Cost-effective Replication-based Storage for Reliability Assurance of Big Data in the Cloud

by

Wenhao Li
B.Eng. (Shandong University)
M.Eng. (Shandong University)

A thesis submitted to
School of Software and Electrical Engineering
Swinburne University of Technology
for the degree of
Doctor of Philosophy

May 2014
To my parents and my wife
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Wenhao Li

May 2014
Acknowledgements

First of all, I would like to express my most sincere and deepest gratitude to my coordinate supervisor, Professor Yun Yang, for his continuous and patient supervision and encouragement throughout my PhD study during the past three and half years. It is his guidance with wisdom that makes me have a comprehensive understanding about doing research. And it is his kindness and upstanding that have deeply influenced me to be a better person. It is my honor to have Professor Yun Yang as my supervisor, research partner and most importantly, a friend. I will cherish our friendship lifelong.

Second, I would like to express my most sincere and deepest gratitude to my family members: I thank my father Jianguo Li, my mother Professor Hong Liu for raising me up and guide me to be a good person, and supporting me to continue my study. I thank my wife Qian Wang for her accompany and her support both mentally and during our daily life.

Third, I would like to thank Swinburne University of Technology for offering me a full research scholarship with tuition fee waiver. I also thank the Research Committee of former Faculty of Information and Communication Technologies for the research publication funding support, so that I can attend conferences. The financial support had greatly facilitated my studying and living in Melbourne. This research is also partly supported by Australian Research Council Discovery Project DP110101340.

Last but not least, I also thank my associate supervisor Associate Professor Jinjun Chen, my review panel members Dr Alan Colman, Prof Jun Han, Dr Caslon Chua, and my research colleagues in Centre for Computing and Engineering Software Systems for their friendship and help during my PhD study, in particular, Dr. Xiao Liu, Dr. Dong Yuan, Dr. Jing Gao, Dr. Gaofeng Zhang, Dahai Cao, Xuyun Zhang, Wei Dong, Antonio Giardina, Feifei Chen.
Abstract

Cloud computing is the latest distributed computing paradigm which provides redundant, inexpensive and scalable resources in a pay-as-you-go fashion to meet various application requirements. Nowadays, with the rapid growth of Cloud computing, the size of Cloud data is expanding at a dramatic speed. A huge amount of data that are big in sizes and large in amounts are generated and processed by Cloud applications with data-intensive characteristics. For maintaining the big data in the cloud, data reliability related issues are considered more important than ever before. However, current data storage and data reliability ensuring strategies based on multiple replicas have become a bottleneck for the big data storage in the Cloud. For storing the massive data in the Cloud, such strategies could consume a huge amount of storage resources on replication, and hence incur a huge storage cost and cause negative effects for both the Cloud storage providers and the storage users. Therefore, a higher demand has been put forward to 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.

In this thesis, we investigate the trade-off between data storage cost and data reliability assurance for the big data in the Cloud. The research is motivated by a scientific application for astrophysics pulsar searching surveys, which is of typical data-intensive characteristics and contains complex and time consuming tasks that process hundreds of terabytes of data. In order to facilitate the migration of the application into the Cloud, our novel research stands from the Cloud storage service providers’ perspective and investigates the issue on how to provide cost-effective data storage while meeting the data reliability requirement throughout the whole Cloud data lifecycle. Our research in this thesis presents four major contributions. According to different stages within the Cloud data lifecycle, these four contributions are presented in the following sequence.

1) For describing data reliability in the Cloud, a novel generic data reliability model is proposed. Based on a Cloud with replication-based data storage scheme, the data reliability model is able to describe the reliability of the Cloud data throughout their lifecycles, in which they are stored with different redundancy levels and stored on different
storage devices in different stages respectively. Compared with existing data reliability models that assumes a constant disk failure rate, our generic data reliability model is able to better describe data reliability over a wide range of failure rate patterns of storage devices.

2) To facilitate data creation, a minimum replication calculation approach for meeting a given data reliability requirement is proposed. Based on the data reliability model, this approach calculates the minimum number of replicas that needs to be created for meeting certain data reliability requirement and predicts the reliability of the data stored for a certain amount of time. In addition, the minimum replication can also act as a benchmark, which can be used for evaluating the cost-effectiveness of various replication-based data storage approaches.

3) In the data maintenance stage, in order to maintain the Cloud data with the minimum replication level in real, a novel cost-effective data reliability assurance mechanism named PRCR (Proactive Replica Checking for Reliability) is proposed. Based on the minimum replication that is created, PRCR is able to maintain the huge amounts of Cloud data with negligible overhead, while a wide variety of data reliability assurance can be provided. Compared with the widely used conventional 3-replica data storage and data reliability ensuring strategy, hence significantly lowering the storage cost in the Cloud. PRCR can reduce from one-third to two-thirds of the Cloud storage space consumption. Even more saving can be achieved compared with data storage strategies with higher replication level.

4) In the data creation and recovery stages, in order to reduce the data transfer cost, a cost-effective strategy named LRCDT (Link Rate Controlled Data Transfer) is proposed. By scheduling bandwidth in a link rate controlled fashion, LRCDT could significantly reduce the energy consumption during the data creation/recovery process in the Cloud network. The result in our simulation indicates that LRCDT is able to reduce energy consumption by up to 63% when compared to existing data transfer strategies.

The research issue of this thesis is significant and has practical value to the Cloud computing technology. Especially, for data-intensive applications that is already migrated or about to migrate into the Cloud, our research could significantly reduce their storage cost while meeting the data reliability requirement, hence has a positive impact on promoting the development of Cloud.
The Author’s Publications

Book:

Conferences:

Journals:
Journal submission (under revision):

Table of Contents

CHAPTER 1 INTRODUCTION ........................................................................................................1

1.1. Data Reliability in the Cloud .........................................................................................1

1.2. Background of Cloud Storage .....................................................................................3

1.2.1. Distinctive Features of Cloud Storage Systems ......................................................3

1.2.2. The Cloud Data Lifecycle .........................................................................................5

1.3. Key Issues of Research ...............................................................................................6

1.4. Thesis Overview ...........................................................................................................8

CHAPTER 2 LITERATURE REVIEW ..................................................................................11

2.1. Data Reliability Assurance in Hardware ......................................................................11

2.1.1. Disk .........................................................................................................................12

2.1.2. Other Storage Medias .............................................................................................15

2.2. Data Reliability Assurance in Software ......................................................................16

2.2.1. Replication for Data Reliability ..............................................................................16

2.2.2. Erasure Coding for Data Reliability ......................................................................18

2.3. Data Transfer for Distributed Systems ........................................................................19

2.4. Summary .....................................................................................................................21

CHAPTER 3 MOTIVATING EXAMPLE AND PROBLEM ANALYSIS .........................22

3.1. Motivating Example .....................................................................................................22

3.1.1. The Pulsar Searching Application Process ...............................................................23

3.1.2. The Pulsar Searching Application Dataflow ..............................................................25

3.1.3. Migrating the Pulsar Searching Application into the Cloud ....................................27

3.2. Problem Analysis .......................................................................................................28
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1.</td>
<td>Two Major Factors of Cloud Storage Cost</td>
<td>28</td>
</tr>
<tr>
<td>3.2.2.</td>
<td>Data Storage Devices and Schemes</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3.</td>
<td>Cloud Network and Data Transfer Activities</td>
<td>31</td>
</tr>
<tr>
<td>3.2.4.</td>
<td>Research Issues</td>
<td>33</td>
</tr>
<tr>
<td>3.3.</td>
<td>Summary</td>
<td>35</td>
</tr>
<tr>
<td>4.1.</td>
<td>Properties of the Data Reliability Model</td>
<td>37</td>
</tr>
<tr>
<td>4.1.1.</td>
<td>Reliability Metrics</td>
<td>38</td>
</tr>
<tr>
<td>4.1.2.</td>
<td>Data Reliability Model Type</td>
<td>39</td>
</tr>
<tr>
<td>4.1.3.</td>
<td>Failure Rate Pattern of Storage Devices</td>
<td>39</td>
</tr>
<tr>
<td>4.2.</td>
<td>Generic Data Reliability Model</td>
<td>40</td>
</tr>
<tr>
<td>4.2.1.</td>
<td>Data Reliability with Static Disk Failure Rate</td>
<td>40</td>
</tr>
<tr>
<td>4.2.2.</td>
<td>Data Reliability with Variable Disk Failure Rate</td>
<td>41</td>
</tr>
<tr>
<td>4.2.3.</td>
<td>Generic Data Reliability Model for Multi-replicas</td>
<td>43</td>
</tr>
<tr>
<td>4.3.</td>
<td>Summary</td>
<td>44</td>
</tr>
<tr>
<td>5.1.</td>
<td>The Minimum Replication Calculation Approach</td>
<td>45</td>
</tr>
<tr>
<td>5.1.1.</td>
<td>Minimum Replication Calculation Formulas</td>
<td>46</td>
</tr>
<tr>
<td>5.1.2.</td>
<td>Optimization of the Minimum Replication Calculation Formulas</td>
<td>49</td>
</tr>
<tr>
<td>5.2.</td>
<td>Minimum Replication Benchmark</td>
<td>51</td>
</tr>
<tr>
<td>5.3.</td>
<td>Evaluation of the Minimum Replication Calculation Approach</td>
<td>52</td>
</tr>
<tr>
<td>5.4.</td>
<td>Summary</td>
<td>53</td>
</tr>
<tr>
<td>6.1.</td>
<td>Proactive Replica Checking</td>
<td>55</td>
</tr>
<tr>
<td>6.2.</td>
<td>Overview of PRCR</td>
<td>56</td>
</tr>
</tbody>
</table>
6.2.1. User Interface........................................................................................................57
6.2.2. PRCR Node...........................................................................................................59

6.3. Working Process of PRCR ......................................................................................60

6.4. Optimization Algorithms in PRCR .......................................................................62
   6.4.1. Minimum Replication Algorithm .....................................................................62
   6.4.2. Metadata Distribution Algorithm ...................................................................64

6.5. Evaluation of PRCR ...............................................................................................67
   6.5.1. Performance of PRCR .....................................................................................68
   6.5.2. Cost-effectiveness of PRCR ............................................................................72
   6.5.3. Summary of the Evaluation ............................................................................75

6.6. Summary ..................................................................................................................76

CHAPTER 7  COST-EFFECTIVE DATA TRANSFER FOR DATA CREATION AND DATA RECOVERY ..............................................................77

7.1. Determining the Deadline for Data Creation and Data Recovery ......................78

7.2. Cloud Network Model ...........................................................................................79
   7.2.1. Overall Network Model ..................................................................................80
   7.2.2. Pipeline Model ...............................................................................................80
   7.2.3. Pipeline Agenda Model ..................................................................................81
   7.2.4. Overall Agenda Model ..................................................................................81

7.3. Energy Consumption Model for Cloud Data Transfer ..........................................83

7.4. Novel Cost-effective Data Transfer Strategy LRCDT .........................................84

7.5. Evaluation of LRCDT .............................................................................................89
   7.5.1. Parameters of Simulation ..............................................................................90
   7.5.2. Energy Consumption Comparison ..................................................................90
   7.5.3. Task Completion Time Comparison .................................................................92

7.6. Summary ..................................................................................................................93
CHAPTER 8 CONCLUSIONS AND FUTURE WORK.................................................94

8.1. Summary of This Thesis ............................................................................94

8.2. Key Contributions of This Thesis ..............................................................96

8.3. Further Discussion and Future Work .........................................................97

8.3.1. Further Discussions ............................................................................97

8.3.2. Future Work .......................................................................................98

BIBLIOGRAPHY ...............................................................................................99

APPENDIX NOTATION INDEX ......................................................................107
List of Figures

Figure 1.1 Cloud data lifecycle ........................................................................................................5
Figure 1.2 Thesis structure ................................................................................................................9
Figure 3.1 Pulsar searching workflow ..........................................................................................23
Figure 3.2 Dataflow graph of a pulsar searching instance for 8 minutes of observation ..........25
Figure 4.1 Failure rate pattern of disk D between time $t_0$ and $t_n$ ........................................41
Figure 6.1 PRCR architecture ......................................................................................................57
Figure 6.2 Working process of PRCR ..........................................................................................60
Figure 6.3 Pseudo code of the minimum replication algorithm ....................................................63
Figure 6.4 Pseudo code of metadata distribution algorithm .......................................................66
Figure 6.5 Original curve and tangent line of $e^{-z}$ ....................................................................69
Figure 6.6 Average replica numbers and data sizes .....................................................................75
Figure 7.1 Overall network model ...............................................................................................80
Figure 7.2 Pipeline model ............................................................................................................80
Figure 7.3 Pipeline agenda model ...............................................................................................81
Figure 7.4 Overall agenda model ................................................................................................82
Figure 7.5 Agenda merge algorithm ............................................................................................83
Figure 7.6 Power-bandwidth function of active routers ..............................................................84
Figure 7.7 Bandwidth scheduling algorithm for ‘lazy’ data transfer ..........................................87
Figure 7.8 Bandwidth scheduling process example .....................................................................89
Figure 7.9 Comparison of average energy consumptions ..........................................................91
Figure 7.10 Comparison of average completion time .................................................................93
List of Tables

Table 6.1  Execution Time and Accuracy Rate of Minimum Replication Algorithm ............71
Table 6.2  Metadata Scanning Time and Proactive Replica Checking Time..........................72
Table 6.3  Maximum Capacity of PRCR Nodes .................................................................73
Table 7.1  Simulation Parameters ..................................................................................90
Chapter 1 Introduction

With the rapid growth in the size of Cloud data, cost-effective data storage has become one of the key issues in Cloud research, yet the reliability of the huge amounts of Cloud data needs to be fully assured. In this thesis, we investigate the trade-off of cost-effective data storage and data reliability assurance in the Cloud. The novel research stands from the Cloud storage service providers’ perspective and investigates the issue on how to provide cost-effective data storage service while meeting the data reliability requirement throughout the whole Cloud data lifecycle. This topic is important and has a practical value to the Cloud computing technology. Especially, for data-intensive application that is already migrated or about to migrate into the Cloud, our research could dramatically reduce its storage cost while meeting the data reliability requirement hence has a positive impact on promoting the deployment of the Cloud.

This chapter introduces the background knowledge and key issues of this research. It is organized as follows. Section 1.1 gives the definition of data reliability and briefly introduces current data reliability assurance technologies in the Cloud. Section 1.2 introduces the background knowledge related to Cloud storage. Section 1.3 outlines the key issues of the research. Finally, Section 1.4 presents an overview for the thesis structure.

1.1. Data Reliability in the Cloud

The term “reliability” is widely used as an aspect of the service quality provided by hardware, systems, Web services, etc. In Standard TL9000, it is defined as “the ability of an item to perform a required function under stated conditions for a stated time period” [13]. For data reliability specifically, which refers to the reliability provided by the data storage services/systems for the stored data, it can be defined as “the probability of the data surviving
in the system for a given period of time” [33]. While term “data reliability” is sometimes used in the industry as a superset of data availability and various other topics, in this thesis we will stick to the definition of data reliability given above.

Data reliability indicates the ability of the storage system to keep data consistent, hence is always one of the key metrics of a data storage/management system. In large-scale distributed systems, due to the big quantity of storage devices being used, failures of storage devices occur frequently [70]. Therefore, the importance of data reliability is prominent, and these systems need better design and management to cope with frequent failures. Increasing the data redundancy level could be a good way for increasing data reliability [39], [23]. Among several major approaches for increasing the data redundancy level, data replication is currently the most popular approach in distributed storage systems. At present, data replication has been widely adopted in many current distributed data storage/management systems in both industry and academia, which include examples such as OceanStore [48], Data Grid [21], Hadoop Distributed File System [15], Google File System [31], Amazon S3 [6], etc. In these storage systems, several replicas are created for each piece of data. These replicas are stored in different storage devices, so that the data have better chance to survive when storage device failures occur.

In recent years, Cloud computing is emerging as a latest distributed computing paradigm which provides redundant, inexpensive and scalable resources in a pay-as-you-go fashion to meet various application requirements [83]. Since the advent of Cloud computing in late 2007 [76], it has fast become one of the most promising distributed solutions in both industry and academia. Nowadays, with the rapid growth of Cloud computing, 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 more data are “touched” by the Cloud within their lifecycles [29]. For maintaining such a large amount of Cloud data, data reliability in the Cloud is considered more important than ever before. 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. For example, storage systems such as Amazon S3, Google File System and Hadoop Distributed File System all adopt similar data replication strategies called the conventional multi-replica replication strategy, in which a fixed number of replicas (normally three) are stored for all data to ensure the reliability requirement. For storage of the huge amounts of
Cloud data, these conventional multi-replica replication strategies consume a lot of storage resources for additional replicas. This could cause negative effects for both the Cloud storage providers and users. On one hand, from Cloud storage provider’s perspective, the excessive consumption of storage resources leads to a big storage overhead and increases the cost for providing the storage service. On the other hand, from Cloud storage user’s perspective, according to the pay-as-you-go pricing model, the excessive storage resource usage will finally be paid by the storage users. For data-intensive Cloud applications specifically, the incurred excessive storage cost could be huge. Therefore, Cloud-based applications have put forward a higher demand for cost-effective management of 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.

1.2. Background of Cloud Storage

In this section, we briefly introduce the background knowledge of Cloud storage. First, we introduce the distinctive features of Cloud storage systems. Second, we introduce the Cloud data lifecycle.

1.2.1. Distinctive Features of Cloud Storage Systems

Data reliability is closely related to the structure of the storage system and how the storage system is being used. Different from other distributed storage systems, the Cloud storage system have some distinctive features that could either be advantages or challenges for the data reliability management of Cloud data.

- **On-demand self-service and pay-as-you-go pricing model**

  The on-demand usage of Cloud storage service and pay-as-you-go payment fashion have greatly facilitated the storage users that they only need to pay for the resources used for storing their data for a needed time period. The cost is easy to be estimated according to the size of data generated [7]. However, based on the pay-as-you-go model, every usage of the resources can be strictly reflected onto the bills payable at the end of the month. Therefore, minimizing resource consumption becomes demanding and critical. This principle is not only applicable to the service users, but also to the Cloud storage service providers. In most current
Cloud storage services, excessive data redundancy is compulsorily generated to ensure data reliability. For data-intensive applications, such excessive data redundancy consumes a large amount of storage resources, and hence incurs very high cost.

- **Redundant and scalable virtualized resources**

  In the Cloud, large amounts of virtualized computing and storage resources are pooled to serve users with various demands [13]. Redundant computing resources make it easy to conduct parallel processing, while the redundant storage resources make it easy to distribute data. For meeting a higher computing/storage demand, the resource pool can be scaled out rapidly, and the virtualization keeps the complex procedures transparent from the service users. However, the virtualization of resources has also led to a challenge that various kinds of data reliability requirement need to be fully assured to make the Cloud storage service trustworthy.

- **Dedicated Cloud network**

  Cloud systems (public Clouds specifically) are primarily running based on data centers with dedicated networks, which interconnect with each other using dedicated links [55]. Such a dedicated feature of the Cloud network has provided the Cloud the potential of full bandwidth control ability. The Cloud storage system could benefit from the dedicated Cloud network that the creation and recovery of data can be conducted in a fully controllable and predictable manner. At the meantime, there is still a great potential that data transfer in the Cloud network could be optimized for being conducted more cost-effectively.

- **Big Data**

  Big data is the term for a collection of data sets so large and complex that it becomes difficult to store and process using traditional data storage and processing approaches. In Cloud storage systems, big data is one of the most distinctive features of the Cloud storage system. These data are generated by a large number of Cloud applications, many of which are data-intensive and computation intensive, and of great importance to these applications. Moreover, the size of the Cloud data is growing even faster. Due to the huge amount of resources consumed by these data, efficient data management could generate huge value. For managing the massive amounts of Cloud data, the Cloud storage system needs to be powerful enough and able to meet the diverse needs of the data of different usages at different stages.
1.2.2. The Cloud Data Lifecycle

![Cloud data lifecycle diagram](image)

**Figure 1.1 Cloud data lifecycle**

The Cloud data lifecycle refers to the period of time starting from the data being created (generated or uploaded) in the Cloud to the data being deleted when the storage space is reclaimed by the Cloud storage system. The lifecycle of each piece of Cloud data consists of four stages, which are the data creation stage, the data maintenance stage, the data recovery stage and the data deletion stage, as depicted in Figure 1.1.

- **Data creation**

  The lifecycle of Cloud data starts from the creation of the data in the Cloud storage system. When the original piece of Cloud data (the original replica for short) is created, certain numbers of additional replicas of the Cloud data also need to be created according to the specific reliability requirement of each piece of data and the storage policy [15], [31]. All these replicas are transferred and stored on specific storage devices in a distributed fashion.

- **Data maintenance**

  After the data are created and stored, the data maintenance stage commences, which occupies the majority of the Cloud data life. At this stage, Cloud data are processed within applications to achieve different goals. However, for most of the time these data are just stored in storage devices waiting for later use. Certain mechanisms can be conducted to maintain all
the replicas so that the service quality is not jeopardized. From the data reliability aspect, the redundancy of Cloud data is maintained at a certain level, so that sufficient data reliability assurance can be offered to meet the storage user’s data reliability requirement.

- **Data recovery**

  At the data maintenance stage of the Cloud data lifecycle, replicas could loss due to storage failures. In order to either restore the redundancy level of the Cloud data or prevent the data from total loss, data recovery is needed. At this stage, certain mechanisms are conducted to recover the lost replicas. For various purposes, these mechanisms follow different data recovery policies and the duration of the data recovery stage could vary. From the data reliability aspect, the data need to be recovered before the data reliability assurance become too low to meet the storage user’s requirement.

