvantage of erasure coding approaches is apparent, i.e., the computation overhead for coding and decoding data can be significant. As will be further explained in Section 3.2, erasure coding is not the best solution for the data storage of certain data-intensive Cloud applications.

The research presented in this paper focuses on the data reliability issue in the Cloud with a replication-based data storage scheme. The issue is closely related to the aspects mentioned above, in which data replication and analysis of disk failure rates are combined to ensure the data reliability. Compared with existing research for data reliability, our research is more comprehensive in supporting data reliability assurance with variable disk failure rate. In terms of our data reliability assurance solution, PRCR minimizes storage consumption based on the premise of not sacrificing data reliability, which is different from the Amazon S3 RRS service. Moreover, our replication solution is more flexible. It allows data to be stored with only one replica for reducing storage consumption and a variety of data reliability requirements can be met without jeopardy when stored with two replicas. Compared with the conventional 3-replica strategy, PRCR is able to provide the same or even higher reliability assurance. In general, PRCR consumes minimum storage with no more than two replicas for all the data in the Cloud with any data reliability requirement.

3 Motivating Example and Problem Analysis

3.1 Motivating Example

Potentially, in order to investigate the data storage issue of massive scientific research data, the scenario of a pulsar searching application has been investigated. The pulsar searching survey is one of the astronomical research programs currently being conducted by the Astrophysics Group at Swinburne University of Technology. With huge amounts of observation data obtained from Parkes Radio Telescope (http://www.parkes.atnf.csiro.au/), the pulsar searching application carries out many complex and time consuming tasks, and easily generates hundreds of terabytes of data during execution. For a better presentation of the pulsar searching application, a small application instance is presented in the Supplementary Material as an example. In the application instance, the raw telescope data for an eight-minute observation is processed. It generates hundreds of GBs of intermediate files and takes tens of hours for data processing. Nevertheless, for achieving different searching goals, the telescope observation time is often much longer, and hence more data needs to be generated and processed. For an observation conducted eight hours a day for 30 days, the size of generated files could reach 543.6TB. Moreover, as the pulsar searching program continues, the number and size of generated files become bigger and bigger, so the cost for data storage becomes much higher.

3.2 Problem Analysis

For data-intensive Cloud applications similar to the pulsar searching application, there is a quite common situation that needs to be faced due to their data-intensive characteristics. During the execution of such applications, large amount of data is generated and processed, so that the storage consumption incurred could be very high. However, whilst the cost for storing data is inevitable, there is still plenty of room to reduce it. On one hand, current Cloud storage systems often use multi-replica data replication strategies for ensuring data reliability, which generate high level data redundancy. For example, storage systems such as Amazon S3, Google File System and Hadoop Distributed File System all adopt the conventional 3-replica strategy, in which by default three replicas are stored for all data. By using the conventional 3-replica strategy, based on the original data (regarded as the first replica), two other replicas are generated at once for storage, i.e., 200 per cent of extra storage consumption. Such strategy causes a huge amount of extra storage consumption due to replication. For data-intensive applications such as the pulsar searching example, the extra storage consumption could be huge. For example, by applying the conventional 3-replica strategy, storing 543.6TB pulsar searching data mentioned above needs about 1630TB storage space, of which two-third of storage space are for data redundancy. On the other hand, in Cloud applications, the data reliability requirements and storage durations of different data are not the same. Taking the pulsar searching application as an example, the generated files can be divided into two primary data types. One type is critical and would be reused for a long period of time, for examples, the extracted beam files, the XML files and the

de-dispersion files. These files record the current state of the universe, which are very important and can be reused for long term analysis. For data of such kind, high data reliability assurance and long-term storage duration are necessary. The other type is only used for shorter term and lacks long-term value, for examples, the accelerated de-dispersion files, seek result files and all the candidate lists. For data of such kind, because of the short storage duration, as will be demonstrated later in Section 4, one replica would suffice to meet both the requirements of data reliability and storage duration. However, by applying the conventional 3-replica strategy, all data is stored with the same number of replicas, which is inappropriate for both types. For the former data type, when a large amount of data is stored, the data reliability assurance of three replicas incurs high storage consumption. For the latter data type, the additional two replicas are simply not needed, thus consuming unnecessary extra storage space.

Instead of using the erasure-coding-based data storage scheme, our research focuses on Cloud with a direct replication-based data storage scheme for two reasons: First, for pulsar searching and a wide range of similar Cloud applications that involve intensive large scale data processing and generation, applying erasure coding approaches is not desirable: both computation and time overheads for encoding and decoding the data are so high (at the level of seconds per MB or even more [13], [21]) that the overall cost saving advantage by reducing storage consumption can be significantly weakened. Second, the replication-based data storage scheme is currently the most widely used Cloud storage scheme, which is adopted by many Cloud service providers. In order to reduce the storage consumption without jeopardizing the data reliability requirement for data-intensive applications in the Cloud, a new replication mechanism needs to be designed to replace the conventional 3-replica strategy.

4 Data Reliability Model

Instead of the conventional 3-replica strategy, there is a way in which we can provide data reliability with fewer replicas. A mathematical model for data reliability has provided the possibility of reducing the number of replicas while meeting the data reliability requirement. In this section we investigate the relationship between data reliability and variable disk failure rate, and propose a generalized data reliability model for data with multiple replicas.

4.1 Data reliability with constant disk failure rate

As mentioned in Section 2, many existing theories assume an exponential disk reliability model, in which the disk failure probability follows the exponential distribution with a constant disk failure rate. In that case, the reliability of a disk over period T (i.e., 1-disk failure probability) can be expressed as $R(T)$, where:

$$R(T) = e^{-\lambda T} \quad (1)$$

The replicas stored in the disk should have the same reliability as the disk. Therefore, (1) is also applicable for calculating the reliability of a single replica when the disk failure rate is a constant. Therefore, $R(T)$ also indicates the data reliability, i.e., the probability of the replica survives over period T with λ as the disk failure rate.

4.2 Data reliability with variable disk failure rate

Although exponential distribution can be used to describe the data reliability when disk failure rate is a constant as mentioned in Section 2, the failure rates of disks actually vary from time to time. In practice, quality control is conducted for each batch of disks before they leave the factory, in which samples should be tested to ensure the quality of the product being consistent, and hence we consider the failure rate pattern of a batch of disks is known. As one of the same batch of disks, the actual failure pattern of the disk should adhere to the batch failure rate pattern quite well statistically. Hence each disk's failure rate pattern should follow the failure rate pattern of the batch of disks, which is known.

Here we investigate the data reliability with a variable disk failure rate. To calculate the data reliability with a variable disk failure rate, we first assume that the data survival probability with a constant disk failure rate follows exponential distribution (i.e., (1) holds). Second, according to the IDEMA standard, when the disk failure rate is a variable, we assume the disk failure rate pattern contains several life stages, and in each life stage of a disk, the disk failure rate does not change. Assume that replica r is stored in disk D between t_0 and t_n, in this period of time, the disk failure rate pattern of disk D contains n life stages, in which the disk failure rates are $\lambda_1, \lambda_2, ..., \lambda_n$ respectively where λ_i indicates the disk failure rate between time t_{i-1} and t_i, $i \in N$. Fig. 1 shows the failure rate pattern of disk D between time t_0 and t_n.

