2021

Data Replication and Its Alignment with Fault Management in the Cloud Environment

Fei Xie

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Data Replication and Its Alignment with Fault Management in the Cloud Environment

Fei Xie 4647750

Supervisors:
Associate Professor Jun Yan
Associate Professor Jun Shen

This thesis is presented as part of the requirement for the conferral of the degree:
Doctor of Philosophy (Integrated)

University of Wollongong

School of Computing and Information Technology

Faculty of Engineering and Information Science

August 2021

Abstract

Nowadays, the exponential data growth becomes one of the major challenges all over the world. It may cause a series of negative impacts such as network overloading, high system complexity, and inadequate data security, etc. Cloud computing is developed to construct a novel paradigm to alleviate massive data processing challenges with its on-demand services and distributed architecture. Data replication has been proposed to strategically distribute the data access load to multiple cloud data centres by creating multiple data copies at multiple cloud data centres. A replica-applied cloud environment not only achieves a decrease in response time, an increase in data availability, and more balanced resource load but also protects the cloud environment against the upcoming faults. The reactive fault tolerance strategy is also required to handle the faults when the faults already occurred. As a result, the data replication strategies should be aligned with the reactive fault tolerance strategies to achieve a complete management chain in the cloud environment.

In this thesis, a data replication and fault management framework is proposed to establish a decentralised overarching management to the cloud environment. Three data replication strategies are firstly proposed based on this framework. A replica creation strategy is proposed to reduce the total cost by jointly considering the data dependency and the access frequency in the replica creation decision making process. Besides, a cloud map oriented and cost efficiency driven replica creation strategy is proposed to achieve the optimal cost reduction per replica in the cloud environment. The local data relationship and the remote data relationship are further analysed by creating two novel data dependency types, Within-DataCentre Data Dependency and Between-DataCentre Data Dependency, according to the data location. Furthermore, a network performance based replica selection strategy is proposed to avoid potential

network overloading problems and to increase the number of concurrent-running instances at the same time.

Three reactive fault tolerance strategies are also proposed for independent tasks and dependent tasks, respectively. A utility-based fault tolerance strategy is firstly proposed for more efficient independent task rescue by considering resource load and task attributes. In addition, a timeline-oriented reactive fault tolerance strategy is also proposed for independent tasks to achieve better cloud resiliency and load balancing performance. It further adds the timeline allocation method to strategically allocate the tasks rescued from the faulty data centre on the timeline of their replica-ready data centres. Finally, a novel PageRank based fault tolerance strategy for workflow rescue is proposed to achieve better task resilience ratio, workflow resilience ratio, and workflow continuity ratio. A modified PageRank algorithm is developed to prioritise the tasks in the workflow instances.

The simulations results show that all of the proposed six strategies achieve better cloud performance in different optimisation domains in comparison with the corresponding comparative strategies.

Acknowledgments

It was a long way from where I began, I always had mixed emotions when I looked back on this amazing PhD research journey. I remembered my smile when my first research paper was published although the first one maybe the worst one of mine. I remembered every single morning when I pushed myself from night to daylight. I remembered my disappointment when my research paper was rejected. All of the things in this research journey are very memorable. Maybe someday in the future, I cannot remember the context of my published papers but I will remember all people who help me and accompany me during this PhD research journey in the rest of my life. Therefore, first and foremost, I would like to show my deepest gratitude and appreciation to these people.

Firstly, I would like to thank my parents, my grandpa Xiaoxian Xie, my dad Limin Xie and my mom Xuanhong Fei. I am very grateful to their support during this long-time PhD research journey. They are all over 60 years old but they understand my choice to start this research journey although I will not be around them for a long time. They are always steadfast to offer all things I need. These things cannot be expressible but all are about love. I also want to thank my grandma Shujie Lv. My grandma has passed away in 2013 but her death is the most important reason for me to start this study journey out of China.

Then I would like to thank my principal supervisor Associate Professor Jun Yan and my co-supervisor Associate Professor Jun Shen who offered me a chance to be their PhD student in 2016. They provide the best guidance about how to become a qualified researcher and show me how to be a great person as well. They assist me to develop the content of my research papers and this thesis. In discussing with them, I could find something I missed or the drawbacks of my research works. The suggestions and

comments from them are useful and indispensable to achieve my research works. I cannot be here without their help in patience. Especially during this COVID-19 tough time, they always support me and offer their help to my study. I will be grateful to them forever in the rest of my life. I also want to thank Dr. Cong Cao, Mrs. Ding Yu and Miss. Ruoxi Sun who helps me a lot in my research and thesis. I also want to thank the University of Wollongong because they are really amazing and always offer so much useful supports to me, especially during this COVID-19 tough time.

I would also like to thank those academic journals or conferences which rejected or accepted my submission. The rejections offered so qualified review comments to me for improving my research works. The acceptances pushed me to work harder and gave the confidence to me. I also want to thank the organizers of ICSOC 2018 which I attended before, I made a lot of friends during the conference time and those friends are always connected to discuss the research works between them and me. It makes me extend my research view. Also, thanks to the examiners who examine my thesis. You are the last audience to my PhD research journey.

This PhD journey is not only about the academics but also about the daily life. I also want to thank my close friends, Miss. Jingxuan Wang, Mrs. Sophie Hazelton, Miss. Na Wu, Mrs. Xiaolei Luo, Mr. Zeyuan Wang, Dr. Kela Xiao, Mrs. Menglin Zheng, Miss. Yunshu Zhu, Miss. Xin Liu, Mr. Yi Lu, Mr. Yuan Sun, and Miss. Jingxian Wu, etc. I cannot list all names here due to the space limitation but they all accompanied and helped me in the daily life when I was a PhD student in the University of Wollongong.

Lastly, thanks to all people I met in the past years. Everything I experienced is better than what I thought.

Certification

I, Fei Xie, declare that this thesis submitted in fulfilment of the requirements for the conferral of the degree Doctor of Philosophy (Integrated), from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This thesis document has not been submitted for qualifications at any other academic institution.

_____

Fei Xie

23rd August 2017

Publication List

Xie, F., Yan, J., & Shen, J. (2017). Towards Cost Reduction in Cloud-Based Workflow Management through Data Replication. *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, 94–99. https://doi.org/10.1109/CBD.2017.24

Xie, F., Yan, J., & Shen, J. (2019). A Data Dependency and Access Threshold Based Replication Strategy for Multi-cloud Workflow Applications. *Service-Oriented Computing – ICSOC 2018 Workshops*, 281–293. https://doi.org/10.1007/978-3-030-17642-6_24

Xie, F., Yan, J., & Shen, J. (2020). A Bandwidth and Latency Based Replica Selection Mechanism for Data-Intensive Workflow Applications in the Multi-Cloud Environment. *Proceedings of the Australasian Computer Science Week Multiconference*, 1–8. https://doi.org/10.1145/3373017.3373030

Xie, F., Yan, J., & Shen, J. (2020). A Utility-Based Fault Handling Approach for Efficient Job Rescue in Clouds. In *Cloud Computing – CLOUD 2020* (pp. 49–63). Springer International Publishing. https://doi.org/10.1007/978-3-030-59635-4_4

Xie, F., Yan, J., & Shen, J. (2021). A Novel PageRank-Based Fault Handling Strategy for Workflow Scheduling in Cloud Data Centers. *International Journal of Web Services Research*, *18*(4), In Press. https://doi.org/10.4018/IJWSR.2021100101

Xie, F., Yan, J., & Shen, J. (2022). A Novel Independent Job Rescheduling Strategy for Cloud Resilience in the Cloud Environment. *Applied Computing and Informatics*, Accepted with Minor Revision.

Abbreviations

The abbreviations used in this thesis are listed as follows.

- FixD: Fixed data
- FlexD: Flexible data
- FFlexD: Free-Flexible Data
- CFlexD: Constrained-Flexible Data
- W-DCD: Within-DataCentre Data Dependency
- B-DCD: Between-DataCentre Data Dependency
- HDD: High-Dependent Data
- HAD: Hot-Access Data
- NPRS: Network Performance Oriented Replica Selection
- UBFT: Utility-Based Reactive Fault Tolerance
- HDFS: Hadoop Distributed File System
- RR: Round Robin
- JSQ: Join the Shortest Queue
- TRR: Task Resilience Ratio
- HEFT: Heterogeneous Earliest-Finish-Time
- TOFT: Timeline-Oriented Reactive Fault Tolerance
- HEFT-T: Heterogeneous Earliest-Finish-Time with TOPSIS
- PRFT: PageRank Based Fault Tolerance
- DAG: Directed Acyclic Graph
- WRR: Workflow Resilience Ratio
- WCR: Workflow Continuity Ratio

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Research Background

In recent years, many organizations face challenges when managing large amount of data generated from various business activities as the business has had a rapid growth due to digitalization development. There are many reports predicting the exponential data growth beyond 2020. For example, as shown in Figure 1.1, according to IDC Global Datasphere in November 2018, the total amount of data around the world has dramatically increased over the past 10 years from 2010 to 2021 and will continue to grow total 171% to reach 175 zettabytes in 2025, with most of the data residing in the cloud environment [93].

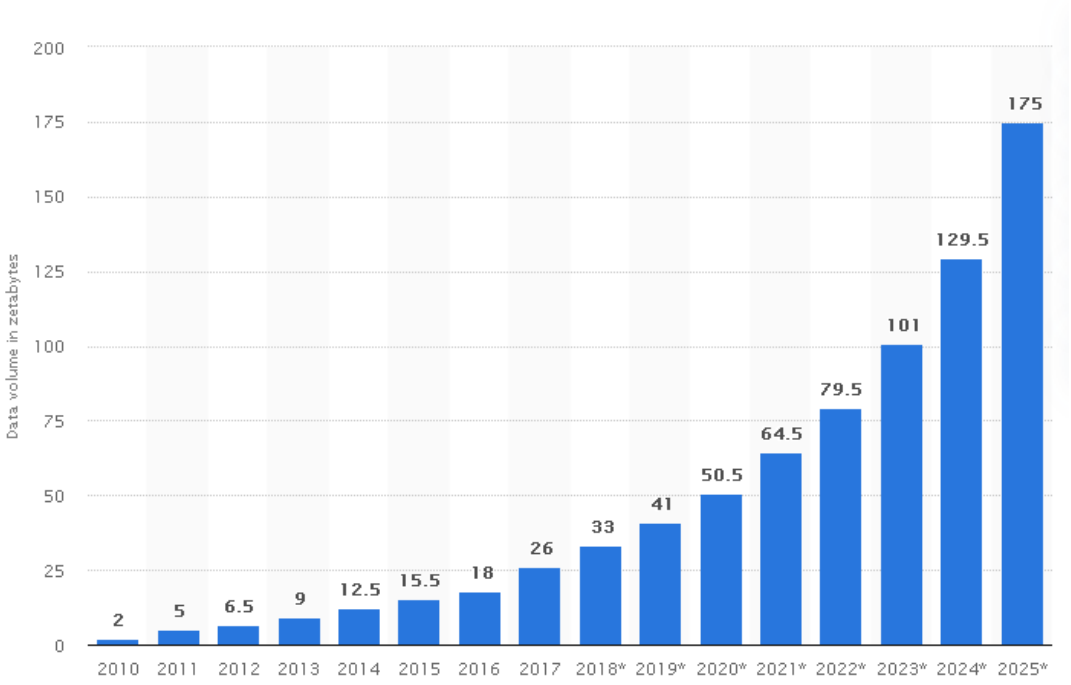

Figure 1.1 The exponential data growth according to IDC Global Datasphere [93]

The exponential data growth in both volume and speed has caused a variety of challenges:

- Network overloading

The exponential data growth may cause dramatical increase of data access load which occupies vast amount of resources. Such load increase may further cause the resource

overloading problems.

- Low data processing efficiency and effectiveness

Processing of the exploding volume of data may take much extra time, especially for those data users who need to cooperate with other data users. It may lead to lower efficiency and effectiveness in the data-intensive applications.

- High system management complexity

The exploding volume of data may need a variety of systems to store, transfer and process them. Therefore, it may lead to the high system management complexity to data service providers and data owners.

- Extra power, cooling and space limitations

More and more data servers should be deployed to store and process the exploding volume of data. All those data servers need to be placed properly. The increased number of data servers need extra power, cooling systems and extra physical space to store and work properly.

- Significant shortage of relevant skills

With the exponential data growth, more and more data need to be processed by users, data owners and data service providers. However, some of users, data owners and data service providers may lack relevant skills.

- Application performance deficiency

With the exponential data growth, the application execution may take longer time as it may need to access more data. This may impact the responsiveness of those applications adversely.

- Out-of-control cost growth

The exploding volume of data will increase different costs, such as data management cost, data transfer cost, and data storage cost, etc. Therefore, the cost may be out-of-control to users, data owners, and data service providers.

- Inadequate data security

The exponential data growth may cause inadequate data security protection because the current data security tools or methods may be outdated or overloaded. Therefore, more advanced security tools or methods may be required to be developed.

- Sluggish agility responding to changing business

The rapid changing business environments exist everywhere. The business agility can be sustained by maintaining and adapting the offered business services to meet the customer requirements. However, the exponential data growth may delay the maintenance and adaption processes and decrease the responsiveness to the changing business environment. Therefore, it may cause sluggish business agility to the cloud users.

With this continuing data explosion, a high-performance computing environment is urgently required. The emergence of cloud computing technologies constructs a novel paradigm to address the problems caused by the data explosion [72]. It allows heterogeneous computing environments to satisfy the global user requirements. The cloud environment can also help users minimise data loss risks and downtime to achieve better quality of service. The heterogeneity of the cloud environments allows many competitive advantages in comparison with the traditional distributed computing environments [66]. For example, from the scale economics perspective, dynamic provisioning and lower capital cost are two of the most significant competitive advantages bringing from the heterogenous cloud environment [22].

To address the potential negative influences of the continuing data explosion in the cloud environment, data replication has been proposed as one of the most significant data management approach to strategically distribute the data access load into multiple cloud data centres. The data replication strategy has been a research area of interest for many years. By creating multiple data copies at multiple cloud data centres, data replication can achieve a variety of benefits such as response time decrease, data availability increase, and more

balanced resource load.

However, the cloud environment is subject to many types of faults, which may lead to a series of negative influences on the cloud data centres in a chain reaction. The data centres in the cloud environment may temporarily be unavailable due to the negative fault impacts. Therefore, fault tolerance becomes one of the biggest challenges in the cloud environment to ensure the quality of service and user satisfaction. The fault tolerance techniques are the major tools being used to achieve a successful and continuous fault handling solution. Many types of fault tolerance techniques have been proposed before.

Particularly, the data replication itself is also a fault tolerance technique to improve the cloud performance when encountering faults. A comprehensive data replication strategy can guide the establishment of a replica-applied cloud environment. The replica-applied cloud environment can protect the cloud environment from being affected by the upcoming faults as much as possible. Tasks at the faulty data centre can be strategically resubmitted or migrated to other proper-working cloud data centres with the required data replicas in the replica-applied cloud environment.

Although the replica-applied cloud environment can protect the cloud environment against the upcoming faults, there are still many types of reactive fault tolerance techniques, such as retry [35], checkpointing/restarting [83] and user defined exception handling [89]. The reactive fault tolerance aims to reduce the negative influences after the faults already occurred.

Task resubmission and task migration are also two significant reactive fault tolerance techniques. The task scheduling method is the core method of task resubmission and task migration. These two reactive fault tolerance techniques enable the automatic task rescue at the faulty data centre, aiming to successfully complete as many as affected tasks.

1.2 Key Research Issues

In this thesis, the key research issues of data replication and its alignment with fault management in the cloud environment are investigated. Data replication strategies and fault tolerance strategies are two key research areas. Several general research questions are listed below:

- Which data should be selected to create its replicas?
- How many replicas of each data should be created in the cloud?
- Where should these replicas be situated?
- Which replica is suitable to select for data access?
- How to develop a suitable reactive fault tolerance strategy for the replica-applied cloud environment?

In more details, this thesis will address the following research problems.