- **Data deletion**

  When the data are no longer needed, they are deleted. The storage space reclamation mechanism of the Cloud (if any) then recycles the pre-occupied storage space, and the lifecycle of the Cloud data ends. Hence this stage of the Cloud data lifecycle will not be discussed in this thesis any further. However, as we will explain later in the thesis, for determining the proper data reliability assurance that meets the storage user’s data reliability requirement, it is preferable that the expected storage duration be given when the data are created.

1.3. **Key Issues of Research**

The research in this thesis involves two major aspects: cost-effective data storage and data reliability. On one hand, the storage cost highly depends on the redundancy level of the data. By reducing the redundancy of the Cloud data, the storage cost could be reduced proportionally. Due to the massive amount of the big data in the Cloud, the storage cost saved can be huge. On the other hand, reducing redundancy also means that the data reliability may be jeopardized, i.e. the data cannot survive until they are deleted (or discarded). In order to provide cost-effective data storage while meeting the data reliability requirement of the Cloud storage users throughout the Cloud data lifecycle, our research involves the following key issues.
1) Data reliability model

First of all, we need a model to describe Cloud data reliability and Cloud data reliability related factors, which is essential for the design of data reliability assurance approach in the Cloud. The data reliability model should be able to describe the reliability of the Cloud data throughout their lifecycles, in which the data are stored with different redundancy levels and stored on different storage devices at different stages respectively.

2) Determination of the minimum replication

In order to reduce the storage cost in the Cloud, we need to determine the minimum data redundancy level for meeting the data reliability requirement. As will be further explained in Chapter 3, our research focuses on the data reliability issue in the Cloud with a replication-based data storage scheme. Therefore, in order to store the Cloud data in a cost-effective fashion, at the data creation stage of the Cloud data lifecycle, the number of replicas created for the Cloud data need to be minimized. Based on the data reliability model, we need an approach that predicts the data reliability under certain given replication level so that the minimum replication that needs to be created can be determined. As a direct consequence, the minimum replication can also act as a benchmark which can be used for evaluating the cost-effectiveness of various replication-based data storage approaches.

3) Cost-effective data reliability assurance

In order to maintain the Cloud data with the minimum replication level, a mechanism that is able to create Cloud data based on the minimum replication calculation approach as well as maintain the created replicas in the Cloud needs to be designed. For effective Cloud data reliability management, this mechanism needs to be able to maintain the big data in the Cloud with a wide variety of data reliability assurance so that all different levels of data reliability requirements can be met. In addition, as a very important aspect, the overhead of such a mechanism also needs to be taken into account.

4) Cost-effective data transfer

When replicas of the Cloud data need to be created or are lost, we need to provide effective data transfer process that could maintain the replication level of the data in a cost-effective fashion. In the data creation and recovery stages of the Cloud data lifecycle, data
transfer activity plays the major role which transfers the data to the appropriate storage devices. Therefore, optimizing the data transfer in Cloud network could be a good solution for cost-effectiveness. By optimizing data transfer, the cost incurred by data creation or recovery can be reduced.

1.4. Thesis Overview

This thesis systematically investigates the challenging issue of providing cost-effective data storage with data reliability assurance, which includes solid theorems and practical algorithms and finally forms a comprehensive solution to deal with the issue. The thesis structure is depicted in Figure 1.2.

In Chapter 2, we introduce existing works in literatures related to our research. To facilitate our research, literatures in three major fields are reviewed. First, from the hardware aspect, to investigate the reliability pattern of storage devices in the Cloud, literatures on hardware reliability theories are reviewed. Second, from the software aspect, to investigate data reliability models, and data redundancy maintenance approaches in the Cloud, literatures on data reliability modeling, data reliability assurance approaches in distributed data storage systems are reviewed. Third, to investigate data recovery approaches in the Cloud, literatures on data recovery and data transfer approaches in distributed systems are reviewed.

In Chapter 3, we present the motivating example of this thesis and analyze our research problem. We first introduce the motivating example of our research, which is a real world scientific application for pulsar searching survey of typical data-intensive characteristics. Based on the motivating example, we analyze the research problem and identify details of our research issues.

In Chapter 4, we present our data reliability model for Cloud data storage. Based on the details of our research issues identified in Chapter 3, first we further determine several properties for our data reliability model, and then our novel generic replication-based data reliability model is presented in detail.
In Chapter 5, we present the minimum replication calculation approach. Based on our generic data reliability model presented in Chapter 4, a minimum replication calculation approach for determining the minimum number of replicas needed for meeting data reliability requirement is proposed. Afterwards, in this chapter we also discuss the usage of the minimum
replication for meeting data reliability requirement as a benchmark, which can be applied for
evaluating the cost-effectiveness and data reliability assurance of various replication-based
data storage approaches. Finally, the evaluation for validating the minimum replication
calculation approach is briefly presented.

In Chapter 6, we present our cost-effective data reliability assurance mechanism
named PRCR (Proactive Replica Checking for Reliability) for maintaining the big data in the
Cloud in a cost-effective fashion. We first present the principle of data reliability assurance by
proactive replica checking. Then the structure and working process of PRCR are presented.
Afterwards, we present algorithms for optimizing PRCR. Finally, evaluations for validating
PRCR are presented in detail, in which the evaluation for the minimum replication algorithm
of PRCR can also reflect the effectiveness of the minimum replication calculation approach.

In Chapter 7, we present our novel energy-efficient data transfer strategy called
LRCDT (Link Rate Controlled Data Transfer) for reducing the data transfer cost incurred
during Cloud data creation and recovery processes. We first present two models for the
strategy, which are the Cloud network model and the energy consumption model of network
devices. Based on these models, we present the principle and detailed design of LRCDT for
reducing data transfer energy consumption by link rate control. Finally, the evaluation for
validating LRCDT is presented in detail.

Finally, in Chapter 8, we summarize the works that have been presented in this thesis
and the major contributions of this research. Further discussions and related research works
are also included.

In order to improve the readability of this thesis, we put the notation index in the
Appendix, which is located at the end of this thesis.
Chapter 2 Literature Review

In this chapter, existing literatures related to the research are reviewed from three major aspects. First, to investigate the data reliability pattern in storage devices, we review literatures on hardware reliability theories and existing reliability models of storage devices. Second, to investigate data reliability models and data redundancy maintenance achieved by using software approaches in the Cloud, literatures on data reliability modeling and data reliability assurance approaches in distributed data storage systems are reviewed. Third, to investigate data transfer for data creation and recovery in the Cloud, literatures on data transfer approaches in distributed systems are reviewed.

The structure of this chapter is organized as follows. In Section 2.1, we summarize existing researches on data reliability assurance in data storage devices in distributed storage systems. In Section 2.2, we review researches on data reliability assurance with software based approaches in distributed storage systems, in which we focus on two major types of data redundancy approaches: data replication and erasure coding. In Section 2.3, we review researches on data transfer approaches in distributed systems. Finally, in Section 2.4, we summarize the works presented in this chapter.

2.1 Data Reliability Assurance in Hardware

In a distributed storage system, there are many factors that could lead to data loss, such as logical errors that refer to non-physical failures (e.g. incorrect software updates, programming errors, etc.) and hardware failures (e.g., disk failures), etc. [19] [13]. However, considering factors that are due to the storage system itself, hardware failure caused by non-human factors is considered to be the major reason for data loss. No matter how fine the system is designed, the occurrence of hardware failures is inevitable, where data loss incurs. In that case, the reliability of data is determined by the storage device on which the data are stored. In this section, we review existing researches on data reliability theories for storage devices.
2.1.1. Disk

It is estimated that over 90% of all new information produced in the world are stored on magnetic media, most of them on hard disk drives [62]. In current Clouds, disks are still the most commonly used storage device for storing the massive amount of Cloud data. Investigations towards the reliability pattern of disks have been conducted for decades in both academia and industry [27], [33], [65], [80]. With the development of distributed systems, such as Clusters [5], P2P systems [81] [67], Grids [21], and Clouds [26] [47] [44], the reliability issues of disks become more important compared to systems with centralized storage due to the big disk amount and more disk failures. During the research for decades, many investigations on issues of disk reliability have been conducted.

Disk failure modes

There are several kinds of disk failure modes. In general, these disk failure modes can be categorized into two categories: partial disk failure and permanent disk failure.

- Partial disk failures

This is a type of disk failures that only affect part of storage space of the disk while the rest is still functional. There are only a few existing works that study partial disk failures. For example, a type of partial disk failure commonly referred to as "bad sectors" has been relatively well studied since 1990s. The bad sectors are seen as inaccessible data blocks or sectors during reading or writing operation. The main cause is due to wear and tear of platter surface, head crash, manufacturing defects and tracking errors. Research on identifying and replacing bad sectors of disks is conducted in [28] [66] [22] respectively, and useful tools have also been produced in industry for a long time. In [41], detailed investigations on partial disk failures are conducted, where several fault-tolerance techniques are proposed to proactively guard against permanent data loss due to partial disk failures. However, research conducted solely on analyzing partial disk failures is rare, as many of the solutions dealing with permanent disk failures can also be used to recover data from a partially failed disk. For example, the data replication approach can be applied on a single disk to avoid bad sectors [40]. Redundant Array of Inexpensive Disks (RAID) can also be used to improve the reliability of data by storing additional parity information on multiple disks, which is generic for both partial and permanent disk failures [61].
• **Permanent disk failures**

The term “permanent disk failure” is used to describe the type of disk failure that the disk is physically not recoverable and requires replacement [70] [62]. The reason for a permanent disk failure could be complex and hard to identify. Damage in internal components such as the printed circuit board, the read-write head and motor or firmware failure could all lead to a permanent disk outage. In general, when permanent disk failures happen, the data stored on the disk is considered to be permanently lost.

Currently, the assumption of permanent disk failure in the disk reliability research and data reliability research is common [70] [62] [23] [50]. In this thesis, the research is conducted based on the permanent disk failure mode. Therefore, we mainly investigate the existing related work about permanent disk failures in the rest of this section.

**Disk reliability metrics**

In general, there are two metrics that are widely used for describing the permanent disk failure rates, which are the Mean Time to Failure (MTTF) and Annualized Failure Rate (AFR). **MTTF** is the length of time that a device or other product is expected to last in operation. It indicates how long the disk can be reasonably expected to work. In industry, the **MTTF** of disks are obtained by running many or even many thousands of units for a specific number of hours and check the number of disks that is permanently failed. Instead of using **MTTF** for describing disk reliability, some hard drive manufacturers now use annualized failure rate [46]. **AFR** is the estimated probability that the disk will fail during a full year of use. Essentially, **AFR** can be seen as another form of **MTTF** expressed in years, which can be obtained according to Equation (2.1) below [78]:

\[
AFR = 1 - \exp(-\frac{8760}{MTTF})
\]  

(2.1)

where 8760 is to convert the time unit from hour to year (1 year = 8760 hours). The advantage of using **AFR** as the disk reliability metric is that it is more intuitive and easier to be understood by non-computer specialists. For example, for a disk with **MTTF** of 300,000, the **AFR** is 2.88% per year, i.e. a probability of 2.88% that the disk is expected to fail during one year of use.
However, in practice, the AFR value is sometimes not consistent with the MTTF value specified in the datasheets of the disks [27] [70]. Because of a variety of factors such as working temperature, work load, etc., actual disk drive reliability may differ from the manufacturer’s specification and vary from user to user [27]. In [70], MTTF and AFR values of disks are comprehensively investigated according to records and logs collected from a number of large production systems for every disk that was replaced in the system. According to the results of these collected records and logs, the AFR of disks typically exceeds 1%, with 2-4% as a norm, and sometimes over 10% can be observed. Meanwhile, however, the datasheet MTTF of those disks only ranges from 1,000,000 to 1,500,000 hours (i.e. an AFR of at most 0.88 %). In [62], disk reliability analysis based on Google's more than one hundred thousand ATA disks also observes an average AFR value higher than 1%, which is from 1.7% for disks that were in their first year of operation to as high as 8.6% for older disks of 3 years old.

Disk reliability patterns

The failure pattern of disks is always a key aspect in the field of disk reliability study. In some early researches on this issue, the failure pattern of disks is assumed to follow exponential distribution [33] [82] due to the continuous and independent occurrence of disk failures. For example, an early study conducted in [33] states that the lifespan of disks can be characterized by exponential distribution. In addition, in order to simplify the calculation, some more recent studies that analyze data reliability also assume an exponential disk/data reliability model [39], [65].

In the exponential disk reliability models, the failure rate of each disk is a constant. However, these reliability models with a constant disk failure rate cannot explain some of the phenomena happening in reality. It has been quite well known that the failure rate of disk drives follows what is often called "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 [32]. This disk failure model underlies many of the more recent models and simplifications, such as in [80] where the disk failure model incorporates the “bathtub” curve to observe the infant mortality phenomenon in large storage systems, etc. In addition to the “bathtub” curve model, some other studies have also obtained results that contradict the constant disk failure rate model. For
example, [27] shows that populations of disks generally do not follow an exponential failure distribution.

Despite of the exponential disk reliability model and the bathtub disk reliability model, there is another type of model that describes the failure pattern of disks in a discrete fashion. For example, the International Disk Drive Equipment and Materials Association (IDEMA) proposed a compromised presentation for disk failure rates that uses discrete disk failure rates [42]. It divides the lifespan of each disk into four different life stages, which are 0–3 months, 3–6 months, 6–12 months, and one year to the End of Design Life (EODL), and disks have different failure rates at different life stages. The discrete disk reliability models have fixed the inconsistency between the exponential disk reliability models, which are of constant disk failure rates, and the variable disk failure rate in reality. Moreover, such models greatly reduced the complexity of the continuous disk reliability model based on “bathtub” curve. Such a discrete disk failure rate model has been demonstrated to be feasible in [80], and a nine-month investigation conducted by Google also obtained results very consistent to this model [62].

2.1.2. Other Storage Medias

This thesis will primarily focus on storage media of disk. However, despite of the widely used disks as the dominating storage devices, there are also several other data storage medias that need to be mentioned.

**Magnetic tape:** it is a data storage device that uses magnetic tapes in the form of cartridges or cassettes for storing large amount of data with very low timely requirement. Currently, the highest capacity tape cartridges can reach the size of 8.5 TB [77], which is quite large. The biggest advantage of magnetic tape data storage is that the storage cost can be significantly reduced as tapes can be much cheaper than disks. Modern usage of magnetic tape storage is primarily as a high capacity medium for backups and archives. However, the poor random access performance and high maintenance overhead of a tape library have limited its usage. Little research has been conducted on investigating the reliability pattern of magnetic tapes. Several researches commit on replacing magnetic tape data storage into disk storage have been spotted in both academia and industry [72] [63] [8].
Solid-State Drive (SSD): it is a data storage device that uses solid-state memory to store data. SSDs are invented for the same purpose as disks and magnetic tapes, but they are made of electronic storage units and do not have any mechanical parts. Unlike disks, SSDs do not store data on spinning platters, but use flash memory instead, such feature have eliminated the possibility of storage failures caused by mechanical problems. Compare to disks, SSDs have several benefits such as much higher data read/write speed and more lightweight. However, it is more expensive per GB of storage and has a lower storage capacity. Each storage unit (memory cell) of SSDs has a strictly limited number of write cycles. Therefore, under certain usage frequency, the failure rate of SSDs could be shown in a continuously rising form, in which the magnitude of increment depends on the writing frequency of SSDS. In addition, some research has also found that SSDs are more vulnerable to power faults compared with disks [87]. In order to enhance the data reliability assurance for storing data on SSDs, RAID-based approaches are investigated in [45] [60].

2.2. Data Reliability Assurance in Software

Apart from researches on reliability theories for storage devices, many efforts for ensuring data reliability have also been made in the software aspect. In this section, we summarize existing literatures on providing data reliability assurance with software-based approaches. Essentially, all the approaches in these literatures achieve the goal of data reliability assurance by adding redundancy to the data. In general, these approaches can be categorized as two major types, which are data replication and erasure coding

1. Both of these approaches have been widely applied to existing distributed storage systems, which form two storage schemas: replication-based data storage schema and erasure coding-based data storage schema. These two storage schemas have their own advantages and disadvantages, and are useful for different scenarios. In this section, these two kinds of approaches with their corresponding storage schemas are reviewed respectively.

2.2.1. Replication for Data Reliability

Among all the existing approaches for adding data redundancy and supporting data reliability, data replication has been considered as a dominant approach in current distributed

1 In addition to these two categories, there do exist some hydrated storage systems that leverage both data replication and erasure coding approaches.
data storage systems. Currently, distributed storage systems that leverage replication for providing data reliability include ThriftStore [30], Farsite [3], TotalRecall [14], Google File System (GFS) [31], Hadoop Distributed File System (HDFS) [15], Amazon S3 [6], PVFS [18], Ceph [75], Freeloader [73] and many others. Specifically, TotalRecall uses replication for small files and erasure coding for large files, Windows Azure Storage [17] uses replication for ‘hot’ data and erasure coding for older yet less used data to reduce the storage cost. Therefore, in these two systems both storage schemes are used.

Data replication related researches have been conducted for many years, and many approaches on this topic for data reliability related issues in distributed storage systems have been proposed [24] [25] [30] [49] [68] [69] [71] [86]. Review articles include such as [25], a detailed survey on reliability issues of Grid systems is presented, in which data replication researches for the reliability of grid systems are comprehensively reviewed and important issues and reviews of data reliability research in grid environments are also identified. In [68], a series of optimistic replication algorithms (or can be understood as ‘lazy’ replication algorithms) is comprehensively surveyed, which synchronizes changes in replicas in the background, discovers conflicts after they happen, and reaches agreement on the final contents incrementally. In this paper, key challenges of optimistic replication systems are also identified, such as ordering operations, detecting and resolving conflicts, propagating changes efficiently, and bounding replica divergence, etc.

For describing data reliability of replication-based systems, in [23] [49] [53] [65] [69], analytical data reliability models are proposed and comprehensively studied. Among these data reliability models, models in [49] [69] are based on simple permutations and combinations to analyze the probability of data loss, while models in [23] [53] [65] are based on more complicated Markov chains to analyze changes in data redundancy level. In [49], data reliability of the system is measured by data missing rate and file missing rate, and the issue of maximizing data reliability with limited storage capacity is investigated. In [69], it proposes an analytical replication model for determining the optimal number of replica servers, catalogue servers, and catalogue sizes to guarantee a given overall data reliability. In [23] [53] [65], researches are conducted on different aspects of similar scenarios. In [23], it investigates the issue of how to dynamically maintain a certain replication level of a large-scale data storage system by gradually creating new replicas. In [53], it proposed an analytical framework to reason and quantify the impact of replica placement policy on system reliability. In [65], it
investigates the issue of maintaining a long-running distributed system using solely data replication. The similarity of these three papers is that they all assume a relatively high replication level \((N \text{ replicas/bricks/data blocks})\) in a large-scale data storage system environment, while replicas are gradually created when needed.

In Cloud computing, data replication technologies have also been widely adopted in current commercial Cloud systems. Some typical examples include Amazon Simple Storage Service (Amazon S3) [6], Google File System (GFS) [31], Hadoop distributed file system (HDFS) [15], etc. Although data replication has been widely used, there is a side effect because 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 cost [6]. However, such cost reduction is realized by sacrificing data reliability. By using RRS, only a lower level of data reliability can be ensured. Some of our works have made contributions in reducing storage cost in the Cloud based on data replication. For example, in [52], we propose 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 cost is limited.

2.2.2. Erasure Coding for Data Reliability

Besides data replication, another type of data storage approaches leverages erasure coding techniques to add data redundancy level so as to reach the data reliability assurance goal. Currently, distributed storage systems with erasure coding-based storage schema include OceanStore [48], Ivy [57], Windows Azure [39], etc.

Erasure coding is a coding approach that reorganizes the original information into another form. In information theory, it creates a mathematical function referred to as polynomial interpolation or oversampling and transforms a message of \(k\) symbols into a longer message (code word) with \(n\) symbols such that the original message can be recovered from a subset of \(n\) symbols [79]. By transforming the message, \(m\) redundant symbols are added to provide protection from storage failures where \(m=n-k\). The redundancy level or code rate is \(n/k\).
The erasure coding approaches have been developed for a long time and widely used for providing data reliability assurance. For example, the simplest even and odd parity is used by RAID 5 to achieve redundancy, in which if a drive in the array fails, remaining data on the other drives can be combined with the parity data (using the Boolean XOR function) to reconstruct the missing data [20]. Reed–Solomon (RS) codes are widely used in producing CDs, DVDs or Blu-ray disks or building RAID 6 data arrays or storing data in mass storage systems [39], etc. Some hybrid researches that combine replication and erasure coding or analyze differences between replication and erasure coding are also conducted [12] [74]. In [12], it proposes a solution, referred to as ‘fusion’ that uses a combination of erasure codes and selective replication for tolerating multiple crash faults over multiple data structures in general distributed systems. In [74], the analysis between replication and erasure coding storage solutions for P2P systems is conducted, where the authors state that erasure coding can significantly reduce the self-repair bandwidth. Recently, researches for erasure coding storage solutions in Clouds are also seen [11], [39]. In [11], an erasure coding approach using the Reed-Solomon 10+4 codes is applied to HDFS-RAID storage systems at Facebook. and in [39], novel LRC 6+3 or 12+4 codes are applied to part of Windows Azure Storage service.

Unlike data replication approaches for storage, erasure coding approaches divide data into several different data blocks, modify the original data and store them with additional erasure coding blocks. By using erasure coding approaches, the data reliability can be assured at a quite high level. Compared to data replication, erasure coding approaches have better performance at reducing storage redundancy and data recovery bandwidth. However, the computing overhead for coding and decoding erasure coded data is very high. For example, in [16], the decoding time for a data block of 16MB using Tornado Z codes is at a magnitude of tens to hundreds of seconds. Such a performance is somewhat even above the average performance of other erasure codes, such as the Reed-Solomon codes, etc.

