

# A cost-effective mechanism for Cloud data reliability management based on proactive replica checking

Wenhao Li\*, Yun Yang\*, Jinjun Chen\* \*\*, Dong Yuan\*

\*Faculty of Information and Communication Technologies, Swinburne University of Technology, Australia  
E-mail: {wli, yyang, dyuan}@swin.edu.au

\*\* School of Systems, Management and Leadership, University of Technology Sydney, Australia  
E-mail: Jinjun.Chen@uts.edu.au

**Abstract**—In current Cloud computing environments, management of data reliability has become a challenge. For data-intensive scientific applications, storing data in the Cloud with the typical 3-replica replication strategy for managing the data reliability would incur huge storage cost. To address this issue, in this paper we present a novel cost-effective data reliability management mechanism named PRCR, which proactively checks the availability of replicas for maintaining the reliability. Our simulation indicates that, comparing with the typical 3-replica replication strategy, PRCR can significantly reduce the storage space consumption, hence storage cost in the Cloud.

**Keywords**- data replication, proactive replica checking, data reliability, cost-effective storage, Cloud computing

## I. INTRODUCTION

The size of Cloud storage is expanding in a dramatic speed. It is estimated that by 2015 the data stored in the Cloud will reach 0.8 ZB, while more data are stored or processed temperately in their journey. Meantime, with development of the Cloud computing paradigm, Cloud-based applications have put forward higher demand for Cloud storage. While the requirement of data reliability should be met in the first place, data in the Cloud need to be stored with higher cost-effectiveness.

Data reliability is defined as the probability that the data is available in the system for a certain period. It is an important issue in data storage systems, which indicates the ability of keeping data persistence and fully accessible. Due to the accelerating growth of Cloud data, the management of data reliability in the Cloud has become a challenge. Currently, data replication is one of the most popular approaches for supporting data reliability. However, the consumption of storage space for replicas would incur huge cost, and it is believed that the cost would be passed on to the users eventually.

In this paper, we present a novel cost-effective data reliability management mechanism based on proactive replica checking named PRCR for reducing the storage space consumption, hence storage cost for data-intensive applications in the Cloud. The objective of PRCR is to reduce the number of replicas stored in the Cloud while meeting the data reliability requirement. Compared with existing data reliability solutions in the Cloud, PRCR has the following features:

- PRCR is able to provide data reliability assurance at the file level.
- PRCR is able to provide data reliability management in a more cost-effective way. By applying PRCR, a wide range of data reliability can be assured with at most two replicas stored in the Cloud.

By applying benchmarks from Amazon Web Services, we evaluate the capacity of PRCR nodes, and simulate the reliability management process using PRCR with the data generated by a data-intensive scientific application. The results are compared with the typical 3-replica replication strategy. It is shown that PRCR can reduce one-third to two-thirds of the storage consumption that currently consumed in the Cloud, hence the storage cost is reduced accordingly.

The rest of the paper is organised as follows. The related works on data reliability, data replication and cost-effective data storage are addressed in Section II. The motivating example and the problem analysis are stated in Section III. The concepts and model of the PRCR mechanism and a high level design for PRCR are presented in Section IV. The detailed design for PRCR is described in Section V. The evaluation of PRCR is illustrated in Section VI. Finally, conclusions and future work are summarised in Section VII.

## II. RELATED WORKS

Data reliability is one of the key research issues in distributed computing. In order to ensure reliable processing and storage of data, many efforts have been made in different aspects [2, 6, 14], and some previous works have provided valuable references. In [5], a detailed survey on grid reliability identifies important issues and reviews the progress of data reliability research in grid environment. In [11, 15], analytical data reliability models are proposed and comprehensively studied. In [11], data reliability of the system is measured by data missing rate and file missing rate, and the issue of maximising data reliability with limited storage capacity is investigated. In [15], it proposes an analytical replication model for determining the optimal number of replica servers, catalogue servers, catalogue sizes to guarantee a given overall data reliability.

Among the existing research for data reliability in distributed systems, data replication has always been considered as an important approach. Some existing works on large-scale distributed storage systems have been proposed, such as [10, 16] investigate data replication in a

Grid environment and [8] investigates data replication in a P2P system, etc. In Cloud computing, some related works have also been conducted. For example, in [17] the Cloud storage and data replication management issues are investigated and optimised mainly from the database perspective. Additionally, data replication technologies are also widely used in current commercial cloud systems for performance and reliability purposes. Some typical examples include Amazon Simple Storage Service (Amazon S3) [3], Google File System [7], Hadoop distributed file system [4], etc.

Although data replication has been widely used, there is a side effect that data replication would consume storage resources and incur additional cost significantly. Currently, the relationship among data reliability, data replication and data storage cost has not been well studied in the Cloud area. Some of our works have made contributions in the cost-effective data reliability management area based on data replication. For example, in [12], we propose a cost-effective dynamic data replication strategy for data reliability in Cloud data centres, 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, this strategy still generates 3 replicas for the data, so that its ability of reducing storage cost is limited.