- Firstly, an appropriate replica creation strategy is necessary and indispensable in a large-scale cloud system [67]. Both external data attributes and internal data attributes have significant influences on the data. The external data attribute refers to the attribute which the data correlates to the external environmental factors such as users and cloud service providers, while the internal data attribute refers to the attribute which the data correlates to other data. However, most of the literature only considers the same type of data attributes to constrain the replica creation. Considering only one type of data attributes may lose the comprehensiveness of the data analysis when developing replica creation approaches. Therefore, both external data attributes and internal data attributes should be jointly considered when developing replica creation process.
- Secondly, each data centre can be recognized as an individual host entity in the cloud environment. A data stored in a cloud data centre may have multiple data relationships to other data inside data centre and outside data centre. The data

relationship between this specific data and its correlated data inside the same data centre can be seen as local data relationship, while the data relationship between this specific data and its correlated data outside the same data centre can be known as remote data relationship. Therefore, the data relationship situations between inside data centre and outside data centre should be distinguished when making the replica creation decision.

- Thirdly, most of the current replica selection strategies lack the consideration of the potential negative impacts among multiple concurrent-running cloud application instances under limited network capability. Hence, those replica selection strategies might not achieve the optimal network performance when there are heavy data access needs in the cloud environment. Therefore, a replica selection strategy for load balancing with the comprehensive analysis of the cloud network capability is urgently required.
- Fourthly, data replication itself is also a fault tolerance technique. The replica-applied cloud environment can protect the cloud environment from being affected by faults beforehand. However, it is not sufficient to achieve the optimal cloud performance by adopting the replica-applied cloud environment only when encountering faults. Therefore, the reactive fault tolerance strategies may also be required to assist with the replica-applied cloud environment to further achieve better cloud performance.
- Fifthly, the data replication strategy always contains three domains, replica creation, replica placement, and replica selection. Besides, there are a variety of fault tolerance techniques. However, there is not a general management framework for cloud environments, which aligns the data replication and the fault management together. Hence, a comprehensive data replication and fault management framework is required to enable a complete management chain to the cloud environment for better cloud

performance, which aligns the data replication strategies with the fault tolerance strategies.

- Sixthly, independent tasks and dependent tasks have different task features. There are no task relationships among independent tasks while there is at least one task relationship among dependent tasks. Therefore, the fault tolerance techniques for independent tasks and dependent tasks should be differentially developed to ensure the applicability of the fault tolerance strategies.

1.3 Research Contributions

To overcome the problems mentioned above, this thesis proposes three data replication strategies and three fault tolerance strategies in the cloud environment by following the proposed data replication and fault management framework. The proposed two replica creation strategies and one replica selection strategy can be used to create a replica-applied cloud environment. Besides, the proposed three fault tolerance strategies are all reactive fault tolerance strategies by aligning with the proposed replica selection method. They aim to reactively protect the task completeness when encountering faults in the cloud environment. The contributions of this thesis are summarised as follows.

- A data replication and fault management framework is proposed to enable a decentralised management chain to the cloud environment in Chapter 3. It adopts multiple user platforms and data centre platforms. The platforms have their inside modules to achieve different management functionalities. Those modules are inter-connected and inter-cooperated to achieve a comprehensive management chain for the cloud environment.
- Three data replication strategies are proposed in the form of two replica creation strategies and one replica selection strategy in Chapter 4. Various evaluation

parameters are considered in these data replication strategies, such as data dependency, data size, access frequency, network performance measurements, and resource load. Different evaluation methods have been applied in these data replication strategies to set the evaluation constraints for constraining different data replication decision-making processes. For example, the threshold-based evaluation method and the normalisation-based evaluation method are two evaluation methods applied in these data replication strategies.

- Three reactive fault tolerance strategies are proposed on the basis of the replica-applied cloud environment in Chapter 5 and Chapter 6. Various evaluation parameters are also considered in these reactive fault tolerance strategies such as network performance measurements, task attributes, task urgency, task utility, resource load, and task dependency, etc. Particularly, different reactive fault tolerance strategies are proposed for independent tasks and dependent tasks, respectively, in the form of two reactive fault tolerance strategies for independent tasks and one reactive fault tolerance strategy for dependent tasks. Different evaluation methods are also applied in these reactive fault tolerance strategies such as the normalisation-based evaluation method and the priority-based evaluation method.
- The case study and the simulations show that the proposed strategies achieve better performance than other relevant comparative strategies in terms of different optimisation objectives.

1.4 Thesis Roadmap

The rest of this thesis is structured as follows.

Chapter 2 presents a literature review of cloud computing technology, data-intensive applications and scientific workflows, data replication strategies, fault tolerance strategies,

and task scheduling strategies. The problem statement and the research insights are also demonstrated.

Chapter 3 introduces the proposed data replication and fault management framework. This framework includes different platforms and modules to achieve a complete management chain for the cloud environment. A set of general notations and basic definitions are also presented in this chapter.

Chapter 4 illustrates three data replication strategies in the form of two replica creation strategies and one replica selection strategy. The first replica creation strategy aims to reduce the total cost of the cloud application execution by considering the data dependency and the access frequency. The second replica creation strategy aims to achieve the optimal cost reduction per replica by identifying a recommended access frequency threshold value. Furthermore, the proposed replica selection approach aims to minimise the potential network overloading problems and increase the number of concurrent-running instances at the same time by considering the network performance at each data centre.

Chapter 5 presents two reactive fault tolerance strategies for the independent tasks in the replica-applied cloud environment. The first fault tolerance strategy fully considers the network performance metrics and the task attributes for more efficient task rescue in the replica-applied cloud environment. The second fault tolerance strategy further adds the timeline allocation into consideration to achieve better cloud resiliency and load balancing performance.

Chapter 6 demonstrates a reactive fault tolerance strategy for the workflows in the replica-applied cloud environment. This strategy aims to achieve better task resilience ratio, workflow resilience ratio, and workflow continuity ratio. It takes the task attributes, the timeline scenario at each data centre, and the overall cloud performance into account. This strategy innovatively incorporates the modified PageRank algorithm into the workflow

scheduling research when handling faults.

Chapter 7 comprehensively discusses the contributions of this thesis and the applicability of each proposed strategy. This chapter also examines the limitations of the proposed strategies, such as optimisation objective diversity, replica placement simplification, workflow type limitation, lack of experiments, and the applicability to the server level or the cloud service provider level.

Chapter 8 concludes the research work in this thesis and discusses some future research directions.

Chapter 2 Literature Review

2.1 Cloud Computing

Cloud computing technology has been widely used to alleviate massive data processing challenges with its on-demand services and distributed architecture. It uses and combines different computing resources such as servers, databases, networks, software applications, and a series of relevant technologies to complete the tasks on demand instead of owning and operating those resources and technologies by organizations themselves [73][126].

In particular, during the tough time of COVID-19, traditional IT shows more and more drawbacks while cloud computing offers a lot of competitive advantages such as remote office work and continuous business operations.

2.1.1 Comparisons between traditional IT and cloud computing

Compared with traditional IT, cloud computing offers a variety of benefits, such as greater cost effectiveness, higher responsiveness to market, better scalability, more flexible elasticity, increased cooperation efficiency, improved reliability, and more durable business continuity.

- Greater cost effectiveness and efficiency

Traditional IT needs the users to construct their computing resources on-premises such as IT infrastructure and software applications by evaluating the data processing requirements inside the organization and outside the organization. It also needs the users to update their computing resources based on their data growth and traffic surge. Purchasing and maintaining computing resources are always costly. Differently, cloud computing enables the “pay-as-you-go” cost schema to pay for the required computing resources only for eliminating the relevant infrastructure costs, the application

development costs and the maintenance cost, etc [7]. This increases the cost effectiveness and efficiency [21][75].

- Higher responsiveness to market

As mentioned above, traditional IT needs users to establish the computing resources on-premises. This might waste time to offer a complete computing capability because purchasing and deploying the computing resources in an organization may take weeks or months. In contrast, cloud computing enables quick deployment of computing resources and thus increases the responsiveness to users [105].

- Better scalability, more flexible elasticity and better security

As mentioned above, traditional IT needs the users to update their computing resources on-premises based on their business growth and traffic surge. Differently, cloud computing enables the users to scale their workload on the cloud servers and automatically adjusts the offering of computing resources for better scalability and more flexible elasticity [123][146].

Besides, as discussed above, traditional IT always needs the users to host their computing resources on-premises. Therefore, there might be many physical and logical security drawbacks and loopholes. Working with a cloud can significantly enhance security because the cloud service provider can keep high-level data security by adopting high-level physical security systems, virtual private clouds, encryptions, and API keys.

- Improved reliability and more durable business continuity

Traditional IT hosts the data on-premises and always has frequent data loss risks and downtime because of the infrastructure issues and the low security. Cloud computing enables a heterogeneous cloud environment with a complete redundancy plan including global networks, data backups, and disaster recovery plans to achieve high reliability.

The cloud service providers will keep the ongoing business success and make the business more affordable and less disruptive.

2.1.2 Cloud service models

There are three cloud service models, Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS) [37][53][88][109][110][112]. Primarily, the logical "data pool" often relies on multiple physical servers in the cloud data centres which are owned by IaaS service providers. Besides, a variety of cloud applications are offered by SaaS service providers. Apart from that, many cloud platforms are constructed to integrate multiple cloud applications with the cloud infrastructure by PaaS service providers. The IaaS service refers to the tangible physical devices which are located in a cloud data centre, including servers, data storage devices, virtual computers, and network devices [32][87]. The IaaS service providers also offer the hardware systems such as air conditioning systems, firefighting systems, backup services, and electrical power systems to ensure the quality of IaaS services. Authorised users can easily access the data stored on the cloud infrastructure via Internet [32].

The SaaS service is a model that the cloud users can order and receive a variety of cloud applications on demand via the Internet instead of installing and updating the applications on their physical computers or servers [52]. There are three major features of the SaaS service, such as multi-tenant efficiency, scalability, and configurability. However, not all cloud applications contain all of these three major features and a cloud application may have one or two features only [9][49].

The PaaS service is the model between IaaS and SaaS as they can integrate the cloud applications on the platform via the Internet and connect the cloud applications to the cloud infrastructure [23]. The cloud users only need to manage the cloud application deployment and the application hosting environment by adopting the PaaS services.

Nowadays, many cloud service providers, such as Amazon, Microsoft, and Google, integrate IaaS, SaaS, and PaaS to offer a comprehensive cloud service.

2.1.3 Cloud architectures

There are a number of common cloud architectures including public cloud, private cloud, community cloud, and hybrid cloud [76].

- Public cloud architecture

The computing resources in a public cloud architecture are owned and operated by the cloud service providers [135]. They are always shared resources, which can be redistributed to multiple tenants via the Internet [104]. The public cloud architecture can achieve a variety of benefits such as operation cost reduction, easy scalability, and low maintenance cost.

- Private cloud architecture

The computing resources in a private cloud architecture are owned and operated by the organizations themselves in their on-premise infrastructures [50][85]. They can also be operated at the leased space in geographically scattered colocation facilities. The private cloud architecture achieves higher customisation and stronger cloud security than the public cloud architecture.

- Community cloud architecture

The community cloud achieves the communities of the consumers that experience the same data [45]. A single organization or multiple organizations can be organized in the cloud community.

- Hybrid cloud architecture

A hybrid cloud architecture integrates the private cloud architecture and the public cloud architecture [79]. It enables both the working efficiency of the public cloud architecture and the high data security of the private cloud architecture. The hybrid

cloud architecture allows organizations to manage their workloads based on their data security requirements. The organizations can freely convert between the private cloud architecture and the public cloud architecture if needed.

2.1.4 Multi-cloud environment

Some users may need to globally deploy their work. A multi-cloud environment uses two or more cloud computing services to share the workload across multiple cloud service providers all over the world. It is commonly used by several popular cloud platforms such as OpenStack and Microsoft Azure.

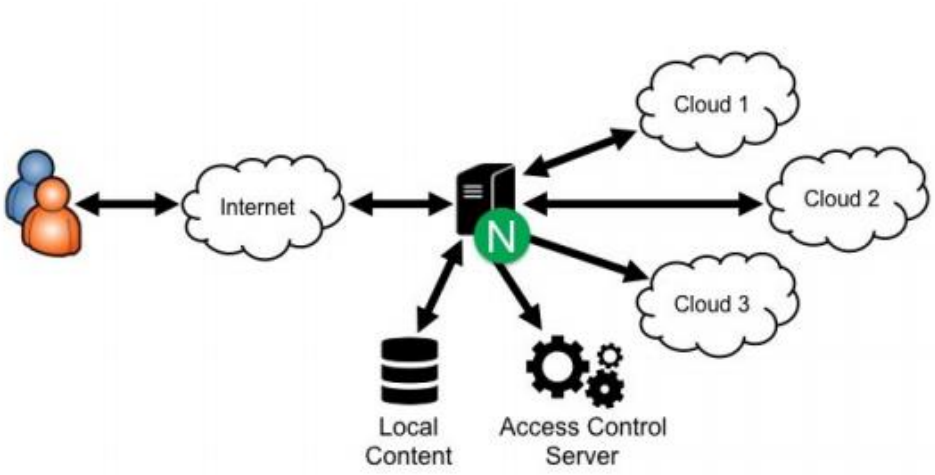

Figure 2.1 An example of multi-cloud architecture

Multi-cloud architecture is always used to support global or cross-regional collaborative work by using cloud infrastructure in multiple cross-regional locations [116]. In this case, it offers more agile and scalable cloud services than using a single cloud service [121]. It helps cloud users avoid the single-vendor lock-in problem. Although the multi-cloud architecture provides an appropriate computing environment to execute the global or cross-regional collaborative work, there are also two critical problems. The first problem is how to appropriately choose the schema of data hosting and task execution, and another problem is how to meet different service requirements [140]. An example

of the multi-cloud architecture is shown in Figure 2.1.

2.2 Data-Intensive Applications and Scientific Workflows

As mentioned above, there are multiple types of cloud architectures such as private cloud, public cloud, and hybrid cloud, etc. These cloud architectures are commonly applied in different cloud service providers for executing data-intensive applications [140].

Data-intensive applications are typically very complex and take a long execution time. They usually contain a large number of independent and dependent tasks. In particular, the scientific workflows as one of the data-intensive applications have been adopted in a wide range of research areas, such as astronomy, high-energy physics, bioinformatics [64], nuclear simulation, and earthquake engineering [120]. The scientific workflows can create an automated way to specify, execute, monitor, and track the data-intensive and highly-structured scientific research processes [24]. They consist of a large set of computational tasks which operate on the input data and generate a set of intermediate data [117].

Because of these features, the cloud environment is one of the most suitable environments for executing scientific workflows [120][124]. Firstly, as the data size increased, scientists only need to request more computing resources from the cloud service provider. Secondly, not all resources are required when performing one task, scientists can request the required resource to perform the task on demand. Thirdly, the total workflow execution cost will depend on how many computing resources the scientific workflows exploited, and then the scientists can compare the total cost with their budget to adjust the resource exploitation. Lastly, as mentioned above, the scientific workflows are always complex, highly-structured, global-collaborative, cross-regional, and data-intensive. Hence, the cloud environment can achieve global

collaborations among scientists all over the world to conduct their research together [138].

There are many famous types of scientific workflows in the real world. The Montage scientific workflow was established by the NASA/IPAC Infrared Science Archive [10]. It is an open-source toolkit, which aims to generate custom mosaics of the sky by using input images in the Flexible Image Transport System format. The CyberShake workflow is adopted in the Southern California Earthquake Centre to characterise the earthquake threats by using the Probabilistic Seismic Hazard Analysis technique [10]. The LIGO Inspiral Analysis workflow aims to detect gravitational waves which are produced by various events in the universe [10]. It is used for the data analysis of the data collected from the coalescing of compact binary systems. The SIPHT program uses an automated search workflow to search the sRNA encoding-genes for all of the bacterial replicons in the National Centre for Biotechnology Information database [10].

2.3 Data Replication

As the data-intensive business increases, an enormous amount of data is generated as shared resources. The size of data is always measured in terabytes or petabytes. Cloud computing is commonly adopted to store and process an enormous amount of data. Data replication has been proposed as a data management approach, which creates multiple data copies into multiple cloud data centres [19]. Data replication has been an area of research interest in the past decade in World Wide Web, peer-to-peer networks, ad-hoc and sensor networking, grid environments, mesh networks, and most importantly cloud environments [1].