2.3. Data Transfer for Distributed Systems

Data recovery is a very important aspect of data reliability management. No matter which data redundancy approach is applied, the lost data must always be recovered when possible so that the redundancy can be maintained at a satisfactory level. Data recovery approaches are highly dependent on the data storage schema of the distributed storage systems.
For systems with either replication-based or erasure coding-based data storage schema, different replication levels or erasure codes could lead to different data recovery strategies [39]. However, for recovering data in a large distributed storage system, there is one universal principle: when the data is lost, the lost data (either already restored to the form of the lost data or not) need to be transferred from somewhere (to somewhere else) to recover the original status of the data, and hence data transfer is considered to be the main procedure during the data recovery process. In order to recover Cloud data in a cost-effective fashion, in this section we focus on data transfer approaches for distributed systems. In addition to data recovery, data transfer is also intensively involved in creating replicas in the Cloud. Therefore, reviews conducted in this section could also benefit our research for creating replicas in the data creation stage.

Data transfer has been considered a very important research issue in the field of high-performance networks and distributed storage systems for a long time [4] [58]. In recent years, the ever developing Cloud and large-scale distributed storage technologies have resulted in higher demand for data transfer from both data transfer speed and energy consumption aspects. Balancing the trade-off between data transfer speed and energy consumption is a significant challenge.

On one hand, to meet the requirements of the large-scale data-intensive applications, the need for high speed yet predictable data transfer is increasing where networks with effective bandwidth controls are required. Due to its fully controlled feature, dedicated networks with bandwidth reservation have drawn more and more attention. Typical examples of dedicated networks include research networks such as the National Lambda Rail [1] and the Internet2 Network [2]. In [64], a bandwidth reservation approach via a centralized resource management platform is proposed for providing predictable performance in research networks. The centralized management pattern has, however, limited scalability and hence constrains the applicability of this approach. In [59], a distributed bandwidth reservation approach for reducing energy consumption in dedicated networks is proposed that can greatly improve the scalability issue compared to [64].

On the other hand, the energy consumption for high speed large-scale data transfer is high. This has become one of the major factors that need to be considered in large-scale storage systems. In recent years, many efforts have been made to reduce the energy consumption incurred in large-scale data transfer. For example, in [43] a standard is developed
for defining management parameters and protocols in energy-efficient Ethernet networks. In [10] and [59], energy consumption models are proposed for switches and general network devices respectively. To reduce the energy consumption over network links, several approaches are proposed. In [23], a replica creation and recovery strategy is proposed where data transfer is conducted with a constant minimum speed to maintain a certain number of replicas. In [37] and [38], energy management approaches, referred to as shutdown approaches, are proposed. In these approaches, devices on the link are shut down when network traffic is too low so that the energy consumption of routers and network links can be reduced. Specifically, in [38], the shutdown approach is conducted in such a way that data are transmitted as fast as possible and the data transfer link is ‘idled’ after data transfer is finished. There could be problems for such approaches, however, as some other tasks might also use the same data transfer link meaning it cannot be shut down. Different from the shutdown approaches that shut down devices to save power, in [35], a phenomenon was observed that less energy is consumed by network devices when operating at lower link rates. In [59], it states that the power of a network device incurs only negligible change when working at a certain link rate. In addition, it is also reported in [85] that the power of routers varies near exponentially with the change of link rate. These findings indicate that, by leveraging link rate control, the energy consumption of network devices can be greatly reduced. This idea has led to the proposition of the technology called Adaptive Link Rate (ALR) [34]. In [36], the issue of applying ALR to a normal Ethernet to reduce energy consumption has been studied where link rates are dynamically adjusted to the load to save energy.

2.4. Summary

In this chapter, the literatures of recent studies related to data reliability management are reviewed. First, we reviewed existing reliability researches on different computer storage devices. Second, we reviewed existing data reliability researches with software based approaches, which dominantly provide data reliability assurance by adding data redundancy. Third, to facilitate our research on cost-effective data transfer for data recovery and creation, some existing researches are reviewed.
Chapter 3 Motivating Example and Problem Analysis

Cloud data storage cost and data reliability are two of the major concerns for storing big data in the Cloud. The ultimate goal of this thesis is to find a solution for providing cost-effective data storage while meeting the data reliability requirement throughout the Cloud data lifecycle. Such a goal could substantially benefit data-intensive applications from eliminating the excessive storage cost for data reliability. The research in this thesis is originally motivated by real-world data-intensive applications for pulsar searching in astrophysics, which could process and generate hundreds of terabytes of data. For storing such massive scientific data in the Cloud, several challenges need to be tackled.

In this chapter, we introduce the motivating example of our research as the scenario for problem analysis and point out the challenges that need to be tackled. In Section 3.1, the motivating example of the pulsar searching survey is described by showing a pulsar searching application instance in detail. In Section 3.2, challenges related to the pulsar searching application when migrating into the Cloud are analyzed with four research issues determined in detail. Finally, in Section 3.3, we summarize the works presented in this chapter.

3.1. Motivating Example

The initial idea of this research is motivated by the astrophysics pulsar searching surveys conducted by Swinburne Astrophysics group using the observation data from Parkes Radio Telescope, NSW (http://www.parkes.atnf.csiro.au/), which is one of the most famous radio telescopes in the world. The application for pulsar searching surveys is a typical data and computation-intensive scientific application. It contains complex and time consuming tasks and needs to process hundreds of terabytes of complicated scientific data, which is of typical big data characteristics.
3.1.1. The Pulsar Searching Application Process

Figure 3.1 Pulsar searching workflow
Figure 3.1 shows the process of the pulsar searching application at the high level in the form of a workflow. There are three major parts in the pulsar searching process, which are raw data recording, data preparation and pulsar seeking [83]:

1) **Raw data recording:**

In Parkes Radio Telescope, there are 13 embedded beam receivers for receiving raw signal data from the universe. Raw signal data are recorded at a rate of 1GB per second by the ATNF Parkes Swinburne Recorder (http://astronomy.swin.edu.au/pulsar/?topic=apsr). Depending on different areas in the universe that the scientists want to conduct the pulsar searching survey, one observation duration is currently from 4 minutes to one hour [83]. The raw signal data are pre-processed by a local cluster at Parkes in real time and archived in tapes for future analysis. These data are then delivered by post to the Swinburne Center for Astrophysics and Supercomputing located in Melbourne.

2) **Data preparation:**

At the beginning of the pulsar searching application workflow, different beam files are extracted from the raw data files and compressed. The outcomes are 13 extracted and compressed beam files. Each of the files is normally 1~20GB in size depending on the observation duration. The beam files contain the pulsar signals which are dispersed by the interstellar medium. Therefore, a de-disperse step needs to be conducted to counteract the effects. In the de-dispersion process, a large number of de-dispersion files are generated with different dispersion trials. For one dispersion trial of one beam file, the size of de-dispersion file is approximately 4.6~80MB depending on the size of the input beam file (1~20GB). A minimum of 1200 dispersion trials are conducted in parallel for each beam file, which in total take 1 to 13 hours to finish and generate around 5~90GB of de-dispersion files. Next, for binary pulsar searching specifically, every de-dispersion file needs a further accelerate step for processing. This step generates the accelerated de-dispersion files of the similar size compared with the de-disperse files.

3) **Pulsar seeking:**

Based on the de-dispersion files, different seeking algorithms can be applied to search pulsar candidates, such as FFT (Fast Fourier Transform) Seeking, FFA (Fast Fold Algorithm) Seeking, Single Pulse Seeking, etc. Take the FFT Seeking algorithm as an example, it takes
7~80 minutes to process the 5~90GB of de-dispersion files, and the outcome of each pulsar seeking algorithm is a seek result file, which is normally 16KB in size. After that, the candidate lists of pulsars in ‘txt’ format with the size of 1KB each generated for each beam file in the same time session are compared so that interferences can be detected and eliminated. For the final pulsar candidates, their feature signals are obtained from the corresponding de-dispersion files, which are then folded into XML files (25KB for each pulsar candidate). This step takes up to one hour according to the number of candidates found. Finally, the XML files are visually displayed for making final decisions on whether a pulsar has been found or not.

3.1.2. The Pulsar Searching Application Dataflow

![Dataflow graph of a pulsar searching instance for 8 minutes of observation](image)

In order to facilitate analyzing the data storage of the pulsar searching application, the dataflow of the pulsar searching application also needs to be described. Figure 3.2 shows the
In Figure 3.2, the amount of data involved in each step of the process is clearly indicated:

- First, in the raw data recording step, the telescope real-time raw data stream is downloaded at the speed of 1GB per second for 8 minutes, and hence 480 GB of raw data are recorded.

- Second, in the data preparation step, 13 extracted and compressed beam files are generated. For the eight-minute observation, the size of each beam file is 2.1GB. Hence the total size of the beam files is 27.3 GB. Next, to counteract the dispersion effect by the interstellar medium, the de-dispersion is conducted at a minimum of 1200 different dispersion trials. Each dispersion trial generates a de-dispersion file of 8.7MB. Hence the total size of the dispersion files is at least 135.72GB (15,600 files). Optionally, for binary pulsar searching, the accelerate step generates 5-10 accelerated de-dispersion files (with the same total size of the original de-dispersion file) for each de-dispersion file. Hence the total size of the accelerated di-dispersion files is also at least 135.72GB (78,000-156,000 files).

- Third, in the pulsar seeking step, based on the (accelerated) de-dispersion files, three different seeking algorithms: FFT (Fast Fourier Transform) Seeking, FFA (Fast Fold Algorithm) Seeking and Single Pulse Seeking are applied to search for pulsar candidates. Each algorithm generates one seek result file (16KB) for each de-dispersion/accelerated de-dispersion file. Therefore, in total about 0.7488-7.488 GB pulsar candidates lists (46,800-468,000 files) are generated. Next, by comparing the candidates generated from 13 beam files (13 ‘txt’ files with a total size of 13KB), final 100 pulsar candidates are selected. These candidates are then folded in XML files with their feature signals, so 100 XML files are generated with a total size of 2.5MB.

In summary, despite of the raw observation data stored on tapes, for the pulsar searching application instance without the accelerate step, this eight-minute pulsar searching application instance generates a total of 31,326 files with a size of about 163GB. If the accelerate step is included, this eight-minute pulsar searching application instance generates a total of at least 327,726 files with a size of about 302GB. In addition, each step of the processing takes hours of time, and hence tens of hours (of supercomputer power) are needed.
for the whole instance. On average, a typical pulsar searching application instance generates more than 100,000 files with the size over 230GB.

### 3.1.3. Migrating the Pulsar Searching Application into the Cloud

The pulsar searching application currently runs on the Swinburne high-performance supercomputing facility. Because the supercomputer is a shared facility that cannot offer sufficient storage capacity to hold the accumulated terabytes of data, 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, at least some of these data should ideally be stored for reuse. For example, the de-dispersion files can be reused to apply different seeking algorithms on finding potential pulsar candidates. Such reuse of the de-dispersion files could save hours of time spent on regeneration, which would not only delay the scientists from conducting their experiments, but also incur a large amount of computation overhead.

The Cloud has offered an excellent storage and computing capacity, which from user’s perspective, is unlimited for storing all the data generated during the execution of applications and processing them in a high performance. This feature of the Cloud is very desirable especially by scientific applications with data-intensive characteristics. When migrating the pulsar searching application into the Cloud, by applying the Cloud storage for the pulsar searching data, the storage limitation can be completely eliminated, and much more generated data can be stored for handy reuse.

If we try to execute the pulsar searching application in the Cloud, the cost for uploading the raw telescope data is the same as before, i.e. raw data stored by tapes and sent to a data center via post. However, a problem has emerged, that is the cost of hiring Cloud storage resources for these data could be huge. As we have mentioned earlier in this section, a typical pulsar searching instance generates more than 100,000 files with the size over 230GB (690GB of data is essentially stored in the Cloud by using the conventional 3-replica replication strategy). According to the latest Amazon S3 storage prices, storing 230GB of data using S3 standard storage service in a “US Standard” region costs US$12.65 per month (i.e. $0.055 per GB/month). This storage cost seems to be a small amount. But in order to meet the needs of pulsar searching applications, we often need to store much more data generated by much longer observation, and several hundreds of such application instances may need to be conducted. For a series of observations conducted eight hours a day for 30 days, the size of
generated files could reach 543.6TB (or 1,630.8TB in the Cloud). According to the latest Amazon S3 storage service price, storing these files using the standard storage service costs about US$29,900 per month ($0.055/GB*month), where two thirds of the money are in fact spent on storing data redundancy for providing data reliability assurance. Moreover, as the pulsar searching program continues, the number and size of the generated files become bigger and bigger, hence the cost for storing data redundancy becomes even higher.

3.2. Problem Analysis

From descriptions of the motivating example in Section 3.1, it can be clearly seen that the storage cost for the big scientific data is one of the biggest barriers for migrating the pulsar searching application into the Cloud. Similar situations are commonly seen in data-intensive applications. Therefore, reducing the data storage cost incurred for storing large amount of data could be one of the most important issues that needs to be solved during the development of Cloud computing paradigm. Storing data in the Cloud consumes storage resources, and hence the cost for data storage in the Cloud is inevitable. But there is still room for reducing it. Based on the motivating example, in this section we first investigate the characteristics of current Cloud storage systems and the data-intensive applications that migrate into the Cloud. Afterwards, further analysis for finding a feasible solution is conducted including analyzing data storage schemes and Cloud networks.

3.2.1. Two Major Factors of Cloud Storage Cost

When we look into the pulsar searching example, there are two major factors that could lead to high storage cost.

- First, current Cloud storage systems generally use data replication for data reliability. As mentioned in Chapter 2, Cloud storage systems such as Amazon S3 [9], Google File System [31], Hadoop Distributed File System [15] and Windows Azure [39] all adopt similar multi-replicas data replication strategies (Windows Azure uses both replication and erasure-coding approaches for data storage). In these Cloud storage systems, by default 3 replicas (including the original copy) are stored for all data. Although more than 3 replicas for each piece of data may be generated when needed, the 3-replica mode is the one that is most commonly used. Therefore, we call these similar data replication strategies...
“conventional 3-replica replication strategy”. By using the conventional 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 consumption of a huge amount of storage resources, and users would have to pay for the cost eventually. By applying the conventional 3-replica replication strategy, storing 1TB of data needs 3TB of data space, in which two thirds of the storage cost are spent for storing data redundancy. For the storage of big data, take the pulsar searching application as an example, the extra money spent would be huge.

- Second, according to the importance and storage duration, the data generated in a pulsar searching application instance can be divided into two major types. One type of data is critical and would 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 application. The extracted beam files record the current state of the universe, which is very important and can be used for long term analysis. The scientists can reuse the extracted beam files for other researches, and reuse the XML files to conduct further analysis to the pulsars. In addition, the de-dispersion files are frequently used generated data [84]. Based on these de-dispersion files, different seeking algorithms can be applied to conduct further pulsar searching activities. 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. Because of the short storage duration of these data, according to the data reliability model that will be described in Chapter 4, one replica would suffice to meet the requirements of data reliability and storage duration. For this type of data, relatively low reliability assurance can be applied and recovery ability is most likely unnecessary. However, by applying the conventional 3-replica strategy, these data are stored with the same number of replicas, which is inappropriate for both data types. For the former type of data, the data reliability assurance by using three replicas incurs a high storage cost especially when large amounts of data are stored. For the latter type of data, the additional two replicas could be simply unneeded, thus incurring unnecessary extra storage cost.

In order to reduce the Cloud storage cost while meeting the data reliability requirement, both abovementioned major factors must be considered. A new data storage as well as data
reliability assurance mechanism should be proposed to replace the conventional 3-replica replication strategy.

3.2.2. Data Storage Devices and Schemes

In current Cloud, disk is the primary storage device for data storage, where a minor proportion of other storage devices are also applied. In Section 2.1, we presented some researches for storage devices such as magnetic tape and solid-state drive, where features of these storage devices were briefly introduced. From the perspective of data reliability management, the primary difference among these storage devices is the failure rate pattern. For example, compared to the disk failure rate pattern, the failure rate pattern of magnetic tapes could have a similar shape but a much slower transform process, whilst the failure rate pattern of solid-state drives could be much more different. In this thesis, the research is conducted primarily based on a Cloud storage environment by using disks. However, by involving a variable failure rate pattern into the data reliability model, providing data reliability assurance by using different storage devices could also be addressed. To facilitate the presentation, we use ‘disk’ for describing all kinds of storage devices in the rest of the thesis.

In addition to analyzing storage devices in the Cloud, the research on Cloud storage and data reliability assurance issues also requires the storage scheme of the Cloud be determined. As mentioned in Section 2.2, there are two major data storage schemes in existing distributed storage systems, which are the replication-based data storage scheme and the erasure coding-based data storage scheme. Instead of using the erasure coding-based data storage scheme, our research still focuses on Cloud with a direct replication-based data storage scheme. The reason for this is twofold:

- First, for pulsar searching and a wide range of similar data-intensive applications that involve intensive large scale data processing and generation, applying erasure coding approaches that are currently used in some of the Cloud storage systems is not practical. For these applications, the term data-intensive does not only mean the requirement of big data storage ability, but also means the requirement of processing data in high performance with low data access delay. In an erasure coding-based data storage environment, the computation and time overheads for coding and decoding the data are so high that the overall cost saving effort in reducing storage cost is significantly weakened.
• Second, the replication-based data storage scheme is currently the most widely used Cloud storage scheme, which is applied by the major Cloud service providers. By conducting research on the Cloud with replication-based data storage scheme, our research could benefit the most for data-intensive applications in the Cloud.

Although Cloud with replication-based data storage scheme is the premise of our research, our data storage and data reliability assurance solution is also applicable for the Cloud with erasure coding-based data storage scheme. We will discuss about it later in Chapter 8 as part of our future work.

3.2.3. Cloud Network and Data Transfer Activities

In the lifecycle of Cloud data, creation and recovery of data are mainly to transfer the replicas within the scale of the Cloud network and store them in the appropriate location. For facilitating our research on maintaining the Cloud data, characteristics of Cloud network and data transfer activities within the Cloud network need to be analyzed. Specifically, for investigating the big data storage in the Cloud, analyses in this area are considered to be important.

• First, for the Cloud network, in Section 2.3, we mentioned that Cloud systems (public Clouds specifically) are primarily running based on data centers with dedicated networks, interconnections with dedicated links. Although bandwidth reservation has not been commonly utilized in current Cloud networks yet, the dedicated feature of Cloud networks, however, makes the bandwidth reservation feasible to be implemented, where the high performance and fully controllable features of bandwidth reserved networks are highly desirable. With the advantages of bandwidth reservation, challenges that commonly exist in Clouds, such as performance prediction of data transfer services and availability of data, can be tackled effectively. Currently, many high-end network routing devices contain bandwidth control and reservation features, hence implementing bandwidth reservation in the Cloud is practical. Therefore, our research assumes a Cloud where bandwidth reservation is enabled on its dedicated Cloud networks. According to the above analysis, we consider the assumption of bandwidth reservation ability to be reasonable.

• Second, data transfer activities in the Cloud network have two major purposes which are data maintenance and data processing. Meanwhile, a Cloud may be composed of several
data centers, and hence the data transfer in the Cloud is conducted both within a data center and on the links between data centers for data center interconnection (DCI). In order to transfer data within the entire Cloud network while meeting the demands of data transfer with different purposes, all these different types of data transfer activities need to be investigated.

1) Case for data maintenance within the data center:

When data are generated, replicas of the data are transferred to appropriate storage devices within the data center [15], [31]. Similarly, when a storage device fails, replicas are recovered via copying other replicas and transfer them to new storage devices. For these data maintenance activities, data transfer does not need to be conducted at the highest speed so that other applications can access the data promptly if needed. Instead, there usually exists a time limit as the upper bound for the duration of data transfer. Failing to complete the data transfer within this time limit could cause problems, for example, violation of service level agreements [50]. Meanwhile, this type of data transfer should not occupy much of the bandwidth over the link as other data transfers with a higher transfer speed requirement may be happening at the same time. We refer to this type of data transfer as ‘lazy’ data transfer.

2) Case for data processing within the data center:

Many data-intensive Cloud applications need to access large amounts of distributed data to conduct data processing tasks. When transferring large amounts of data, these applications often demand a high data transfer speed. Data transfer could be the major factor influencing the performance of the application. The time spent on shifting these data needs to be as short as possible so that the calculation process of the application is not impacted. In contrast to ‘lazy’ data transfer, we refer to this type of data transfer as ‘eager’ data transfer.

3) Case for across data centers with DCI activities:

---

2 Here we do not mean that all data transfer for data maintenance is of a ‘lazy’ fashion, and all data transfer for data processing is of an ‘eager’ manner. Some of the data maintenance activities (e.g. many data creation activities, recovering the data that needs to be processed, etc.) may also need to be conducted in an ‘eager’ fashion. Actually, whether the data is transferred in a ‘lazy’ manner is not determined by the purpose of the data but the time constraint of the data transfer task and the speed limit of the data transfer.
In spite of data transfer for replica maintenance and data processing in a single data center, there are also large amounts of data transfer activities with the same purposes conducted between data centers. Although conducted for the same purposes, these DCI data transfers are quite different when compared to those conducted within a data center. Most of such data transfer activities between data centers are dominated by the ‘lazy’ style, non-interactive bulk data transfer. These data can range in size from several terabytes to petabytes [55]. Due to the large data size, both the transfer speed and energy consumption need to be fully considered. According to [59], the shutdown approaches are often used for this type of data transfer to reduce energy consumption of network devices.

3.2.4. Research Issues

The above analysis consists of three parts. First, we analyzed the two major factors that have led to excessive storage cost for data-intensive applications in current Cloud. Second, we determined on which data storage scheme that our research should be conducted, so that our motivating example and data-intensive applications of similar nature could benefit the most. Third, we also analyzed characteristics of the Cloud network and data transfer activities within the Cloud network. Based on the analysis, our research issues are further determined with details below.