In this paper, our research is closely related to the aspects mentioned above, Compare with the data replication strategy in [12], PRCR stores no more than 2 replicas for each file in the Cloud. In addition, PRCR also incorporates the distinctive characteristics of Cloud storage environment. First, Cloud storage systems are formed by several large-scale data centres rather than many heterogeneous nodes. Second, data properties such as data and metadata locations can not be derived by users because of virtualisation. These two characteristics have led to different approaches of achieving data reliability in the Cloud from other distributed systems.

### III. MOTIVATING EXAMPLE AND PROBLEM ANALYSIS

#### A. Motivating example

The Cloud has offered an excellent storage capacity, which from user’s perspective, is unlimited for storing all the data generated during the execution of applications. This feature of the Cloud is well desired especially by scientific applications with data-intensive characteristic. For example, a series of pulsar searching applications has been conducted by the Astrophysics Group at Swinburne University of Technology. Depending on terabytes of observation data obtained from Parkers Radio Telescope [1], many complex and time consuming tasks have been carried out and large volumes of data are generated during their execution. Figure 1 shows the dataflow of a pulsar searching workflow example [13]. There are several steps in the pulsar searching workflow. First, from the raw signal data received from the Parkes Telescope, 13 beam files are extracted and compressed. The beam files contain the pulsar signals which are dispersed by the interstellar medium. For counteracting this effect, in the second step, the de-dispersion process

conducts different dispersion trials and a minimum of 1200 de-dispersion files are generated for storing the results. Next, for binary pulsar searching purposes, every de-dispersion file needs another accelerate step, which generates 5-10 accelerated di-dispersion files for each di-dispersion file. Based on the accelerated di-dispersion files, the seeking algorithm can be applied to search for pulsar candidates. In each beam, a candidate list of pulsars is generated. Furthermore, by comparing the candidates generated from different beam files, the top 100 pulsar candidates are selected. These final pulsar candidates are folded in XML files with their feature signals, and finally submitted to scientists as the final result. The pulsar searching applications currently run on the Swinburne high-performance supercomputing facility. Due to the storage limitation, all the generated data are deleted after having been used once and only the beam data which are extracted from the raw telescope data are stored. However, by applying the Cloud for storage of the pulsar searching data, the storage limitation for these applications can be completely eliminated, and much more generated data can be stored for reuse.

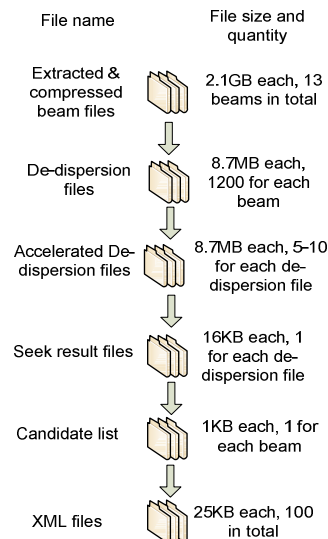


Figure 1. Dataflow of a pulsar searching workflow instance

When we were trying to store all pulsar searching data into the Cloud, a problem has emerged, that is the cost of hiring Cloud storage resources for these data could be huge. For example, in a typical pulsar searching workflow instance as shown in Figure 1, more than 100000 files with the size over 230GB are generated. According to the latest Amazon S3 storage prices, storing these data using S3 standard storage service costs about US\$12.65 per month (\$0.055 per GB/month). In order to meet the needs of pulsar searching applications, we probably need to store data generated by several hundreds of such workflow instances, hence thousands of dollars will be spent on data storage every month.

## B. Problem analysis

The cost for storing data is inevitable, but there is still room for reducing it. When we look into the pulsar searching example, there are two major factors that could lead to the high storage cost.

First, current Cloud storage systems generally use data replication for data reliability. For example, storage systems such as Amazon S3 [3], Google File System [7] and Hadoop Distributed File System [4] all adopt similar multi-replicas data replication strategies, which typically store three replicas by default for all data. By using the typical 3-replica replication strategy, three replicas are generated at once at the beginning of the storage and stored at three different places. Such replication strategy causes a huge amount of storage space consumed, and users have to pay for the cost eventually. By applying the typical 3-replica replication strategy, storing one TB of data needs three TBs of data space, which would incur a 200% additional cost. For data-intensive applications such as the pulsar searching example, the extra money spent could be huge.