In current cloud environments, different data replication strategies are commonly deployed in different cloud data centres. Thus, accessing data can be strategically distributed to multiple cloud data centres to optimise the data access load and the overall

cloud performance by adopting different data replication algorithms [78]. Data replication can offer the following benefits:

- The data replication strategy guarantees fast data access for the tasks in the cloud environment, especially for the tasks in the data-intensive applications. Multiple concurrent-running instances may access the same data at one specific cloud data centre. The resource contention may cause a performance bottleneck to this data centre. This bottleneck can be eliminated by applying data replication strategies, which results in more balanced resource load in the entire cloud environment [12][13]. At the same time, multiple replicas can also improve data availability [31][65][136].
- The data replication strategy can reduce the data access distance to the required data [46] [61]. Some required data may be able to be replicated to the local data centre so that those data can be accessed locally to reduce the data movement [82]. By doing this, the data management cost [38][59] and the response time [115] can also be reduced.
- In the cloud environment, unexpected faults can happen at any time [103]. The data replication strategy can guarantee data reliability because the tasks at the faulty data centre might be rescued and completed in time by accessing other required data replicas after the fault occurred [54][56][57][71].

In most of the literature, data replication is commonly classified into two main groups, static data replication strategies and dynamic data replication strategies [6]. Static data replication strategies rely on deterministic policies. The number of replicas and the host nodes for the replicas are commonly well-defined and pre-determined at the build-time stage. The static data strategies can be easily implemented but it is not often applied because of its limited adaptability to the dynamic environment [78]. Dynamic data

replication strategies dynamically make the intelligent data replication solutions depending upon the dynamic environment situations [78].

Replica creation, replica placement, and replica selection have been identified as three major sub-areas of data replication research. Replica creation is the strategy of creating a suitable number of data replicas for the necessary data [108]. Normally, replica creation strategies include some of the following phases:

- Analysing and modelling the relationship between the number of replicas and the system availability
- Identifying the data importance and triggering the replica creation process when the data satisfies the replica creation constraints
- Determining a suitable number of replicas to satisfy the system requirement

Replica placement is the strategy of placing data replicas to the appropriate cloud data centres [69]. Multiple placement constraints can be set to guide the appropriate cloud data centres to place the replica.

Replica selection is the strategy of selecting the optimal replica-ready data access routes for the tasks in the cloud environment [142]. Various parameters have been considered in the different replica selection algorithms, such as data access cost, data maintenance cost, access latency, resource load, workload, storage load, task execution time, and response time, etc.

2.3.1 Replica creation

Many replica creation strategies have been proposed in the past decade. In [91], the authors propose a Fair-Share Replication (FSR) strategy that takes both access load and storage load into account to determine the replica creation. An average access frequency is used to compare with the access frequency of the targeted datasets for identifying the popular file and ranking the file.

In [18], the authors propose a Latest Access Largest Weight (LALW) strategy in order to select a popular file and calculate a suitable number of copies and grid sites for data replication in data grids by considering the access frequency to exhibit the importance for the access history in different time intervals.

In [56], the authors propose a Cost-Effective Incremental Replication (CIR) strategy to manage the data reliability of each data centre in a cost-effective way. An incremental replication method is applied to determine when the replica should be created. The number of replicas is minimised by predicting the additional replica creation to ensure the reliability requirement for achieving the cost-effective replica management goal.

In [118], the authors propose a threshold-based file replication strategy to dynamically make the file replica creation based on the file popularity and the file request processing in case of node failure without the user intervention. The threshold-based file replication strategy carries out the file replication when the total number of the access requests for a particular file reaches the threshold value.

In [30], a Dynamic Cost-aware Re-replication and Re-balancing strategy (DCR2S) is proposed for the knapsack problems in three phases, by identifying the suitable data file and the number of data file replicas to replicate to appropriate locations and determining the additional required replication for satisfying the available requirement.

In [125], the authors propose a CDRM strategy as a cost-effective dynamic replication management scheme. They propose a novel way to capture the relationship between availability and replica number. The minimum replica number is computed under a certain availability. The purpose of CDRM strategy aims to provide the cost-effective availability and improve the load balancing performance. By analysing the workload change and the storage resource, the CDRM strategy dynamically re-distributes the workloads among different data centres. At the same time, it maintains the number of

replicas at each data centre to satisfy the availability requirement of each data centre at low cost. Besides, the replicas are dynamically placed into data centres to distribute the workload based on the resource load and the utilisation intensity of each data centre. This is achieved by calculating the capacity and the blocking probability of each data centre.

In [8], the authors propose a three-level replica management strategy called RTRM to improve the network utilisation and service response time. The RTRM strategy consists of replica creation level, replica placement level and replica selection level. It evaluates the average response time to automatically control the replica creation and the number of replicas by adopting a threshold-based method. The bandwidth situation is predicted among the replica servers based on the upcoming requests in the RTRM strategy. It will be combined with the number of replicas and the network transfer time to control the replica placement and selection processes.

In [107], the authors propose a dynamic data replication strategy called D2RS with three phases. These three phases cover two data replication research area, replica creation and replica placement. They address two major research questions: which, when and where data file should be replicated; and how many replicas should be created. In the D2RS strategy, the data access information is used to identify a popular file. A threshold-based method is used to compare the file popularity, which aims to identify which data file should be replicated. Besides, the number of replicas is determined upon the reasonable growth of the file availability. Then a balanced replica placement method is applied by evaluating the data access information from the directly connected data centres.

In [127], the authors propose a cost-effective data replication strategy to approximately minimise the data management cost. The access frequency and the average response time are considered to determine which data should be replicated in the cloud

environment by applying a threshold-based evaluation method. Besides, the number of replicas and the storage destinations can be decided according to the location problem graph and the minimum management cost.

In [31], the authors propose a dynamic, cost-aware data replication strategy by identifying the minimum number of replicas to satisfy the desired availability, to get the maximum value and to keep the total weight less than or equal to the peak budget at the same time.

In [143], the authors propose a dynamic file heat and node load based replica creation strategy to solve the excessive replica creation problem by jointly considering the characteristics of the hybrid cloud environment. They firstly propose an initialisation strategy to dynamically decide the number of replicas based on the user requirements. Then file heat history, access frequency, and file change rate are considered for file heat formulation. Besides, a dynamic file heat and node load based adjustment schema is created to dynamically adjust the number of replicas to further reduce the average response time and improve the overall cloud performance.

In [58], a dynamic data replication strategy is proposed with the consideration of both the tenant budget and the provider profit to satisfy the data availability and the performance requirements. A cost model is developed to calculate the minimum number of replicas to maintain the optimal data availability. The replica creation will be initiated when the pre-calculated number of replicas or the response time is not satisfied and the profit can be positive to the cloud service provider. The replica placement uses query scheduling techniques to balance the parameters between the load balancing and the tenant budget.

In [36], the authors propose a novel dynamic predicted replication strategy (DPRS) to predict the future file access and periodically calculate the number of replicas based on

the real access history and the future access. A calculation model is proposed to calculate the optimal number of replicas based on the number of accesses. A single exponential smoothing method is applied to predict future file access.

In [80], a data replication strategy called RSPC is proposed to satisfy the cloud performance, the minimum availability, and the cloud service provider profit at the same time. A threshold-based replica creation method is applied to initialize the RSPC data replication strategy. Then a new replica will be created if a suitable replica placement solution can be heuristically identified based on the evaluation of response time and the cloud service provider profit. The penalties and the data replication cost are considered in the estimation process of the cloud service provider revenue and expenditure.

2.3.2 Replica placement

Many replica placement strategies have been presented in the past years. In [101], a dynamic popularity based replica placement (PBRP) strategy is proposed for hierarchical data grids to shorten the job execution time and reduce the bandwidth consumption. A threshold-based popularity-driven guide model is developed to guide the replica placement. The authors present the Adaptive-PBRP (APBRP) algorithm to dynamically set the popularity threshold value according to the data request arrival rates.

In [59], the authors propose a data replication strategy to solve the QoS-aware data replication (QADR) problems to minimise the data replication cost and the number of QoS-violated data replicas. To solve the QADR problems, a greedy algorithm called high-QoS first-replication (HQFR) is proposed to assign the precedence for the cloud applications in the cloud environment. Besides, the authors find that the optimal solution of the QADR problem can be identified by formulating the QADR problem as

an integer linear programming formulation. Therefore, they transfer the QADR problem to the minimum-cost maximum-flow problem and propose a novel algorithm to solve the minimum-cost maximum-flow problem to identify the optimal replica placement solution based on the QoS requirements.

In [64], the authors propose a group based genetic replica placement algorithm collaborated with the analysis of the scientific application characteristics to reduce the data transmission in the cloud environment by considering the data size and the bandwidth situations among data centres.

In [65], a Multi-objective Optimized Replication Management strategy (MORM) is proposed to balance the trade-off among five optimisation objectives, including mean service time, mean file unavailability, load variance mean access latency, and energy consumption, to make a near-optimal data replication solution. Authors develop the mathematical models to formulate the five optimisation objectives with the consideration of multiple parameters such as data size, data access rate, failure probability, data transfer rate, and resource capacity at the same time. The feasible file is founded by identifying the replication factor based on the integrity constraint and the capacity constraint. The feasible individuals can be placed among data centres, which consider the five optimisation objectives. A suitable number of replicas is maintained to achieve the optimal performance with respect to five optimisation objectives for each feasible individual.

In [128], the authors propose a QoS-aware data replication and placement strategy to approximately evaluate the big data analytics query in the cloud environment. The QoS-aware data replication and placement strategy aims to strategically create and place the data samples to the data centres in the cloud environment by considering a trade-off between the query evaluation cost and the query evaluation error bound. Two efficient

algorithms are developed for single approximate query and multiple approximate queries, respectively. Then a heuristic algorithm is proposed to evaluate a set of approximate queries to minimise the evaluation cost and the delay requirements.

In [70], the authors propose a hierarchical data replication strategy (HDRS) to reduce the response time and the bandwidth usage. A multi-tier structure for data replication is firstly proposed to offer flexible and scalable management for vast files. Then the HDRS strategy develops the replica creation strategy, the replica placement strategy, and the replica replacement strategy. The replica creation strategy identifies the popular file according to the exponential growth or the decay rate and then creates the replicas. The access load and the labelling technique are applied to decide the replica placement. The replica replacement is based on the evaluation of the number of future file access and the file size.

In [27], the authors propose an energy-aware data replication strategy to decide the number of required replicas and the locations for those replicas. A hybrid metaheuristic algorithm, named HPSOTS, is developed to generate high-quality data replication solutions by combing the Particle Swarm Optimization algorithm and the local search capability of the Tabu Search.

2.3.3 Replica selection

Many replica selection strategies have been proposed in the past decade. The K-RSDG replica selection strategy is proposed in [3] for data grids, which considered: (i) two higher-valued attributes: security and file availability and (ii) two lower-valued attributes: price and response time and (iii) two unimportant attributes for each file. The replica location service is used to gather the replica location information based on the user request. The k-means clustering algorithm is used to cluster the labels from which a decision table is created. The grey based rough set theory is applied as input data by

using the replicas information only and then the grey-based k-means clustering algorithm is applied to the input data to make the replica selection decisions.

In [60], the authors propose a network coordinate based nearest replica selection service called Rigel. The best replica is selected from a site that has the smallest round-trip time. Rigel provides a lightweight and scalable solution to select the optimal replica for grid users.

In [90], the authors propose a dynamic data replication strategy with a replica management system. The proposed strategy concentrates on data availability by developing the replica placement and selection algorithm. It has two phases, where the first phase creates replicas by using catalogue and index, and the second phase stores the replicas. The replica selection strategy selects the replica with the minimum cost and bandwidth utilisation.

In [131], a 2PhaseEnhancing is proposed by the design of DNS to reduce the file request time in two phases. The first phase reduces the catalogue search time by using a local file that collects the historical file request of each user. The second phase considers a selection criterion to make the best replica selection choice.

In [42], the authors implement various classical replica selection algorithms such as the random algorithm, the round-robin algorithm, and the least response time algorithm, etc. They also analyse the performance of those replica selection algorithms for the current classic key-value stores in the cloud environment.

In [4], the authors propose a multithreaded and integrated maximum flow based optimal replica selection strategy for heterogeneous data storage architectures. They propose both sequential and parallel integrated maximum flow algorithms to find the optimal response time retrieval. The algorithms support both distributed storage architecture and centralized storage architecture.

In [55], the authors propose a comprehensive data replication strategy including all three replica management strategies. The replica creation strategy is based on the access tendency, named DRC-AT. The replica placement strategy is according to the user request response time and the storage capacity, named DRPRS. The replica selection is based on the response time, named DRS-RT. The DRC-AT strategy periodically calculates the file access tendency based on the file popularity and the period value of the file popularity to create and delete replicas. The DRP-RS strategy analyses the response time of the user requests and the storage capacity to select the best node set to place the created replica. The DRS-RT strategy offers the information about the replica-ready node with the strongest service capability to the users and guides the users to select that node to access the data.

2.4 Fault Tolerance and Task Scheduling

The cloud environment is subject to many types of faults, which might lead to a data centre or the network links to a data centre being unavailable [62][95][106]. For example, electricity interruption, data house collapse, cable damage, and natural disaster are all huge faults to cloud data centres [92][113]. When such a fault occurs, the tasks that require access to the data at the faulty data centre might be seriously impacted, resulting in deteriorated performance or access disruption [98]. Hence, it is critical to own the ability to handle the faults for all cloud data centres [34][40]. An appropriate fault tolerance strategy can reduce and even eliminate the negative influence of the faults.

The fault tolerance techniques are typically divided into two categories, proactive fault tolerance techniques and reactive fault tolerance techniques [84]. The proactive fault tolerance techniques try to proactively predict the faults and protect the system environment to avoid the faults from occurring while the reactive fault tolerance

techniques reduce the negative influence of the faults when the faults already occurred [83]. For example, MapReduce uses self-healing and pre-emptive migration for achieving proactive fault tolerance purposes [84]. Besides, the examples of the reactive fault tolerance techniques include checkpoint, retry, rescue workflow, user defined exception handling, task resubmission, and task migration, etc [89].

2.4.1 Fault tolerance techniques

There are many fault tolerance techniques applied in the fault tolerance strategies. Some classical fault tolerance techniques are described as follows.

- Self-healing

The self-healing technique allows the system to automatically detect, diagnose, and repair software faults and hardware faults. It deploys the application instances onto multiple virtual machines for achieving automatic handling [2].

- Pre-emptive migration

The pre-emptive migration technique enables the capability to migrate the cloud application executions away from the suspicious computing nodes to the stable computing nodes [39]. It is achieved by continuous system monitoring.

- Software rejuvenation

The software rejuvenation is developed for system periodic restarts. The periodic restart of the system can enable a clean state of the system [83].

- Load balancing

The upper resource utilisation limit is set in this technique. The resource load will be distributed to other computing nodes to avoid overloading problems if the resource load exceeds the upper resource utilisation limit in one of the computing nodes [51].

- Checkpointing/restarting

The checkpointing/restarting technique aims to continuously save the system state in the

event of a fault. The task execution can be restarted from the most recent state [137][141].

- Task resubmission and task migration

The task resubmission technique or the task migration technique allows the task to resubmit or migrate to the same or similar computing resources for achieving continuous task executions when encountering faults [77].

- Data replication

The data replication technique enables the replica-applied system environment to protect the system environment against the upcoming faults [16][26]. Many famous distributed computing environments have been adopted the data replication technique to create a replica-applied distributed environment for system robustness, such as HDFS, Google Cloud, and Amazon S3 [44][68]. In case that the primary data becomes inaccessible, the task can also follow the replica selection strategy and the task resubmission strategy to remain away from the interruption by accessing one of the required data replicas.

2.4.2 Fault tolerance strategies

Many contemporary fault tolerance strategies focus on resolving the faulty problem. In [62], a proactive fault tolerance strategy is proposed by considering the multi-VM coordination to satisfy the completion requirement of the parallel application. A particle swarm optimisation algorithm is proposed to migrate the VMs on the deteriorating physical machine to an optimal physical machine. The CPU temperature evaluation is applied to detect the deteriorating physical machine.