1) Data reliability model:

First of all, we need a model to describe Cloud data reliability and Cloud data reliability related factors, which is essential for the design of data reliability assurance approach in the Cloud. The data reliability model should be able to describe the reliability of the Cloud data throughout their lifecycles, in which they are stored with different redundancy levels and stored on different storage devices with different failure rate patterns in different stages respectively.

In order to facilitate our research, our data reliability model should be consistent with the analysis conducted above as well as literature reviews conducted in Chapter 2. Therefore, first, from the hardware aspect, our data reliability model should be able to precisely describe the relationship between data reliability and failure pattern of storage devices. As we have mentioned in Section 2.1, storage device failure is the source of storage failure and data loss. Precise description of the impact of storage devices to data reliability could substantially
improve the ability of the model to predict data reliability, i.e. the data loss rate, after the data are stored for a certain period of time. Second, the data reliability model must be able to describe the reliability of Cloud data stored in the form of replicas. The number of replicas represents the redundancy level of the data. In the data reliability model, the relationship between data reliability level and the number of replicas needs to be reflected. Third, in order to describe the reliability of Cloud data throughout their lifecycles, the model must be able to reflect the changes in replica number, i.e. data redundancy level, so as to correspond to the lifecycle stages of data creation, data maintenance and data recovery.

2) Minimum replication calculation and benchmark:

When metadata such as data size, expected data storage duration, data reliability requirement are collected and the corresponding storage device is determined, the interface between the Cloud and the storage user, if necessary, needs to determine the minimum replica number that is needed for the purpose of creating data replicas. The calculation should be fast and of low overhead. Moreover, in order to facilitate the data maintenance mechanism, it is necessary that the minimum replication calculation approach also predicts the reliability of the data that are stored for a certain period of time. However, with a variable disk failure rate pattern, the overhead of such a calculation could be a concern, and hence optimization needs to be conducted to reduce the overhead of the data reliability prediction process.

3) Cost-effective data reliability assurance mechanism:

For the maintenance of the Cloud data throughout the Cloud data lifecycle, we need to design a data reliability assurance mechanism that could replace the conventional 3-replica data storage strategy in current Clouds. There are three major challenges as follows for the design of a cost-effective data reliability assurance mechanism in the Cloud.

- First, the mechanism should be running in a cost-effective fashion so that the Cloud data storage cost can be reduced. This does require not only the reduction on replica number, but also the overhead incurred for conducting the mechanism to be considered.

- Second, the mechanism should be able to effectively utilize the computation and storage power of the Cloud, so that the big data in the Cloud could be managed properly.
• Third, according to the previous problem analysis, it can be seen that the data reliability assurance mechanism is the core of the whole solution. Therefore, it needs to be designed to be able to coordinate with the data creation and data recovery processes effectively.

Only after all the three challenges of the mechanism are solved, the goal of cost-effective data storage and data reliability assurance can be achieved.

4) Cost-effective data transfer strategy for data creation and data recovery:

In order to transfer the big data in the Cloud in a cost-effective fashion, the overhead, i.e. the energy consumption incurred during the data transfer process, needs to be reduced. Such optimization could benefit our data storage solution in both the data creation stage and the data recovery stage. According to the analysis in Section 3.2 for Cloud network characteristics and data transfer activities in the Cloud, there are two other issues that also need to be fully considered during the design of the data transfer strategy.

• First, for data transfer in the Cloud, the time constraint of data transfer tasks (i.e. the data transfer deadline) must not be jeopardized so as to suit the high demand of time accuracy in a dedicated Cloud network as well as ensure that the data reliability requirement of Cloud data can always be met.

• Second, due to the “lazy” nature of data transfer for maintaining the Cloud data, our optimization focuses on the ‘lazy’ data transfer type to reduce the data transfer energy consumption. Meanwhile, negative impact of the optimization towards other types of data transfer activities should be avoided. For example, the maximum speed data transfer requirement of ‘eager’ data transfer should not be jeopardized, and the existing link traffic should not be affected, etc.

3.3. Summary

In this Chapter, we presented the motivating example of our research and analyzed characteristics of the motivating example, the Cloud storage and the Cloud network respectively to further illustrate the details of our research issues. Based on the problem analysis in this Chapter, our cost-effective data storage and data reliability assurance issues are
finally identified. In the rest of the thesis, the solutions for four research issues identified will be presented one by one from Chapter 4 to Chapter 7 respectively.
Chapter 4  Generic Data Reliability Model in the Cloud

In this chapter, we present our replication-based data reliability model in detail. In this model, the relationships among reliability of Cloud data, storage duration, storage device failure pattern and replication level are well described. By conducting further investigation on properties of the data reliability model and describing detailed derivation of the generic data reliability model step by step, this chapter clearly presents the solid theoretical foundation of our research, which can also be easily understood by the readers.

The structure of this chapter is organized as follows. In Section 4.1, further analysis on properties of the data reliability model is conducted, in which several aspects of the model, such as properties of the model and the disk failure rate pattern are determined. In Section 4.2, the detailed derivation of the generic data reliability model is presented step by step, in which data reliability with static disk failure rate, data reliability of a single replica with variable disk failure rate and finally the generic data reliability model for multiple replicas with variable disk failure rate are described respectively. Finally, in Section 4.3, we summarize the works presented in this chapter.

4.1. Properties of the Data Reliability Model

In order to design a data reliability model with all the requirements listed in Chapters 1 and 3 being met, further analyses on more detailed properties of the data reliability model need to be conducted. Therefore, prior to the presentation of our generic data reliability model, in this section we determine the reliability metrics that we use, the model type that we use for the model design as well as the failure rate pattern of storage devices that are applied for describing Cloud data reliability and storage devices’ reliability respectively, and explain the reason why these specific properties are selected.
4.1.1. Reliability Metrics

As mentioned in Section 2.1, there are two fundamental disk reliability metrics that are currently used for describing the permanent disk failure rates, which are the Mean Time to Failure (MTTF) and Annualized Failure Rate (AFR). In this thesis, we apply the AFR as the disk reliability metric to our research due to the following two reasons.

- First, AFR is easier to be understood by non-expert readers. The representation of MTTF is by time, which is calculated according to the equation 
  \[ MTTF = \frac{\text{TestDiskNumber} \times \text{TestHours}}{\text{DiskFailures}}. \]
  For example, a disk manufacturer tested a sample of 1,000 disks for a period of 1,000 hours (i.e., 41.5 days) and within that period of time one disk failure occurred. According to the equation, the MTTF value is 1,000,000 hours. From the reader’s point of view, the MTTF value that equals to 114 years would be hard to understand because no single disk could survive for that long. In contrast, the representation of AFR is by percentage, which indicates the expected probability of disk failure occurrence during one year of usage. For the MTTF value of 1,000,000 hours, according to Equation (2.1) in Section 2.1, the equivalent AFR value is 0.87%, meaning that 0.87% of all the disks are expected to fail during one year of usage. Compared with MTTF, the advantage of AFR on readability can be easily seen.

- Second, as mentioned in Section 2.1, MTTF is obtained in the industrial test by running many disks for a specific period of time. On the contrary, AFR is obtained from the real scenario by checking the running history of disks in the system via system logs. Therefore, the AFR value could better reflect the actual reliability level of disks in a real storage system. In addition, much existing research conducted by industry researchers applies AFR for disk reliability evaluation. In this thesis, results from existing industrial research are well investigated and applied in our evaluation as well.

Based on the AFR disk reliability metric, the data reliability is presented in the similar style. In our novel reliability model, the data reliability is described in the form of annual survival rate, which indicates the proportion of the data that survives during one year storage.
4.1.2. Data Reliability Model Type

As mentioned in Section 2.2, two types of data reliability model have been spotted among all the literatures reviewed, which are based on simple permutations and combinations and complicated Markov chains, respectively. In this thesis, we apply the former to our novel data reliability model due to the following two reasons.

- First, by using the design based on simple permutations and combinations, the variable disk failure rate can be added into the model relatively easily compared with the Markov chain type. In existing Markov chain reliability models that we have reviewed, the disk failure rates are all considered as a constant [23] [53] [65]. The complexity of the models could be one of the major reasons for this. In order to solve the extremely complicated functions of the Markov chain reliability model, many complex matrix operations are involved, which could incur large computing overhead. Although we have not tested the complexity of solving a Markov chain reliability model with variable failure rates, we can foresee that the complexity of solving it could be substantially increased, which is not desirable for our data reliability assurance mechanism.

- Second, in our research we pursue reduction on the number of replicas stored for the Cloud data. As will be mentioned later, in our data reliability assurance mechanism, we only store no more than 2 replicas for each piece of Cloud data. Therefore, the data reliability model based on simple permutations and combinations is sufficient for doing the job\(^3\), while building the complicated state diagram of the Markov chain reliability model for analyzing a very high data redundancy level becomes unnecessary.

4.1.3. Failure Rate Pattern of Storage Devices

As mentioned in Section 2.1, there are three different styles of failure rate patterns that have been applied in existing researches and industry standards, which are (1) the constant failure rate applied in reliability models strictly following exponential distribution, (2) the continuous variable failure rate pattern applied in ‘bathtub’ (and extended models based on the ‘bathtub’ theory) reliability models, and (3) the discrete failure rate pattern applied in several researches and industry standards. In this thesis, we describe the disk failure rate pattern with

\(^3\) In fact, as will be explained later, our novel data reliability model is also able to describe the reliability of data with more replicas.
discrete failure rates, which divides the lifespan of disks into discrete life stages with discrete disk failure rates. By using the discrete disk failure rate pattern, the constant disk failure rates in different disk life stages and the trend of changes in disk reliability are well combined. Compare to the continuous variable failure rate pattern, the discrete disk failure rate pattern could greatly simplify the computational complexity of the reliability model, and hence reduce the computing overhead for calculating parameters in the data reliability model. In addition, by using the discrete disk failure rate pattern, we could also apply existing research results in industry to our research. In Chapter 6, we conduct the evaluation for our research based on the discrete disk failure rates provided by IDEMA standards and Google’s nine-month disk failure trend study.

4.2. Generic Data Reliability Model

In this section we describe our generic data reliability model. The relationship between data reliability and the variable disk failure rate is demonstrated by presenting data reliability with static disk failure rate and data reliability of a single replica with variable disk failure rate respectively. Finally, a generic data reliability model is presented in detail. For easy description, we use the term “data file” to represent a Cloud data storage unit, where actually any types of data storage unit, such as data object and data block, are also applicable.

4.2.1. Data Reliability with Static Disk Failure Rate

For many existing theories that assume disk reliability following the exponential distribution, the failure rate of each disk is a constant. In that case, the reliability of a disk over period $T$ can be expressed as Equation (4.1):

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

(4.1)

In this equation, $R(T)$ is the function of the disk reliability over period $T$. $\lambda$ is the disk failure rate. The replicas stored in the disk should have the same reliability as the disk. In other words, if a data center experiences 100 disk failures from 10000 disks for a year, the average disk failure rate is 1% per year, and thus the reliability of each replica stored in the data center should be 99% per year. Therefore, Equation (4.1) is still applicable for calculating the reliability of a single replica when the disk failure rate is a constant.
4.2.2. Data Reliability with Variable Disk Failure Rate

From the previous discussion, it can be seen that exponential distribution is able to describe data reliability when disk failure rate is a constant. However, as mentioned in Section 2.1, the failure rates of disks in reality vary from time to time. In practice, quality control is conducted for each batch of disks before they leave the factory. Hence we consider the failure 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. Hence we assume that each disk’s failure rate pattern 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, based on the discussion earlier, we first assume that when the disk failure rate is a constant, data reliability follows exponential distribution (i.e. Equation (4.1) holds). Second, when the disk failure rate is a variable, by using the discrete disk failure rate pattern, we assume the disk failure rate pattern contains several life stages of disks. In each life stage of a disk, the disk failure rate does not change.

![Figure 4.1 Failure rate pattern of disk D between time t0 and tn](image)

Assume that replica r is stored in disk D between t0 and tn, 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 \( \lambda_i \) indicates the disk failure rate between time \( t_{i-1} \) and \( t_i \), \( i \in N \).

Figure 4.1 shows the failure rate pattern of disk \( D \) between time \( t_0 \) and \( t_n \).

We derive the data reliability of a single replica with a variable disk failure rate below:

\[
R(T) = e^{-\lambda T}
\]  

(4.2)

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), and \( T = \sum_{j=1}^{n} T_j \) is the sum of all storage durations, which is the lifespan of the data file. The derivation of Equation (4.2) is presented as follows.

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:

\[
P(A_n A_{n-1} ... A_1) = P(A_n | A_{n-1} ... A_1) P(A_{n-1} A_{n-2} ... A_1) \\
= ... = P(A_n | A_{n-1} ... A_1) P(A_{n-1} | A_{n-2} ... A_1) ... \\
= P(A_2 | A_1) P(A_1)
\]

where \( P(A_j | A_{j-1} A_{j-2} ... A_1) \) 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 \( D \), \( P(A_j | A_{j-1} A_{j-2} ... A_1) = R_j \) where \( R_j \) is the reliability of the data file stored from \( t_{j-1} \) to \( t_j \). Therefore, we have

\[
P(A_n A_{n-1} ... A_1) = R_1 R_2 ... R_n
\]

According to Equation (4.1), we have

\[
R_j = e^{-\lambda_j (t_j - t_{j-1})}
\]

Let \( T_j = t_j - t_{j-1} \), hence we have:
\[ P(A_n A_{n+1} \ldots A_k) = e^{-\lambda T} e^{-\lambda T} \ldots e^{-\lambda T} \]
\[ = \exp((-\sum_{j=1}^{n} \lambda_j T_j / \sum_{j=1}^{n} T_j) \sum_{j=1}^{n} T_j) \]

Because \( P(A_n A_{n+1} \ldots A_k) = R(T) \), the above equation can be denoted as:

\[ R(T) = e^{-2\bar{T}} \]

where \( \bar{T} = \sum_{j=1}^{n} \lambda_j T_j / \sum_{j=1}^{n} T_j \), and \( T = \sum_{j=1}^{n} T_j \).

From Equation (4.2), it can be seen that the data reliability of one replica 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 lifespan. Therefore, Equation (4.1) can be considered as a special case of Equation (4.2) when the disk failure rate is a constant.

### 4.2.3. Generic Data Reliability Model for Multi-replicas

In previous sub-sections we discussed the data reliability of storing one replica. Based on the discussions above, the novel generic data reliability model with a variable disk failure rate for multiple replicas is proposed. In this model, due to the assumption that each disk could theoretically have its own failure rate pattern, we assume that the disk failures are independent. Assume that replicas of the same data file are stored in different disks. The data reliability with multiple replicas can be derived from Equation (4.3):

\[ R(T_k) = 1 - \prod_{i=1}^{k} (1 - e^{-\bar{\lambda} r_i}) \]  

(4.3)

In Equation (4.3), the data reliability level with multiple replicas is described based on permutations and combinations principles. In this equation, \( k \) represents the number of replicas, \( \bar{\lambda_i} \) is the weighted average failure rate of the disk storing replica \( r_i \), \( T_k \) is the storage duration of the data file 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 (4.3) reveals the relationship among data reliability level, the number of replicas, failure rates of disks and the storage duration of the data file. 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 Equation (4.2) is a special case of Equation (4.3) when $k=1$.

4.3. Summary

In this chapter, we first determined several properties in our data reliability model, which includes the reliability metrics used for describing disk and data reliability levels, the model type used in our data reliability model design and the style of the disk failure rate pattern used for describing the disk reliability. Afterwards, we presented our novel generic data reliability model in detail. Based on this model, the relationship among data reliability levels, the number of replicas, failure rates of disks and the storage durations of the data are able to be well described, where calculations among these parameters are defined.
Chapter 5  Minimum Replication for Meeting the Data Reliability Requirement

In this chapter we present the approach for calculating the minimum replication for meeting the data reliability requirement. Essentially, based on the generic data reliability model, this approach provides a practical and efficient calculation method with given parameters. When the data reliability requirement is determined and the expected storage duration is provided, this approach could quickly calculate the minimum replicas that are needed as well as predict the longest storage duration of the data file for meeting the data reliability requirement. These outcomes of our approach are the key for our data reliability assurance solution, based on which the whole series of approaches during different data lifecycle stages can be conducted. In addition, as a direct consequence, the minimum replication can also act as a benchmark, which can be used for evaluating cost-effectiveness for data reliability assurance of various replication-based data storage approaches.

The structure of this chapter is organized as follows. In Section 5.1, details of the minimum replication calculation approach are presented. In Section 5.2, we discuss about the application of the minimum replication benchmark for evaluation of replication-based data storage approaches. In Section 5.3, the outcomes of the evaluation for the minimum replication calculation approach are briefly presented. Finally, in Section 5.4, we summarize the works presented in this chapter.

5.1. The Minimum Replication Calculation Approach

As mentioned above, our minimum replication calculation approach has two purposes. First, it determines the minimum replica number for ensuring the data reliability requirement. Second, given a certain data reliability requirement, it predicts the longest storage duration of the data file while the data reliability requirement is met. By solving our generic data
reliability model presented in Chapter 4, the longest storage duration of Cloud data files with any number of replicas can be predicted. However, considering that no more than two replicas for each data file are needed in our data storage solution. Therefore, in this section, we only present the investigations conducted for the Cloud data files stored with a single replica or two replicas.

5.1.1. Minimum Replication Calculation Formulas

In a commercial storage system such as that of the Cloud, “data reliability” has two aspects of meaning, which are the data reliability requirement $RR(t)$ and the data reliability assurance $RA(t)$. $RR(t)$ indicates the data reliability that storage users wish to achieve within the storage duration of $t$, while $RA(t)$ indicates the data reliability that the system is able to provide within the storage duration of $t$. As we use $AFR$ to describe disk reliability and annual survival rate to describe data reliability, $RR(t)$ is provided under unit time (i.e. $RR(1)$). Meanwhile, $RA(t)$ 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 amount of Cloud data survive the whole lifecycle must not be lower than that of the user’s expectation.

**Rule 2:** The data reliability assurance follows the generic data reliability model.

$$RA(t) = 1 - \prod_{i=1}^{t} (1 - e^{-\lambda_i t})$$  \hspace{1cm} (5.1)

According to Rule 1, the average data loss rate during the storage lifespan cannot be bigger than that of user’s expectation, which is:

$$1 - RA(lifespan) \leq 1 - RR(lifespan)$$

From the user’s perspective, the data reliability requirement is considered to indicate a data loss process with constant rate. However, from the storage provider’s prospective, the average data loss rate is achieved from a storage lifespan that consists a bunch of separate storage periods. At each of the periods, replicas of the Cloud data files are stored in different storage devices with different failure rates and different redundancy level, and hence the data
reliability assurance provided could be different. Therefore, the above inequation can be transformed into the following form:

\[ \sum_{i=1}^{k} SD_k(1 - RA_k(1)) \leq lifespan(1 - RR(1)), \text{ where } \sum_{i=1}^{k} SD_k = lifespan \]

For providing the lowest data reliability assurance that is needed, we have:

\[ \sum_{i=1}^{k} SD_k(1 - RA_k(1)) = lifespan(1 - RR(1)), \text{ where } \sum_{i=1}^{k} SD_k = lifespan \] (5.2)

Equation (5.2) reveals the actual data storage process that is conducted in the storage system, where each storage duration \( SD_k \) indicates a period that the data redundancy level remains the same, and \( RA_k(1) \) indicates the data reliability assurance that should be provided during that period. The value of \( RA_k(1) \) could be different from \( RR(1) \) to tolerate changes in the data redundancy level. Considering the data storage with no more than two replicas, during the storage duration with two replicas, \( RA_k(1) \) should be bigger than \( RR(1) \). Only that, when the data redundancy is low (i.e. one replica), there is a time window for the redundancy to be increased (i.e. create new replica or recover the lost replica) and the data storage with one replica does not jeopardize the data reliability requirement in overall terms. As we only present the approach for calculating the minimum replication approach in this chapter, we put the discussion on determining the data reliability assurance \( RA_k(1) \) in Chapter 6. In each storage duration \( SD_k \), the data reliability assurance should always be no smaller than \( RA_k(1) \). Then we have:

\[ 1 - RA_k(1) / 1 \geq 1 - RA(t) / t \]

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

\[ RA(t) \geq 1 - (1 - RA_k(1))t \] (5.3)

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

With single replica, \( RA(t) = e^{-2at} \) (5.4)
With two replicas, \[ RA(t) = 1 - (1 - e^{-\frac{t}{\lambda_d}})(1 - e^{-\frac{t}{\lambda_f}}) \] (5.5)

Inequation (5.3) is the key for building the relationship between the data reliability requirement and the storage duration of data files. After combining Inequation (5.3) and Equation (5.4) above, we have:

\[
RA_k(1) \leq \frac{(e^{-\frac{t}{\lambda_d}} + t - 1)}{t}
\]

For the storage with one replica, because the redundancy level cannot be changed, we have \( RA_k(1) = RR(1) \). Therefore:

\[
RR(1) \leq \frac{(e^{-\frac{t}{\lambda_d}} + t - 1)}{t}
\] (5.6)

Inequation (5.6) shows the relationship between data reliability requirement and the storage duration of data file with single replica. Assume that \( t \) be the expected storage duration of the data file, \( \lambda_1 \) be the average disk failure rate of the corresponding disk storing the replica, if Inequation (5.6) holds, then a single replica suffices to provide data reliability assurance that meets the data reliability requirement. Otherwise, if this inequation does not hold, the storage with single replica may jeopardize the data reliability requirement, and hence creating another replica is necessary.