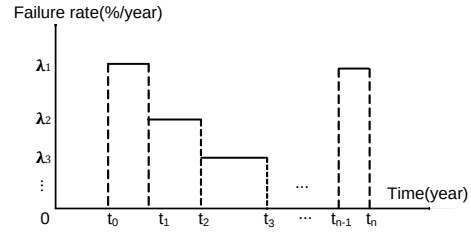

Fig. 1. The failure rate pattern of disk D between time t_0 and t_n

Let event A_j be disk D surviving from t_{j-1} to t_j, where $j \in N$, the probability that disk D survives from t_0 to t_n can be described as $P(A_n A_{n-1} ... A_1)$. According to the property of conditional probability, we have:

$$P(A_n A_{n-1} ... A_1) = P(A_n \mid A_{n-1} ... A_1) P(A_{n-1} A_{n-2} ... A_1)$$
$$\cdot = ... = P(A_n \mid A_{n-1} ... A_1) P(A_{n-1} \mid A_{n-2} ... A_1) ...$$
$$\cdot P(A_2 \mid A_1) P(A_1)$$

in which $P(A_j \mid A_{j-1} A_{j-2} ... A_1)$ indicates the probability of

disk *D* surviving (i.e., the reliability of disk *D*) between t_{j-1} and t_j, given that *D* is alive at time t_{j-1}. Because replica *r* has the same reliability as disk *D*, $P(A_j \mid A_{j-1}A_{j-2}...A_1) = R_{tj}$, where R_{tj} is the reliability of data stored from t_{j-1} to t_j. Therefore, we have $P(A_nA_{n-1}...A_1) = R_{t1}R_{t2}...R_{tn}$.

According to (1), $R_{tj} = e^{-\lambda_j(t_j - t_{j-1})}$. Let $T_j = t_j - t_{j-1}$, hence we have:

$$P(A_nA_{n-1}...A_1) = e^{-\lambda_1 T_1}e^{-\lambda_2 T_2}...e^{-\lambda_n T_n}$$
$$\cdot = \exp((-\sum_{j=1}^{n}\lambda_j T_j / \sum_{j=1}^{n} T_j)\sum_{j=1}^{n} T_j)$$

Because of $P(A_nA_{n-1}...A_1) = R(T)$, this equation can be denoted as:

$$R(T) = e^{-\bar{\lambda}T} \quad (2)$$

In (2), several life stages of the disk during the whole lifespan of the replica are transformed into a single variable $\bar{\lambda}$, where $\bar{\lambda} = \sum_{j=1}^{n}\lambda_j T_j / \sum_{j=1}^{n} T_j$ is the weighted mean of the disk failure rate with storage durations as weights ("weighted average failure rate" for short). $T = \sum_{j=1}^{n} T_j$ is the sum of all storage durations, which is the lifespan of the data. From (2), we can tell that data reliability, i.e., the probability of one replica survives, with a variable disk failure rate, also follows the exponential distribution, while the disk failure rate becomes the weighted mean of all the disk failure rates during the storage. It can be seen that (1) is a special case of (2) when the disk failure rate is a constant.

4.3 Generalized data reliability model

In previous sub-sections we discussed the data reliability of storing one replica. Based on these discussions, a generalized data reliability model with a variable disk failure rate for multiple replicas is proposed. As indicated in Section 4.2, we assume that the failure rate pattern of each disk could theoretically be known by following the failure rate pattern of the batch of disks. In that case, each disk failure can be considered independent. Assume that replicas of the same data be stored in different disks. According to (2), the data reliability with multiple replicas can be derived from (3):

$$R(T_k) = 1 - \prod_{i=1}^{k}(1 - e^{-\bar{\lambda}_i T_k}) \quad (3)$$

In this equation, *k* is the number of replicas, $\bar{\lambda}_i$ is the weighted average failure rate of the disk storing replica r_i and T_k is the storage duration of the data with *k* replicas. The right-hand side of the equation describes the probability that at least one of the *k* replicas survives during the storage duration of T_k. Equation (3) reveals the relationship between data reliability, the number of replicas, disk failure rates and storage duration. If the number of replicas and the failure rates of disks are known, the relationship between storage duration and data reliability can then be derived. It can be seen that (2) is a special case of (3) when *k=1*.

5 Cost-Effective Mechanism of PRCR

In this section, the idea of proactive replica checking is described. Based on this idea, we propose a cost-effective data reliability management mechanism for Cloud data storage named PRCR (Proactive Replica Checking for Reliability), and describe it at the high level.

5.1 Proactive Replica Checking

There is a well-known property of exponential distribution called the memory-less property, which is that for all $s,t > 0$, there are $P(T > s+t \mid T > s) = P(T > t)$. Because the data reliability, i.e., the probability of a single replica surviving, follows exponential distribution, and the data reliability of each replica is independent, this property should also apply to our generalized data reliability model for multiple replicas. This property denotes that the data reliability from time *s* to *s+t* is equivalent to that from time *0* to *t*. According to this property, as long as we can guarantee that the data is not lost at a certain moment, the data reliability for any period from that moment can be calculated. More importantly, according to (3), shorter storage duration results in lower probability of data loss. Thus, the basic idea of managing data reliability based on proactive replica checking is formed. While data is stored on disks, each replica of the data is proactively checked periodically and the loss of replicas is discovered and recovered within each period. By conducting proactive replica checking at a certain frequency, a certain level of data reliability assurance can be provided.

Based on this idea, the PRCR mechanism is proposed. In PRCR, the data in the Cloud is managed in different types according to its expected storage duration and reliability requirements. For data that is only for short-term storage and/or requires the data reliability that a single replica can supply, one replica would be enough; for data that is for long-term use and/or has a data reliability requirement higher than the reliability assurance of a single replica, two replicas are stored which are periodically and proactively checked. During the proactive replica checking, replicas of the data are accessed to determine their existence[1]. The proactive replica checking tasks must always be conducted before the reliability assurance drops below the reliability requirement. Once found, any single replica loss can be quickly recovered according to certain strategy, such as [6], [18], so that the reliability of the data can be ensured.

In some extreme cases, both replicas may be lost in a small time window. The probability of such a situation is already incorporated in the data reliability model. Given

[1] As the proactive replica checking is conducted inside the Cloud, we believe that the instability of the network is minimized, hence the replica is considered lost when it cannot be accessed.

a certain data reliability requirement, PRCR is responsible for the probability of data loss being within the agreed range, so that the data reliability requirement is met. For example, given the data reliability requirement of 99.99 per cent per year, PRCR ensures that the data loss rate is no bigger than 0.01 per cent of all the data per year, and hence does not jeopardize the reliability assurance in overall terms.

5.2 Overview of PRCR

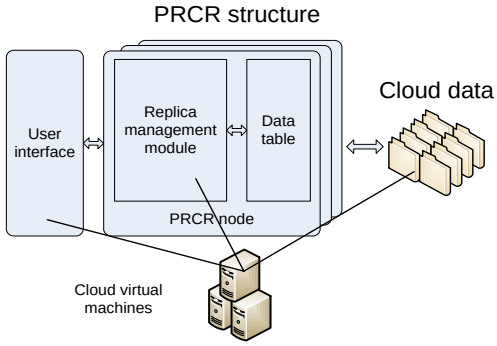

Fig. 2. PRCR architecture

PRCR is a data reliability management mechanism that can manage large amounts of data in the Cloud. By using PRCR, Cloud data can be stored with minimum replication while meeting the data reliability requirement which can also serve as a cost effectiveness benchmark. For the ease of description, we simply use the term "file" as the data storage unit managed by PRCR. PRCR is normally conducted as a data reliability management service provided by the Cloud storage providers. It runs on virtual machines in the Cloud. Fig. 2 shows the architecture of PRCR. Note that Cloud virtual machines are for running user interface, PRCR nodes and conducting proactive replica checking respectively at the storage provider's cost.