In [145], the authors combine three algorithms to achieve a redundant VM placement optimisation strategy for improving service reliability. The first algorithm selects a set of VM-hosting servers from a large host server candidate pool based on the network

topology. The second algorithm is to place the primary and backup VMs with the *k*-fault-tolerance assurance from the selected VM-hosting servers. The last algorithm is a heuristic algorithm to solve the task reassignment problem by finding a maximum weight matching in bipartite graphs.

In [25], an offloading system is proposed to make the robust offloading decisions for mobile services and optimise the execution time and the energy consumption with the consideration of the dependency relationships among services when the faults occurred.

In [47], the authors present a novel FT-HCC fault-tolerant task clustering strategy to enhance the workflow execution performance and improve the current task clustering strategies under a faulty cloud environment if the transient failures meet the proposed failure model. The FT-HCC fault-tolerant task clustering strategy considers the workflow execution time and the workflow execution cost as two major constraints to specify the deadline requirements during the workflow scheduling stage.

In [147], the authors propose a real-time workflow fault-tolerant model with the consideration of the cloud characteristics which extends from the traditional PB fault-tolerant model. A task allocation and message transmission analysis model is also proposed to assist the fault-tolerant workflow execution. Authors apply the overlapping and VM migration mechanisms when doing task scheduling to enable fault tolerance and achieve high resource efficiency at the same time. The authors also propose a resource elastic provisioning mechanism for full idle resource utilisation, fast resource provisioning, and the avoidance of unnecessary frequent resource allocation changes.

In [74], an energy-aware fault-tolerant dynamic scheduling scheme (EFDTS) is developed to assign and schedule the tasks with a fault-tolerant mechanism to optimise resource utilisation and energy consumption. A task classification method is proposed to partition the coming tasks and allocate the tasks to the suitable virtual machine

according to their task classes and energy consumption to reduce the mean response time. The replication method is also used to minimise the task rejection ratio caused by machine failure and delay. An elastic resource provisioning mechanism helps to improve resource utilisation and energy efficiency.

In [99], a Checkpointing and Replication based on Clustering Heuristics (CRCH) is proposed to achieve the fault tolerance purpose by using replication, resubmission, and checkpointing methods. Authors develop an unsupervised way to learn the task replication counts and a checkpointing mechanism to support the dynamic task resubmissions on the most optimum resource.

In [133], the authors present a novel fault-tolerant workflow scheduling (ICFWS) algorithm for the cloud environment to achieve the fault tolerance purpose by considering both resubmission and replication method and the workflow deadline. The algorithm firstly breaks the workflow deadline into multiple sub-deadlines for all tasks in the workflow. Then, a suitable fault-tolerant strategy is selected and the suitable resource will be reserved by analysing the sub-deadline competitions of the tasks and adopting the on-demand cloud resource provisioning. After that, the authors design an online scheduling and reservation adjustment scheme to select a suitable resource for the tasks. This online scheduling and reservation adjustment scheme can also adjust the sub-deadlines of the current-running tasks and the selected fault-tolerant strategy for the upcoming tasks to be executed.

According to the above strategy in [133], a deadline-constrained Hybrid Fault-Tolerant Scheduling Algorithm (HFTSA) for independent tasks in the cloud environment by integrating both resubmission and replication method is further proposed in [134]. Similar to [133], HFTSA selects the fault-tolerant strategy by using resubmission and replication for each task according to the task attributes and the cloud resource

situations. Then it reserves the suitable resources for each task execution. An online adjustment scheme is also developed to adjust the selected fault-tolerant strategy and an elastic resource provisioning mechanism is designed to dynamically adjust the resources for executing the tasks.

2.4.3 Task scheduling strategies

As mentioned above, task resubmission and task migration are two of the most important fault tolerance techniques. The task scheduling method is the core method of task submission and task migration. The task scheduling strategies can enable a reasonable task allocation solution when rescuing the tasks at the faulty data centre. Many task scheduling strategies have been proposed in the past years. Various constraint parameters have been considered to optimise different objectives. In particular, the deadline-constrained task scheduling strategies are one of the common types of task scheduling strategies to satisfy the deadline requirements.

The HEFT series strategies are one of the most significant series of deadline-constrained task scheduling strategies, which are published from 2002 to date [11]. In [114], the authors develop a Heterogeneous Earliest-Finish-Time (HEFT) algorithm to minimise its earliest finish time with an inserted-based policy. It firstly assigns the priority to each task in the scheduling list and then assigns each task to the first available server which can enable the task to finish the earliest. In [144], the authors propose a Budget and Deadline Constrained scheduling algorithm named BEFT to find the optimal workflow scheduling solution to satisfy both deadline and budget constraints for avoiding SLA violations. Specifically, the BEFT algorithm only works by reserving and billing a fixed number of resources in heterogeneous grid computing systems. In [5], a novel list-based task scheduling algorithm is proposed called Predict Earliest Finish Time (PHEFT) to improve the makespan and the efficiency to compare with the HEFT, LDCP, and

LHEFT strategies. At the same time, this algorithm keeps the same time complexity to the HEFT strategy. In [119], the authors extend the classic HEFT strategy in [114] and the BHEFT strategy in [144]. They develop a Budget and Deadline Constrained Heterogeneous Earliest Finish Time (BDHEFT) algorithm. The BDHEFT considers six major variables, such as spare workflow budget, spare workflow deadline, current task budget, current task deadline, budget adjustment factor, and deadline adjustment factor, to generate a budget and deadline constrained scheduling plan. In [100], an Enriched-Look ahead HEFT (E-LHEFT) algorithm is proposed to optimise both QoS and load balancing without considering any constraints. It utilises Mobile Assistance Using Infrastructure architecture to execute the tasks. The E-LHEFT algorithm updates the processor selection phase of the LHEFT algorithm by applying the task grouping and the Pareto theory for more effective load balancing performance. In [63], the tasks with both unconstrained and time deadline constrained cases are considered by applying a HEFT technique for the order preference called the HEFT-T algorithm. A three-stage non-dominated sorting strategy is applied to identify the optimal solutions for the unconstrained case, and an adaptive weight adjustment strategy is proposed to adjust the weight value for time for addressing the deadline-constrained case. In [29], a workflow scheduling algorithm named Greedy Resource Provisioning and Modified HEFT (GRP-HEFT) is proposed with a resource provisioning mechanism. The resource provisioning mechanism generates the instance type list based on the efficiency ratio of different instance types and selects the most efficient instances constrained by a pre-defined budget. The modified HEFT algorithm employs the optimal configuration of the instance types with their number of created VMs to obtain the task scheduling plan. In [96], the authors propose a Dynamic Variant Rank HEFT (DVR-HEFT) algorithm to reduce the scheduler's makespan without increasing the algorithm's time complexity to

compare with the classic HEFT strategy.

There are still many other deadline-constrained task scheduling strategies. In [15], a deadline-constrained workflow scheduling algorithm called DCWS is proposed to reduce the monetary cost. The DCWS algorithm is a list-based algorithm that considers the probabilities of the task combinations to place together to improve resource utilisation and satisfy the deadline constraint.

In [132], a deadline-constrained energy-aware task scheduling method is proposed by exploiting the computing parallelism of the divisible task. The urgency level is developed to prioritize the real-time task order to be processed. Two proposed energy-aware task scheduling algorithms consider whether the task load is divisible.

The deadline is not the only parameter considered in the task scheduling strategies. Many other constraint parameters have been used. For example, in [97], the authors develop an energy-efficient task scheduling strategy for cloud data centres. They formulate the task scheduling problem as an integer programming problem, which aims to minimise the data centre energy consumption and maximise the residue energy capacities of the data centres. A greedy task scheduler is deployed to minimise the number of active servers.

In [146], a task rescheduling method has been proposed to minimise network resource consumption. Three algorithms are developed for identifying a set of good virtual machines from the virtual machine candidate pool by using the skyline operation. The task importance is analysed by taking the data size and the task emergency, and the optimal task insertion point into account.

There are also some task scheduling strategies that jointly coordinate with the data management strategies to enhance the cloud performance and satisfy the user requirements. For example, in [122], a novel data placement and task scheduling

optimisation algorithm is proposed for the scientific workflows in the cloud environment to optimise the data placement and task scheduling performance. A *k*-means algorithm based build-time data placement strategy is proposed to reduce the data movement at the workflow build-time stage by considering the data dependency and the data size. Then a multi-level task replication method based run-time task scheduling strategy is proposed to reduce the intermediate data transfer among cloud data centres at the run-time stage.

In [48], the authors propose the SLA-aware task scheduling strategy cooperated with a data replication strategy to satisfy the requirements of the response time and the minimum availability and enhance the profit to the cloud service providers. A novel Bottleneck Value Scheduling (BVS) process is developed to couple with a proposed Correlation and Economic Model-based Replication (CEMR) strategy.

2.5 Problem Statement and Research Insight

Firstly, there are two common types of data attributes, external data attributes and internal data attributes. The external data attribute refers to the attribute which the data correlates to the external environmental factors such as users, cloud service providers, and cloud environment, while the internal data attribute refers to the attribute which the data correlates to other data. Both external data attributes and internal data attributes have significant influences on the data. For example, access frequency is one of the most important external data attributes in the past literature to constrain the replica creation processes for identifying which data is hot-accessed by users. Besides, data dependency is also one of the most important internal data attributes, which refers to the relationship between a pair of data. It can identify the potential influences between a pair of data when doing replica creation. Both external data attributes and internal data attributes have been considered to constrain the replica creation in the past literature.

However, they are used in some simple parameter combinations. Some of the parameter combinations only consider the same type of data attributes. For example, as shown in Table 2.1, most of the replica creation strategies lack the consideration of the internal data attributes.

For another example, in [127], they only consider the access frequency and the average response time as two major constraint parameters, which both the access frequency and the average response time belong to the external data attribute. Considering only one type of data attributes may lose the comprehensiveness of the data attribute analysis when developing replica creation strategies. Therefore, the external data attributes and the internal data attributes should be jointly considered to constrain the replica creation decision making. A more general replica creation strategy, which considers both external data attributes and internal data attributes, needs to be investigated in order to comprehensively determine the replica creation and further improve the optimisation objectives.

Secondly, the cloud map should be taken into consideration to make a more precise replica creation decision. Each data centre can be recognized as an individual host entity in the cloud environment. A data may have multiple data relationships to other data inside the same data location and outside the same data location. The data relationship between this specific data and the correlated data inside the same data centre can be seen as local data relationship, while the data relationship between this specific data and the correlated data outside the same data centre can be known as remote data relationship. The local data relationship and the remote data relationship are hardly considered in the most of current replica creation strategies, as shown in Table 2.1. Therefore, the data relationship situations inside data centre and outside data centre should be distinguished when making the replica creation decision.

Table 2.1 The comparison of replica creation strategies

The Comparison of Replica Creation Strategies										
Replica Creation Strategies	Year	External Data Relationship			Internal Data Relationship		Other Measurements			
		Access Frequency	Average Response Time	Resource Load	Data Dependency	Data Dependency Inside and Outside Data Center	The Number of Replicas	Cost	File Change Rate	Failure Rate
FSR [91]	2008	✓	✗	✓	✗	✗	✗	✗	✗	✗
LALW [18]	2008	✓	✗	✗	✗	✗	✓	✗	✗	✗
CDRM [125]	2010	✓	✓	✓	✗	✗	✓	✗	✗	✓
CIR [56]	2011	✗	✗	✓	✗	✗	✓	✓	✗	✓
Threshold-based file replication mechanism [118]	2012	✓	✗	✗	✗	✗	✗	✗	✗	✗
DCR2S [30]	2014	✓	✗	✗	✗	✗	✓	✓	✗	✗
Dynamic Cost-Aware Data Replication Strategy [31]	2016	✗	✗	✗	✗	✗	✓	✓	✗	✗
Dynamic File Heat and Node Load Based Replica Creation Strategy [143]	2017	✓	✓	✓	✗	✗	✓	✗	✓	✗
DRAPP [58]	2019	✗	✓	✓	✗	✗	✓	✓	✗	✗
RSPC [80]	2020	✗	✓	✓	✗	✗	✓	✓	✗	✗
DPRS [36]	2020	✓	✗	✗	✗	✗	✓	✗	✗	✗

Thirdly, although the existing research has made significant progress to replica selection, there are still research gaps to be filled. Most of the current replica selection

strategies focus on how to select a data replica to access without considering the potential impacts among multiple concurrent-running instances under limited network capability. In particular, they might not be suitable to apply in a cloud environment with heavy data access needs and a large number of application instances when the data access needs and the number of application instances result in overloading in certain parts of the cloud network. Therefore, a replica selection strategy is urgently required by considering the potential impacts among multiple concurrent-running instances and the limited network capability.

Fourthly, different network performance metrics should be jointly considered to achieve a comprehensive evaluation of the network situations at each cloud data centre. Most of the current data replication strategies model the network performance metrics in an isolated way. Therefore, a suitable evaluation method should be developed to jointly evaluate different types of network performance metrics.

Fifthly, as shown in Table 2.2, most of the fault tolerance strategies pay insufficient attention to both the network performance and the attributes of the affected tasks. When the data access requests are resubmitted to other replica sites or when new data replicas are created, the impacts to the overall cloud environment performance have been largely overlooked.

If a system executes many task resubmission operations or replica re-creation operations, it will significantly increase the resource load on certain data centres [102]. In addition, some tasks may miss the deadline even if they have been resubmitted to access the required replicas without considering the attributes of the affected tasks. As a result, this may cause a series of negative influences, such as user dissatisfaction, reputation damage, future profit reduction, and economic compensation. Therefore, the insufficient consideration of both the network performance and the task attributes may

largely degrade the overall cloud performance [129]. Thus, it is desirable to have a fault tolerance strategy that fully considers both the network performance and the attributes of the affected tasks.

Table 2.2 The comparison of fault handling strategies

The Comparison of Fault Tolerance Strategies

Fault Tolerance Strategies	Year	Network Performance Metrics			Task Attributes					Other Measurements
		Bandwidth	Latency	Error Rate	Task Size	Deadline	Task Execution Duration	Task Dependency	Task Operation Profit	
TBFR [118]	2012	X	X	X	X	X	X	X	X	Task request count
HDFS framework with erasure coded replication scheme [50]	2014	X	X	X	X	X	X	X	X	Storage load
Computation offloading strategy [25]	2015	X	X	X	X	X	✓	✓	X	Queue waiting time, energy consumption, current workload
FASTER [147]	2016	X	X	X	✓	✓	✓	✓	X	Timeline scenarios
Location-aware data block allocation strategy [129]	2016	✓	X	X	X	X	X	X	X	Data block location, write cost
Redundant VM placement optimization strategy [145]	2017	X	✓	X	X	X	X	X	X	Available host servers, current workload, strategy cost
Software-based selective replication technique for HPC applications [106]	2017	X	X	X	X	X	✓	X	X	Failure probability, current workload, Reliability
PCFT [62]	2018	✓	X	X	X	X	X	X	X	CPU temperature, CPU idle capacity, memory
EFDTS [74]	2019	✓	X	X	✓	✓	X	X	X	Energy consumption
FT-HCC [47]	2020	X	X	X	✓	✓	✓	✓	✓	Critical Path

Sixthly, although the replica-applied cloud environment can protect the cloud environment against the upcoming faults, a suitable reactive fault tolerance strategy can further enhance the cloud resiliency as well as the overall cloud performance. At the same time, the deadline contention and the resource contention problems may also exist when handling independent tasks and dependent tasks. An independent task denotes the task has no dependencies to other tasks, while a dependent task denotes the task has at least one dependency to other tasks. The success of an independent task is only related to itself while the success of a dependent task always relies on the success of its preceding tasks. The dependent tasks should be assigned the task priority when allocating the tasks to the cloud data centres because a parent task may influence all its succeeding tasks. Therefore, the task dependencies among tasks should be considered when handling dependent tasks. The fault tolerance strategies for dependent task rescue should be developed in different ways in comparison with the independent task rescue.