After combining Inequation (5.3) and Equation (5.5), we have:

\[
RA_k(1) \leq 1 - (1 - e^{-\frac{t}{\lambda_d}})(1 - e^{-\frac{t}{\lambda_f}}) / t
\]

This inequation shows the relationship between the data reliability requirement and the storage duration of data file with two replicas. The right hand side of the inequation is a monotonically decreasing function of \( t \). Therefore, while \( RA_k(1) \) is determined, variable \( t \) cannot exceed a certain value. When the longest storage duration of the data file is reached, the right hand side of the inequation equals to \( RA_k(1) \), which is

\[
RA_k(1) = 1 - (1 - e^{-\frac{t}{\lambda_d}})(1 - e^{-\frac{t}{\lambda_f}}) / t
\] (5.7)

Therefore, by solving Equation (5.7), the longest storage duration of the data file can be obtained.
In general, our minimum replication calculation approach determines the minimum replication for meeting the data reliability requirement based on Inequation (5.6) and Equation (5.7). By using Inequation (5.6), we are able to justify whether the storage with one replica suffices to meet the data reliability requirement, and hence the minimum replica number (i.e., either one replica or two replicas) could be determined. Given \( \lambda_1, \lambda_2 \) as the average failure rates of the corresponding disks, by solving Equation (5.7), the longest storage duration of the data file while meeting \( R_{A_t}(1) \) can be predicted.

5.1.2. Optimization of the Minimum Replication Calculation Formulas

Inequation (5.6) and Equation (5.7) are the keys for calculating the minimum replication for meeting the data reliability requirement. However, it could be difficult to solve Equation (5.7) in its current form due to two reasons:

- First, due to the variable nature of the average disk failure rate, it changes along with the storage duration and the exact age of the disk. Therefore, \( \lambda \) becomes a function of variable \( t \). Considering this factor, the solving process of Equation (5.7) becomes very complicated.

- Second, as a direct consequence of the variable average disk failure rate, the longest storage duration of the data file changes from time to time. Therefore, every time when the longest storage duration of the data file is needed, the process of solving Equation (5.7) needs to be conducted again. As will be mentioned later, in our data reliability assurance solution the longest storage duration of the data file could be used many times throughout the lifespan of each data file. Therefore, the overhead for solving Equation (5.7) could be very big.

In general, solving complicated Equation (5.7) could be a time consuming and expensive process. In particular, the involvement of function \( \lambda(t) \) and calculation of the longest storage duration of data files for more than once have made the problem even worse. To

---

4 We do not predict the longest storage duration of data file with a single replica. If Inequation (5.6) does not hold, it means that one replica simply cannot provide satisfactory data reliability assurance for any time; and if Inequation (5.6) holds, it means that one replica is sufficient for storing the data file for the expected storage duration, and hence the prediction is not needed.
address this issue, optimizations need to be conducted for the minimum replication calculation approach.

In order to simplify the computation of Equation (5.7), our solution contains two major steps:

- First, based on the discrete disk failure rate pattern that is applied in our generic data reliability model, the average disk failure rate can be converted into a piecewise function \( \bar{\lambda}(t) \) of storage duration. According to the disk failure rate pattern of the disk 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 different disk failure rates contained in the disk failure rate pattern after the start time. By doing this, Equation (5.7) is transformed into an equation in which \( t \) is the only independent variable, with variable \( \bar{\lambda} \) being eliminated.

- Second, after the first conversion of Equation (5.7), the previous equation has now been converted into a piecewise function, which equals to several functions, each covering a specific period of storage duration. Due to the increment in the number of equations that need to be solved to obtain the longest storage duration value, the solving process is still time consuming and expensive in terms of overhead. To optimize the performance of the solving process, the data reliability equation is further simplified for reducing the computation complexity. 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 in 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. Assuming that the function of the substituted straight line is \( f(\bar{\lambda}t) = a\bar{\lambda}t + b \), Equation (5.7) can be simplified into Equation (5.8):

\[
RA_x(1) = 1 - (1 - a\bar{\lambda}t - b)(1 - a\bar{\lambda}t - b) / t
\]  

(5.8)

As the average disk failure rate can be expressed as a first degree piecewise function of \( t \), Equation (5.8) is essentially a quartic function of \( t \). Compared to many complicated equation solving methods such as trust-region equation solving algorithms [56], etc., for solving the original non-polynomial Equation (5.7), the simplified Equation (5.8) can be solved by the
methods for solving polynomial equations, which are much more efficient, and hence the calculation overhead can be significantly reduced.

In addition to the simplification described above, addressing the issue of solving the equation for multiple times, optimizations are also conducted. In order to avoid any excessive overhead incurred for solving Equation (5.8) for multiple times, the multiple calculations are conducted in one go when the data file is first created in the Cloud. As long as replicas of the data file are not lost, the solving process does not need to be conducted again, and hence resulting in better efficiency.

In Chapter 6, the minimum replication calculation approach is applied for our generic data reliability assurance mechanism where we present the pseudo code of the approach with the mechanism together then.

5.2. Minimum Replication Benchmark

By solving the corresponding inequalities and equations mentioned in Section 5.1, the minimum replication, i.e. the minimum number of replicas required for meeting the data reliability requirement is determined. In addition to finding the minimum number of replicas for data storage in our data reliability assurance solution, the minimum replication could also be used as a benchmark for evaluating different approaches. It shows the theoretical minimum data redundancy level of a replication-based data storage system without jeopardizing the data reliability requirement. By using this benchmark, the cost-effectiveness as well as the ability of providing data reliability assurance of a replication based data storage system can be clearly presented as described next.

Given data file set $F(f_1, f_2, f_3, \ldots, f_m)$ managed by replication based system $S(d_1, d_2, d_3, \ldots, d_n)$ with the data reliability requirement set of $RR(r_1, r_2, r_3, \ldots, r_m)$, where $f_i(r_{i1}, r_{i2}, r_{i3}, \ldots, r_{ip})$ indicates a data file in $F$ and $d_q$ indicates a disk in $S$. $r_{ij}(d_q)$ indicates the $j$th replica of $f_i$ which is stored in disk $d_q$. In order to avoid searching the disks for storing all the replicas of the data file, the disk failure rate patterns are obtained from randomly selected disks. For each, Apply the minimum replication approach for each $f_i$ in $F$, and the minimum replication $min_i$ for each $f_i$, can be obtained. The minimum replication level for storing data file set $F$ can be described as Equation (5.9):
\[ MIN_s = \sum_{i=1}^{m} \min \frac{1}{m} \] (5.9)

When the current replication level in system \( S \) is close to \( MIN_s \), it means that the data stored in the system are maintained cost effectively. However, when the current replication level is lower than \( MIN_s \), it means that the data redundancy level of the system is too low to provide sufficient data reliability assurance, so that the data reliability requirement could be jeopardized.

### 5.3. Evaluation of the Minimum Replication Calculation Approach

In this section, we briefly present the results of our evaluation on the minimum replication calculation approach so as to provide an intuitive understanding of the effectiveness of the approach. The evaluation is conducted by running a minimum replication algorithm. The algorithm is essentially the implementation of the minimum replication approach, which runs as a part of our data reliability assurance mechanism to be presented in Chapter 6. As the minimum replication algorithm is described in Chapter 6, details of the experiments will be presented in Chapter 6 as well.

During the evaluation we evaluate the algorithm under different data reliability requirements and with different configurations including failure rate types and calculation equations. The evaluation is conducted from the aspects of execution time of the algorithm and the accuracy rate of the output of the optimized algorithm compared with the original algorithm (see Section 6.5 for more details). The execution time of the algorithm addresses the computing overhead of the minimum replication calculation approach, whilst the accuracy rate of the algorithm output addresses the effectiveness of our optimization to the minimum replication calculation approach presented in Section 5.1.

From the aspect of execution time, the results show that the time for determining the minimum replica number (i.e. one or two replicas) is less than 1 millisecond. The time for predicting the longest storage duration of the data file for meeting the data reliability requirement is at the magnitude of several milliseconds to tens of milliseconds, which varies according to the data reliability requirement and whether if the optimized formulas are applied. Specifically, when the optimized formulas are applied in the algorithm, the execution time
could be significantly reduced. In general, the results indicate that the minimum replication calculation approach is able to effectively determine the minimum number of replicas for the Cloud data storage as well as predicting the longest storage duration of the data file for meeting the data reliability requirement whilst incurring a relatively small computing overhead in terms of execution time.

From the aspect of accuracy rate of the optimized algorithm output, among different versions of the algorithm, the results show that the output of the optimized algorithm is very close to the original algorithm. The accuracy rate is lower (90%) when the data reliability requirement is low, and quickly increases with the increment of the data reliability requirement (99.9% when the data reliability requirement is 99.999% per year). In general, this result indicates that our optimization to the minimum replication calculation approach could generate results that are of little difference compared to the original one.

5.4. Summary

Based on the generic data reliability model presented in Chapter 4, in this chapter we presented our approach of calculating the minimum replication for meeting a given data reliability requirement. We first presented the formulas for determining the minimum replica number, and also presented our optimization solution for solving the equations so that the prediction process of the longest storage duration of the data file can be conducted efficiently to reduce the overhead for managing the Cloud data. Then, we discussed the issue of using the minimum replication as a benchmark for evaluating the cost-effectiveness and data reliability assurance ability of a replication-based storage system. Finally, we briefly presented the satisfactory outcomes of the evaluation for the minimum replication calculation approach.
Chapter 6  Cost-Effective Data Reliability Assurance for Data Maintenance

In this Chapter, we present our novel cost-effective data reliability assurance mechanism named PRCR (Proactive Replica Checking for Reliability), which is for maintaining the Cloud data in a cost-effective fashion. PRCR has the following features:

- First, by coordinating with the minimum replication calculation approach and data recovery approach, PRCR maintains the Cloud data files with the minimum replication level, in which no more than two replicas are created for each data file.

- Second, by using the abundant Cloud computing resources in the form of Cloud compute instances, PRCR is able to maintain the big data in the Cloud with a huge number of Cloud data files with flexibility, while a wide variety of data reliability assurance can be provided to meet storage user’s reliability requirement.

- Third, by checking the replicas of each data file regularly in a proactive fashion, PRCR is able to detect any replica loss incident and cooperate with the data recovery process. In this way, PRCR makes sure the data reliability assurance is not jeopardized in overall terms.

- Compared with the huge number of Cloud data files that PRCR is able to maintain, the running overhead of PRCR is very small that can be neglected. By using PRCR for the data reliability management, the excessively generated data replicas in current Clouds can be minimized, so that the storage cost could be significantly reduced.

The structure of this chapter is organized as follows. In Section 6.1, we explain how proactive replica checking can be used for providing data reliability assurance. In Section 6.2, we present the high level structure of PRCR. In Section 6.3, more detailed design of PRCR is presented, in which we present the working process of PRCR for maintaining a Cloud data file throughout its lifecycle. In Section 6.4, two algorithms for optimizing PRCR are presented including the minimum replication algorithm for determining the minimum number of replicas.
and the metadata distribution algorithm for maximizing the utilization of the PRCR capacity. In Section 6.5, evaluation for PRCR is presented, in which we evaluate PRCR from aspects of performance and cost-effectiveness. Finally, in Section 6.6 we summarize the works presented in this chapter. This section is mainly based on our work presented in [50].

6.1. Proactive Replica Checking

There is a well-known property of exponential distribution called the memoryless property, which is that for all $s, t > 0$, there are $P(T > s + t | T > s) = P(T > t)$. In other words, for given $T > s$, the probability distribution of $T$ from time $s$ to $s+t$ is equivalent to that from time $0$ to $t$. For data reliability specifically, this property denotes that as long as we know the data file is not lost at any given moment, the probability of the data file survive for the next time $t$ follows the same probability distribution.

In Section 4.2, we illustrated that the data reliability of a single replica with a variable disk failure rate follows exponential distribution, and hence the memory-less property still holds. As the data reliability of each replica is independent, the memoryless property should also hold to our generic data reliability model for multiple replicas. According to this property, the data reliability for any period from any given moment can be calculated. More importantly, according to our generic data reliability model, shorter storage duration results in lower probability of data loss. Thus, the basic idea of managing data reliability based on proactive replica checking can be formed: While a data file is stored in the Cloud, each replica of the data file is checked periodically. The loss of replicas can be discovered and then recovered within each allowed period, and this process is repeated during the storage. By changing the duration of such a period as well as the frequency of proactive replica checking, a range of data reliability assurances can be provided. Based on this idea the PRCR mechanism can be proposed.

By using PRCR, Cloud data files can be managed in different styles according to their expected storage duration and reliability requirements: for data files that are only for short-term storage and/or require the data reliability that a single replica can offer, one replica is sufficient for the data file; for data files that are for long-term use and/or have a data reliability requirement higher than the reliability assurance of a single replica, two replicas are stored while being periodically and proactively checked. During the proactive replica checking,
replicas of the data files are accessed to check their existence\(^5\). The proactive replica checking tasks are always conducted before the reliability assurance drops below the reliability requirement. Any single replica loss can be recovered in time when found, so that the reliability of the data files can be ensured.

In some extreme cases, both replicas may be lost at the same time or within a small time window (i.e. between two successive proactive checking tasks for the data file). The probability of such a situation is already incorporated in the data reliability model. Given a certain data reliability requirement, PRCR is responsible for maintaining the data loss probability within the agreed range. For example, given the data reliability requirement of 99.99% per year, PRCR ensures that the data loss rate is no bigger than 0.01% per year for all the data files, and hence the loss of both replicas does not jeopardize the reliability assurance in overall terms.

### 6.2. Overview of PRCR

PRCR is a data reliability assurance/replica management mechanism designed for managing the big data in the Cloud with a huge number of Cloud data files. It is normally conducted as a data reliability management service provided by the Cloud storage providers. By using PRCR, Cloud data files can be stored with minimum replication while meeting the data reliability requirement.

As shown in Figure 6.1, there are two major parts of PRCR, which are the user interface and the PRCR node. Each of the components is deployed onto a Cloud compute instance. For providing different data reliability assurances and managing the huge number of Cloud data files, the entire PRCR mechanism could be composed of one user interface and multiple PRCR nodes. Specifically, each PRCR node proactively checks data files with a certain frequency, so that different data reliability assurances can be provided by different PRCR nodes, which correspond to \( RA_k(1) \) as demonstrated in Chapter 5. The number of data files that a PRCR node manages is referred to as the capacity of the PRCR node. As the maximum capacity of each PRCR node is limited, each PRCR node could only manage a certain number of data files. Each PRCR node works independently of one another, so that it

---

\(^5\)As the proactive replica checking is conducted within the same Cloud provider, we believe that the instability of the network is minimized. Therefore, the replica is considered to be lost when it cannot be accessed.
can be easily created and removed according to the number of data files that needs to be managed and the data reliability assurance that needs to be provided.

![PRCR structure diagram](image)

**Figure 6.1 PRCR architecture**

### 6.2.1. User Interface

It is the component of PRCR responsible for determining the minimum replica number, creating replicas by coordinating with the data creation service (for storage with two replicas), creating and distributing metadata of data files, etc.

- First, when the original replica of a data file is created (generated or uploaded) in the Cloud, based on the minimum replication approach presented in Chapter 5, the user interface determines the minimum number of replicas (i.e. one or two replicas).

- Second, if a data file needs to be stored with two replicas, the user interface calls Cloud service to create the second replica for the data file, where the data transfer approach that we propose (see Chapter 7) can be applied.

- Third, if a data file is stored with two replicas, the metadata of the data file are created and distributed to the corresponding PRCR node.
For all data 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 addresses.

**File ID**: it is the unique identification of the data file.

**Time stamp**: it records the time when the last proactive replica checking task for the data file was conducted.

**The data reliability requirement and expected storage duration**: these are requirements for the storage qualities.

**Checking interval**: it is the maximum time interval allowed between two consecutive proactive replica checking tasks for the same data file.

**Replica address**: it records the location of each replica.

In order to obtain all these metadata attributes, the file ID and replica addresses are automatically given when the original and second replicas of the data file are created. Time stamp is initialized with the current time and then updated when the proactive replica checking task is conducted. The data reliability requirement and expected storage duration can be given by the storage user and maintained for rebuilding metadata in case of replica loss. The checking interval can be calculated by using the minimum replication calculation approach.

Among these attributes, the data reliability requirement and expected storage duration are the only attributes provided by the storage user (default values may apply if they are not given). Despite of that, all the other storage structure related attributes are transparent to the storage user. The checking interval equals to the longest storage duration of the data file while meeting the data reliability requirement. Therefore, starting from the time that last proactive replica checking task is conducted, within the checking interval period, PRCR must check the data file at least once so as to ensure the data reliability assurance is higher than the data reliability requirement. As mentioned in Section 5.1, because of the variable disk failure rate, the longest storage duration of a Cloud data file varies. Therefore, one or more checking interval values may apply throughout the lifespan of the data file in the Cloud. Depending on the attributes of time stamp and checking interval, PRCR is able to determine the time that each data file needs to be checked. According to the replica addresses, all replicas of the data file can be spotted.
6.2.2. PRCR Node

It is the core component of PRCR responsible for the management of the metadata and replicas of data files. Within each PRCR node, there are two parts: data table and replica management module, which are for maintaining the metadata of data files and conducting the proactive replica checking tasks, respectively.

**Data Table**\(^6\): For all data files that each PRCR node manages, the metadata attributes are maintained in the data table. To ensure the data reliability of data files, all the metadata are periodically scanned by the replica management module. The so called “scan” inspects the metadata of a data file in the data table and determines whether proactive 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 are sequentially scanned once. In order to proactively check all the managed data files in a certain frequency, the time for the scan cycle of each PRCR node is set to a fixed value. By doing so, the scanning frequency of metadata can be determined so that proactive replica checking tasks can be conducted well in time. However, due to the limited performance of the compute instance that the PRCR node is running on, the time constraint of scan cycle means that the maximum capacity of the PRCR node is also limited.

**Replica Management Module**: It is responsible for scanning the metadata in the data table and co-operating with the Cloud compute instances 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 data file needs to be checked. If a data file needs to be checked, the replica management module obtains its metadata from the data table and sends it to a Cloud compute instance 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 files with different usages, contents and formats, different data recovery strategies can be applied to achieve certain goals. As will be presented in Chapter 7, our data transfer approach for data creation and data recovery can be applied to reduce the energy consumption during the data recovery process.

\(^6\)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.
6.3. Working Process of PRCR

By tightly integrating all the functions of PRCR components, a series of activities for maintaining Cloud data files with data reliability assurance is conducted, in which the complex
working process of PRCR is formed. In Figure 6.2, we illustrate this process by following the lifecycle of a data file managed by PRCR in the Cloud.

1) The process starts at the time that the original replica of the data file is created in the Cloud. According to the minimum replication approach, the user interface determines the minimum number of replicas, i.e. to store the data file with one replica or two replicas. Specifically, if one replica cannot satisfy the data reliability and storage duration requirements of the data file, the user interface requests to create a second replica by calling Cloud service (see Chapter 7), and calculates the checking interval values of the data file.

2) According to the checking interval values of the data file, its metadata are distributed to the replica management module of the corresponding PRCR node. Otherwise, when one replica is sufficient to meet the data reliability requirement, only the original replica is stored and the metadata of the data file need not be created.

3) Metadata attributes of the data file are stored into the data table.

4) Metadata are scanned once in each scan cycle of the PRCR node. When the metadata are scanned, PRCR determines whether proactive replica checking is needed according to the time stamp and checking interval of the data file.

5) If proactive replica checking is needed, the replica management module obtains the metadata of the data file from the data table.

6) The replica management module assigns the proactive replica checking task to one of the unoccupied Cloud compute instances that is created in advance. The Cloud compute instance executes the task, in which both replicas of the data file are checked.

7) The Cloud compute instance conducts further action according to the result of the proactive replica checking task: if both replicas are alive (or lost which is very rare, but yet within the data reliability assurance range in overall terms), go to step 8; if only one replica is lost, the data recovery process needs to be initiated, where the compute instance calls the Cloud service (see Chapter 7) to generate a new replica based on the replica that is alive.
8) The Cloud compute instance returns the result of the proactive replica checking task. If both replicas are not lost (or recovered from losing one replica), the time stamp, checking interval and the new replica address (if applicable) of the data file are updated in the data table. Otherwise, a data loss alert will be issued.

**Note:** Steps 4 to 8 form a continuous loop until the expected storage duration is reached or the data 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 data file and stop the proactive replica checking process.

### 6.4. Optimization Algorithms in PRCR

In Sections 6.2 and 6.3, we presented the high-level design of PRCR and its working process in detail. During the working process, additional algorithms are required so that all the data files could be maintained properly. In this section, we present two algorithms for supporting the data reliability assurance and optimizing the utilization of PRCR resources. First, we present the minimum replication algorithm for determining the minimum number of replicas. Second, we present the metadata distribution algorithm for maximizing the utilization of the PRCR capacity. Both algorithms work within the user interface of PRCR.

#### 6.4.1. Minimum Replication Algorithm

Based on the minimum replication approach presented in Chapter 5, the minimum replication algorithm is proposed. Based on this algorithm, minimum replicas are created for each data file, while the checking interval values throughout the expected storage duration of the data file are calculated. This algorithm is essentially the implementation of the optimized version of the minimum replication approach described in Chapter 5. To facilitate the practical application of the approach, the equation in the algorithm has been revised. In the original minimum replication calculation approach, the longest storage duration of a data file for meeting the data reliability requirement is predicted by following Equation (5.8), where $RA_k(I)$ is the data reliability assurance that is not yet determined. In the minimum replication algorithm, the calculation of checking interval follows Equation (6.1):

$$RR(I) = 1 - (1 - e^{-\lambda t})(1 - e^{-\lambda t}) / t$$  \hspace{1cm} (6.1)
By revising the equation, the checking interval can be determined before $RA_k(1)$ is determined. As $RR(l)$ is the data reliability requirement provided by the storage user, the output of the algorithm, i.e. checking interval of the data file, can still ensure that the data reliability requirement is not jeopardized. Figure 6.3 shows the pseudo code of the minimum replication algorithm.