Second, according to the importance and storage duration, the data generated in a pulsar searching workflow instance can be divided into two major types. One type of data is critical and can be reused for a long time. For example, the extracted beam files and the XML files are the input and output of a pulsar searching workflow; the de-dispersion files are frequently used generated data [19], based on which different seeking algorithms can be applied and some other generated data can be derived. These data record the current state of the universe, which is very important and can be used for long term analysis. For this kind of data, high data reliability assurance and recovery ability are necessary. Another type of data is only used for a short term and lack of long-term value. For example, the accelerated de-dispersion files, seek result files and the candidate lists all belong to this type. For this type of data, relatively low reliability assurance can be applied and recovery ability is most likely unnecessary. However, by applying the typical 3-replica replication strategy, all these data are replicated for the same number of times, which would incur some unnecessary extra storage cost.

In order to reduce the storage cost for data-intensive scientific applications in the Cloud, both above mentioned major factors must be solved, and a new replication mechanism should be proposed to replace the typical 3-replica replication strategy. In the rest of the paper, we provide a feasible solution.

## IV. PROACTIVE RELIABILITY MANAGEMENT MECHANISM

Instead of the typical 3-replica replication strategy, there is another way in which we can provide data reliability with fewer replicas. In this section we propose a novel cost-effective reliability management mechanism for Cloud data storage named PRCR (Proactive Replica Checking for Reliability) at the high level with detailed design presented in Section V.

## A. Concepts and reliability management model of PRCR

The idea of PRCR follows the principles described below.

First, the reliability of data in the Cloud follows the exponential distribution. As the basic component of the Cloud, data centres contain a large amount of commodity storage facilities which are unified as Cloud storage units. The data in the Cloud are stored in these storage units, and each copy of the data is called a replica of the data. According to classical theories [9, 18], we assume that the reliability of a Cloud storage unit over period  $T$  follows the exponential distribution:  $F(T) = e^{-\lambda T}$ , where  $\lambda$  is the failure rate of a Cloud storage unit, which stands for the expected number of Cloud storage unit failures in a certain period. The replicas in Cloud storage units should have the same reliability as the storage units, because data loss is primarily caused by the failure of Cloud storage units. For example, if a data centre experiences 100 disk failures out of 10000 disks a year, the average failure rate is 0.01/year, and thus the reliability of each replica should be 99% over one year. In addition, the reliability assurance of data changes according to the storage duration of the data. If the data are stored for more than one year, the reliability assurance of the data will decrease while the probability of data loss will increase.

Second, the storage duration for meeting the reliability requirement can be deduced. Assume that multiple replicas be stored in different storage units. According to the reliability equation of a Cloud storage unit (which is also the reliability equation of a single replica), the reliability of data with multiple replicas can be derived from equation (1):

$$X = 1 - (1 - e^{-\lambda T_k})^k \quad (1)$$

In this equation, variable  $X$  is the data reliability requirement, variable  $k$  is the number of replicas and variable  $T_k$  is the storage duration of the data with  $k$  replicas. The right-hand side of this equation describes the probability that no failure happens during the storage duration of  $T_k$ , when  $k$  replicas are stored in Cloud storage units with failure rate  $\lambda$ . From this equation, it can be found that if we get the number of replicas and the failure rate of storage units, the storage duration of data while meeting the reliability requirement can be derived.

Third, the reliability of the data has the memory-less property. As the exponential distribution has the memory-less property, the reliability of data in the storage units should also have the memory-less property, in which  $P(T > s + t | T > s) = P(T > t)$  for all  $s, t > 0$ . This is to say that the reliability of data from time  $s$  to  $s+t$  is equivalent to the reliability of data from time  $0$  to  $t$ . According to this property of data reliability, as long as we can guarantee that the data is available at this very moment, the reliability of data for any period from that moment can be calculated.

Based on the principles described above, the PRCR mechanism is proposed. In PRCR, the data in the Cloud are managed in different types according to their expected storage duration and reliability requirement: for data that is only for short-term storage and require low data reliability, one replica is stored in the Cloud; for data that is for long-term use and have a data reliability requirement higher than

the reliability assurance of single replica, two replicas are stored in the Cloud where the management of two replicas is based on a proactive checking manner. In this proactive checking manner, the availability of data is checked before the reliability assurance dropped to less than the reliability requirement. If one of the replicas is found to be unavailable, it can be recovered from another replica. By applying PRCR, the reliability of all the data can be maintained to meet the reliability requirement. Compared with the typical 3-replica replication strategy, PRCR stores no more than two replicas for each file, so that the storage cost can be reduced significantly.