Seventhly, the HEFT series strategies tend to select the first available server to enable the earliest finish time when doing timeline allocation. Although the HEFT series strategies were developed over a long time period, selecting the first available server might not be the optimal configuration when handling faults [11][94][99]. It may cause the deadline contention and the resource contention problems in which the task rescue with the high priority may unnecessarily impact the task rescue with the low priority. Moreover, selecting the first available server may cause a temporary dramatic load increase at certain time points on the timeline, which might lead to the performance bottleneck to cloud data centres. Therefore, a time allocation method should be developed to balance the resource load and eliminate the deadline contention and the resource contention problems as much as possible.

Chapter 3 Data Replication and Fault Management Framework

In this chapter, the data replication and fault management framework is proposed and some basic definitions and general notations used in this thesis are introduced. The data replication and fault management framework is described in Section 3.1. The basic definitions and the general notations are demonstrated in Section 3.2.

3.1 Data Replication and Fault Management Framework

The cloud environment always contains at least one cloud service provider. Each cloud service provider may also have at least one data centre. Each data centre has its specific environment configurations. Each data centre can be seen as an independent host entity in the cloud environment. Therefore, a decentralised management framework is more suitable to apply in the cloud environment. The decentralised management framework can enable self-management in each data centre side and will not be influenced by the management configuration of other data centres. The proposed data replication and fault management framework establishes a decentralised overarching management to offer the “anyone, anytime and anywhere” flexibility, the adaptability, and the geo-diversity for the global collaborators in the cloud environment. To execute multiple concurrent-running cloud application instances, such a management framework is easier to handle the modular growth and takes advantage of the geo-elasticity and the geo-diversity. New data centres, cloud service providers, or cloud application instances can be integrated into the current cloud network without affecting the operations of other data centres, cloud service providers, and application instances. Normally, the general cloud environment can be shown in Figure 3.1.

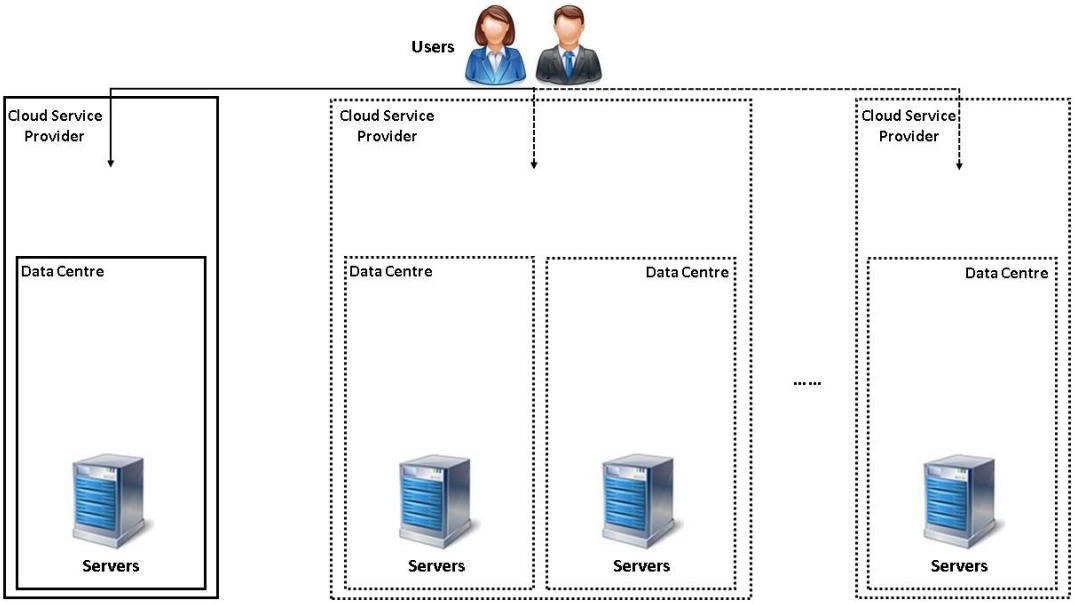

Figure 3.1 The general cloud environment

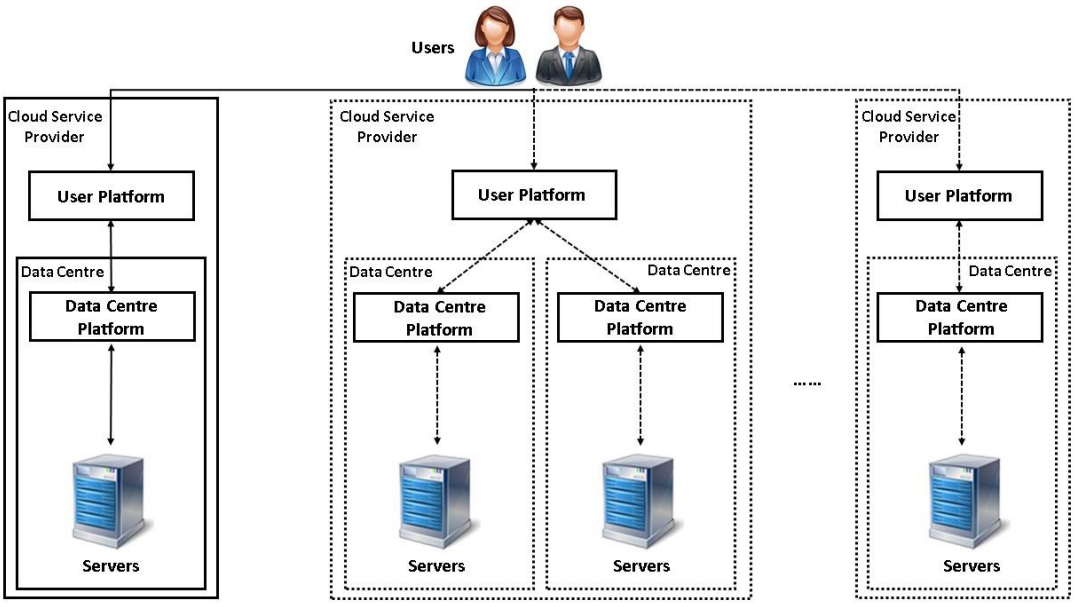

Figure 3.2 The data replication and fault management framework

The proposed data replication and fault management framework is shown in Figure 3.2. It is a decentralised cloud management framework that contains two types of platforms at the user side and the data centre side, respectively. Each cloud service provider has its unique user platform and data centre platform because they may have different functionalities applied to the user platform and the data centre platform. Each cloud

service provider has only one user platform. Each user can access the specific user platform to send the task execution requests or the data access requests to the cloud service provider. Each cloud service provider may have multiple data centre platforms at each cloud data centres which belong to the cloud service provider. Each data centre has only one data centre platform for replica management, fault management, and data centre control.

The data replication strategies and the fault management strategies may need the information about the performance characteristics of each cloud service provider [78]. In the cloud environment, different cloud service providers may have different scenarios inside [14][17]. Therefore, a decentralised analysis of each single cloud service provider is required. Each data centre platform acts on behalf of a cloud service provider and is responsible for interacting with the user platforms. The data centre platform can collect the characteristics information in the cloud data centre, such as the response time of the data centre, the available bandwidth of the data centre, the storage capacity of the data centre, and the location of the target replica, etc. It avoids the problems related to the privacy policy difference among different cloud service providers because each cloud service provider only hosts a uniform type of data centre platform obeying its own privacy policy. Different data centre platforms can collect the required information based on the different privacy policies in different cloud service providers. The collected information can be used as measurements when making the data replication decisions or the fault management decisions.

The detailed interior structure of the user platform and the data centre platform is shown in Figure 3.3, which also indicates a complete data replication and fault management framework between the cloud users and a single data centre. It also shows the corresponding relationship between the required modules and the context locations

where the module applies in this thesis.

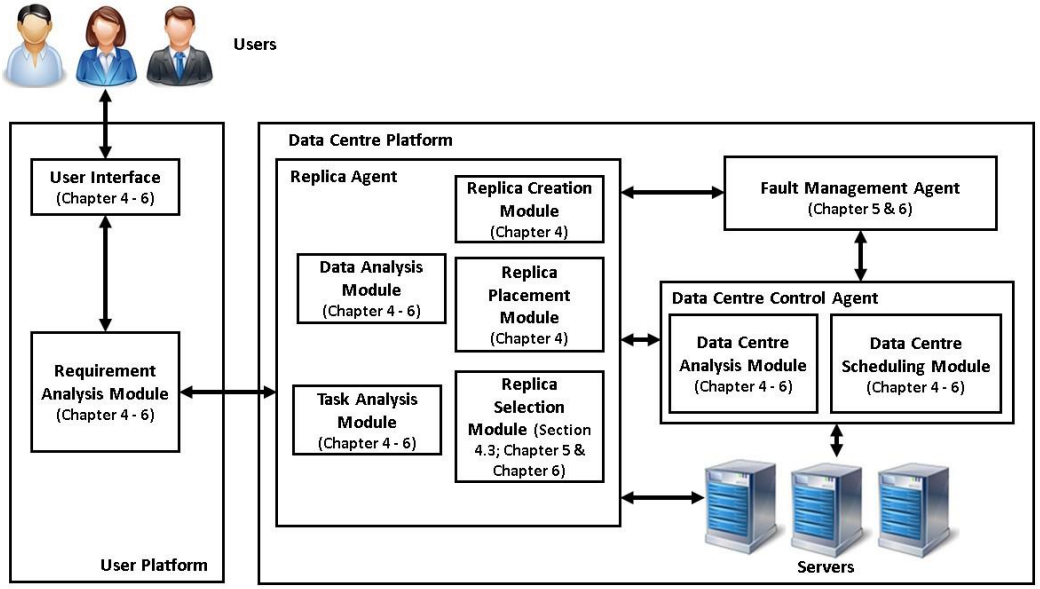

Figure 3.3 The interior structure of the user platform and the data centre platform

3.1.1 User platform

Each **user platform** contains two modules, **user interface** and **requirement analysis module**. The **user interface** is responsible for interacting with the users to collect the data access requests or the task execution requests. Then the **user interface** will transfer the user requests to the **requirement analysis module**. The **requirement analysis module** further includes two operation units, **data requirement analysis unit** and **task requirement analysis unit**, as shown in Figure 3.4. The **data requirement analysis unit** aims to analyse the user data requirement and the **task requirement analysis unit** focuses on the user task requirement analysis. Collectively, the **requirement analysis module** analyses the user requirements to answer the following questions.

- Which task is being executed?
- Which data should be accessed?
- Where is the required data replicas situated?
- Where is the task situated?

After the analysis of the user requirements, a list of target data centres will be generated including the required replica names, the replica locations, the task names, and the task locations. Then the relevant data centre platforms will receive the corresponding data access information and the task execution information from the user platform.

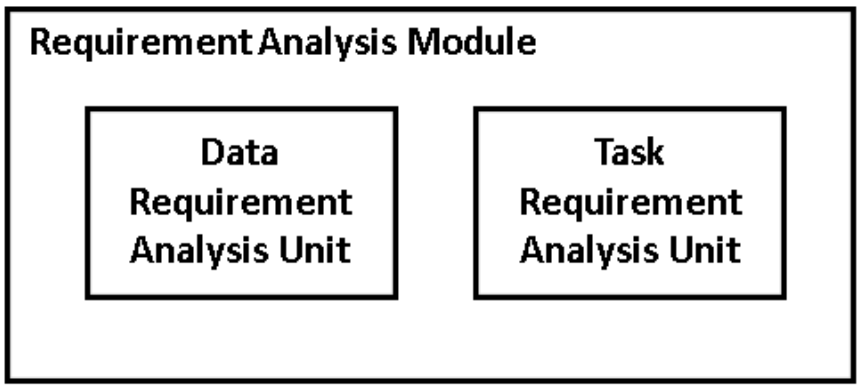

Figure 3.4 The interior structure of the requirement analysis module

3.1.2 Data centre platform

Each **data centre platform** contains three management agents and a pool of cloud servers. These agents and cloud servers are interconnected. The **replica agent** is responsible for creating the replica-applied cloud environment and analysing the required replica to be accessed and the task to be executed. It includes five modules, **replica creation module**, **replica placement module**, **replica selection module**, **data analysis module**, and **task analysis module**. The **replica creation module** and the **replica placement module** are used to create a replica-applied cloud environment. The **replica creation module** enables the replica creation processes to create multiple replicas into multiple cloud data centres based on the applied replica creation strategy. Then the newly created data replicas will be offered a destination to be situated by the **replica placement module** based on the applied replica placement strategy. The **replica selection module** aims to guide the access to the optimal required replicas. Particularly, it assists with the **fault management agent** to guide the task rescheduling when

handling faults. The **replica creation module**, the **replica placement module**, and the **replica selection module** can achieve a replica management chain in the replica-applied cloud environment.

The **data analysis module** aims to collect the value of the required data attributes and analyse the data dependency of the required data. The **data analysis module** includes two operation units, **data attribute analysis unit** and **data dependency analysis unit**, as shown in Figure 3.5. The **data attribute analysis unit** is used to collect the value of different data attributes related to the required data. The **data dependency analysis unit** aims to analyse the data dependency of the required data.

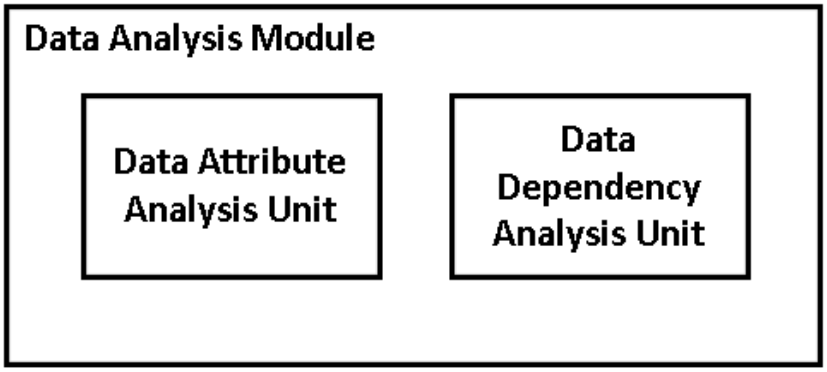

Figure 3.5 The interior structure of the data analysis module

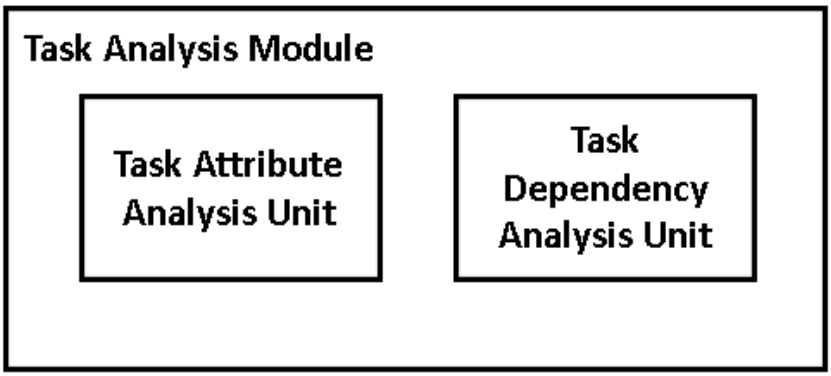

Figure 3.6 The interior structure of the task analysis module

The **task analysis module** aims to analyse the task dependency of the tasks to be executed and the relevant task attributes. The **task analysis module** includes two

operation units, **task attribute analysis unit** and **task dependency analysis unit**, as shown in Figure 3.6. The **task dependency analysis unit** is developed to analyse the task dependency of the tasks to be executed. The **task attribute analysis unit** is used to analyse the attributes of the tasks to be executed.

The **data analysis module** and the **task analysis module** will contribute to the development of different cloud management strategies, such as replica creation strategy, replica placement strategy, replica selection strategy, task scheduling strategy, and fault tolerance strategy, by offering the required data information or task information.

As mentioned in Chapter 2, unexpected faults are unpredictable. The **replica agent** can create a replication-applied cloud environment to protect the cloud environment against the upcoming faults. However, it is not sufficient to reduce or even eliminate the negative fault impacts. The **fault management agent** is responsible for reactively and strategically handling the fault scenarios to further improve the cloud performance when encountering a fault. It helps the faulty data centre handle the tasks which cannot be continued in this data centre and need to migrate to other computing nodes. It includes two operation units, **fault detection unit** and **fault handling guide unit**, as shown in Figure 3.7.