In Figure 6.3, $ET$ is the expected storage duration of the data file. $P_1$ and $P_2$ are the disk failure rate patterns of the two disks for storing two replicas of the data file. $StartT$ is the time when the algorithm starts. $CIS$ is the result set containing all checking interval values. The algorithm first calculates the average failure rate of the data file stored on disk 1 for the duration of $ET$ (line 1). According to this value and Inequation (5.6), it determines the number of replicas that needs to be stored, i.e. to store the data file with one replica or two (line 2). If storing the data file with two replicas is necessary, based on the piecewise functions for the variable failure rate patterns of disk 1 and disk 2, the algorithm calculates all checking interval values throughout the expected storage duration of the data file in one go, and returns the checking interval values set as the result (lines 3-11).

Algorithm: Minimum Replication algorithm

Input: $ET$;                      // Expected storage duration
       $RR(1)$;                     // Data reliability requirement
       $P_1, P_2$;                  // Disk failure rate patterns of disk 1 and disk 2
       $startTime$;                // Start time of the algorithm

Output: $CIS$;                      // Set of checking interval values

01. $\lambda \leftarrow$ calculateAverageFailureRate($P_1, startTime, ET$);
02. if ($e^{-\lambda*ET} + ET - 1/RR(1)) < 0$) // Minimum replica number
03. $T = startTime$;                // The start time of each storage period
04. while ($T <= ET$) {
05.    $\lambda(t) \leftarrow$ obtainPiecewiseFunction($P_1, T$);
06.    $\lambda(t) \leftarrow$ obtainPiecewiseFunction($P_2, T$);
07.    solve Equation (6.1);
08.    $Ck \leftarrow$ the positive real root of Equation (6.1); // Checking interval
09.    $T = T + Ck$;
10.    $CIS \leftarrow Ck$;
11. } return $CIS$;
12. } else return -1;            // The file can be stored with only one replica

Figure 6.3 Pseudo code of the minimum replication algorithm

In addition to the application of the algorithm for data storage with variable disk failure rate, it is also applicable when the disk failure rate is a constant (e.g. virtual disks located over the virtual layer of the Cloud could apply such reliability model). In that case, the
minimum replication 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 Equation (6.1) only needs to be conducted once, and the checking interval obtained does not change unless any replica of the data file is lost and the corresponding disk is changed.

6.4.2. Metadata Distribution Algorithm

In order to manage the large amount of data files in the Cloud, PRCR must have a practically sufficient capacity. Meanwhile, in order to fully utilize the capacity of PRCR, the utilization of PRCR nodes must be maximized. To address this issue, we propose our metadata distribution algorithm. There are two purposes of the algorithm. First, it maximizes the utilization of PRCR, so that the running cost of PRCR for maintaining each data file is minimized. Second, it distributes the metadata of data files to the appropriate PRCR nodes, so that a sufficient data reliability assurance $RA_k(1)$ can be provided for meeting the data reliability requirement.

The Maximum Capacity of PRCR

The maximum capacity of PRCR stands for the maximum number of data files that PRCR is able to manage. In PRCR, the main component for replica management is the PRCR node. As mentioned in Section 6.2, PRCR may contain multiple PRCR nodes. Therefore, the maximum capacity of PRCR is the sum of the maximum capacities of all PRCR nodes. The maximum capacity of each PRCR node is determined by two parameters, which are the metadata scanning time and the scan cycle of the PRCR node. Note that the metadata scanning time is the time taken for scanning the metadata of a data file in the data table. The maximum capacity of PRCR can be presented by Equation (6.2). In the equation, $C$ indicates the maximum 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.

$$C = \sum_{i=1}^{N} \frac{T_{cycle}^i}{T_{scan}^i}$$

(6.2)
Provision of Sufficient Data Reliability Assurance

Although the maximum capacity of PRCR nodes can be calculated as mentioned above, in order to provide sufficient data reliability assurance to the data files, the scan cycle of the PRCR node must be no bigger than the checking interval values of data files. Therefore, each data file should be managed by the PRCR node with a scan cycle of the proper length. The scan cycle constraint of the PRCR node could lead to certain underutilization of PRCR.

In order to maximize the utilization of PRCR while providing sufficient data reliability assurance to the data files, according to the checking interval values of the data files and the scan cycles of PRCR nodes, the metadata distribution algorithm distributes the metadata of each data file to the most appropriate PRCR node. The principle of the algorithm is simple: it compares the checking interval values of the data file with the scan cycle of each PRCR node. Among the PRCR nodes with a scan cycle smaller than the checking interval values of the data file, the metadata are distributed to the node (or a random one of several nodes) that has the biggest scan cycle. The difference between the scan cycle of a PRCR node and the checking interval of the data file indicates the length of time for which the proactive replica checking task is conducted before the checking interval is reached. When this difference is minimized, the metadata scanning and proactive replica checking tasks can be least frequently conducted to each data file, so that the number of data files that a PRCR node is able to manage can be maximized.

The proof of the effectiveness of the metadata distribution algorithm is presented below:

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

**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 data file \( f \) maintained by PRCR node \( A \) and any other PRCR node \( I \) with scan cycle bigger than \( A \), let \( CI(f) \) be the minimum checking interval of data file \( f \), we have \( \text{ScanCycle}(A) \leq CI(f) < \text{ScanCycle}(I) \). Without losing generality, we randomly create another metadata distribution other than the current one by swapping the metadata of a pair...
of data files. Assume two PRCR nodes $B$ and $C$, in which $ScanCycle(B) > ScanCycle(C)$. Assume that data files $f_1$ and $f_2$ be managed by PRCR node $B$ and PRCR node $C$ respectively. Swap their managing PRCR nodes. Since $CI(f_2) < ScanCycle(B)$, the data reliability requirement of $f_2$ cannot be met. Therefore, data 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. Hence the theorem holds.

Algorithm: Metadata distribution algorithm
Input: $CI$; // Minimum checking interval of the data 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 $i \in S$ & scancycle($i$) < $CI$)
03. \hspace{0.5cm} diff $\leftarrow$ $CI$ - scancycle($i$);
 \hspace{0.5cm} // calculate the $CI$ - scancycle value for all available PRCR nodes
04. for (each $j \in S$ & scancycle($i$) < $CI$) {
05. \hspace{0.5cm} if ($CI$ - scancycle($j$) = min(diff))
06. \hspace{0.5cm} nodes $\leftarrow$ $j$;
}
 \hspace{0.5cm} // find the nodes with the smallest $CI$ - scancycle value
07. node $\leftarrow$ random(nodes); // randomly return one of the nodes
08. return node;

Figure 6.4 Pseudo code of metadata distribution algorithm

Figure 6.4 shows the pseudo code of the metadata distribution algorithm. In the figure, $CI$ indicates the minimum checking interval of the data file. $S$ indicates the set of all the PRCR nodes. The algorithm first calculates the differences between CI and the scan cycles of all available PRCR nodes (lines 2-3). Then, from all the PRCR nodes with a scan cycle smaller than CI, 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 situation where multiple PRCR nodes have the same scan cycle. The metadata distribution algorithm is able to effectively optimize the utilization of all the PRCR nodes. However, in addition to the algorithm, to distribute metadata, there are several issues that need to be further addressed.
• First, the capacity of each PRCR node is limited; when more and more data files are managed by PRCR, the capacity of PRCR nodes could gradually run out. To address this issue, 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 time for the scan cycle of the new PRCR node can be set to the same as the fully occupied PRCR node, which should be considered according to the data management requirement.

• Second, the data reliability model with a variable disk failure rate has led to the side effect that there exist multiple checking interval values for each data file, i.e. the checking interval changes from time to time. Once the checking interval 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 data file is managed by the PRCR node with a scan cycle smaller than all the checking interval values that the data files could have. Or, if the metadata of data files need to be redistributed no matter how, the redistribution could be conducted in a batch mode to reduce its impact and computation overhead.

• Third, the metadata are distributed according to the calculation of the minimum replication 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 checking interval is overestimated. In general, the situation of prediction errors is very similar to the second issue. Therefore, the solutions for the second issue are also applicable to prediction errors. In addition, the disk failure rates can be adjusted by statistics on the disks, etc.

6.5. Evaluation of PRCR

Based on the results of several experiments conducted on both a local computer and Amazon Web Services, in this section we evaluate PRCR from the aspects of performance and cost-effectiveness.
6.5.1. Performance of PRCR

Instead of being used to describe the speed of data access, term “performance” of PRCR refers to the running overhead as well as the maximum capacity of PRCR, which are the key indicators that show the ability of cost-effectively managing the Cloud data files. To evaluate the performance of PRCR, despite the time for creating replicas, we find that the minimum replication algorithm, the metadata scanning procedure and the proactive replica checking procedure are the major procedures which most affect the performance of PRCR. Therefore, investigations of these three procedures are conducted.

The Minimum Replication Algorithm

We evaluate the minimum replication algorithm due to two reasons:

- First, this evaluation is considered as the evaluation for the minimum replication approach described in Chapter 5.
- Second, as part of the user interface component in PRCR, this algorithm is of great significance for conducting the first and second steps of the PRCR working process as depicted in Figure 6.2.

The evaluation is conducted on aspects of execution time and result accuracy. In addition, as presented in Chapter 5, optimization approaches have been conducted for simplifying the calculation process. In order to fully investigate the minimum replication 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 version and ORV version) of the algorithm calculate the checking interval by solving Equation (6.3):

$$RA_k (1) = 1 - (1 - e^{-\lambda t})(1 - e^{-2\lambda t}) / t \quad (6.3)$$

which is the revised version of Equation (5.7). This equation calculates the checking interval in the unoptimized form. Meanwhile, while the optimized versions (i.e. OPC version and OPV version) calculate the checking interval by solving Equation (6.1). The constant disk failure
rate versions (i.e. ORC version and OPC version) of the algorithm are for storage with a constant disk failure rate, while the variable disk failure rate versions (ORV version and OPV version) are for storage with a variable disk failure rate. The evaluation of the constant disk failure rate versions of the algorithm corresponds to the discussion in Section 6.4 about the algorithm working in a constant failure rate environment.

In Equation (6.1) we use the tangent line of $e^{-\lambda t}$ at point $(0, 1)$ as a substitution for the original curve $e^{-\lambda t}$. The function of the tangent line is $f(\lambda t) = 1 - \lambda t$, which is a special case of $f(\lambda t) = a\lambda t + b$ mentioned in Section 5.1, while $a=-1$ and $b=1$. Figure 6.5 shows both the original curve of $e^{-\lambda t}$ and the substitution curve of tangent line $f(\lambda t) = 1 - \lambda t$. In the figure, the substitution curve is located at the lower side of the original curve of $e^{-\lambda t}$. According to the maximum disk failure rate of the IDEMA standard (i.e. 4.38 %/year or 0.5 %/1000 hours) [42] and the disk nominal lifespan of five years [70], the range of $\lambda t$ 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 $\lambda t$, the deviation gets even smaller. After this substitution, Equation (6.1) is further simplified into function $RR(1) = 1 - \lambda_1\lambda_2 t$. Compared to Equation (6.3), 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 Equation (6.1), the result (i.e. the checking interval values of the data file) is always conservatively underestimated slightly, 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.

The execution time and accuracy rate of the algorithm for all four versions are tested under the same data file and disk settings. Note that the accuracy rate stands for the ratio between the results calculated by the optimized versions of the algorithm and the original versions of the algorithm, which indicates the accuracy of the results produced by the optimized versions of the algorithm. The results of the experiments are shown in Table 6.1. The upper half of Table 6.1 shows the average execution time of all four versions of the algorithm. In addition, the number of checking interval values calculated in each run of the algorithm is also shown in brackets 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 same execution time incurred when data reliability is 99%, at which one replica suffices to meet the data reliability and the codes of all versions of the algorithm run are the same, the execution time of the optimized versions of the algorithm is significantly smaller than that of the original versions of the algorithm. Second, although the overall trend for all versions of the algorithm is that the execution time increases with the increment of data reliability requirement, the execution time of the optimized versions of the algorithm increases much slower than that of the original versions. However, in the execution time part of Table 6.1, it also shows that, whether or not optimized, with the increment of the data reliability requirement, the execution time of the variable disk failure versions of the algorithm increases fast. The reason for this is because with a higher data reliability requirement, the checking interval of the data file becomes shorter, so that during the expected storage duration of the data file, more checking interval values need to be calculated. In the accuracy rate part of Table 6.1, due to storage with a single replica, the accuracy rate for data reliability of 99% is not applicable. Despite of that, the lowest accuracy rate occurred is 89.52% when the data reliability requirement is 99.9%. The accuracy rates increase with the increment of the data reliability requirement, which is consistent with the trend of deviation between the tangent line and the original curve of \( e^{-\lambda t} \) as shown in Figure 6.5. According to the existing results shown in Table 6.1, the accuracy rates of the optimized versions of the algorithm reach 99.9%. In fact, this value can be even larger when the data reliability requirement becomes higher.
Table 6.1 Execution Time and Accuracy Rate of Minimum Replication Algorithm

<table>
<thead>
<tr>
<th>Reliability</th>
<th>One replica</th>
<th>Two replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORC</td>
<td>0.69</td>
<td>15.34</td>
</tr>
<tr>
<td>OPC</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>ORV</td>
<td>0.72(1)</td>
<td>16.26(1)</td>
</tr>
<tr>
<td>OPV</td>
<td>0.72(1)</td>
<td>4(1)</td>
</tr>
</tbody>
</table>

Accuracy rate

| OPC         | NA          | 89.52%      | 99.00%      | 99.90%      |
| OPV         | NA          | 89.61%      | 99.00%      | 99.90%      |

The results in Table 6.1 show that, depending on the data reliability assurance provided, the minimum replication algorithm is able to calculate the checking interval values of data files between a few milliseconds to hundreds of milliseconds. However, the reliability assurance is not restricted to that shown in the table, which can be even higher (e.g. the data reliability assurance of 99.9999% is the same as that of the conventional 3-replica strategy) and easily changed according to the data reliability requirement. To provide higher data reliability assurance, more time could be taken to conduct the minimum replication algorithm, and more checking interval values need to be calculated. The execution time of the optimized versions of the algorithm is generally much shorter than that of the original versions, but the accuracy rate is somewhat lower when the reliability requirement is lower and increases while the data reliability requirement increases.

Metadata Scanning and Proactive Replica Checking

An experimental PRCR is implemented based on the Amazon Web Services including Amazon S3, Amazon EC2, and Amazon Beanstalk mainly for evaluating the metadata scanning procedure and the proactive replica checking procedure. The structure of the experimental PRCR consists of one user interface and one PRCR node, both of which run on a single EC2 instance. Based on the above, execution times of the metadata scanning process and the proactive replica checking task are obtained.

In the experiments, the metadata scanning procedure and the proactive replica checking procedure are both simulated with several configurations. We hire four types of EC2
compute instances for the management of 3000 S3 objects (i.e. data files) stored with Standard Storage Service and Reduce Redundancy Storage Service respectively. Table 6.2 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.

<table>
<thead>
<tr>
<th></th>
<th>t1.micro</th>
<th>m1.small</th>
<th>m1.large</th>
<th>m1.xlarge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scanning Time</strong></td>
<td>≈700ns</td>
<td>≈400ns</td>
<td>≈700ns</td>
<td>≈850ns</td>
</tr>
<tr>
<td><strong>Checking Time (standard)</strong></td>
<td>≈27ms</td>
<td>≈27ms</td>
<td>≈30ms</td>
<td>≈27ms</td>
</tr>
<tr>
<td><strong>Checking Time (reduce redundancy)</strong></td>
<td>≈25ms</td>
<td>≈24ms</td>
<td>≈37ms</td>
<td>≈23ms</td>
</tr>
</tbody>
</table>

6.5.2. Cost-effectiveness of PRCR

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

**Running Overhead**

First, the running overhead of PRCR is evaluated. Our major concern is about what proportion the running overhead takes from the total cost for each data file. For the huge number of Cloud data files, PRCR nodes would normally be well loaded. The running overhead of each data file can be derived by dividing the total PRCR running cost by the maximum capacity of PRCR nodes.

Therefore, the maximum capacities of PRCR nodes are presented. According to the results shown in Table 6.2, for illustration we choose 700ns and 30ms as the standard execution times for the metadata scanning process and the proactive replica checking task respectively. The micro EC2 instance (t1.micro) is chosen as the default Cloud compute instance. Based on the standard execution times, the maximum capacity of the PRCR nodes is calculated and presented. We calculate the maximum capacity of PRCR nodes for storing data files with the data reliability requirements of 99%, 99.9%, 99.99% and 99.999% per year under different storage unit failure rates. In Table 6.3, the relationships among the reliability
requirement, the average failure rate of a single replica $\lambda$ and the maximum capacity of PRCR nodes are clearly revealed. With different failure rates of a single replica and reliability requirements, each PRCR node is able to manage from $4.5 \times 10^{10}$ to $2.8 \times 10^{15}$ data files, which is quite large. Although the maximum capacity of PRCR nodes reduces with the increment of disk failure rate and data reliability requirement, the maximum capacity of PRCR nodes is deemed big enough to be practical for the management of the huge number of Cloud data files.

Table 6.3 Maximum Capacity of PRCR Nodes

<table>
<thead>
<tr>
<th>RA</th>
<th>10%</th>
<th>5%</th>
<th>2%</th>
<th>1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>$5 \times 10^{13}$</td>
<td>$2.3 \times 10^{14}$</td>
<td>$2.8 \times 10^{15}$</td>
<td>NA</td>
</tr>
<tr>
<td>99.9%</td>
<td>$4.5 \times 10^{12}$</td>
<td>$1.8 \times 10^{13}$</td>
<td>$1.2 \times 10^{14}$</td>
<td>$5 \times 10^{14}$</td>
</tr>
<tr>
<td>99.99%</td>
<td>$4.5 \times 10^{11}$</td>
<td>$1.8 \times 10^{12}$</td>
<td>$1.1 \times 10^{13}$</td>
<td>$4.5 \times 10^{13}$</td>
</tr>
<tr>
<td>99.999%</td>
<td>$4.5 \times 10^{10}$</td>
<td>$1.8 \times 10^{11}$</td>
<td>$1.1 \times 10^{12}$</td>
<td>$4.5 \times 10^{12}$</td>
</tr>
</tbody>
</table>

The total PRCR running cost is composed of the running cost for user interface, PRCR nodes and Cloud compute instances for proactive replica checking. According to the latest Amazon EC2 prices, the corresponding cost of an EC2 micro instance takes only $14.40 (0.02/hour * 24 hours/day * 30 days/month) per month each. Therefore, for complete PRCR running over AWS with 1 PRCR node, the running cost is $43.20 ($14.40*3) per month. When divided by the maximum capacity of a PRCR node, it can be seen that the running overhead for each data file is very small (no more than $10^9$/data file/month according to Table 6.3) and can be negligible. For example, the storage of a data file with the size of 1GB has a PRCR running overhead about $10^7$ times cheaper than the storage cost (several cents/month).

Data Storage Cost

Next, the data storage cost using PRCR is investigated. We simulate the data reliability management process of PRCR to manage the data files of the pulsar searching example presented in Section 3.1. In the simulation, the storage costs are compared with the conventional 3-replica strategy which is widely used in current Clouds.

In the simulation, data files generated by the pulsar searching application mentioned in Section 3.1 are applied for illustration. In order to compare the storage using PRCR and without using PRCR, three storage modes are provided. When PRCR is applied, two storage
modes are provided: One is a 2-replica mode, which is for high data reliability and/or long-term storage; and the other is a 1-replica mode, which is for low data reliability and/or short-term storage. The former stores data files with two replicas and manages data files with PRCR, while the latter only stores data files with one replica without PRCR involved. When PRCR is not applied, the conventional 3-replica mode is applied. By combining these storage modes for storage, we test four different storage plans: 1-replica, 1+2 replica, 2-replica and 3-replica. The 1-replica plan stores all data files in the 1-replica mode, which represents the data storage without any replication. The 2-replica plan stores all data files in the 2-replica mode, so that all of the data files are stored with high data reliability assurance and/or long-term storage. The 3-replica plan stores all data files with three replicas, which represents the conventional 3-replica strategy. The 1+2 replica plan divides all the data files into two categories, in which both the 1-replica mode and 2-replica mode are used. This 1+2 replica plan represents the actual data reliability management of PRCR. To meet different data reliability requirements and different storage durations of data files, the minimum number of replicas is created. As mentioned in Section 3.1, in the pulsar searching application instance, among all the data files, 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 in the 2-replica mode. The other data files, for temporary usage purposes, are stored in the 1-replica mode.

Figure 6.6 shows the average replica numbers and total data sizes with different storage plans for the data files processed and generated in the pulsar searching example. 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 data 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 for the pulsar searching application instance, each application instance has around 16,000 data files stored with two replicas, while all the rest, more than 310,000 data files, are stored with one replica. The ratio between the number of the two types of data files reaches a staggering 1:41, and the ratio between the sizes of two types of data files is about 2.34:1. Compared with the 2-replica plan, more than 95% of replicas with 23% of the total data size are reduced. Compared with the 1-replica plan, the 1+2 replica plan generates only 53% additional storage space for storing all the pulsar searching data files (i.e. the data redundancy level is 1.53), and the data reliability requirement of all the data files is guaranteed without jeopardy. For other Cloud
applications with different data composition and data reliability requirements, the data redundancy varies, and could be even lower than 1.53.