User interface: It is the component of PRCR responsible for determining the minimum replica number, creating replicas (if necessary), creating and distributing metadata of files. First, when the original replica of a file is created (generated or uploaded) in the Cloud, the user interface determines the minimum number of replicas (i.e., one or two replicas). Second, if a file needs to be stored with two replicas, the user interface calls Cloud service to create the second replica for the file. Third, if a file is stored with two replicas, the metadata of the file is created and distributed to an appropriate PRCR node. For all files managed by PRCR, there are in total six types of metadata attributes, which are file ID, time stamp, data reliability requirement, expected storage duration, checking interval, and replica address. File ID is the unique identification of the file. Time stamp records the time when the last proactive replica checking task for the file was conducted. The data reliability requirement and expected storage duration are requirements for the storage qualities. Checking interval is the time interval between two consecutive proactive replica checking tasks for the same file. Replica address indicates the location of each replica. The file ID and replica address are automatically given when the original and second replica of the file are created. Time stamp is updated when the proactive replica checking task is conducted. The data reliability requirement and expected storage duration are given by the storage user, and maintained for rebuilding metadata in case of replica loss. These two attributes are the only information that can be provided by the storage user, while default value may apply if they are not given (e.g., 99.9999% per year for data reliability requirement and 1 year for expected storage duration). All the other storage structure related attributes are transparent to them. The checking interval is obtained based on the data reliability requirement and the expected storage duration, by using the storage duration prediction algorithm that will be mentioned in the next section. One or more checking intervals may apply throughout the lifespan of the file in the Cloud. Depending on the time stamp and the checking interval, PRCR is able to determine the time that files need to be checked. All replicas of the file can be found through their addresses.

PRCR node: It is the core component of PRCR responsible for the management of metadata and replicas. In order to provide data reliability assurance to meet a wide range of data reliability requirement with different storage durations, the PRCR normally should be composed of one user interface and multiple PRCR nodes. PRCR nodes work independently, so can easily be created and destroyed, as required by variation in the amount of data to be managed. Each PRCR node contains two sub-components: data table and replica management module.

Data Table: For all files that each PRCR node manages, the above mentioned metadata attributes are maintained in the data table. To ensure the data reliability of files, all metadata are periodically scanned by the replica management module. The so called "scan" inspects the metadata of a file in the data table to determine whether replica checking is necessary. In the data table, each round of the scan is called a scan cycle, in which all of the metadata in the data table is sequentially scanned once. The scan cycle of each PRCR node is set to a fixed value to scan files at certain frequency. By doing so, the frequency of conducting proactive replica checking tasks can be determined, which corresponds to certain data reliability assurance that can be provided. However, due to the limited performance of the virtual machine that the PRCR node is running on, the time constraint of scan cycle means that the maximum capacity, i.e., maximum number of managed files, of the PRCR node is also limited. The reliability of the data table itself is beyond the scope of this paper. In fact, a conventional primary-secondary backup mechanism may well serve the purpose.

Replica Management Module: It is responsible for

scanning the metadata in the data table and co-operating with the Cloud virtual machines to process the proactive replica checking tasks. In each scan cycle, the replica management module scans the metadata in the data table and determines whether the file needs to be checked. If a file needs to be checked, the replica management module obtains its metadata from the data table and sends it to a Cloud virtual machine for proactive replica checking. After the proactive replica checking task is finished, the replica management module conducts further actions according to the returned result. In particular, if any replica is lost, the replica management module initializes the recovery process for creating a new replica. For the recovery of data with different usages, contents and formats, different data recovery strategies can be applied to achieve required goals [6], [18].

6 Design of PRCR

In this section, the design details of PRCR are presented at a high level.

6.1 Working Process of PRCR

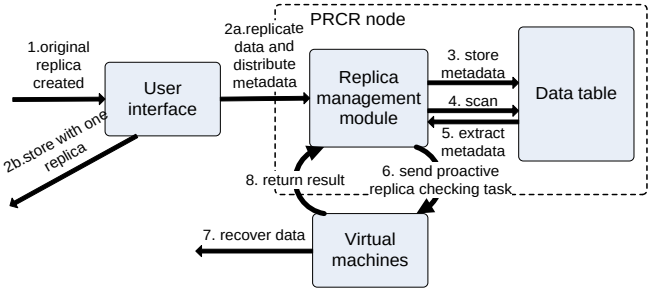

Fig. 3. High level working process of PRCR

In Fig. 3, we illustrate the high level working process by following the lifecycle of a file managed by PRCR in the Cloud without unnecessary implementation details.

1. The process starts at the time when the original replica of the file is created in the Cloud. According to the disk failure rate, the expected storage duration and data reliability requirement, the user interface determines whether to store the file with one replica or two.
2. According to the calculation in the user interface, if one replica cannot satisfy the data reliability and storage duration requirements of the file, the user interface creates a second replica by calling Cloud services, and calculates the checking interval(s) of the file. Its metadata is then distributed to the appropriate PRCR node (2a). If one replica is sufficient, only the original replica is stored and the metadata of the file is not created (2b).
3. Metadata attributes of the file are stored in the data table of the corresponding PRCR node.
4. Metadata is scanned periodically according to the scan cycle of the PRCR node. According to file's time stamp and the current checking interval, PRCR determines whether proactive replica checking is needed.
5. If proactive replica checking is needed, the replica management module obtains the metadata of the file from the data table.
6. The replica management module assigns the proactive replica checking task to one of the Cloud virtual machines for proactive replica checking. The Cloud virtual machine executes the task, in which both replicas of the file are checked.
7. The Cloud virtual machine conducts further action according to the result of the proactive replica checking task: if both replicas are alive or lost, go to step 8; if only one replica is lost, the virtual machine calls the Cloud services to generate a new replica based on the replica that is alive.
8. The Cloud virtual machine returns the result of the proactive replica checking task, while in the data table, the time stamp and checking interval(s) are updated. Specifically, (1) if both replicas are not lost, the next checking interval is put forward as the current checking interval; and (2) if a replica is lost and recovered on a new disk, the new replica address is stored and all the checking interval(s) are recalculated. Otherwise, further steps could be conducted, for example, a data loss alert could be issued.

Note: Steps 4 to 8 form a continuous loop until the expected storage duration is reached or the file is deleted. If the expected storage duration is reached, either the storage user could renew the PRCR service or PRCR could delete the metadata of the file and stop the proactive replica checking process.

6.2 Key Algorithms for PRCR

Within the whole working process of PRCR, in order to determine the minimum replica number and improve the performance of PRCR including computation overhead and utilization of the data management capacity, two algorithms are proposed, which are storage duration prediction algorithm and metadata distribution algorithm respectively.