The **fault detection unit** is used to continuously detect the fault in the data centre and then report it to the **fault handling guide unit** to initiate the fault handling process. The **fault handling guide unit** offers the guidance of the whole fault handling process based on the applied fault tolerance strategy. It guides the detailed task rescue operations to the **data centre control agent** when encountering a fault. The **fault handling guide unit** also references the replica selection strategy from the **replica selection module** in the **replica agent** to guide the task resubmission and migration operations under fault scenarios.

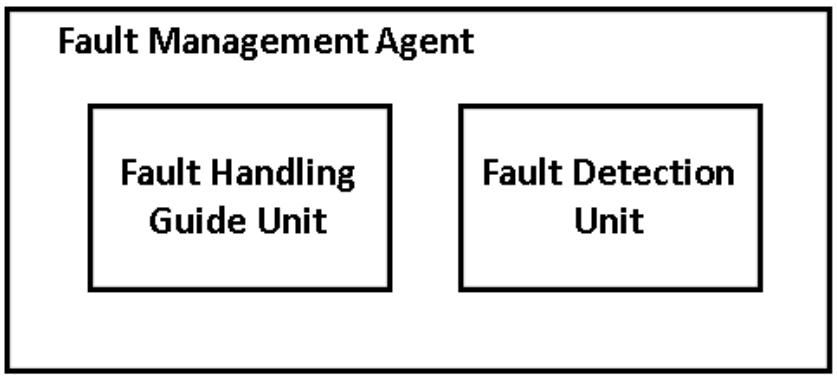

Figure 3.7 The interior structure of the fault management agent

The **data centre control agent** is the console of the data centre scheduling operations and the cloud environment provisioning. It contains two modules, **data centre analysis module** and **data centre scheduling module**. The **data centre analysis module** is used to monitor and analyse the environment information of the data centre. It collects the value of different types of cloud resources such as bandwidth, latency, error rates, and time slot utilisation situations. This can help the development of the data replication strategy applied in the **replica creation module**, the **replica placement module**, and the **replica selection module**. At the same time, this can also contribute to the development of the fault tolerance strategy applied in the **fault management module** as well as the resource provisioning in the **data centre scheduling module**.

The **data centre scheduling module** is used to schedule the data and the tasks. It contains two operation units, **task scheduling unit** and **replica scheduling unit**, as shown in Figure 3.8. The **task scheduling unit** is used to schedule the tasks under normal circumstances and reschedule the tasks when encountering a fault, upon the applied task scheduling strategy in this unit. This unit can cooperate with the **replica selection module** to generate a task scheduling solution under normal circumstances. The operations of the task resubmission and migration can also be completed in this unit under fault scenarios, upon the cooperation with the **fault management agent**. The

replica scheduling unit is responsible for scheduling the newly created replicas to different locations by cooperating with the **replica creation module** and the **replica placement module**. This unit implements the practical operations of the replica creation solution and the replica placement solution.

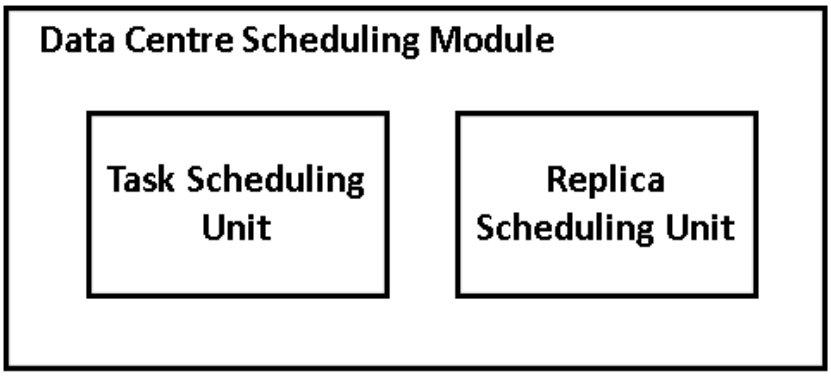

Figure 3.8 The interior structure of the data centre scheduling module

3.2 Basic Definitions and General Notations

Several basic definitions and general notations of the cloud environment are listed below to use in the following proposed strategies.

Definition 1. Cloud environment. A cloud environment is a computing environment that enables on-demand access to the computing resources, such as applications, servers (physical servers and virtual servers), development tools, networking capabilities, and more relevant resources. These computing resources are hosted at each data centre in the cloud environment which is managed by a specific cloud service provider. Therefore, a cloud environment can be represented as a 2-tuple (CSP,DC), where

- CSP is the set of cloud service providers in the cloud environment.
- $DC = \{dc_1, dc_2, \ldots, dc_z\}$ is the set of data centres in the cloud environment. dc_y denotes the yth data centre in DC.
- There may exist multiple data centres with multiple cloud service providers in

the cloud environment. Each $dc \in DC$ has only one $csp \in CSP$, while one csp may have at least one dc.

Definition 2. Task and data. A set of cloud applications can be deployed in the cloud environment by users or cloud service providers. They may contain a set of independent tasks and dependent tasks. Each task corresponds to a set of required data to be accessed. Therefore, a set of tasks J and a set of data D is defined for the cloud environment, where

- J:$\{j_1, j_2, \ldots, j_m\}$ is the set of tasks scheduled in the cloud environment. j_m denotes the mth task in J.
- D:$\{d_1, d_2, \ldots, d_n\}$ is the set of data stored in the cloud environment. d_n denotes the nth data in D.

Definition 3. Workflow applications. The cloud environment may contain a set of dependent tasks which may perform in different workflow applications. Therefore, in general, a workflow application $G = (N, E)$ is modelled as a Directed Acyclic Graph (DAG), where N is the set of nodes $\{Nod_0, Nod_1, ..., Nod_q\}$ as tasks and E is a set of edges as the control dependencies among the workflow tasks. For each pair of nodes $Nod_p, Nod_q \in N$, $edge(Nod_p, Nod_q)$ denotes the edge between Nod_p and Nod_q. The cloud environment may contain a set of x workflow applications $\{G_1, G_2, \ldots, G_x\}$ scheduled in the cloud environment.

The notation tables are also made in Table A1.1, Table A1.2, Table A1.3, and Table A1.4 in Appendix 1 which these notations will be used in the descriptions, the equations, and the pseudocodes of the following six strategies.

Chapter 4 The Development of Data Replication Strategies

Data replication strategies can help cloud service providers establish a replica-applied cloud environment. As mentioned in Chapter 2, the data replication strategies bring many benefits for improving the overall cloud performance, such as fast data access, low response time, balanced workload and resource load, and increased data availability and reliability. In addition, the tasks at the faulty data centre can be rescued to continuously execute by strategically accessing other required data replicas if the replica-applied cloud environment is deployed. In this chapter, three data replication strategies are proposed to manage replica creation, replica placement, and replica selection. The first two strategies focus on replica creation including the replica placement of newly created replicas. The last strategy contributes to replica selection. The proposed replica creation strategy and replica selection strategy can also be aligned together to create a replica-applied cloud environment and guide the data access for executing the tasks in the cloud environment.

4.1 Replica Creation for Total Cost Reduction in Clouds

The replica creation strategy is the basis of all data replication strategies because the replica creation strategy is responsible for creating multiple data copies into multiple cloud data centres. As demonstrated in Chapter 2, both external data attributes and internal data attributes have significant impacts on the replica creation process according to the past literature. The joint consideration of the external data attributes and the internal data attributes is important for the replica creation decision-making process. Data dependency is one of the most significant internal data attributes, as it reveals the data relationship between a specific data and other data. Access frequency is an external data attribute to check whether the data is being accessed in hot. Data size is

also an internal data attribute, which might largely influence the data storage, data transmission, and data allocation, etc.

In this thesis, data dependency, access frequency, and data size are taken into account for jointly applying the internal data attribute and the external attribute to constrain the replica creation. Besides, the data types are classified into three sub-categories to denote the replica creation feasibility of a data to a specific data centre. A replica creation algorithm is also developed to create multiple data replicas into the target data centres.

4.1.1 Data classification

In data replication, the data are commonly classified into two categories, fixed data and flexible data. The fixed data (FixD) cannot be replicated because of the constraints of its own data attributes, such as data ownership or privacy concerns, while the flexible data (FlexD) can be freely replicated across geographical data centres as well as inside data centre. In this strategy, the flexible data is further classified into two new sub-categories, free-flexible data (FFlexD) and constrained-flexible data (CFlexD). The data dependency constraint, the access frequency constraint, and the data size constraint are applied as three replica creation constraints during the replica creation decision making. A data $d \in D$ can be finally classified to FFlexD to a specific data centre dc when the data d can satisfy all three constraints to this data centre. In other words, it means that the data d can be freely replicated to the data centre dc when the data d is a FFlexD to this data centre. Otherwise, the data d will be classified into CFlexD to a data centre dc when the data d cannot meet at least one of three replica creation constraints to this data centre.

The data in CFlexD to a data centre $dc \in DC$ is still a flexible data to other data centres in DC, thus it may not be CFlexD to other data centres in DC. For example, if the data $d_i \in D$ cannot satisfy at least one of the data dependency constraint and the access

frequency constraint, it will be CFlexD to all data centres in the cloud environment. However, if the data d_i can satisfy both the data dependency constraint and the access frequency constraint, except that it cannot satisfy the data size constraint to a specific data centre $dc_z \in DC$, then the data d_i will be CFlexD to the data centre dc_z only. For the same data d_i, if the data d_i can satisfy all three replica creation constraints to another data centre $dc_y \in DC$, then the data d_i will be FFlexD to the data centre dc_y.

4.1.2 Data dependency and access frequency

The data dependency and the access frequency are defined as two constraint parameters when initiating the replica creation. The data dependency is the relationship between each two data and the access frequency refers to the frequency of access in a specific time duration by users. The data dependency between two data d_i and d_k is defined as the number of tasks that use both d_i and d_k [139]. The data dependency between two data d_i and d_k has two expressions, $Dep(d_i, d_k)$ and $Dep(d_k, d_i)$, which can be formulated in Eq. 4.1, where $J(d_i)$ denotes the set of tasks which access the data d_i. $Dep(d_i, d_k)$ refers to the data dependency of the data d_i to the data d_k, while $Dep(d_k, d_i)$ refers to the data dependency of the data d_k to the data d_i. The numerical value of $Dep(d_i, d_k)$ and $Dep(d_k, d_i)$ is same, as also shown in Eq. 4.1.

$$\begin{cases} Dep(d_i, d_k) = Count(J(d_i) \cap J(d_k)) \\ Dep(d_k, d_i) = Count(J(d_k) \cap J(d_i)) \\ Dep(d_i, d_k) = Dep(d_k, d_i) \end{cases} \tag{4.1}$$

The access frequency of the data d_i can be formulated in Eq. 4.2, where $AF(d_i)$ denotes the access frequency of the data d_i, $AT(d_i)$ denotes the number of access times of the data d_i, and $AI(d_i)$ denotes the access time interval to the data d_i.

$$AF(d_i) = \frac{AT(d_i)}{AI(d_i)} \tag{4.2}$$

A threshold-based evaluation method is adopted to evaluate the data importance for further making the replica creation decision. A threshold parameter ω is set for the data

dependency constraint. The data dependency $Dep(d_i, d_k)$ should satisfy $Dep(d_i, d_k) \geq \omega$, which is one of the mandatory constraints to replicate d_i. Similarly, the data dependency $Dep(d_k, d_i)$ should satisfy $Dep(d_k, d_i) \geq \omega$, which is also one of the mandatory constraints to replicate d_k. The data dependency threshold parameter ω can be ranged from the minimum data dependency value to the maximum data dependency value of the data in D.

An access frequency threshold parameter $\emptyset$ is also set for the access frequency constraint. The access frequency of d_i and d_k should satisfy either $AF(d_i)$ or $AF(d_k) \geq \delta$ at least, which is another mandatory constraint to replicate d_i or d_k. The access frequency threshold parameter $\emptyset$ can be ranged from the minimum access frequency value to the maximum access frequency value of the data in D.

4.1.3 Data size constraint

In this strategy, the data size constraint is also applied to constrain the replica creation process. The replica creation should follow the data size constraint defined in Eq. 4.3, where $Size(d_i)$ denotes the data size of d_i and $ASS(dc)$ denotes the available storage capacity in the data centre dc.

$$Size(d_i) \leq ASS(dc) \tag{4.3}$$

4.1.4 Cost

For $d_i \in D$, the total cost $TC(d_i)$ can be the sum of the data storage cost $DSC(d_i)$ and the data transfer cost $DTC(d_i)$ as shown in Eq. 4.4.

$$TC(d_i) = DSC(d_i) + DTC(d_i) \tag{4.4}$$

The data storage cost of d_i at a data centre dc depends on many parameters such as the data storage price of this data centre $SP(dc)$, the data size $Size(d_i)$, and the data storage time interval at this data centre $ST(d_i)^{dc}$. For a data d_i stored at the data centre dc, the data storage cost of d_i at this data centre dc can be $SP(dc)$ * $Size(d_i)$ *

$ST(d_i)^{dc}$. However, the data d_i may store in multiple data centres. Therefore, the total data storage cost for a data d_i, $DSC(d_i)$, can be formulated as in Eq. 4.5, where μ is a determinant variable for calculating the data storage cost. If dc_y is the data location of the data d_i, then μ equals to 1. Otherwise, μ equals to 0.

$$DSC(d_i) = \sum_{y=1}^{z} \mu * SP(dc_y) * Size(d_i) * ST(d_i)^{dc_y} \quad (4.5)$$

The data transfer cost of d_i, $DTC(d_i)$, depends on the transfer cost ratio α per data unit, the data size $Size(d_i)$, the determinant variable β and the number of access times $AT(d_i)$. The determinant variable β will be 1 if the cloud users require to access the data from a remote data centre, while it will be 0 if the cloud users only need to access the data locally. Therefore, the data transfer cost of d_i, $DTC(d_i)$, can be formulated as in Eq. 4.6.

$$DTC(d_i) = \alpha * Size(d_i) * AT(d_i) * \beta \quad (4.6)$$

Then the overall total cost TC of all data in D can be formulated in Eq. 4.7.

$$TC = \sum_{i=0}^{n} (DSC(d_i) + DTC(d_i)) \quad (4.7)$$

4.1.5 Assumed scenarios

This research assumes that the initial data placement and the initial task placement have been completed by using the strategy proposed in [139]. The set of data D are allocated into data centres based on the data placement rules from [139]. At the same time, the set of data D can be initially categorised into fixed data and flexible data based on their own attributes. The set of tasks J are also randomly allocated to different data centres in the cloud environment.

4.1.6 Replica creation strategy

In this research, a replica creation strategy is proposed to create multiple replicas into appropriate data centres by satisfying the data dependency constraint, the access frequency constraint, and the data size constraint. After all data placement and task

placement completed, the set of tasks $J(d)$ are located, which needs to access each data $d \in D$. Each data $d \in D$ will be firstly classified into FixD and FlexD. Then the data in FixD will not be considered to be replicated because those data cannot be replicated.