![Figure 6.6 Average replica numbers and data sizes](image)

In Figure 6.6 we only discuss the case of processing data files for eight minutes of observation by the pulsar searching application. However, regarding the case presented in Section 3.1 that processes the data files for the observation of eight hours a day, 30 days per month, assuming that Amazon S3 standard storage uses the 3-replica strategy, the storage cost per month is reduced from US$29,900 to US$15,300. Meanwhile, the running cost of PRCR for managing such data amount is only tens of dollars per month. It can be seen that the storage cost saved by using PRCR could be huge. Moreover, here we only compared PRCR with the conventional 3-replica strategy. To manage data files with very high data reliability requirement by using the conventional strategy, even three replicas may not be enough. According to the nature of PRCR that stores no more than two replicas, the storage cost reduced by using PRCR could be even more.

6.5.3. Summary of the Evaluation

We evaluated PRCR from aspects of performance and cost-effectiveness. As for performance, we have tested the major procedures of PRCR’s working process, including the minimum replication algorithm, the metadata scanning process, and the proactive replica checking process. Specifically, the evaluation of minimum replication algorithm is also for evaluating the minimum replication calculation approach presented in Chapter 5. We conclude
that PRCR is able to provide data reliability management with a wide range of data reliability requirements at a high performance. With regard to the cost-effectiveness, we have found that the maximum capacity of PRCR suffices to provide data reliability management for the big data in the Cloud with a huge number of Cloud data files with a very low running overhead. According to the data reliability management simulation conducted, PRCR is able to minimize the storage cost without violating the data reliability requirements. Compared to the storage using the conventional 3-replica strategy, our PRCR can reduce between two-thirds to one-third of the storage cost, while the running overhead for PRCR itself is negligible.

6.6. Summary

In this chapter, we presented our data reliability assurance solution for the big data in the Cloud with a huge amount of data files during the data maintenance stage, which is a novel cost-effective data reliability assurance mechanism named PRCR (Proactive Replica Checking for Reliability). We first explained the theoretical basis of PRCR that the data reliability can be predicted and ensured by regularly checking the data files and recovering any data loss. Then we described the structure and working process of PRCR in detail. Afterwards, in order to maintain the large amount of data files in the Cloud properly as well as optimize the function of PRCR, two algorithms for supporting the data reliability assurance and optimizing the utilization of PRCR resources have been presented respectively.
Chapter 7 Cost-Effective Data Transfer for Data Creation and Data Recovery

Data transfer activities are intensively involved in the replication-based Cloud data storage process especially for data creation and data recovery. For cost-effective big data storage in the Cloud while meeting the data reliability throughout the Cloud data lifecycle, the energy consumption of data transfer must be taken into account. Based on previous analysis in Section 2.3 and Section 3.2 on the features of the Cloud network and data transfer in the Cloud, in this chapter we present our data transfer solution that cooperates with PRCR for maintaining the huge number of Cloud data files.

In this chapter, we present in detail our novel energy-efficient data transfer strategy called LRCDT (Link Rate Controlled Data Transfer). By scheduling bandwidth in a link rate controlled fashion, LRCDT aims to reduce the energy consumption by “lazy” data transfer that does not require data to be transferred at the maximum speed, so that the cost incurred for both data creation and data recovery processes can be reduced.

The structure of this chapter is organized as follows. In Section 7.1, we explain how the data transfer deadline is determined for data creation and data recovery processes respectively. In Section 7.2, we describe our network model for Cloud networks with bandwidth reservation. In Section 7.3, we describe the energy consumption model of network devices. In Section 7.4, the data transfer strategy LRCDT is presented in detail. In Section 7.5, evaluation for LRCDT is conducted from the aspects of energy consumption and task completion time respectively. Finally, in Section 7.6, we summarize the works presented in this chapter.

This section is mainly based on our work presented in [51].
7.1 Determining the Deadline for Data Creation and Data Recovery

As discussed in Chapter 5, there is a time window for the system to increase the data redundancy level when a data file is first created or needs to be recovered. In order to conduct the data transfer activity for data creation or data recovery, in this section we discuss how to determine the time window, i.e. the data transfer deadline. Although the process of data creation and data recovery are similar, due to different start time for conducting the data transfer tasks, the deadline for data creation and data recovery are determined based on different equations.

- When the Cloud data file is first created, if it needs to be stored with two replicas, there is a deadline for generating the second replica. For the data files that are first created in the Cloud, as the original replicas are newly generated/uploaded and never get lost before the data creation process, according to Equation (5.2), the deadline for creating the replica can be calculated according to equation shown below:

\[
dealine_{\text{cre}} \cdot \lambda = \text{lifespan}(1 - RR(1))
\]

where \( \lambda \) indicates the failure rate of the disk for storing the original replica of the data file.

- When a Cloud data file needs to be recovered, there is also a deadline for the new replica to be transferred to the corresponding storage device. For such data files that have been stored for a certain period in the Cloud, as there could have some data files that have already been lost, the deadline need to be determined based on previous storage history of the whole data space. The data space is the group of data files that PRCR provides data reliability assurance to. According to Equation (5.2), the deadline of the recovery process can be calculated according to equation shown below:

\[
dealine_{\text{rec}} \cdot \lambda + \frac{D_{\text{org}} - D_{\text{now}}}{D_{\text{org}}} = \text{lifespan}(1 - RR(1))
\]

In this equation, \( D_{\text{org}} \) indicates the original size of the data set and \( D_{\text{now}} \) is the current size of the data set.
When the data transfer can be completed within the *deadline* calculated by the above equations, the possibility of data loss during the data creation and data recovery processes is lower than expected, so that the data reliability requirement can be met.

7.2. Cloud Network Model

The Cloud network consists of many devices such as routers, switches, optical fibers, twisted-pair wires and network interface cards. Each of these devices has its own working schema with different parameters. In order to focus on the data transfer aspects, the specific working schema of each device should be simplified and abstracted. In this section, we present the network model for data transfer in the Cloud.

As addressed in Section 3.2, our research assumes a Cloud where bandwidth reservation is enabled on its dedicated Cloud networks. Under this assumption, we propose an end-to-end network model for data transfer in the Cloud. This model describes the data transfer link between the data source and the target. Since the data transfer link and bandwidth scheduling is determined in advance in a bandwidth reserved dedicated network, it is impractical to dynamically reroute and reschedule the data transfer link. Therefore, this model is sufficient to describe the data transfer link from point A to point B with a single routing path. However, this does not mean that the routing path is unalterable. For data transfer processes with more than one path, the bandwidth reservation and scheduling can be conducted at the desired routing path respectively.

The network model consists of four sub models: the **overall network model**, the **pipeline model**, the **pipeline agenda model** and the **overall agenda model**. Among these sub models, the overall network model presents the entire data transfer link at high level with all the network devices from the data source to the target; the pipeline model describes the connecting status between two network devices over the link; the pipeline agenda model describes the detailed bandwidth usage and schedule of a pipeline; the overall agenda model describes the detailed bandwidth usage and schedule of the entire data transfer link.
7.2.1. Overall Network Model

Figure 7.1 Overall network model

Figure 7.1 shows an example of the overall network model. There is one source and one target at each end of the link, which indicates the start and end of the data transfer. Each of the two ends of the data transfer link can be a storage device or a subset of the dedicated network. Between the source and target, there are several routing devices over the data transfer link. Without losing generality, these routing devices are abstracted as ‘router’ throughout the thesis for ease of description. We assume that these routers have the capacity of changing link rates. Each router has one input port and one output port connected to the link. The connection between the output port of a router (or source) and the input port of the subsequent router (or target) forms a ‘pipeline’, in which the link rate and available bandwidth vary over time. As shown in the figure, the boxes indicate the link rate of the pipeline while the dark parts indicate the available bandwidth.

7.2.2. Pipeline Model

Figure 7.2 Pipeline model

Figure 7.2 shows the pipeline model between two routers. The status of a pipeline can be described as a set \((LR, \text{availableBW}, t)\), in which \(LR\) is the link rate, \(\text{availableBW}\) is the available bandwidth and \(t\) is the time. For bandwidth reservation purposes, routers at both ends
of the pipeline record and maintain the pipeline status. Each of the records is called a pipeline agenda.

7.2.3. Pipeline Agenda Model

Figure 7.3 shows the pipeline agenda model. From the pipeline agenda, the existing bandwidth schedule of the pipeline can be clearly seen. In this model, we call the period a timeslot, where the available bandwidth and link rate remain the same. A timeslot can be denoted as $TS(t_i)$ in which $t_i$ is the start time of the timeslot. The length of each timeslot depends on the existing bandwidth schedule on the pipeline. For example, in $TS(t_2)$, both the link rate and available bandwidth drop to 0, which indicates a shutdown period. During a shutdown period, one router or both routers connected to the pipeline are shut down. The beginning or the end of each timeslot is called an event $E(t_i)$ indicating that the status of the pipeline is about to change.

7.2.4. Overall Agenda Model

Figure 7.4 shows the overall agenda model of the data transfer link. The overall agenda of the link is created by collecting all the pipeline agendas on the link. Based on the overall agenda, bandwidth scheduling for the entire data transfer link can be conducted. The format of the overall agenda model is similar to the pipeline agenda model except:
- First, instead of showing the available bandwidth and link rate of each timeslot, in the overall agenda model each timeslot contains a list of the available bandwidth and link rate of all the routers. The list is sorted according to the available bandwidth under the current link rate;

- Second, instead of indicating the available bandwidth of the pipeline under the current link rate, the dark bar in each timeslot indicates the available bandwidth of the link. It is the minimum available bandwidth of all the routers on the link at the time;

- Third, instead of indicating a shutdown period of the router at either side of the pipeline, the shutdown period in the overall agenda model (i.e., between $t_2$ and $t_3$) indicates a shutdown period of the link. This means that at least one of the routers on the link is shut down.

For creating the overall agenda of a link, all the pipeline agendas on the link need to be collected. We apply the agenda collection approach proposed in [59]. According to this approach, all the pipeline agendas are finally transferred to the target of the link. Therefore, the agenda merge algorithm is conducted at this node.

The overall agenda consists of a sorted event list $L$ for describing timeslots and a two-dimensional data structure $Ranking$ for recording the status of each router at each timeslot. Figure 7.5 shows the pseudo code of the pipeline agenda merge algorithm. First (lines 1-3), all events from all pipeline agendas are sorted in $L$ in chronological order. Then (lines 4-6), grouped by events, the available bandwidth and link rate of each router is stored in $Ranking$. 

![Figure 7.4 Overall agenda model](image_url)
which represents the status of each router in the timeslot. In addition (line 7), for each event, all the routers on the link are sorted by the available bandwidths under current link rates in ascending order. The complexity of the agenda merge algorithm is $O(n^2 m \log n)$ where $n$ is the total number of events (or timeslots) and $m$ is the number of pipelines (or routers) on the link.

Algorithm: Agenda merge algorithm
Input:   Agendas; // All pipeline agendas on the link
Output: OA; // The overall agenda of the link

1. For each agenda in Agendas
2.   For each event $E(ti)$ in agenda
3.       List $L$ <- $E(ti)$ sorted in chronological order;
4.   For each event $E(ti)$ in $L$
5.       For each agenda in Agendas {
6.         Ranking(ti) <- the availableBW and LR of each agenda at $ti$;
7.       Sort Ranking(ti) by availableBW in ascending order; }
8. OA <- $L$ & Ranking

Figure 7.5 Agenda merge algorithm

7.3. Energy Consumption Model for Cloud Data Transfer

In order to investigate the energy consumption of Cloud data transfer, a deep understanding of how the energy is consumed in Cloud network devices is required. In this section, we adopt an end-to-end energy consumption model called ECOFEN that was proposed in [59] to support our research. Details of the ECOFEN model can be found in the original paper.

In general, the ECOFEN model consists of three parts:

- **Energy consumption equation:** $E(T) = \int_0^T Power(t) dt$. This equation defines the energy consumption during the time period of $T$ to be the accumulation of the power consumption function $Power(t)$ all over the period.

- **Router energy consumption model:** $E = E_{boot} + E_{work} + E_{halt}$. This defines the energy consumption of a router to be the accumulation of the energy consumption when the router is booting, working and halting.
**Power-bandwidth function**: In Figure 7.6, we only show the phased power-bandwidth function of active routers. This function shows the power variation of the routers along with the occupied bandwidth of the link. From this figure, it can be seen that with the increment of occupied bandwidth, the power consumption of the router only incurs a negligible increment. Until the occupied bandwidth reaches a certain level, the power of the router will incur a significant increase indicating that the link rate is increased.

According to this energy consumption model, the power-bandwidth function specifically, the power consumption of active routers shows a clear step-like pattern with the link rate. Based on this model, the relationship between link rate and energy consumption of routers can be clearly seen, and the potential possibilities of reducing data transfer energy consumption by link rate control is clearly revealed.

### 7.4. Novel Cost-effective Data Transfer Strategy LRCVT

Based on the network model and data transfer energy consumption model, in this section we present details of our novel LRCDT (Link Rate Controlled Data Transfer) strategy for energy-efficient data transfer in the Cloud. Under the assumption of the dedicated Cloud network with bandwidth reservation, our LRCDT schedules the bandwidth for each data transfer task. By leveraging the phenomenon that the power level of network devices only changes when the link rate is changed but is insensitive to the utilization of the bandwidth, LRCDT conducts data transfer in a link rate controlled fashion. This means that the link rate of network devices is limited to the minimum level while as much available bandwidth as possible (within the link rate) is scheduled for data transfer tasks. In LRCDT, data transfer
tasks are divided into two types according to the data transfer speed requirement. These are ‘eager’ data transfer and ‘lazy’ data transfer. The ‘eager’ data transfer requires the maximum transfer speed while the ‘lazy’ data transfer does not. By dividing data transfer into two types, LRCDT schedules link bandwidth respectively to improve energy consumption while meeting the data transfer speed requirement at the same time. It is able to significantly reduce energy consumption specifically for data transfer tasks that do not require the maximum transfer speed, referred to as ‘lazy’ data transfer, so that the overall energy efficient data transfer goal can be achieved. Compared to the data transfer strategies mentioned in Section 2.3, first, LRCDT provides a much faster transfer speed in comparison to the minimum-speed strategy as proposed in [23]. Second, LRCDT only schedules the bandwidth on active links (i.e., the period when the link is active) so that the shutdown approach is still allowed on the same link when LRCDT is already applied. Third, LRCDT consumes much less energy during data transfer in comparison to both the minimum-speed strategy and the maximum-speed data transfer strategy proposed in [38]. Fourth, unlike ALR that monitors bandwidth usage and changes link rate afterwards, LRCDT schedules bandwidth before data transfer is conducted. It fully utilizes the advantages of the dedicated network of the Cloud so that data transfer can be fully controlled. Meanwhile, LRCDT divides data transfer tasks into two types, according to the transfer speed requirement, so that the energy consumption can be improved while the data transfer speed requirement can be met at the same time.

In LRCDT several features have been designed accordingly for meeting the needs of energy-efficient data transfer as well as addressing the considerations illustrated in Section 3.2.

- First, in order to reduce energy consumption, the basic idea of LRCDT is to limit the routers’ link rates to the minimum level available. This ensures that the power consumption of the routers is minimized. Meanwhile, by providing as much available bandwidth as possible (without changing the link rate), the data file can be delivered as fast as possible.

- Second, to address the first issue raised in Section 3.2 for cost-effective data transfer strategy, in LRCDT, a \((\text{startTime}, \text{deadline})\) pair is set for each data transfer task. Within a bandwidth reserved network, the \((\text{startTime}, \text{deadline})\) pair indicates the expected period of link occupation, which is crucial to the bandwidth scheduling process. However, if these parameters are not provided by the application, a default deadline value could be set. According to the size of the data file to be transferred in each task, LRCDT allocates
sufficient bandwidth within the \((\text{startTime}, \text{deadline})\) period to ensure that the task can be completed in time. In addition, considering the medium that receives the data file, an upper boundary \(\text{maximumBW}\) for the data transfer bandwidth is set for each data transfer task.

- Third, to address the second issue raised in Section 3.2, in LRCDT, for ‘lazy’ data transfer, the energy-efficient data transfer is conducted where the link rate is minimized. For ‘eager’ data transfer the data file is transferred as quickly as possible while LRCDT schedules the maximum bandwidth for data transfer without considering the link rate. By conducting these two different types of data transfer, LRCDT is able to meet the requirements of both types of data transfer in the Cloud. To avoid affecting the existing link traffic on the Cloud network, in LRCDT, the bandwidth is allocated based on the existing agenda of the link. No data transfer bandwidth is allocated during the already scheduled \textit{shutdown} period unless the data transfer task cannot be completed within the maximum transfer duration and can be completed if the \textit{shutdown} period is occupied.

According to the Cloud network model, all agendas of the routers on the link can be merged into an overall agenda. Based on the overall agenda, the bandwidth scheduling is conducted. According to [85], the link rate switching time ranges from 10ms to 100ms. This is quite considerable for high-performance data transfers. Therefore, to eliminate additional link rate switching caused by LRCDT, the bandwidth scheduling is conducted with timeslot as the minimum schedule unit. In this way, the link rate switching caused by LRCDT could be done where the link rate is already planned to switch. In LRCDT, the bandwidth scheduling follows a simple ‘lower boundary policy’. At each non-\textit{shutdown} timeslot, the scheduled bandwidth should not be smaller than a certain lower-boundary called ‘\text{minimumBW}\’ unless the maximum available bandwidth of the link is smaller than it. \(\text{minimumBW}\) is the lowest average bandwidth for ensuring the conduct of the data transfer task. It can be obtained from equation \(\text{minimumBW} = \frac{\text{dataSize}}{\text{maximumTransferDuration}}\). This policy aims to ensure that the data transfer task will be completed before the deadline. If the available bandwidth of the link is smaller than \(\text{minimumBW}\), the link rates of routers increase to provide more available bandwidth. The order of link rate increment is according to the available bandwidth of each router. The router that has the minimum available bandwidth increases its link rate first so that the available bandwidth of the entire link increases. The link rate increment stops when the available bandwidth is larger than, or equal to, \(\text{minimumBW}\). Afterwards, the smaller one between \(\text{maximumBW}\) and the maximum available bandwidth under the current link rate is
Algorithm: Bandwidth allocation algorithm for “lazy” data transfer
Input: OA; //Overall agenda of the link
deathline, startTime;
dataSize; //Size of the data
maximumBW; //Transfer speed upper bound between device A and B
Output: BA; //Bandwidth allocation

01. While (the bandwidth of dataSize is not all allocated & there is still available bandwidth that can be allocated) {
02. \( transferDuration = \text{deadline} - \text{startTime} - \text{shutdown period} \); // Data transfer duration
03. \( \text{minimumBW} = \frac{\text{dataSize}}{\text{transferDuration}} \); //Minimum transfer bandwidth
04. List \( TS < - \) all timeslots between \( \text{startTime} \) and \( \text{deadline} \) according to \( OA.L \);
05. For each timeslot \( TS(t) \) in \( TS \)
06. If \( (ts, is \text{ shutdown period}) \) Skip \( TS(t) \);
07. If \( (\text{availableBW of the link } < \text{minimumBW} \& \text{ can still be increased} ) \)
08. Repeat { Increase the link rate of the router with the smallest available bandwidth;
09. Recalculate available bandwidth of the router;
10. Recalculate \( \text{availableBW of the link} \);
11. } until \( (\text{availableBW of the link } \geq \text{minimumBW} \lor \text{availableBW of the link can not be increased anymore}) \)
12. If \( (\text{availableBW}<\text{maximumBW}) \) BA<-allocate availableBW;
13. Else BA<-allocate maximumBW;
14. Update OA;
}

Figure 7.7 Bandwidth scheduling algorithm for ‘lazy’ data transfer

As the major part of the LRCDT strategy, Figure 7.7 shows the pseudo code of the bandwidth scheduling algorithm for ‘lazy’ data transfer. The bandwidth scheduling starts with the initialisation of several parameters: first (line 2), the \( (\text{startTime}, \text{deadline}) \) pair is set and the maximum data transfer duration is initialised to be the time between \( \text{deadline} \) and \( \text{startTime} \) minus the \text{shutdown} period. Second (line 3), according to the maximum data transfer duration and size of the data file, the bandwidth lower boundary \( \text{minimumBW} \) can be calculated. Third (line 4), for initialising list \( TS \), the timeslots of all agendas during the data transfer process are obtained according to event list \( L \) of the overall agenda. After the initialisation part, in the main part of the algorithm (lines 5-13), it allocates bandwidth for each timeslot between \( \text{startTime} \) and \( \text{deadline} \): first, the \text{shutdown} period is skipped; second, if \( \text{available bandwidth} \) of the link is smaller than the lower boundary \( \text{minimumBW} \) and still has not reached the maximum available bandwidth of the link, the algorithm repeats the process of
link rate increase for the router with the smallest available bandwidth (i.e., the bottleneck router that constrain the available bandwidth of the link). The repeating process finishes when available bandwidth of the link is bigger than minimumBW or reaches the maximum level. Third, if the available bandwidth of the link is larger than maximumBW, only maximumBW bandwidth is allocated to the timeslot. Otherwise all available bandwidth is allocated to the timeslot. Due to the bandwidth scheduling lower boundary policy, in a very rare case, the algorithm cannot allocate sufficient bandwidth for completing the data transfer task. In this case, the algorithm sets a loop (lines 2-14) where, if there is still available bandwidth that can be allocated between startTime and deadline, the bandwidth scheduling is conducted based on the new overall agenda (line 14) until all the bandwidth for transferring the data file is allocated. The complexity of the bandwidth scheduling algorithm is $O(nm^2)$.