### B. Overview of PRCR

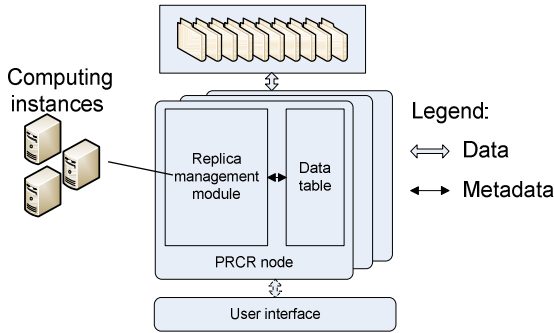


Figure 2. PRCR architecture

PRCR runs on top of the virtual layer of the Cloud and manages data which are stored as files in different data centres. Figure 2 shows the architecture of PRCR. There are two major parts, which are the PRCR node and the user interface.

**PRCR node:** It is the core component of PRCR, which is responsible for the management of the replicas. As will be mentioned later, according to the number of files in the Cloud, PRCR may contain one or more PRCR nodes which are independent of each other. The PRCR node is composed of two major components: data table and replica management module.

**Data Table<sup>1</sup>:** It maintains the metadata of all data that each PRCR node manages. For each file in the storage, there are four types of metadata attributes maintained in the data table: file ID, time stamp, scan interval, and replica address. File ID is a unique identification of the file. Time stamp records the time when the last replica checking is conducted on the file. The scan interval is the time interval between two replica checkings of each file. Depending on the time stamp and the scan interval, PRCR is able to determine the files that need to be checked, and according to the replica addresses, all replicas of the data are able to be found. In the data table, each round of the scan is called a scan cycle. In each scan cycle, all of the metadata in the data table are sequentially scanned once.

<sup>1</sup> The reliability of data table itself is beyond the scope of this paper. A conventional primary-secondary backup mechanism may well serve the purpose.

**Replica Management Module:** It is the control component of the PRCR node, which is responsible for managing the metadata in the data table and cooperating with the Cloud computing instance<sup>2</sup> to process the 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. For files that need to be checked, the replica management module extracts their metadata from the data table and sends these metadata to the Cloud computing instances. After the Cloud computing instances have finished the checking tasks, the replica management module receives the returned results and conduct further action accordingly. In particular, if any replica is not available, the replica management module sends a request for replication to the Cloud which creates a new replica for the data, and updates the data table accordingly.

**User interface:** It is a very important component of PRCR, which is responsible for justifying the reliability management requests and distributing the accepted requests to different PRCR nodes. The reliability management request is a request for PRCR to manage the file. In the request, the metadata of the file is required. Despite the four types of metadata attributes in the data table, a metadata attribute which is called the expected storage duration is needed. It is an optional metadata attribute of the file, which indicates the storage duration that the user expects. According to the expected storage duration, the user interface is able to justify whether the file need to be managed by PRCR or not. In particular, if such management is not necessary, the reliability management request is declined and the file is stored with only one replica in the Cloud.

## V. DETAILED DESIGN OF PRCR

In section IV we presented the high level design of PRCR. In this section, the detailed design is presented. First, we demonstrate the capacity of PRCR for managing files in the Cloud. Then, in order to maximise the capacity of PRCR, a metadata distribution algorithm is demonstrated. At last, by following the life cycle of a file, the working process of PRCR is presented.

### A. Capacity of PRCR

The capacity of PRCR stands for the maximum number of files that PRCR is able to manage. In order to manage the reliability of Cloud data, PRCR must have an excellent capacity in order to manage the huge number of files in the Cloud. Due to the limitation of computing power of a single PRCR node and different reliability requirements, PRCR usually contains several PRCR nodes, and the capacity of PRCR is the sum of the capacities of all PRCR nodes.

The capacity of PRCR is mainly determined by two parameters, which are the metadata scanning time and the scan cycle of the PRCR node. The metadata scanning time is the time for scanning each metadata in the data table, and the scan cycle is the time that all metadata in the data table of a

<sup>2</sup> In this paper we use the term Cloud computing instance to represent the virtual computing unit in the Cloud.

PRCR node are sequentially scanned once. The capacity of PRCR can be presented with equation (2):

$$C_{app} = \sum_{i=1}^N T_{cycle}^i / T_{scan}^i \quad (2)$$

In this equation,  $C_{app}$  is the capacity of PRCR,  $T_{cycle}^i$  is the scan cycle of PRCR node  $i$ ,  $T_{scan}^i$  is the metadata scanning time of PRCR node  $i$  and  $N$  is the number of PRCR nodes in PRCR. From this equation, it can be seen that with faster metadata scanning time and scan cycle, more files can be managed by PRCR.

### B. Metadata distribution algorithm

Due to different reliability requirements for scan cycles of PRCR nodes, the capacity of the PRCR may fluctuate significantly. By maximising the capacity of PRCR, the cost for running PRCR can be minimised, and thus the reliability management can be conducted cost-effectively. To address this issue, we design a metadata distribution algorithm for the PRCR.