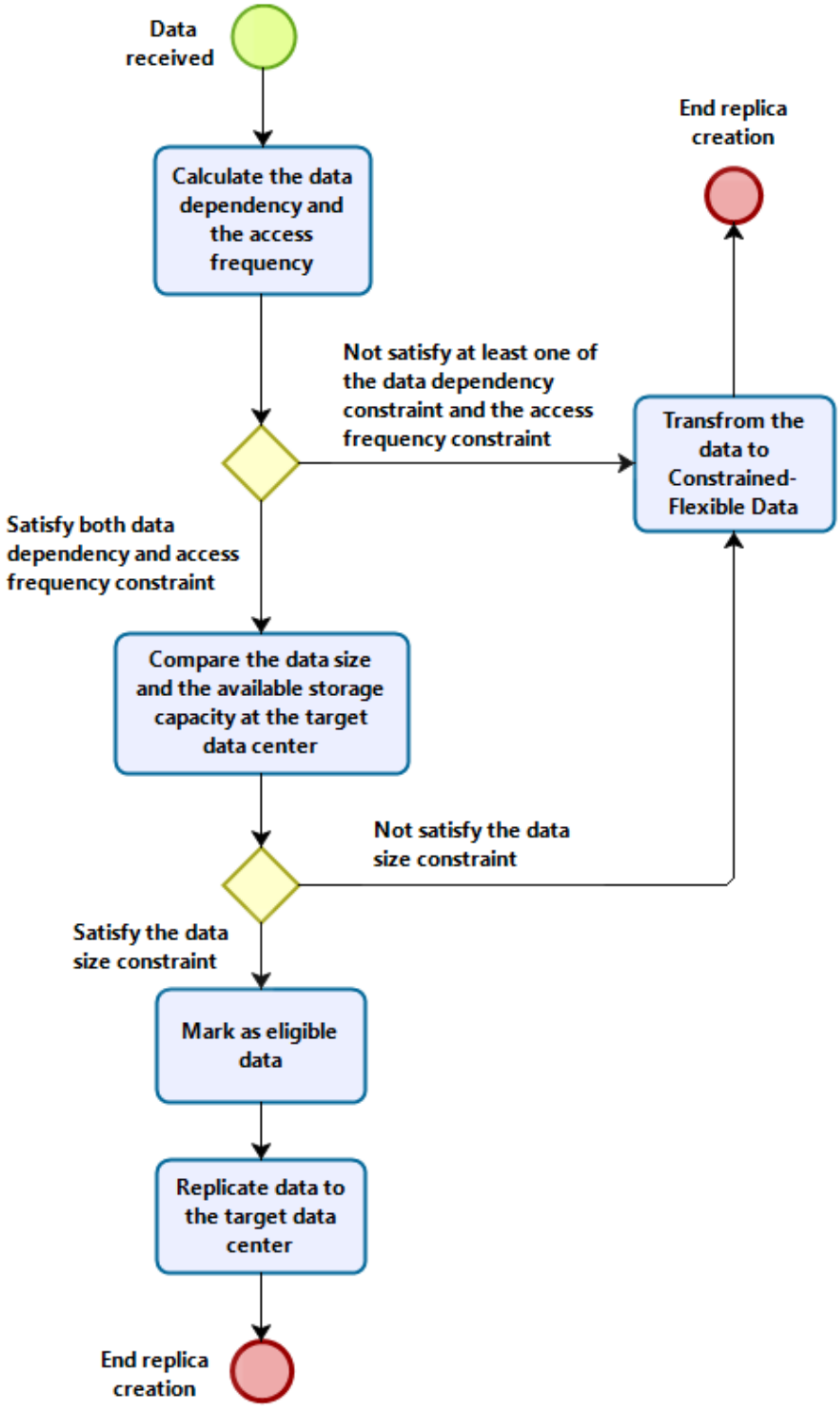

Figure 4.1 Replica creation decision-making process

The eligible data identification for a data d to the data centres where its relevant tasks $J(d)$ located will follow the proposed replica creation decision-making process, as shown in Figure 4.1. Figure 4.1 shows the eligible data identification process for a single data to a single relevant task location. For a data centre $dc \in DC$ in the cloud environment where its relevant tasks $J(d)$ located, the data dependency and the access frequency will be firstly calculated for the data $d \in D$. Then the data d will be checked with the data dependency constraint, the access frequency constraint and the data size constraint. The data d will be marked as the eligible data to this data centre dc if it satisfies all three replica creation constraints and the data type of this data d will be transferred to FFlexD to this data centre dc. If a data d is an eligible data to the data centre dc, then the data d will be replicated to this data centre dc. Otherwise, the data type of this data d to this data centre dc will be transferred to CFlexD.

The proposed replica creation algorithm is shown in Algorithm 4.1. It aims to find the replica creation solution for each data in the cloud environment. The algorithm is firstly initialised by emptying all data types from **Line 1** to **Line 2**. Then the set of data *D* is classified in the cloud environment into FixD and FlexD at **Line 3**. After all steps above, the replica creation decision will be made from **Line 4** to **Line 94** for each pair of data in FlexD. The time complexity of Algorithm 4.1 is O(n^2).

Nine different scenarios are processed in Algorithm 4.1 for each pair of data d_i and d_k, as follows.

1. If $j(d_i)$ and $j(d_k)$ locate in the same location, the scenarios will be as follows.

- Both d_i and d_k satisfy the data dependency constraint, the access frequency constraint and the data size constraint. (**Line 16** to **Line 19**)
- Both d_i and d_k satisfy the data dependency constraint and the access frequency constraint but at least one of them cannot satisfy the data size constraint. (**Line**

20 to **Line 33**; **Line 41** to **Line 43**)

- Both d_i and d_k satisfy all three constraints but the rest available resource cannot accommodate d_i and d_k at the same time (**Line 34** to **Line 40**)

2. If $j(d_i)$ and $j(d_k)$ locate in different locations, the scenarios will be as follows.

- Both d_i and d_k satisfy all three constraints. (**Line 45** to **Line 48**)
- Both d_i and d_k satisfy the data dependency constraint and the access frequency constraint but at least one of them cannot satisfy the data size constraint. (**Line 49** to **Line 61**)

3. There are also some other scenarios as follows.

- Both d_i and d_k satisfy the data dependency constraint. However, only one of d_i and d_k satisfies the access frequency constraint. The data which satisfies the data dependency constraint and the access frequency constraint can also satisfy the data size constraint. (**Line 65** to **Line 75** except **Line 69** to **Line 71** and **Line 77** to **Line 88** except **Line 81** to **Line 83**)
- Both d_i and d_k satisfy the data dependency constraint. However, only one of d_i and d_k satisfies the access frequency constraint. The data which satisfies the data dependency constraint and the access frequency constraint cannot satisfy the data size constraint. (**Line 69** to **Line 71**; **Line 81** to **Line 83**)
- Both d_i and d_k satisfy the data dependency constraint but they cannot satisfy the access frequency constraint. (**Line 89** to **Line 91**)
- Both d_i and d_k cannot satisfy the data dependency constraint. (**Line 92** to **Line 94**)

Algorithm 4.1: Replica Creation Algorithm

Input: DC, J, D

Output: Replica creation solution

1. Initialization { Create $tl[]$, $dl[]$
2. Empty FixD, FlexD, FFlexD, CFlexD}
3. Classify(D) //Classify data into FixD and FlexD
4. **for** each data d_i in FlexD, $d_i \in D$
5. **for** each data d_k in FlexD, $d_k \in D$
6. Calculate $Dep(d_i, d_k), Dep(d_k, d_i), i \neq k$
7. Calculate $AF(d_i)$ and $AF(d_k)$
8. Empty $tl[]$, $dl[]$
9. **if** $Dep(d_i, d_k), Dep(d_k, d_i) \geq \omega$
10. **if** $AF(d_i) \geq \emptyset$ and $AF(d_k) \geq \emptyset$
11. Search dc in DC where $J(d_i)$ located and add into $tl[]$
12. Search dc in DC where $J(d_k)$ located and add into $dl[]$
13. **for** each element $tl[u]$ in $tl[]$ **do**
14. **for** each element $dl[r]$ in $dl[]$ **do**
15. **if** $tl[u] = dl[r]$
16. **if** $Size(d_i), Size(d_k), Size(d_i) + Size(d_k) \leq tl[u]$
17. Transform d_i and d_k from FlexD to FFlexD
18. Replicate d_i and d_k to $tl[u]$
19. Update $ASS(tl[u])$
20. **else if** $Size(d_i) \leq ASS(tl[u])$
21. **if** $Size(d_k), Size(d_i) + Size(d_k) > ASS(tl[u])$
22. Transform d_k from FlexD to CFlexD
23. Transform d_i from FlexD to FFlexD
24. Replicate d_i to $tl[u]$
25. Update $ASS(tl[u])$
26. **end if**
27. **else if** $Size(d_k) \leq ASS(tl[u])$
28. **if** $Size(d_i), Size(d_i) + Size(d_k) > ASS(tl[u])$
29. Transform d_i from FlexD to CFlexD
30. Transform d_k from FlexD to FFlexD
31. Replicate d_k to $tl[u]$
32. Update $ASS(tl[u])$
33. **end if**

34. **else if** $Size(d_i), Size(d_k) \leq ASS(tl[u])$
35. **if** $Size(d_i) + Size(d_k) > ASS(tl[u])$
36. Random transform d_i or d_k from FlexD to CFlexD
37. Transform the rest one from FlexD to FFlexD
38. Replicate d_i or d_k in FFlexD to $tl[u]$
39. Update $ASS(tl[u])$
40. **end if**
41. **else if** $Size(d_i)$, $Size(d_k)$, $Size(d_i) + Size(d_k) > ASS(tl[u])$
42. Transform d_i and d_k from FlexD to CFlexD
43. **end if**
44. **else if** $tl[u] \neq dl[r]$
45. **if** $Size(d_i) \leq ASS(tl[u]), Size(d_k) \leq ASS(dl[r])$
46. Transform d_i and d_k from FlexD to FFlexD
47. Replicate d_i to $tl[u]$ and d_k to $dl[r]$
48. Update $ASS(tl[u])$, $ASS(dl[r])$
49. **else if** $Size(d_i) \leq ASS(tl[u])$, $Size(d_k) > ASS(dl[r])$
50. Transform d_i from FlexD to FFlexD
51. Transform d_k from FlexD to CFlexD
52. Replicate d_i to $tl[u]$
53. Update $ASS(tl[u])$
54. **else if** $Size(d_k) \leq ASS(dl[r])$, $Size(d_i) > ASS(tl[u])$
55. Transform d_k from FlexD to FFlexD
56. Transform d_i from FlexD to CFlexD
57. Replicate d_k to $dl[r]$
58. Update $ASS(dl[r])$
59. **else if** $Size(d_i) > ASS(tl[u])$, $Size(d_k) > ASS(dl[r])$
60. Transform d_i and d_k from FlexD to CFlexD
61. **end if**
62. **end if**
63. **end for**
64. **end for**
65. **else if** $AF(d_i) \geq \emptyset$ and $AF(d_k) < \emptyset$
66. Search dc in DC where $J(d_i)$ located and add into $tl[]$

67. Transform d_k from $FlexD$ to $CFlexD$
68. **for** each element $tl[u]$ in $tl[]$ **do**
69. **if** $Size(d_i) > ASS(tl[u])$
70. Transform d_i from FlexD to CFlexD
71. **else**
72. Transform d_i from FlexD to FFlexD
73. Replicate d_i to $tl[u]$
74. Update $ASS(tl[u])$
75. **end if**
76. **end for**
77. **else if** $AF(d_i) < \emptyset$ and $AF(d_k) \geq \emptyset$
78. Search dc in DC where $J(d_k)$ located and add into $dl[]$
79. Transform d_i from FlexD to CFlexD
80. **for** each element $dl[r]$ in $dl[]$ **do**
81. **if** $Size(d_k) > ASS(dl[r])$
82. Transform d_k from FlexD to CFlexD
83. **else**
84. Transform d_k from FlexD to FFlexD
85. Replicate d_k to $dl[r]$
86. Update $ASS(dl[r])$
87. **end if**
88. **end for**
89. **else if** $AF(d_i) < \emptyset$ and $AF(d_k) < \emptyset$
90. Transformd_i and d_k from FlexD to CFlexD
91. **end if**
92. **else if** $Dep(d_i, d_k), Dep(d_k, d_i) < \omega$
93. Transform d_i and d_k from FlexD to CFlexD
94. **end if**
95. **end for**
96. **end for**

4.1.7 Case study and discussions

A sample workflow in [139] is studied as a case to evaluate the total cost with and without the proposed replica creation strategy. The sample workflow is shown in Figure

4.2. In this case, the storage capacity at each data centre is assumed large enough.

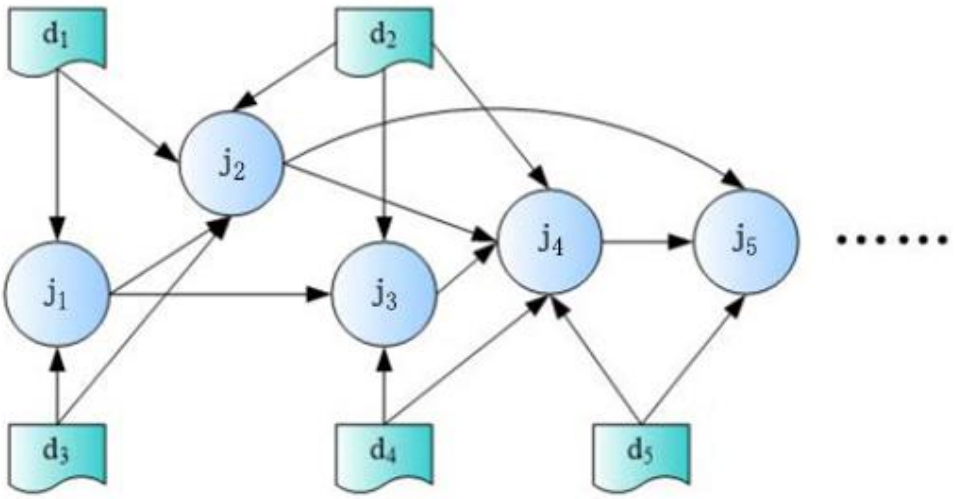

Figure 4.2 Sample workflow [139]

	d_1	d_2	d_3	d_4	d_5
d_1	2	1	2	0	0
d_2	1	3	1	2	1
d_3	2	1	2	0	0
d_4	0	2	0	2	1
d_5	0	1	0	1	2

Figure 4.3 Data dependency matrix [139]

Firstly, the data dependency of each pair of data is calculated for this sample workflow. The result of the data dependency calculation is stored in a data dependency matrix as shown in Figure 4.3. According to this data dependency matrix and the data dependency

threshold parameter ω, the data which satisfies the data dependency constraint can be identified. For example, if the threshold parameter of the data dependency constraint ω is set to 1, then $Dep(d_1, d_2)$ satisfy the data dependency constraint. The access frequency of each data is also calculated for the sample workflow. According to the access frequency of each data and the access frequency threshold parameter $\emptyset$, the data which satisfies the access frequency constraint can also be identified.

Table 4.1 The settings of the main parameters

Parameters	**Value**
$SP(dc)$	0.175 per data unit
α	0.173 per data unit
$Size(d_1)$	10 data unit
$Size(d_2)$	20 data unit
$Size(d_3)$	5 data unit
$Size(d_4)$	10 data unit
$Size(d_5)$	15 data unit
$AF(d_1)$	2 times per time unit
$AF(d_2)$	8 times per time unit
$AF(d_3)$	4 times per time unit
$AF(d_4)$	5 times per time unit
$AF(d_5)$	10 times per time unit

To evaluate the effectiveness of the proposed replica creation strategy, the main parameters are set, as shown in Table 4.1. The data storage time interval for each data will be set to 1 time unit in a consistent value for the calculation convenience. The data storage cost and the data transfer cost are referenced from the cloud storage service pricing model in the Microsoft Azure Australia East area. The data dependency

threshold parameter ω is randomly set to 1 and the access frequency threshold parameter $\emptyset$ is randomly set to 4 times per time unit.

It assumes that the initial data placement and task placement for this sample workflow is already done by [139]. In Figure 4.4, d_1 and d_3 are located in dc_1, d_2 and d_4 are located in dc_2, and d_5 is located in dc_3. j_1 and j_2 are located in dc_1, j_3 and j_4 are located in dc_2, and j_5 is located in dc_3.