In Figure 7.8, we present a bandwidth scheduling process to better illustrate the algorithm. Take the agenda example shown in Figure 7.4 as the overall agenda of the link for transferring data file D from device A to device B. maximumBW is initialised as the maximum transfer speed between A and B whereas minimumBW is calculated according to the maximum data transfer duration and the data file size. The start time $t_1$ and deadline are set as shown in Figure 7.8 and the bandwidth scheduling process starts from $t_1$:

1) Between $t_1$ and $t_2$, the initial available bandwidth $BW_2$ is smaller than minimumBW so that the link rate increases twice where the available bandwidth of the link increases to $BW_2'$ and $BW_2''$ respectively. Because $BW_2''$ is larger than minimumBW and smaller than maximumBW, $BW_2''$ is allocated in this time slot (striped area as shown in the figure);

2) $TS(t_2)$ is skipped because of shutdown period;

3) Between $t_3$ and $t_4$, the available bandwidth of the link is already larger than minimumBW and smaller than maximumBW. Hence, link rates remain unchanged and the current available bandwidth is allocated;

4) The bandwidth scheduling process repeats for each timeslot. Between $t_{n-1}$ and deadline, the available bandwidth is larger than maximumBW so maximumBW is allocated;

5) All the bandwidth is allocated in which the data transfer task is expected to be completed between $t_{n-1}$ and deadline.
7.5. Evaluation of LRCDT

In this section, we present the evaluation of our LRCDT strategy. In order to validate the effectiveness of the strategy in reducing energy consumption, we compare LRCDT with two other existing popular strategies proposed in [23] and [38] from the aspects of energy consumption and task completion time respectively. As mentioned in Section 2.3, the strategy proposed in [23] is to transfer the data file in typical ‘lazy’ fashion, where data transfer is conducted with a constant minimum speed and completes by the deadline. Meanwhile, the strategy proposed in [38] transfers the data file in a typical ‘eager’ fashion that data transfer is conducted at the maximum speed available. According to the characteristics of these two strategies, in this paper we name them as the minimum-speed strategy and the maximum-speed strategy respectively.

In the evaluation, we build an environment to simulate data transfer links of a real Cloud network. All the three data transfer strategies are simulated as three different bandwidth scheduling processes following different rules. Simulations of all three strategies are conducted based on randomly generated data transfer links with random traffic conditions. We generate multiple data transfer links with different parameters, and conduct the bandwidth scheduling processes with different rules on each data transfer link. We obtained the bandwidth usage of data transfer links and calculated the overall energy consumption during the period of task execution. Each simulation with certain parameter sets was conducted for several times, and all the simulation results are the average results of the simulations.
7.5.1. Parameters of Simulation

Table 7.1 Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>0</td>
</tr>
<tr>
<td>deadline</td>
<td>80, 800, 8000 Seconds</td>
</tr>
<tr>
<td>dataSize</td>
<td>1 MB-10GB (8 Mbits-80Gbits)</td>
</tr>
<tr>
<td>maximumBW</td>
<td>10Mbps, 100Mbps, 1Gbps</td>
</tr>
<tr>
<td>routers</td>
<td>10</td>
</tr>
<tr>
<td>events</td>
<td>500</td>
</tr>
<tr>
<td>linkRates</td>
<td>10Mbps, 100Mbps, 1Gbps</td>
</tr>
<tr>
<td>routerPower</td>
<td>0.1W, 0.4W, 3.6W</td>
</tr>
</tbody>
</table>

We have simulated all the three strategies based on randomly generated data transfer links. In order to simulate the traffic conditions of a real data transfer link in the Cloud, in each overall agenda of the generated links, the timeslots, available bandwidth and link rate of each router are generated based on parameters including $(startTime, deadline)$ pair, $dataSize$, $maximumBW$, $routers$, $events$ and $linkRates$. Among these parameters, $maximumBW$ is set to different values representing different data transfer speeds between different devices. $routers$ indicates the number of routers on the link, $events$ indicates the accumulated number of events of all the pipeline agendas, $linkRates$ indicates the link rate that the router can be working at and $routerPower$ is set according to the link rate of the router. The $routerPower$ value is obtained based on the research presented in [85]. Table 7.1 shows the range of these parameters in the simulation.

In this simulation, we do not involve the shutdown period as a parameter. By involving it the result is obvious: the minimum-speed strategy and maximum-speed strategy would consume more energy by allocating bandwidth during the shutdown period while LCRDT does not. This only magnifies the proportion of energy consumption reduced by LCRDT.

7.5.2. Energy Consumption Comparison

Figure 7.9 shows the average energy consumptions of data transfer by applying LRCDT and the other two strategies.
In the figure, each sub graph shows the energy consumptions for transferring data files ranging from 1MB to 10GB with different maximumBW transfer speed upper boundaries. It
can be seen that LRCDT is able to transfer data files with the least energy consumption for all sizes of data files under all three maximumBW values. Comparing to the other two data transfer strategies, LRCDT reduces at least 27.6% of the energy consumption. Under different maximumBW, however, the energy saving effect of LRCDT is different. Specifically, comparing to the maximum-speed strategy when maximumBW is higher, LRCDT is able to reduce more energy. It consumes on average 37.8% less energy when maximumBW is 10Mbps compared to 63% less energy when maximumBW is 1Gbps. In contrast, compared to the minimum-speed strategy, LRCDT is able to reduce more energy consumption when maximumBW is lower. On average, 27.6% energy consumption of data transfer can be saved when maximumBW is 1Gbps whereas 33.7% can be saved when maximumBW is 10Mbps. In addition to the discussions above, Figure 7.9 also shows that the energy saving effect of LRCDT gradually decreases when the data size becomes too high. This is because the maximum data size that can be transferred within the transfer period is limited. When the data size is close to this limit, minimumBW becomes close to maximumBW and all bandwidth that can be of use has to be allocated. In extreme cases, when the data size reaches the transfer limit, the energy consumption of all data transfer strategies becomes the same.

7.5.3. Task Completion Time Comparison

In addition to the energy consumption comparison mentioned above, we also compared the task completion time by applying the three data transfer strategies. Based on simulations conducted with different maximumBW and deadline values, Figure 7.10 shows the average completion time of transferring data files by using the three data transfer strategies. The horizontal axis is the data size and the vertical axis is the proportion of time from startTime to the end of the data transfer task in the maximum data transfer duration. From this comparison we find that the energy saving of LRCDT strategy is achieved without sacrificing too much data transfer time. Although it can be seen in the figure that the task completion time using LRCDT is more than the maximum-speed strategy, the task completion time is still much shorter than the maximum data transfer duration. In general, compared to the maximum-speed strategy that transfers data files with the shortest time, the average transfer time increment of LRCDT is 37.9%. Apart from the fast increment for transferring data files between 1MB and 1GB, for transferring the data files from 1GB to 10GB the average transfer time increment of LRCDT drops by 27.8%. Meanwhile, the average transfer time of LRCDT is 27.5% shorter in comparison to the minimum-speed strategy. This means that LRCDT can
finish the data transfer task with 27.5% less time. The reason for the fast increment in task completion time when data size is between 1MB and 1GB is that, when the data size is small, minimumBW is smaller than the current available bandwidth of most of the routers. Hence, LRCDT is able to allocate bandwidth without increasing the link rates of routers. The available bandwidth of each timeslot is allocated quickly so that the task completion time incurs a rapid increase. When the data size reaches about 1GB, the link rates of routers are increased so that more bandwidth can be allocated and the task completion time drops quickly.

![Figure 7.10 Comparison of average completion time](image)

7.6. Summary

In this Chapter, we presented a novel energy-efficient data transfer strategy called LRCDT (Link Rate Controlled Data Transfer) for improving the cost-effectiveness of Cloud data creation and recovery. Based on the assumption of the dedicated Cloud network with bandwidth reservation, LRCDT schedules bandwidth in a link rate controlled fashion to reduce the energy consumption specifically for ‘lazy’ data transfer, which is consistent with the characteristics of data transfer for data creation and recovery activities. In this way, LRCDT cooperates with PRCR for maintaining the big data in the Cloud and achieving the cost-effective data creation and recovery goal. We first presented the Cloud network model as well as the energy consumption model for network devices, and then the LRCDT was presented in detail.
Chapter 8  Conclusions and Future Work

In this chapter, we summarize the whole thesis. The structure of this chapter is organized as follows. In Section 8.1, we overview the contents of this thesis. In Section 8.2, we summarize the key contributions of this thesis. In Section 8.3, we present further discussions to the research and our future work.

8.1. Summary of This Thesis

The content of this thesis was presented in the following order.

- In Chapter 1, we introduced the background knowledge for our research, which are the definition of data reliability, current development of data reliability in the Cloud, distinctive features of Cloud storage and the lifecycle of Cloud data. Afterwards, we outlined the key issues of this research and presented a high level overview of the whole thesis.

- In Chapter 2, we intensively reviewed literatures on existing technologies related to the research. First, we presented our review for existing hardware reliability theories, in which disk reliability theories were reviewed specifically. Second, we presented our review for existing software-based data reliability assurance approaches, in which reviews for replication-based approaches and erasure coding-based data storage approaches were presented respectively. Third, we presented our review for existing data transfer approaches in distributed systems.

- In Chapter 3, we presented a motivating example with the analysis of our major research issues. First, we presented the details of the pulsar searching scientific application as the motivating example of our research. Second, we presented our analysis on the problem of cost-effective big data storage in the Cloud with data reliability assurance in detail, in which major factors of Cloud storage cost, data storage devices and schemes, Cloud network for data transfer during data creation and data recovery are addressed. Third, based on the analysis, we presented the details of our research issues.
• In Chapter 4, we presented our novel generic replication-based data reliability model in detail for describing the Cloud data reliability. First, despite of previous investigations, we determined further details of the model including reliability metrics, presentation type and failure rate pattern. Second, according to the reasoning process, we demonstrated the data reliability model in detail.

• In Chapter 5, based on the generic data reliability model, we presented our approach for calculating the minimum replication for meeting the data reliability requirement. First, we presented related formulas of the calculation approach as well as optimization of the formulas for reducing the computation complexity. Second, we discussed the application of the minimum replication as a benchmark for evaluation of replication-based data storage approaches. Third, we briefly summarized the satisfactory outcomes of the evaluation for the minimum replication calculation approach.

• In Chapter 6, we presented our novel cost-effective data reliability assurance mechanism PRCR (Proactive Replica Checking for Reliability) for maintaining the big data in the Cloud with a huge number of data files in a cost-effective fashion. First, we explained the principle of maintaining data reliability by proactive replica checking. Second, we presented the structure of PRCR, in which the two major parts: the user interface and the PRCR node are presented in detail. Third, we presented the working process of PRCR by following the lifecycle of a data file managed by PRCR in the Cloud. Forth, we presented two algorithms for optimizing the performance of PRCR, which are the minimum replication algorithm and the metadata distribution algorithm. Finally, we presented the evaluation of PRCR, in which the performance and cost-effectiveness of PRCR are evaluated by comparing with the widely used conventional 3-replica data storage strategy.

• In Chapter 7, we presented our novel energy-efficient data transfer strategy LRCDT (Link Rate Controlled Data Transfer) for reducing the data transfer cost incurred during data creation or data recovery processes. First, we presented the formulas for calculating data transfer deadline for data creation and data recovery processes respectively. Second, we presented the Cloud network model for Cloud with bandwidth reservation, in which four sub-models are presented. Third, we presented the energy consumption model of network devices in the Cloud. Fourth, we presented the LRCDT strategy in detail. Finally, we presented the evaluation of LRCDT, in which the energy consumption and task completion
time of the strategy are evaluated by comparing with the existing minimum-speed and maximum speed data transfer strategies.

By presenting all the contents as above, our cost-effective replication-based Cloud storage solution for reliability assurance of big data is comprehensively presented. Each part of the solution, including the data reliability model, algorithm, cost-effective data reliability assurance approaches during data creation stage, data maintenance stage and data recovery stage of the Cloud data lifecycle are unfolded to the readers comprehensively.

8.2. Key Contributions of This Thesis

In this thesis, our research focuses on the issue of providing cost-effective storage while meeting the reliability requirement for the big data in the Cloud. Based on systematic investigations to the existing distributed storage technologies and Cloud storage and network environments, we provide a systematic cost-effective Cloud data storage solution, in which the data reliability requirement of each data file is considered throughout the whole data lifecycle. Confronting the rapid development of data-intensive applications in the Cloud and the growth of Cloud data in a dramatic speed, the significance of this research is obvious. In particular, the major contributions of this thesis can be concluded as follows in four parts:

- First, a novel generic data reliability model for Cloud data storage is proposed for describing the reliability of Cloud data with multiple replicas stored on devices with variable failure patterns. As far as we know, this model is one of the few that investigate the data replication techniques with a variable disk failure rate.
- Second, a new minimum replication calculation approach is proposed for calculating the minimum replication that is needed for meeting the data reliability requirement. In addition, the minimum replication can also act as a benchmark for evaluating the cost-effectiveness of various replication-based data storage approaches. This approach is able to effectively determine the minimum number of replicas for the Cloud data storage with relatively small computing overhead (i.e. execution time).
- Third, an innovative generic data reliability assurance mechanism named PRCR (Proactive Replica Checking for Reliability) is proposed for maintaining the big data in the Cloud in a cost-effective fashion, while appropriate data reliability assurances are offered. It is able to provide data reliability management with a wide range of data
reliability requirements efficiently. Compared to the storage using the conventional 3-replica strategy, our PRCR can reduce between two-thirds to one-third of the storage cost, while the running overhead for PRCR itself is negligibly small.

- Fourth, an innovative energy-efficient data transfer strategy named LRCDT (Link Rate Controlled Data Transfer) is proposed for reducing the cost of data transfer activities that are intensively involved in data creation and recovery processes. The strategy could balance the trade-off between data transfer speed and energy consumption, and hence could benefit the cost-effective storage for data reliability in both the data creation stage and the data recovery stage. LRCDT is able to significantly reduce the data transfer energy consumption during data creation and data recovery processes, in which up to 33.7% of the energy consumption by using the minimum-speed strategy or 63% by using the maximum-speed strategy can be reduced. Such an energy saving outcome is achieved by sacrificing some data transfer time but without jeopardizing the deadline.

8.3. Further Discussion and Future Work

In this section, we first present some further discussions related to the PRCR (Proactive Replica Checking for Reliability) mechanism, and then the future work of the research in this thesis is presented.

8.3.1. Further Discussions

Cloud data storage concerns not only reliability but also other concerns such as availability and data access performance. These other concerns are not yet addressed in this thesis. With the “no more than 2 replicas” storage fashion of PRCR, there could potentially be some side-effects so that the data availability and data access performance are affected. However, it does not mean that storing more than two replicas for the data are not feasible with PRCR. Based on certain needs, any number of replicas can be created, and PRCR can certainly maintain all of them.

Despite of that, another thing that needs to be discussed is the generality of PRCR. As mentioned in Section 3.2, our research is based on the Cloud with a replication-based data storage scheme. However, PRCR is generic rather than specifically for a replication-based data scheme only. The combination of PRCR with an erasure coding-based data storage
scheme could be feasible for increasing the reliability of erasure coded data. Similar to what PRCR does to data replicas, it could proactively check the erasure coded data blocks periodically, and recover lost data blocks when found. Erasure coded data could be recovered before \( k \) data blocks are lost and less than \( n \) data blocks are available. By applying PRCR, the probability of losing \( k \) data blocks can be reduced, so that the reliability of the data could be improved. For the combination of PRCR with erasure coding based data storage, the data reliability model and data recovery process can be further investigated.

### 8.3.2. Future Work

Based on the current work in this thesis, future work can be conducted from the following aspects.

- First, at present, the entire cost-effective Cloud data storage solution is still at its validation stage, where the approaches provided are based on experimental environments. In the near future, our work will focus on implementing a prototype of the solution in the Cloud. Specifically, as the core of the entire solution, a PRCR prototype can be implemented based on such as Amazon Web Services, in which some further design of the mechanism may be conducted.

- Second, as mentioned in Section 8.2.1, the impact of our solution on data availability and data access performance due to reduction in data redundancy is not addressed at current stage. In the near future, comprehensive analysis as well as evaluations will be conducted for this issue, where effective approaches to minimize such impact may be proposed.


[70] 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.


### Appendix  Notation Index

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFR</td>
<td>Annualized Failure Rate</td>
</tr>
<tr>
<td>$A_j$</td>
<td>Disk $D$ surviving from $t_{j-1}$ to $t_j$</td>
</tr>
<tr>
<td>availableBW</td>
<td>Available bandwidth of the data transfer link</td>
</tr>
<tr>
<td>BA</td>
<td>Bandwidth allocation</td>
</tr>
<tr>
<td>$C$</td>
<td>Capacity of PRCR</td>
</tr>
<tr>
<td>CI($f$)</td>
<td>Checking interval of data file $f$</td>
</tr>
<tr>
<td>CIS</td>
<td>Set of checking interval values</td>
</tr>
<tr>
<td>$D$</td>
<td>Disk</td>
</tr>
<tr>
<td>deadline&lt;sub&gt;cre&lt;/sub&gt;</td>
<td>The deadline for data creation process</td>
</tr>
<tr>
<td>deadline&lt;sub&gt;rec&lt;/sub&gt;</td>
<td>The deadline for data recovery process</td>
</tr>
<tr>
<td>diff</td>
<td>Set of CI – scan cycle values of PRCR nodes</td>
</tr>
<tr>
<td>$d_q$</td>
<td>Disk $q$ in the storage system $S(d_1, d_2, d_3, \ldots, d_n)$</td>
</tr>
<tr>
<td>$E_{boot}$</td>
<td>Energy consumption for booting the router</td>
</tr>
<tr>
<td>$E_{halt}$</td>
<td>Energy consumption when the router is on halt</td>
</tr>
<tr>
<td>ET</td>
<td>Expected storage duration</td>
</tr>
<tr>
<td>$E(t_i)$</td>
<td>Pipeline status changing event at $t_i$</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$E_{work}$</td>
<td>Energy consumption when the router is working</td>
</tr>
<tr>
<td>$F(f_1, f_2, f_3, \ldots, f_m)$</td>
<td>Data file set composed of data files $f_1, f_2, f_3, \ldots, f_m$</td>
</tr>
<tr>
<td>$f_i(r_{i1}, r_{i2}, r_{i3}, \ldots, r_{ip})$</td>
<td>Data file $i$ stored with replicas $r_{i1}, r_{i2}, r_{i3}, \ldots, r_{ip}$</td>
</tr>
<tr>
<td>$L$</td>
<td>Event list</td>
</tr>
<tr>
<td>$LR$</td>
<td>Link rate</td>
</tr>
<tr>
<td>$MIN_S$</td>
<td>The minimum replication level for storing data file set $F(f_1, f_2, f_3, \ldots, f_m)$ in storage system $S(d_1, d_2, d_3, \ldots, d_n)$</td>
</tr>
<tr>
<td>$MTTF$</td>
<td>Mean Time to Failure</td>
</tr>
<tr>
<td>$\min_i$</td>
<td>The minimum replication level for data file $f_i$</td>
</tr>
<tr>
<td>$maximumBW$</td>
<td>Maximum bandwidth for the data transfer task</td>
</tr>
<tr>
<td>$minimumBW$</td>
<td>Minimum bandwidth for the data transfer task</td>
</tr>
<tr>
<td>node</td>
<td>PRCR node</td>
</tr>
<tr>
<td>nodes</td>
<td>Set of PRCR nodes</td>
</tr>
<tr>
<td>OA</td>
<td>Overall agenda of the data transfer link</td>
</tr>
<tr>
<td>$Power(t)$</td>
<td>Power consumption function with variable $t$</td>
</tr>
<tr>
<td>$P_i$</td>
<td>The disk failure rate pattern of disk $i$</td>
</tr>
<tr>
<td>Ranking</td>
<td>A two-dimensional data structure for recording and sorting the status of each router.</td>
</tr>
<tr>
<td><strong>$RA_k(t)$</strong></td>
<td>The data reliability assurance that should be provided for a storage duration of $t$ within the $k^{th}$ period of storage</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>$R_{ij}$</strong></td>
<td>Reliability of the data file stored from $t_{j-1}$ to $t_j$</td>
</tr>
<tr>
<td><strong>$RR(r_1, r_2, r_3, \ldots, r_m)$</strong></td>
<td>Set of reliability requirement for replicas $r_1, r_2, r_3, \ldots, r_m$</td>
</tr>
<tr>
<td><strong>$R(t)$</strong></td>
<td>Disk reliability function with variable $t$</td>
</tr>
<tr>
<td><strong>$RR(t)$</strong></td>
<td>Data reliability requirement for the storage duration of $t$</td>
</tr>
<tr>
<td><strong>$r_j(d_q)$</strong></td>
<td>The $j^{th}$ replica of $f_i$ which is stored in disk $d_q$</td>
</tr>
<tr>
<td><strong>$Scancycle(B)$</strong></td>
<td>Scan cycle of PRCR node $B$</td>
</tr>
<tr>
<td><strong>$SD_k$</strong></td>
<td>The length of the $k^{th}$ period of storage</td>
</tr>
<tr>
<td><strong>$S(d_1, d_2, d_3, \ldots, d_n)$</strong></td>
<td>Storage system composed of disk $d_1, d_2, d_3, \ldots, d_n$</td>
</tr>
<tr>
<td><strong>$S$</strong></td>
<td>Set of all the PRCR nodes in PRCR</td>
</tr>
<tr>
<td><strong>startTime</strong></td>
<td>Start time of the algorithm</td>
</tr>
<tr>
<td><strong>$T_k$</strong></td>
<td>Storage duration of the data file with $k$ replicas</td>
</tr>
<tr>
<td><strong>$T_{cycle}^i$</strong></td>
<td>Scan cycle of PRCR node $i$</td>
</tr>
<tr>
<td><strong>$T_{scan}^i$</strong></td>
<td>Metadata scanning time of PRCR node $i$</td>
</tr>
<tr>
<td><strong>$TS(t_i)$</strong></td>
<td>The timeslot start from $t_i$</td>
</tr>
<tr>
<td><strong>$TS$</strong></td>
<td>Set of timeslots</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Disk failure rate (constant)</td>
</tr>
<tr>
<td>$\bar{\lambda}$</td>
<td>Average disk failure rate (variable)</td>
</tr>
</tbody>
</table>