The metadata distribution algorithm is conducted within the user interface component of the PRCR. It calculates the scan interval attribute and if needed, distributes the reliability management request to one of the PRCR nodes. In order to maximise the capacity of PRCR, this algorithm follows such an idea, that each file should be managed by a most appropriate PRCR node. When a reliability management request is received, the metadata distribution algorithm first calculates the longest duration that the file can be stored. If replication is needed, it then sends the request to the PRCR node which has the scan cycle smaller but closest to the longest duration. The longest duration that the file can be stored without proactive replica checking (“the longest storage duration” for short) indicates the time that the reliability assurance drops to the level that the reliability requirement can no longer be met. The difference between the scan cycle of a PRCR node and the longest duration of a file indicates such a meaning, that how long the proactive replica checking is conducted before the longest storage duration is reached. The reasons for sending the request to the PRCR node with the scan cycle smaller but closest to the longest duration are: first, PRCR must conduct proactive replica checking before the longest storage duration of each file; second, conducting proactive replica checking only when the longest storage duration of the file is about to reach can maximise the capacity of the PRCR service. In addition to what is stated above, the design of the metadata distribution algorithm has also considered the different types of data in the storage. As mentioned in Section III.B, there are two types of data generated by the scientific applications in the Cloud: short term and long term. The metadata distribution algorithm justifies the reliability management request by comparing the file’s longest storage duration and the expected storage duration. If the expected storage duration is shorter than the longest storage duration of one replica, no replication is needed, hence the reliability management request is declined.

The longest storage duration of the file can be derived from equation (1) in Section III. Specifically, for no more than two replicas stored in the Cloud, the equation can be transformed into equation (3) and equation (4) respectively: When  $k=1$ ,

$$f(T_1) = e^{-\lambda T_1} / X \quad (3)$$

When  $k=2$ ,

$$f(T_2) = Xa^2 - 2a + 1 \quad (4)$$

where  $a = e^{\lambda T_2}$ . In equations (3) and (4),  $T_1$  and  $T_2$  indicate the longest durations of data with one replica and two replicas respectively.

Algorithm: Metadata distribution algorithm

Input: Expected storage duration  $e$ ; // -1 if not exist

Failure rate  $\lambda$ ;

Reliability requirement  $X$ ;

PRCR nodes set  $S$ ; // include all the  $N$  PRCR nodes

Output:  $scanInterval$ ; // the longest storage duration without detection

PRCR node  $node$  // the destination PRCR node

---

```

01. if ( $e \neq -1$ ) {
02.   solve equation  $f(T_1) = e^{-\lambda T_1} / X$ 
03.    $T_1 \leftarrow$  the positive real root of  $f(T_1)$ 
04.   if ( $e < T_1$ ) return -1 // reject the request if short term storage
05. else {
06.   solve equation  $f(a) = Xa^2 - 2a + 1$ 
07.    $a \leftarrow$  the positive real root of  $f(a)$ 
08.    $T_2 \leftarrow \log a / \lambda$ 
09.    $scanInterval \leftarrow T_2$  // longest storage duration with 2 replicas
10. for (each  $i \in S$  &  $scancycle(i) < scanInterval$ )
11.   Set  $diff \leftarrow scanInterval - scancycle(i)$ 
// calculate the  $scanInterval - scancycle$  values
12. for (each  $j \in S$  &  $scancycle(j) < scanInterval$ ) {
13.   if ( $scanInterval - scancycle(j) = \min(diff)$ )
14.     Set  $nodes \leftarrow j$ 
// find the nodes with the smallest  $scanInterval - scancycle$  value
15.  $node \leftarrow \text{random}(nodes)$  // randomly return one of the nodes as the
destination
16. return  $scanInterval, node$ 

```

Figure 3. Pseudo code of the metadata distribution process

The pseudo code of the algorithm is shown in Figure 3. When the reliability management request encapsulating the metadata arrives, the algorithm computes the longest storage duration of one replica  $T_1$  first. By solving equation (3),  $T_1$  is derived. If the expected storage duration attribute exists and is shorter than  $T_1$ , the reliability management request is rejected and the file is stored with only one replica in the Cloud. Otherwise, the reliability management request is accepted and the file is stored in the Cloud with two replicas. The longest storage duration of two replicas  $T_2$  is derived from equation (4). After  $T_2$  is derived, it is stored as the scan interval attribute. Afterward, the process calculates the scan interval minus the scan cycle value for the PRCR node of which the scan cycle is smaller than the scan interval of the file. Then, a randomly selected PRCR node that has the smallest scan interval minus the scan cycle value is decided as the destination node. Finally, the scan interval attribute together with the destination PRCR node are returned as the result. In this process, we randomly pick one node from the nodes set to deal with the situation that

two PRCR nodes have the same scan cycle. By applying the metadata distribution algorithm, PRCR can reach its maximum capacity, and the cost for managing files can be minimised.

### C. Working process of PRCR

In Figure 4, we illustrate the working process of PRCR in the Cloud by following the life cycle of a file.