Storage duration prediction algorithm

The storage duration prediction algorithm has two purposes. First, it determines the minimum replica number (i.e., one or two) for meeting the data reliability requirement. Second, given a certain data reliability requirement, it calculates the longest storage duration of the data while the data reliability requirement is met ("LSDWP" for short). It equals to the checking interval in PRCR but it indicates the time that the reliability assurance drops to the level that the reliability requirement can no longer be met. In PRCR, the scan cycle should be no bigger than LSDWPs of each file so as to provide sufficient data reliability assurance. The algorithm is applied to both the user interface and PRCR nodes. In the user interface it is used for determining the minimum replica number and calculating LSDWPs of

files. In PRCR nodes it is used to recalculate LSDWPs of files after replicas are recovered on new disks.

In a commercialized storage system such as that of the Cloud, "data reliability" has two aspects of meaning - data reliability requirement *RR(t)* and data reliability assurance *RA(t)*. *RR(t)* indicates the data reliability that storage users need to achieve in the duration of *t*, while *RA(t)* indicates the data reliability that the system is able to provide within the duration of *t*. Usually, *RR(t)* is provided under unit time (i.e., *RR(1)*), while *RA(t)* is used to determine whether the data reliability requirement is met. In order to meet the data reliability requirement, a storage system must comply with the following rules:

Rule 1: The data reliability assurance must not be lower than the data reliability requirement.

Rule 2: The data reliability assurance should follow the generalized data reliability model.

$$RA(t) = 1 - \prod_{i=1}^{k} (1 - e^{-\bar{\lambda}_i t})$$

According to Rule 1, the loss rate of files must be smaller than that of user expectation. Therefore, we have:

$$1 - RR(1) \Big/ 1 \geq 1 - RA(t) \Big/ t$$

Note that value "1" on the denominator of the left hand side indicates the unit storage duration of 1 year. The above inequality can be transformed to:

$$RA(t) \geq 1 - (1 - RR(1))t \quad (4)$$

According to Rule 2, the data reliability assurance with single replica and two replicas can be derived, which are:

with single replica, $$RA(t) = e^{-\bar{\lambda}_1 t} \quad (5)$$

with two replicas, $$RA(t) = 1 - (1 - e^{-\bar{\lambda}_1 t})(1 - e^{-\bar{\lambda}_2 t}) \quad (6)$$

Inequality (4) is the key to build the relationship between the data reliability requirement and the storage duration in the storage system. After combining (4) and (5) above, we have:

$$RR(1) \leq (e^{-\bar{\lambda}_1 t} + t - 1) \Big/ t \quad (7)$$

This inequality shows the relationship between data reliability requirement and the storage duration with single replica. If this inequality holds, it means that single replica suffices to meet the data reliability requirement. Otherwise, if this inequality does not hold, the storage with single replica may jeopardize the data reliability requirement, and the storage with two replicas is necessary.

After combining (4) and (6) above, we have:

$$RR(1) \leq 1 - (1 - e^{-\bar{\lambda}_1 t})(1 - e^{-\bar{\lambda}_2 t}) / t$$

This inequality shows the relationship between data reliability requirement and LSDWP with two replicas. From this inequality, it can be found that while the data reliability requirement is given, LSDWP cannot exceed a certain value. By solving (8) below, this certain storage duration can be obtained.

$$RR(1) = 1 - (1 - e^{-\bar{\lambda}_1 t})(1 - e^{-\bar{\lambda}_2 t}) / t \quad (8)$$

Due to the variable nature of the average disk failure rate, variable $\bar{\lambda}$ changes along with the storage duration and the exact age of the disk. In addition, the LSDWP of a file is also not a constant. Therefore, the process of solving (8) needs to be conducted more than once to obtain all the LSDWPs throughout the lifespan of the file.

Due to the variable nature of the average disk failure rate, there are two difficulties to solve (8) in Section 6.2. The first one is that $\bar{\lambda}$ changes along with the storage duration and the exact age of the disk. The second one is that the LSDWP of a file is also not a constant, so that the process of solving (8) needs to be conducted more than once to obtain all the LSDWPs throughout the lifespan of the file. PRCR includes solutions to overcome the two difficulties. For the first one, the average disk failure rate is converted into a piecewise function $\bar{\lambda}(t)$. According to the disk failure rate pattern of the disk (which is described in IDEMA style) and the start time of the storage period, the average disk failure rate can be calculated by following a piecewise function containing *n* sub functions, in which *n* is the number of life stages contained in the disk failure rate pattern after the start time. By doing this, (8) is transformed into an equation in which *t* is the only independent variable and variable $\bar{\lambda}$ is eliminated. For the second one, we notice that when replicas are first created and stored, the disks for storing these replicas and the disk failure rate patterns are determined. Therefore, the algorithm calculates the LSDWPs of a file in one go when the file is first created in the Cloud. As long as replicas of the data are not lost, the algorithm does not need to be conducted again, resulting in better efficiency.

```
Algorithm:  Data Reliability Prediction Algorithm
Input:   ET;              // Expected storage duration
         RR(1);           // Data reliability requirement
         P1,P2;           // Disk failure rate patterns of disks 1 and 2
         StartT;          // Start time of the algorithm
Output: SDS;              // Set of longest storage durations
```

```
01. λ̄1 ←calculateAverageFailureRate(P1,StartT,ET);
02. if ((e^(-λ̄1·ET) + ET - 1)/ET - RR(1) < 0){   // Determine replica number
03.    T=StartT;                  // The start time of each storage period
04.    while (T<=ET+StartT) {
05.       λ1(t) ← obtainPiecewiseFunction(P1,T);
06.       λ2(t) ← obtainPiecewiseFunction(P2,T);
07.       solve (8);
08.       SD←the positive real root of (8);// Longest storage duration
09.       T=T+SD;
10.       SDS←SD;
11.    } return SDS;
12. } else return -1;          // The file can be stored with only one replica
```

Fig. 4. Pseudo code of storage duration prediction algorithm

Fig. 4 shows the pseudo code of the storage duration prediction algorithm. In Fig. 4, *ET* is the expected storage duration of the file. P_1 and P_2 are the disk failure rate patterns of the two disks for storing two replicas of the file, where P_1 is the pattern of the disk storing the original replica and P_2 is the pattern of the randomly picked disk for the new replica. *StartT* is the time that the

replicas are stored in the Cloud. *SDS* is the result set containing all LSDWPs. The algorithm first calculates the average failure rate of the file stored on disk 1 for the duration of *ET* (line 1). According to this value and (7), it determines the number of replicas that need to be stored, i.e., to store the file with one replica or two (line 2). If two replicas need to be stored, the algorithm calculates all LSDWPs throughout the expected storage duration of the file in one go, and returns the LSDWPs set as a result (lines 3-11).

To address the earlier statement mentioned in the introduction that the generalized data reliability model is able to provide data reliability management for both virtual and physical disks, the storage duration prediction algorithm is also applicable when the disk failure rate is a constant (e.g., Amazon S3 [2]). In that case, the storage duration prediction algorithm is significantly simplified, as the steps of calculating average failure rate (line 1) and obtaining piecewise functions (lines 5-6) can be omitted. The process of solving (8) only needs to be conducted once, and the LSDWP obtained does not change unless any replica of the file is lost.