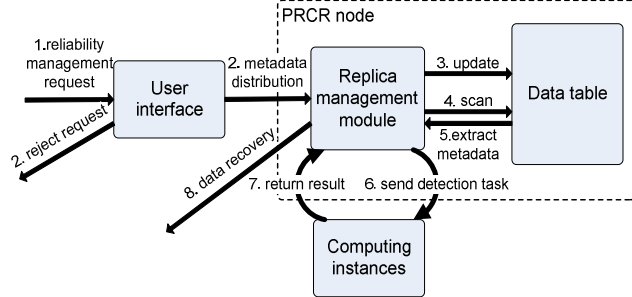


Figure 4. Working process of PRCR

1. The process starts from the submission of a reliability management request to PRCR. In the request, the metadata of the file is encapsulated. According to the metadata, user interface justifies whether the management of the file in PRCR is necessary.
2. According to the metadata distribution algorithm, if the proactive replica checking is necessary, the request is distributed to the replica management module of the corresponding PRCR node. Otherwise, the reliability management request is declined (rejected) when the file only needs one replica stored in the Cloud to satisfy the reliability requirement.
3. The replica management module calls cloud storage service to do the data replication. After data replication is finished, the metadata are updated in the data table.
4. In each scan cycle of the PRCR node, the metadata are scanned sequentially. According to the metadata, PRCR determines whether the file needs to be checked or not.
5. During the scanning process, if a file needs to be checked, the replica management module extracts its metadata from the data table.
6. The replica management module sends the proactive replica checking task to one of the Cloud computing instances. The Cloud computing instance executes the task, which checks the availability of both replicas of the file.
7. After the checking task is completed, the Cloud computing instance returns the result. The replica management module receives the returned result and conducts further action accordingly as follows.
8. If one of the replicas is not available<sup>3</sup>, the replica management module sends a replica generation request to the Cloud. Using the metadata of the other replica, a

<sup>3</sup> In some extreme cases, both replicas may be lost which would be within the reliability range, hence not jeopardising the reliability assurance in overall terms.

new replica is generated and thus the data can be recovered.

Note: Steps 4 to 8 form a continuous loop until the file is deleted, i.e. end of life cycle.

## VI. EVALUATION

### A. Parameter setting for the PRCR simulation

We have conducted several tests based on the Amazon Web Services including Amazon S3, Amazon EC2, and Amazon Beanstalk. Based on tests conducted on Amazon, we have obtained benchmarking parameters from a real Cloud environment.

The tests are conducted on an experimental PRCR implemented in Java. It is composed of one user interface and one PRCR node, and both of them run on a single EC2 instance. In order to ease the operation, the user interface is designed in the form of a Web site. In the experimental PRCR, the metadata scanning process and the proactive replica checking tasks are conducted. There are mainly two parameters collected from the tests on the Amazon Web Services: the metadata scanning time and the proactive replica checking task processing time. The metadata scanning time is a very important parameter. As mentioned in Section V.A, it is directly related to the capacity of each PRCR node. The proactive replica checking task processing time is the time spent for each proactive replica checking task. In this experimental PRCR, the content of each proactive replica checking task is to access both replicas of the file and check the availability. The proactive replica checking task processing time is an important parameter for the evaluation of PRCR, because the replica checking tasks occupy the most resources of PRCR, and we need this parameter to compute the running cost of PRCR. The tests are conducted on the experimental PRCR with several different configurations. We hired four types of EC2 computing instances for the management of 3000 S3 objects (e.g. files) stored with standard storage and reduce redundancy storage respectively.

The results are shown in Table I. It can be seen that the scanning time for each metadata is at a hundreds of nanoseconds level, and the checking task processing time of each file is at a tens of milliseconds level. To our surprise, the results seem to indicate that the performance of PRCR does not follow the trend of the increasing performance of the Cloud computing instances. As shown in Table I, the performance of the Cloud computing instances increases from left to right, but the scanning time and proactive replica checking task processing time do not increase accordingly. We speculate that the reason for this is that we have not optimised the code for multi-core processors, and the Cloud virtual partition technology may have reduced the overall performance. According to the results shown in Table I, in the evaluation of PRCR we chose 700ns and 30 ms as the benchmarks for metadata scanning time and proactive replica checking time respectively, and the micro EC2 instance (t1.micro) is chosen to be the default Cloud computing instance.

TABLE I. DATA SCANNING TIME AND CHECKING TASK PROCESSING TIME

	t1.micro	m1.Small	m1.large	m1.xlarge
Scanning time	≈700ns	≈400ns	≈700ns	≈850ns
CTP time(standard)	≈27ms	≈27ms	≈30ms	≈27ms
CTP time(reduce redundancy)	≈25ms	≈24ms	≈37ms	≈23ms

### B. Cost-effectiveness

In order to demonstrate the effectiveness of PRCR, we evaluate the capacity of PRCR nodes, and simulate the reliability management process using PRCR for data generated by pulsar searching workflow instances. The results are compared with the widely used 3-replica replication strategy.