Optimization of the algorithm

In the storage duration prediction algorithm, solving the complicated data reliability equation is a time consuming and expensive process. In particular, the involvement of piecewise function $\bar{\lambda}(t)$ and the calculation for more than one LSDWP would make the time overhead more significant. To optimize the performance of the algorithm, the data reliability equations need to be simplified to reduce the computation complexity. During the design of the algorithm, it is observed that the curve of data reliability with a single replica (i.e., $e^{-\bar{\lambda}t}$) changes almost linearly when $\bar{\lambda}t$ is within a certain range. Therefore, in this value range, the curve can be substituted by a straight line with $\bar{\lambda}t$ being the dependent variable without sacrificing much accuracy of the result as detailed in Section 7.1. Assuming that the function of the substituted straight line is $f(\bar{\lambda}t) = a\bar{\lambda}t + b$, (8) can be simplified to (9):

$$RR(1) = 1-(1-a\bar{\lambda}_1 t-b)(1-a\bar{\lambda}_2 t-b)/t \qquad (9)$$

As the average disk failure rate can be expressed as a first degree piecewise function of *t*, (9) is essentially a quartic function of *t*. Compared to the original non-polynomial equation of (8), the simplified equation of (9) can be solved by the methods for solving polynomial equations, which are much more efficient, hence the performance of the storage duration prediction algorithm can be optimized.

[2] This is based on the durability and reliability statement of Amazon S3 available at http://aws.amazon.com/s3/details/.

Metadata distribution algorithm

Due to the limit that the scan cycle of the PRCR node must be no bigger than LSDWPs of files, each file should be managed by a proper PRCR node. In order to maximize the utilization of PRCR, the metadata distribution algorithm is proposed.

According to the LSDWP of the file and the scan cycles of PRCR nodes, the metadata distribution algorithm is able to distribute the metadata of the file to the most appropriate PRCR node so that the capacity of PRCR can be maximized. The principle of the algorithm is simple: it compares the LSDWP of the file with the scan cycle of each PRCR node. Among the PRCR nodes with a scan cycle smaller than the LSDWP of the file, the metadata is distributed to the node (or a random one of the nodes) that has the biggest scan cycle.

Theorem. *Given several PRCR nodes with different scan cycles, the distribution of metadata following the metadata distribution algorithm maximizes the utilization of PRCR.*

Proof. Assume that all PRCR nodes reach the maximum capacity while all the metadata are distributed by following the metadata distribution algorithm. Therefore, for any file f managed by PRCR node *A* and any PRCR node *i* with scan cycle bigger than *A*,: $LSDWP(f) \geq ScanCycle(A) \,\&\, LSDWP(f) < ScanCycle(i)$ Without losing generality, we randomly create another metadata distribution other than the current metadata distribution by swapping the metadata of a pair of files. Assume two PRCR nodes *B* and *C*, in which $ScanCycle(B) > ScanCycle(C)$ and assume that files f_1 and f_2 be managed by PRCR node *B* and PRCR node *C* respectively, swap their managing PRCR nodes, since $LSDWP(f_2) < ScanCycle(B)$, the data reliability requirement of f_2 cannot be met. Therefore, file f_2 cannot be managed by PRCR by following the new metadata distribution. Therefore, the utilization of PRCR nodes by following this new distribution is lower than that by following the metadata distribution algorithm. According to the above reasoning, it can be deduced that there is no other metadata distribution that has higher utilization than the distribution by following the metadata distribution algorithm. Hence the theorem holds.

The metadata distribution algorithm is conducted at the user interface of PRCR. Fig. 5 shows the pseudo code of the algorithm. *SD* indicates the current LSDWP of the file. *S* indicates the set of all the PRCR nodes. The algorithm first calculates the differences between *SD* and the scan cycles of all available PRCR nodes (lines 2-3). Then, from all the PRCR nodes with a scan cycle smaller than *SD*, the ones with the smallest difference values are selected as the candidates of the destination node (lines 4-6). Finally, one of the candidates is randomly chosen as

the destination node (line 7). The reason for randomly choosing one node from the node set is to deal with the case where two PRCR nodes have the same scan cycle.

```
Algorithm:  Metadata distribution algorithm
Input:   SD;      // Current longest storage duration of the file
         S;       // The set of all the PRCR nodes
Output: node;     // the destination PRCR node
---------------------------------------------------------------
01. Set diff, nodes; // define two sets
02. for (each i ∈ S & scancycle(i) < SD)
03.     diff ← SD - scancycle(i);
 // calculate the SD - scancycle value for all available PRCR nodes
04. for (each j ∈ S & scancycle(i) < SD) {
05.     if (SD - scancycle(j) = min(diff))
06.     nodes ← j; }
// find the nodes with the smallest SD - scancycle value
07. node ← random(nodes); // randomly return one of the nodes
08. return node;
```

Fig. 5. Pseudo code of metadata distribution algorithm

The metadata distribution algorithm is able to effectively optimize the utilization of all the PRCR nodes. However, there are three issues that need to be further addressed. First, the capacity of each PRCR node is limited. When more and more files are managed by PRCR, the capacity of PRCR nodes could gradually run out. Fortunately, the independence of each PRCR node has provided great elasticity to the organization of PRCR. When one of the PRCR nodes is reaching or about to reach its maximized capacity, a new PRCR node is created, where the scan cycle of the new PRCR node can be set to the same length. Second, the data reliability model with a variable disk failure rate has led to the side effect that the LSDWP of each file changes from time to time. Once the LSDWP increases to a threshold that is equal to the scan cycle of another PRCR node, current metadata distribution becomes sub-optimal. To address this issue, several solutions could be applied. For example, the scan cycles of PRCR nodes need to be well organized so that each file is managed by the PRCR node with a scan cycle smaller than all the LSDWPs the files could have. Or, if the metadata of files needs to be redistributed regardless, the redistribution could be conducted in a batch mode to reduce its impact and computation overhead. Third, the metadata is distributed according to the calculation of the storage duration prediction algorithm. However, the predicted storage duration could be different from that of the disks in reality, and hence prediction errors could occur. Such a situation is most likely caused by the deviation of disk failure rates, and the only type of error that could possibly jeopardize data reliability is that the disk failure rates are being underestimated, so that the LSDWP value is overestimated. In general, the situation of prediction errors is very similar to the second issue. Therefore, the solutions for the second issue are still applicable to prediction errors.

7 Evaluation

In this section we evaluate PRCR from the aspects of performance and cost effectiveness.

7.1 Performance of PRCR

To evaluate the performance of PRCR, first of all, we evaluate the major procedures in PRCR. We find that the calculation of LSDWP (i.e., the storage duration prediction algorithm), metadata scanning and proactive replica checking are the three major procedures which most affect the performance of PRCR. Therefore, investigations of these three procedures are conducted respectively. In addition, we also evaluate the impact of using PRCR on data access performance compared with the conventional 3-replica strategy.

Evaluation of storage duration prediction algorithm

The storage duration prediction algorithm is mainly conducted at the beginning of the storage of a file. The performance of this algorithm is of great significance to the PRCR for determining the minimum number of replicas and calculating LSDWPs for each file.

In order to fully investigate the storage duration prediction algorithm and the effect of our optimization, the evaluation is carried out as follows: four versions of the algorithm are implemented, which are the original constant disk failure rate version (version ORC), the optimized constant disk failure rate version (version OPC), the original variable disk failure version (version ORV) and the optimized variable disk failure version (version OPV). The original versions (i.e., ORC and ORV) of the algorithm calculate LSDWPs by solving (8), while the optimized versions (i.e., OPC and OPV) calculate LSDWPs by solving (9). The evaluation of the constant disk failure rate versions of the algorithm corresponds to the discussion in Section 6.2 about the algorithm working in a constant failure rate environment.