We calculate the capacity of PRCR nodes for storing files with the data reliability requirements of 99% over a year, 99.99% over a year and 99.99999% over a year under different storage unit failure rates. In Table II, the relationship among the reliability requirement  $X$ , failure rate of single replica  $\lambda$  and the capacity of PRCR nodes can be clearly seen. With different single replica failure rate and reliability requirement, each PRCR node is able to manage from  $1.4 * 10^{10}$  files to  $4.7 * 10^{15}$  files which are rather large.

TABLE II. PRCR CAPACITY

$X \backslash \lambda$	0.1	0.01	0.001	0.0001
99%	$4.7 * 10^{12}$	$4.7 * 10^{13}$	$4.7 * 10^{14}$	$4.7 * 10^{15}$
99.99%	$4.5 * 10^{11}$	$4.5 * 10^{12}$	$4.5 * 10^{13}$	$4.5 * 10^{14}$
99.99999%	$1.4 * 10^{10}$	$1.4 * 10^{11}$	$1.4 * 10^{12}$	$1.4 * 10^{13}$

In a further step of the evaluation we simulate the reliability management process using PRCR for data generated by pulsar searching workflow instances. In the simulation we assume that all the pulsar searching workflow instances be the same. The configuration is set as follows. The storage unit failure rate is 0.01 per year. Two types of Cloud storage services are provided, which are 2-replicas storage with reliability assurance of 99.99999% over a year and 1-replica storage with reliability assurance of 99% over a year which is normally enough for short term data. The data stored in the former are managed by PRCR.

In the simulation, we applied four different types of storage plans respectively, which are 1-replica plan, 1+2 replica plan, 2-replica plan and 3-replica plan. Among these four storage plans, the 1+2 replica plan divides all the data into two categories, in which the data with higher reliability requirement are stored with 2 replicas and the rest are stored with 1 replica. As mentioned in Section III.B, in the pulsar searching workflow example, among all the files, the extracted & compressed beam files, the XML files and the de-dispersion files are stored for long-term use and have

higher reliability requirements, so that they are stored with 2-replica storage. The rest files are stored with the 1-replica storage. Figures 5 and 6 have shown the results of the simulation.

Figure 5 shows the average replica numbers and file sizes generated by the pulsar searching workflow instance. The results show that by applying the 2-replica storage plan, 1/3 of the generated data size can be reduced compared with the 3-replica replication strategy, and the average replica number for each file is reduced accordingly. By applying the 1+2 replica storage plan, the combination of two storage types further reduced the consumption of storage space. In our simulation, by applying the 1+2 replica storage plan for the pulsar searching workflow instance, each workflow instance has around 16000 files stored with 2 replicas, while more than 90000 files are stored with 1 replica. The ratio between the volumes of two types of files is about 4.8:1, and compared with the 2-replica plan, about 70GB data is reduced. With a large number of pulsar searching instances, the overall saving would be huge.

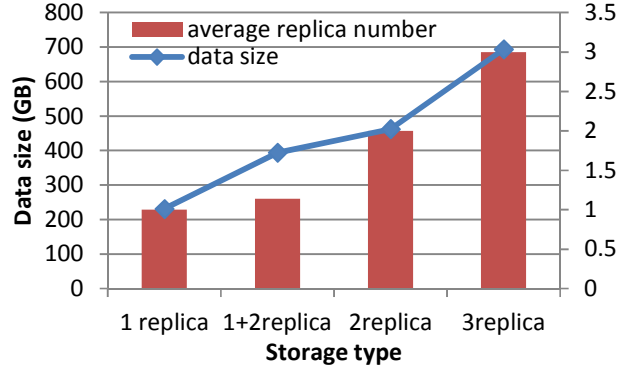


Figure 5. Average replica numbers and data sizes

For comparing the general cost of data storage using PRCR and the 3-replica replication strategy, we adopt the Amazon pricing model for Amazon EC2 instance and S3 storage. Assuming that Amazon S3 standard storage also use the 3-replica replication strategy, Figure 6 shows the cost for storing 1GB of data for one month in the Cloud. In this figure, the horizontal axis represents the number of pulsar searching workflow instances that is conducted. The results have indicated the cost-effectiveness of PRCR for the reliability management of different sizes of data. By using the micro EC2 instance, the cost for running PRCR with one PRCR node in the Cloud is \$14.4 per month. When only a small amount of data is managed by PRCR, the cost of hiring EC2 instances for PRCR has taken a significant proportion of the total data storage cost, and the cost-effectiveness of using the default 3-replica replication strategy in the Cloud is higher. With the increase of data amount, the proportion of cost for running PRCR become smaller, thus the unit storage cost decreases rapidly. Consider the situation that PRCR would normally well loaded, the cost for running PRCR is small so that it can be negligible. The unit storage cost can be reduced to from 2/3 to 1/3 of the Amazon S3 storage cost.