In (9) as addressed in Section 6.2, we use the tangent line of $e^{-\bar{\lambda}t}$ at point (0, 1) as a substitution for the original curve $e^{-\bar{\lambda}t}$. The function of the tangent line is $f(\bar{\lambda}t)=1-\bar{\lambda}t$, which is a special case of $f(\bar{\lambda}t)=a\bar{\lambda}t+b$ mentioned in Section 6.2, where a=-1 and b=1. Fig. 6 shows both the original curve of $e^{-\bar{\lambda}t}$ and the substitution curve of tangent line $f(\bar{\lambda}t)=1-\bar{\lambda}t$, where the unit of x axis is calculated as "% per year * year", which has no explicit time unit. In the figure, the substitution curve is located at the lower side of the original curve of $e^{-\bar{\lambda}t}$. According to the disk failure rate range of the IDEMA standard and the disk nominal lifespan of five years[3], the range of $\bar{\lambda}t$ is (0, 0.219). In this range, it can be seen that the deviation of the tangent line is relatively small. With the decrease of $\bar{\lambda}t$, the deviation gets even smaller. After this substitution, (9) is further simplified into function

[3] B. Schroeder and G. Gibson, "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?," in *USENIX Conference on File and Storage Technologies*, pp. 1-16, 2007.

$RR(1) = 1 - \bar{\lambda}_1 \bar{\lambda}_2 t$. Compared to the original equation, the simplification of the complexity of the equation is obvious. In addition to reducing the complexity of the equation, there is another advantage of using the tangent line as a substitution. By solving (9), the result (i.e., the LSDWP of the file) is always conservatively underestimated, so that the deviation caused by the substitution does not reduce the data reliability assurance that PRCR provides. In fact, by using the tangent line substitution, the data reliability assurance PRCR provides is always higher than the calculated result.

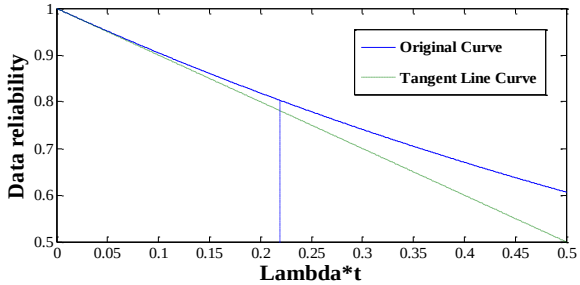

Fig. 6. Original curve and tangent line of $e^{-\bar{\lambda}t}$

The execution time and accuracy rate of the algorithm for all four versions are tested under the same file settings and the same disk settings for constant failure rate versions and variable failure rate versions respectively (ET=1 year, $\bar{\lambda}$ for ORC and OPC is 1% and disk failure rate pattern for ORV and OPV ranges from 0.5% to 4.38% which is based on the IDEMA standard). Note that the accuracy rate stands for the ratio between LSDWPs of optimized versions of the algorithm and original versions of the algorithm, which indicates the accuracy of the results produced by the optimized versions of the algorithm. The results of one representative experiment are shown in Table 1.

TABLE 1

EXECUTION TIME AND ACCURACY RATE OF STORAGE DURATION PREDICTION ALGORITHM

	One replica	Two replicas		
Average Execution Time (ms) & (Number of LSDWPs)				
Reliability	**99%**	**99.9%**	**99.99%**	**99.999%**
ORC	0.69	15.34	15.62	16.20
OPC	0.69	0.69	0.69	0.69
ORV	0.72(1)	16.26(1)	16.30(1)	155.82(10)
OPV	0.72(1)	4(1)	7.81(2)	41.52(11)
Accuracy Rate				
OPC	NA	89.52%	99.00%	99.90%
OPV	NA	89.61%	99.00%	99.90%

The upper half of Table 1 shows the average execution time of all four versions of the algorithm. In addition, the number of LSDWPs calculated in each run of the algorithm is also shown in parentheses for ORV and OPV versions of the algorithm, respectively. It can be seen that the optimized versions of the algorithm outperform the original versions in several respects. First, despite the equal execution time when data reliability is 99 per cent, at which one replica suffices to meet the data reliability, in other cases the execution time of optimized versions of the algorithm (i.e., OPC and OPV) is significantly smaller than that of original versions of the algorithm (i.e., ORC and ORV) respectively. Second, although the overall trend for all versions of the algorithm is that the execution time increases with the increase in data reliability requirement, the execution time of optimized versions of the algorithm increases much slower than that of original versions. In the accuracy rate part of Table 1, due to storage with a single replica, the accuracy rate for data reliability of 99 percent is not applicable. In other cases, the accuracy rate increases with the increase in data reliability requirement. In Table 1, the accuracy rates of optimized versions of the algorithm reach 99.9 per cent. In fact, this value can be even larger when the data reliability requirement becomes higher.

In general, the results in Table 1 show that, depending on the data reliability assurance provided, the storage duration prediction algorithm is able to calculate LSDWPs of files between a few milliseconds to hundreds of milliseconds. However, the reliability assurance shown in the table could be even higher. To provide higher data reliability assurance, more time could be taken to conduct the storage duration prediction algorithm as more LSDWP values are calculated. The execution time of optimized versions of the algorithm is much shorter than that of original versions, and the accuracy rate increases with increasing data reliability assurance.

Evaluation of metadata scanning and proactive replica checking

To evaluate the metadata scanning and proactive replica checking procedures, an experimental PRCR is implemented with Amazon Web Services (AWS). The structure of the experimental PRCR consists of one user interface, one PRCR node and one more Cloud virtual machine for proactive replica checking, each runs on a single AWS EC2 instance. Based on the experimental PRCR, the metadata scanning time and the proactive replica checking time are measured on independent EC2 instances respectively.

TABLE 2

METADATA SCANNING TIME AND PROACTIVE REPLICA CHECKING TIME

	t1.micro	**m1.small**	**m1.large**	**m1.xlarge**
Scanning Time	≈700ns	≈400ns	≈700ns	≈850ns
Checking Time	≈27ms	≈27ms	≈30ms	≈27ms

In the experiments, the metadata scanning procedure and the proactive replica checking procedure are simulated on four types of EC2 computing instances (i.e., virtual machines in EC2) for the management of 3000 S3 objects (i.e., files). Table 2 shows the results of the

experiments. It can be seen that the metadata scanning time is at a magnitude of hundreds of nanoseconds, and the proactive replica checking time is at a magnitude of tens of milliseconds.

Impact of PRCR on data access performance

Compared with the conventional 3-replica strategy, by using PRCR, intuitively, data access performance could potentially be affected. Specifically, such impact could primarily be on the data transfer speed. However, such impact may vary due to different Cloud data access strategies and different data storage plans. In this subsection, we only briefly discuss this issue as it is essentially not how PRCR impacts on data access performance but how storing no-more-than-two replicas could impact on data access performance in comparison to storing three replicas.

For evaluation, we conducted data access speed tests with Amazon S3 services so as to analyze the impact of storing no-more-than-two replicas compared with storing conventional three replicas. As Amazon has stated that "latency and throughput for reduced redundancy storage are the same as for standard storage" (*http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingRRS.html*), we conducted data access performance tests for Amazon S3 standard storage only. Specifically, we created Amazon AWS EC2 instances in different AWS regions as well as used a local computer at Swinburne University of Technology in Melbourne, Australia to experiment the data transfer speed for accessing files stored in S3 in different regions. In each test, a 10MB file is transferred from the data source to the target location.

The results are shown in Table 3. Based on the results, a major observation can be clearly seen, i.e., data transfer within the same region is always of the highest speed, where the data are transferred much quicker than that between different places (>3000 KB/s vs. <300 KB/s).

TABLE 3

TRANSFER SPEED FOR ACCESSING DATA IN AMAZON S3

Source / Target	Oregon	Ireland	Singapore	Sydney	Local
Oregon	3372KB/s	170KB/s	184KB/s	172KB/s	86KB/s
Ireland	231KB/s	3284KB/s	211KB/s	36KB/s	54KB/s
Singapore	190KB/s	209KB/s	3466KB/s	202KB/s	107KB/s
Sydney	137KB/s	110KB/s	230KB/s	3205KB/s	224KB/s

According to the above results, in summary, we have the following conclusions for the impact of using PRCR on data access performance compared with the conventional 3-replica strategy.

First, in the case that all replicas are stored in one region in practice as in Amazon S3, as addressed above, according to the description of Amazon S3, latency and throughput of standard storage and reduced redundancy storage are the same. This in fact means that on Amazon S3 the data access performance by using 3 replicas is the same to that by using less than 3 replicas. Therefore, for PRCR which uses no more than 2 replicas without jeopardizing reliability, it would have the same data access performance with S3, and hence no performance degradation in general.

Second, in the case that all replicas are stored in different regions, there would be some impact on data transfer speed for some users. For example, on Amazon S3, if 3 replicas are stored in different regions such as Oregon, Ireland and Sydney in a traditional manner whilst in PRCR only 2 replicas are stored in regions such as Oregon and Ireland (i.e.,, no replica stored in Sydney), there would be performance degradation to some users from such as Australia to access data because they would suffer slower data transfer speed to either Oregon or Ireland (at 86KB/s or 54KB/s respectively from Swinburne) in comparison to faster access to Sydney (at 224Kb/s from Swinburne). Similarly, the impact on a single replica can be analyzed. To solve this issue, on one hand, research in another area on data placement can be conducted to minimize the performance impact, e.g., the replica accessed least and/or slowest can be eliminated. On the other hand, in the case that access performance for certain data is of ultimate goal, extra replica(s) can be added at the extra cost, which would not jeopardize the effectiveness of PRCR for data reliability.

7.2 Cost effectiveness of PRCR

The cost effectiveness of PRCR in managing a large number of files is evaluated. There are two major costs incurred for managing data with PRCR: the running overhead of PRCR and the cost for storing data replicas.

Running overhead of PRCR

Considering the large amounts of data in the Cloud, PRCR nodes would normally be well loaded, i.e., being or being close to their maximum capacity. Therefore, the running overhead of each file can be derived by dividing the total PRCR running cost by the maximum capacity of PRCR nodes.

TABLE 4

MAXIMUM CAPACITY OF PRCR NODES (NUMBER OF FILES)

RA \ $\overline{\lambda}$	**0.1**	**0.05**	**0.02**	**0.01**
99%	$5*10^{13}$ files	$2.3*10^{14}$ files	$2.8*10^{15}$ files	**NA**
99.9%	$4.5*10^{12}$ files	$1.8*10^{13}$ files	$1.2*10^{14}$ files	$5*10^{14}$ files
99.99%	$4.5*10^{11}$ files	$1.8*10^{12}$ files	$1.1*10^{13}$ files	$4.5*10^{13}$ files
99.999%	$4.5*10^{10}$ files	$1.8*10^{11}$ files	$1.1*10^{12}$ files	$4.5*10^{12}$ files

Based on Table 2, for the ease of illustration, we choose 700ns as the standard execution time for the metadata scanning process. Given the reliability assurance and disk failure rate, we are able to calculate the LSDWP of the file, and hence the biggest scan cycle of the corresponding PRCR node. Further, with the

standard execution times, the maximum capacity, i.e., number of files to be managed, of the PRCR nodes is calculated (Maximum capacity = Scan cycle / Metadata scanning time) and presented in Table 4. In the table, the relationships among the reliability requirement, the average failure rate of a single replica λ and the maximum capacity of PRCR nodes are clearly revealed. With different single replica failure rates and reliability requirements, each PRCR node is able to manage from $4.5*10^{10}$ to $2.8*10^{15}$ files, which is quite large. Although the maximum capacity of PRCR nodes reduces with the increment of disk failure rates and data reliability requirements, the maximum capacity of PRCR nodes is deemed big enough to be practical for the management of a large number of files in the Cloud.

The total PRCR running cost is composed of the running cost for user interface, PRCR nodes and Cloud virtual machines for proactive replica checking. According to Amazon EC2 prices, the corresponding cost of an EC2 micro instance is only $14.40/month each ($0.02/hour * 24 hours/day * 30 days/month). Therefore, for a complete PRCR running over AWS, the running cost could be as little as several tens of dollars per month. According to the maximum capacity of PRCR, the running overhead for each file is very small, which is no more than $\$10^{-9}$/file*month (i.e., $\$14.40*3/(4.5*10^{10}$files)). Such an overhead is so small in comparison to the cost for data storage, which is at the level of $\$10^{-2}$/GB*month. For example, the storage of a file with the size of 1GB has a running overhead about 10^7 times cheaper than the storage cost (several cents/month). Therefore, the cost of running PRCR is negligible.

Data storage saving by using PRCR

The data storage saving using PRCR is investigated. We simulate the data reliability management process of PRCR to manage the files of the pulsar searching example as illustrated in Supplementary Material. In the simulation, the storage consumption is compared with the conventional 3-replica strategy, which is widely used in current Clouds.

In the simulation, four different storage plans are tested: 1-replica, 1+2 replica, 2-replica and 3-replica. The 1-replica plan stores all files with one replica, which stands for the data storage without any replication. The 2-replica plan stores all files with two replicas. These two storage plans can be redeemed as two data storage strategies, but also represent the lower and upper bounds of storage consumption by using PRCR. The 3-replica plan stores all files with three replicas, which represents the conventional 3-replica strategy. The 1+2 replica plan divides all the files into two categories and stores them with one replica or two replicas, respectively, which represents the actual data management of PRCR. According to the pulsar searching example, the extracted and compressed beam files, the XML files and the de-dispersion files should be stored for long-term use and have higher reliability requirements, so they are stored with two replicas. The other files that are for temporary usage are stored in the 1-replica mode.

Fig. 7 shows the average replica numbers and total data sizes with different storage plans for the pulsar searching data. By applying the 2-replica plan, one-third of the generated data size can be reduced in comparison to the 3-replica strategy, and the average replica number for each file is reduced accordingly. By applying the 1+2 replica plan, the consumption of storage space is further reduced and minimized. In our simulation, by applying the 1+2 replica plan, the ratio between the number of the two types of files for pulsar searching application reaches a staggering 1:41, and the ratio between the accumulated sizes of two types of files is about 2.34:1. Compared with the 2-replica plan, more than 95 per cent of replicas with 23 per cent of the total data size are reduced. Compared with the 1-replica plan, the 1+2 replica plan generates only 53 per cent additional storage space for all the pulsar searching data (i.e., the data redundancy is 1.53), and the data reliability requirement of all the files can be met without jeopardy.