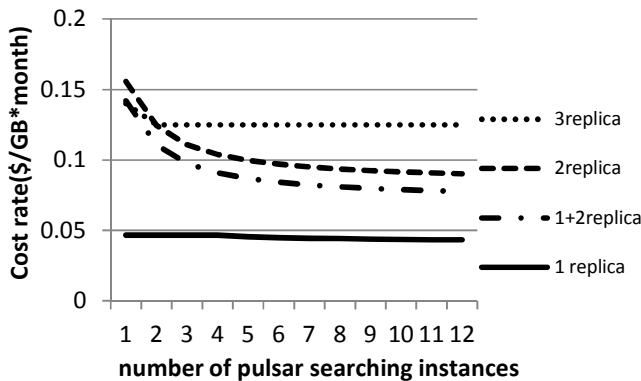


Figure 6. Cost rates for storing data

## VII. CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel cost-effective reliability management mechanism called PRCR. It applies a proactive replica checking approach to check the availability of each file, and the data reliability can be assured with no more than two replicas. Simulation of the reliability management process using PRCR for data generated by pulsar searching workflow instances has demonstrated that this mechanism can significantly reduce the Cloud storage space consumption and hence Cloud storage cost, while the data reliability requirement can be fully met.

Our current work considers only cost and reliability factors for data storage. Therefore, in this paper we did not discuss the data access performance issue. In the future, we will incorporate the performance factor in our research.

## REFERENCES

- [1] The Parkes Radio Telescope. Available: <http://www.parkes.atnf.csiro.au/>
- [2] J. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," in International Parallel and Distributed Processing Symposium, pp. 238–244, 2004.
- [3] Amazon. (2011). Amazon Simple Storage Service (Amazon S3). Available: <http://aws.amazon.com/s3/>
- [4] D. Borthakur. (2007). The Hadoop distributed file system: Architecture and design. Available: [http://hadoop.apache.org/common/docs/r0.18.3/hdfs\\_design.html](http://hadoop.apache.org/common/docs/r0.18.3/hdfs_design.html)
- [5] C. Dabrowski, "Reliability in grid computing systems," *Concurrency and Computation: Practice and Experience*, vol. 21, pp. 927–959, 2009.
- [6] R. Duan, R. Prodan, and T. Fahringer, "Data mining-based fault prediction and detection on the Grid," in IEEE International Symposium on High Performance Distributed Computing, pp. 305–308, 2006.
- [7] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in ACM Symposium on Operating Systems Principles pp. 29 - 43, 2003.
- [8] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher, "Adaptive Replication in Peer-to-Peer Systems," in International Conference on Distributed Computing Systems, pp. 360 - 369, 2004
- [9] H. Jin, X. H. Sun, and Z. Zheng, "Performance under failures of DAG-based parallel computing," in IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 236-243, 2009.
- [10] H. Lamehamed, B. Szymanski, Z. Shentu, and E. Deelman, "Data replication strategies in grid environments," in International Conference on Algorithms and Architectures for Parallel Processing, pp. 378 - 383, 2002.
- [11] 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.
- [12] W. Li, Y. Yang, and D. Yuan, "A novel cost-effective dynamic data replication strategy for reliability in cloud data centres," presented at the International Conference on Cloud and Green Computing, Sydney, Australia, 2011.
- [13] X. Liu, D. Yuan, G. Zhang, W. Li, D. Cao, Q. He, J. Chen, and Y. Yang, *The Design of Cloud Workflow Systems*: Springer, 2012.
- [14] D. Patterson, G. Gibson, and R. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in ACM SIGMOD International Conference on the Management of Data, pp. 109–116 1988.
- [15] F. Schintke and R. Alexander, "Modeling replica availability in large data grids," *Journal of Grid Computing*, vol. 1, pp. 219 - 227, 2003.
- [16] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman, and B. Tierney, "File and object replication in data grids," *Journal of Cluster Computing* vol. 5, pp. 305-314, 2002.
- [17] H. Vo, C. Chen, and B. Ooi, "Towards Elastic Transactional Cloud Storage with Range Query Support," *Proceedings of the VLDB Endowment*, vol. 3, 2010.
- [18] J. W. Young, "A first order approximation to the optimal checkpoint interval," *Communications of the ACM*, vol. 17, pp. 530-531, 1974.
- [19] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen, "A data dependency based strategy for intermediate data storage in scientific cloud workflow systems," *Concurrency and Computation: Practice and Experience*, 2010.