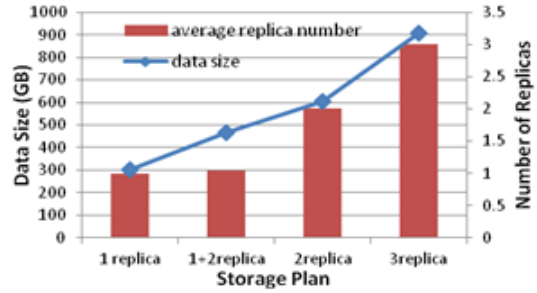

Fig. 7. Average replica numbers and data sizes

In general, by using PRCR, the storage consumption could be reduced significantly. For the pulsar searching application, up to 49% (i.e., (3-1.53)/3) of the storage space can be saved compared with the conventional 3-replica data storage strategy, i.e., 277TB of storage space can be saved for an observation of eight hours a day for a month. Meanwhile, the running cost of PRCR for managing such an amount of data is only tens of dollars per month. For other Cloud applications with different data composition, say more data with less data reliability requirement, the storage space reduction effect of PRCR could be even higher. Moreover, here we only compared PRCR with the conventional 3-replica strategy. To manage data with even more replicas needed [6], according to the nature of PRCR that stores no more than two replicas, the storage saving by using PRCR could be even more.

8 Conclusions and Future Work

In this paper, we have presented a cost-effective reliability management mechanism (PRCR) based on a generalized data reliability model. It applies an innovative proactive replica checking approach to ensure the data reliability while the data can be maintained with the minimum number of replicas (serving as a cost

effectiveness benchmark for evaluation), which is no more than two. Evaluation of PRCR has demonstrated that this mechanism is able to manage large amounts of data in the Cloud, significantly reduce the Cloud storage space consumption at a negligible overhead.

In the near future, this research can be extended in two directions. First, a more detailed design of PRCR will be conducted including further optimization. Second, as PRCR inevitably reduces the replication level of Cloud data, the location of replicas becomes more important which deserves further research on improving data access performance.

ACKNOWLEDGEMENT

This work is partly supported by Australian Research Council grants under Discovery Project DP110101340 and Linkage Project LP130100324. The work was done when W. Li and D. Yuan were with Swinburne University of Technology and Y. Yang is the corresponding author.

REFERENCES

[1] Amazon. (2011). *Amazon simple storage service (Amazon S3)*. Available: http://aws.amazon.com/s3/

[2] R. Bachwani, L. Gryz, R. Bianchini, and C. Dubnicki, "Dynamically quantifying and improving the reliability of distributed storage systems," in *IEEE Symposium on Reliable Distributed Systems*, pp. 85-94, 2008.

[3] B. Balasubramanian and V. Garg, "Fault tolerance in distributed systems using fused data structures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, pp. 701-715, 2013.

[4] E. Bauer and R. Adams, Reliability and availability of cloud computing: IEEE Press, 2012.

[5] D. Borthakur. (2007). *The Hadoop distributed file system: Architecture and design*. Available: http://hadoop.apache.org/common/docs/r0.18.3/hdfs_design.html

[6] B. G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *Symposium on Networked Systems Design & Implementation*, pp. 45-58, 2006.

[7] J. G. Elerath and S. Shah, "Server class disk drives: how reliable are they?," in *Annual Symposium on Reliability and Maintainability*, pp. 151-156, 2004.

[8] J. Gantz and D. Reinsel, "Extracting value from chaos," International Data Corporation (IDC), 2011.

[9] A. Gharaibeh, S. Al-Kiswany, and M. Ripeanu, "ThriftStore: finessing reliability trade-offs in replicated storage systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 22, pp. 910-923, 2011.

[10] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *ACM Symposium on Operating Systems Principles,* pp. 29-43, 2003.

[11] G. Gibson, "Redundant disk arrays: reliable, parallel secondary storage," University of California, Berkeley. Technical Report UCB/CSD 91/613, 1991.

[12] G. Gibson and D. Patterson, "Designing disk arrays for high data reliability," *Journal of Parallel and Distributed Computing,* vol. 17, pp. 4-27, 1993.

[13] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in Windows Azure storage," in *USENIX Annual Technical Conference*, pp. 2-13, 2012.

[14] IDEMA, "R2-98: specification of hard disk drive reliability," *IDEMA Standards*, 1998.

[15] M. Lei, S. V. Vrbsky, and Z. Qi, "Online grid replication optimizers to improve system reliability," in *IEEE International Parallel and Distributed Processing Symposium*, pp. 1-8, 2007.

[16] W. Li, Y. Yang, J. Chen, and D. Yuan, "A cost-effective mechanism for Cloud data reliability management based on proactive replica checking," in *International Symposium on Cluster, Cloud and Grid Computing*, pp. 564-571, 2012.

[17] W. Li, Y. Yang, and D. Yuan, "A novel cost-effective dynamic data replication strategy for reliability in cloud data centres," in *International Conference on Cloud and Green Computing*, pp. 496-502, 2011.

[18] W. Li, Y. Yang, and D. Yuan, "An energy-efficient data transfer strategy with link rate control for Cloud," (http://www.ict.swin.edu.au/personal/yyang/papers/IJAACS-Li.pdf) *International Journal of Autonomous and Adaptive Communications Systems,* Accepted on Oct. 11, 2013.

[19] E. Pinheiro, W. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *USENIX Conference on File and Storage Technologies*, pp. 17-29, 2007.

[20] S. Ramabhadran and J. Pasquale, "Analysis of long-running replicated systems," in *IEEE Conference on Computer Communications*, pp. 1-9, 2006.

[21] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *SIGCOMM*, pp. 331-342, 2014.

[22] Q. Xin, T. J. E. Schwarz, and E. L. Miller, "Disk infant mortality in large storage systems," in *IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems,* pp. 125-134, 2005.

[23] D. Yuan, Y. Yang, X. Liu, W. Li, L. Cui, M. Xu, and J. Chen, "A highly practical approach towards achieving minimum datasets storage cost in the Cloud," *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, pp. 1234-1244, 2013.

BIOGRAPHIES

Wenhao Li received the BEng and MEng degrees from Shandong University, China in 2007 and 2010, respectively, and the PhD degree from Swinburne University of Technology, Australia, in 2014. He is currently a research fellow in the School of Computer Science and Technology, Shandong University, China. His research interests include parallel and distributed computing, cloud computing, data management in distributed computing environment and big data

Yun Yang received the BSci degree from Anhui University, China, in 1984, the MEng degree from the University of Science and Technology of China, China, in 1987, and the PhD degree from the University of Queensland, Australia, in 1992. He is currently a full professor in the School of Software and Electrical Engineering at Swinburne University of Technology, Australia. His current research interests include software engineering, cloud computing, workflow systems, and service-oriented computing.

Dong Yuan received the BEng and MEng Degrees from Shandong University, Jinan, China, in 2005 and 2008, the PhD degree from Swinburne University of Technology, Australia, in 2012, all in computer science. He is currently a Lecturer in School of Electrical and Information Engineering, University of Sydney. His research interests include cloud computing, data management in parallel and distributed systems, scheduling and resource management, business process management and workflow systems.