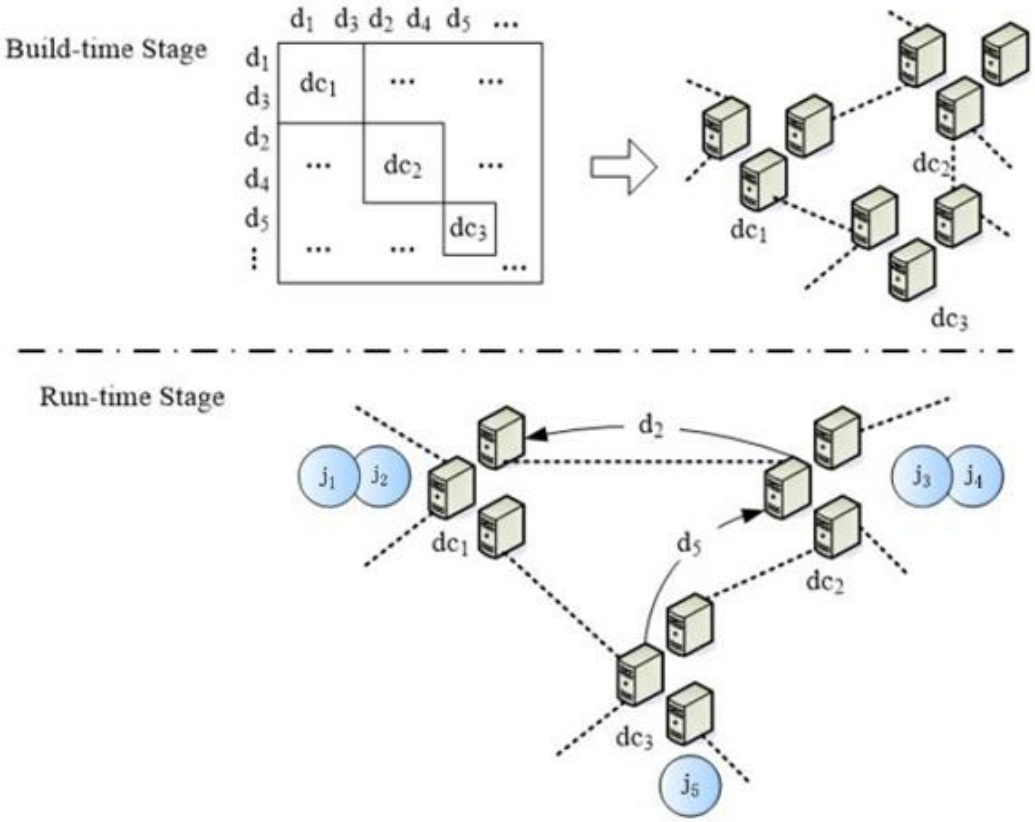

Figure 4.4 Initial data placement in sample workflow [139]

It is also clear in Figure 4.4 that, if the proposed replica creation strategy is not applied, d_2 is located at dc_2 and should be accessed remotely by j_2 which is located in dc_1. d_5 is located at dc_3 and should be accessed remotely by j_4 which is located in dc_2.

By applying the proposed replica creation strategy in this case, d_2 and d_5 are two eligible data for replica creation. Hence, d_2 and d_5 should be replicated to dc_1 and dc_2, respectively. After that, the new replicas of d_2 and d_5 can be accessed locally by j_2 and

j_4, respectively.

The total cost is firstly tested by applying the proposed replica creation strategy into the cloud environment. In this case, d_2 and d_5 will be replicated to dc_1 and dc_2, respectively, as abovementioned. Besides, the total cost without the proposed replica creation strategy applied is calculated. In this case, d_2 will be accessed 8 times remotely by j_2 located in dc_1 and d_5 will be accessed 10 times remotely by j_4 in dc_2. The cost of other data is ignored because they will be accessed locally. The total cost comparison is shown in Figure 4.5.

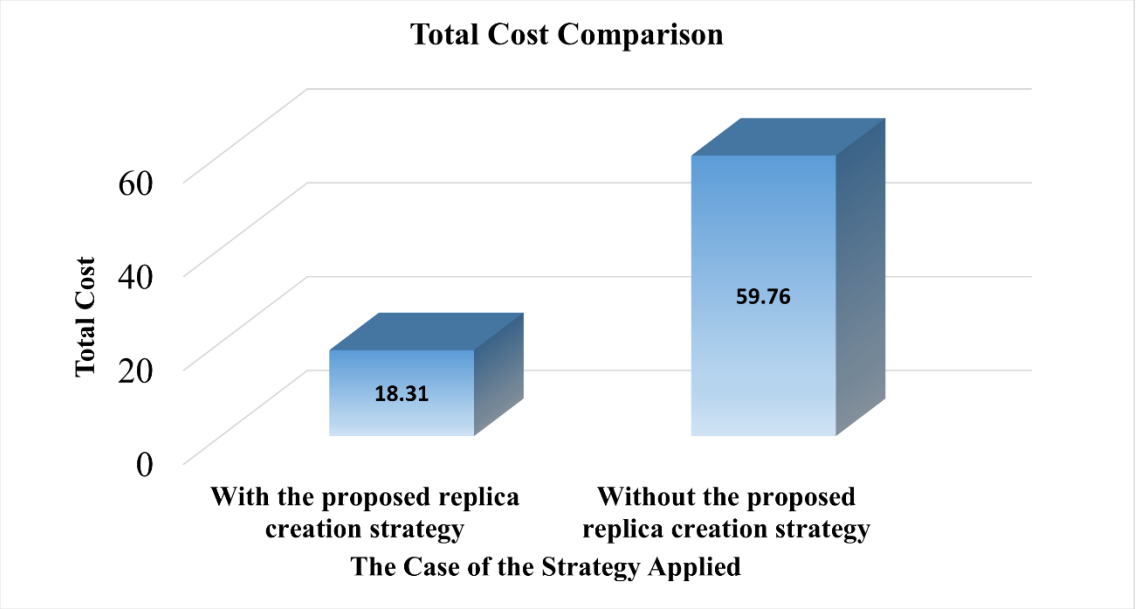

Figure 4.5 Total cost comparison

It is evident that the total cost has a sharp decrease by applying the proposed replica creation strategy. There is a 69.36% decrease in terms of total cost from 59.76 to 18.31 by applying the proposed replica creation strategy as shown in Figure 4.5. As a result, the proposed replica creation strategy can significantly reduce the total cost for cloud applications.

4.2 Cloud Map Oriented and Cost Efficiency Driven Replica Creation

Based on the findings from the replica creation strategy in Section 4.1, data dependency

and access frequency can significantly influence the replica creation process. Data dependency refers to the data relationship between a pair of data. Access frequency refers to the frequency of access in a specific time duration.

In this research, data dependency and access frequency are still followed to use as two of the replica creation constraints. However, as discussed in Chapter 2, the cloud map should be considered to make a more precise replica creation decision. Each data centre can be recognized as an individual host entity in the cloud environment. A specific data in one specific data centre may have multiple data relationships to other data inside this data centre and outside this data centre. The data relationship can be further categorised into local data relationship and remote data relationship. Therefore, a detailed analysis of the data relationship inside data centre and outside data centre is required to identify the local data relationship and the remote data relationship, as this research is conducted at the data centre level. Therefore, different to Section 4.1, the data dependency is analysed and classified into two new categories, Within-DataCentre Data Dependency and Between-DataCentre Data Dependency, to identify the local data relationship and the remote data relationship for the data in the cloud environment, respectively.

Besides, nine different replica creation scenarios are addressed in Section 4.1. However, those replica creation scenarios increase the complexity of the proposed replica creation algorithm in Section 4.1. Thus, this research develops two eligible data candidate pools to reduce the algorithm complexity. The two eligible data candidate pools enable fast eligible data identification for replica creation. By identifying the overlapping elements in these two eligible data candidate pools, the data which is highly-dependent and hot-accessed can be directly collected as the eligible data for replica creation. The data which cannot satisfy the data dependency constraint or the access frequency constraint will not be processed in the replica creation algorithm. Thus, the complexity of the

replica creation algorithm can be reduced.

Apart from that, the proposed replica creation strategy in Section 4.1 focuses on cost reduction as the optimisation objective only. It aims to place the newly created replicas to the locations of all its relevant tasks. The proposed replica creation strategy in Section 4.1 lacks control to the number of replicas. Therefore, the number of replicas might not be the optimal case sometimes. Thus, how to control the number of replicas while cutting costs is a major research question in this research. A recommended access frequency threshold value will be identified to achieve the optimal cost reduction per replica.

4.2.1 Assumed scenarios

Before the start of the proposed replica creation strategy in this research, the same assumptions are made by following Section 4.1, in which initial data placement and task placement have been completed by using the strategy from [139]. Data and tasks are allocated into geographical data centres in DC. Besides, this strategy assumes that each data centre has enough resources to store the data replicas. Thus, the data size constraint can be ignored in this strategy. Apart from that, this research assumes that all data in D are flexible data.

4.2.2 System model

As mentioned in Section 4.1, the data dependency represents the data relationship between each pair of data. The data dependency between d_i and d_k is calculated same to the strategy in Section 4.1, as shown in Eq. 4.1.

This replica creation strategy is a cloud map oriented strategy which aims to analyse the local data relationship and the remote data relationship according to the cloud map. Two novel data dependency categories are defined, Within-DataCentre Data Dependency (W-DCD) and Between-DataCentre Data Dependency (B-DCD) for further analysing

the data relationship inside data centre and outside data centre. For a data $d_i \in D$, W-DCD(d_i) is the data dependency between the data d_i and all other correlated data in D within the same location of d_i. B-DCD(d_i) is the data dependency between the data d_i and all other correlated data in D outside the same location of d_i.

A $DCD(dc, d_i)$ function is used to calculate W-DCD(d_i) and B-DCD(d_i) for the data d_i at the data centre dc. W-DCD(d_i) and B-DCD(d_i) can be calculated using Eq. 4.8 and Eq. 4.9.

W-DCD(d_i) = $\sum_{k=1}^{n} Dep(d_i, d_k)$, $i \neq k$ (d_i and d_k store at the same location) (4.8)

B-DCD(d_i) = $\sum_{k=1}^{n} Dep(d_i, d_k)$, $i \neq k$ (d_i and d_k store at different locations) (4.9)

For a data d_i, if B-DCD(d_i) > W-DCD(d_i), this data will be added into a new data set called High-Dependent Data (HDD). A $DepCompare(d_i)$ function is used to compare between W-DCD(d_i) and B-DCD(d_i) for the data $d_i \in D$.

The access frequency of each data is calculated same as proposed in Section 4.1. The access frequency $AF(d_i)$ is counted for each data $d_i \in D$. Then the sum of the access frequency of all data AF_{total} is calculated as in Eq. 4.10. Then the average access frequency of all data, AF_{avg}, is calculated as in Eq. 4.11, where $Num(D)$ denotes the total amount of data in D. A $AFCalculation()$ function is used to calculate the value of AF_{total} and AF_{avg}. An access frequency threshold value $\emptyset$ is set for the access frequency constraint. The access frequency threshold value $\emptyset$ can be dynamically changed from 0 to $Num(D)$ in order to identify an optimal $\emptyset$ value which enables the optimal cost reduction per replica with balancing the total cost and the number of replicas.

$$AF_{total} = \sum_{i=1}^{n} AF(d_i),\ d_i \in D \quad (4.10)$$

$$AF_{avg} = \frac{AF_{total}}{Num(D)} \quad (4.11)$$

If $AF(d_i) > \emptyset * AF_{avg}$, then the data d_i will be added into a new data set called Hot-Access Data (HAD). A $AFCompare(d_i)$ function is designed to compare the value between $AF(d_i)$ and $\emptyset * AF_{avg}$ in order to determine whether a data d_i should be categorised into HAD. In addition, the cost model in Section 4.1 is also followed to use in this research.

4.2.3 Eligible data candidate pool for replica creation

This research develops two new types of data sets, High-Dependent Data (HDD) and Hot-Access Data (HAD). These two data sets are compared to identify the eligible data candidates for making the replica creation decision. In particular, the HAD candidate pool can be enlarged or shrunk by dynamically changing the access frequency threshold value$\emptyset$.

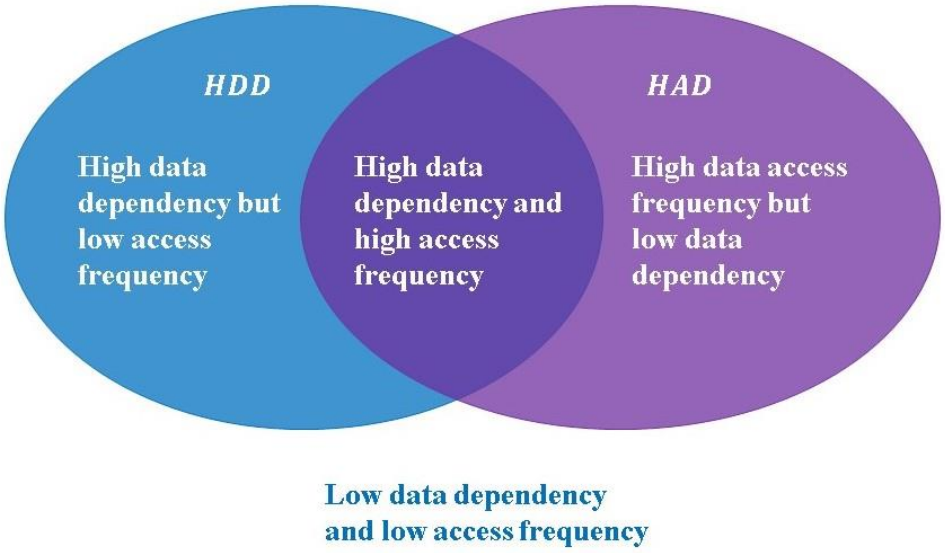

Figure 4.6 Four different situations segmented by HDD and HAD

The eligible data candidates can be identified by analysing the overlapping elements in HDD and HAD. These eligible data candidates are both highly-dependent and hot-accessed.

The replicas of these eligible data candidates should be created and placed into

appropriate data centres by using the same replica placement strategy proposed in Section 4.1. The eligible data candidate pool for replica creation is shown in Figure 4.6. HDD and HAD segment the whole data pool in four different situations as shown in Figure 4.6 and the four different situations can be described in Table 4.2.

Table 4.2 Four different data situations

Situations	**Data belongs to …**
High data dependency but low access frequency	HDD but not in HAD
High access frequency but low data dependency	HAD but not in HDD
High data dependency and high access frequency	Both HDD and HAD
Low data dependency and low access frequency	Not in both HDD and HAD

4.2.4 Recommended value of $\emptyset$

This replica creation strategy is also a cost efficiency driven strategy which aims to achieve the optimal cost efficiency performance in terms of the cost reduction per replica. A recommended value of the access frequency threshold value $\emptyset$ will be returned when the result of the following Eq. 4.12 is optimal, where $TC_{initial}$ denotes the total cost when there is no replica creation strategy applied, and $TC_{current}$ and $NR_{current}$ denote the current total cost value and the current number of replicas, respectively, when the access frequency threshold value $\emptyset$ stays at a specific value. A cost efficiency evaluation parameter CE is introduced to evaluate the cost efficiency in terms of the cost reduction per replica, which can be calculated as in Eq. 4.12. It means the cost reduction per replica is optimal when CE is maximum value at a specific value

of $\emptyset$. Then, this value of $\emptyset$ can be returned as the recommended value of $\emptyset$.

$$CE = \frac{TC_{initial} - TC_{current}}{NR_{current}} \quad (4.12)$$

4.2.5 Replica creation algorithms

Two algorithms are proposed in this cloud map oriented and cost efficiency driven replica creation strategy. Algorithm 4.2 aims to control the replica creation process for each eligible data candidate. It can also contribute to obtaining the recommended value of $\emptyset$. Firstly, the array $rec[]$ for storing the recommended value $\emptyset$ and the array $eva[]$ for storing the evaluation parameter CE will be created at **Line 1**. The size of these two arrays is set to 1 at **Line 2**. Then $rec[]$ will be emptied and $eva[0]$ will be initially set to 0 at **Line 3**.

Then $\emptyset$ is dynamically changed from 0 to $Num(D)$ by stepping a self-defined increment at **Line 4** to evaluate the cost reduction per replica under each $\emptyset$. For each $\emptyset$, Algorithm 4.3 is initiated to identify all eligible data for replica creation at **Line 5**. After that, the new replicas for all eligible data are created at **Line 6** and they are placed to their relevant task locations at **Line 7** by adopting the same replica placement method proposed in Section 4.1. The number of replicas will be counted at **Line 8**. $TC_{current}$ and $TC_{initial}$ will also be calculated at **Line 9**. Finally, the evaluation parameter CE will be calculated at **Line 10** for each $\emptyset$. The comparison between CE and $eva[]$ is conducted from **Line 11** to **Line 16**. If $CE > eva[]$ then, CE will be replaced into $eva[]$ at **Line 12** and its corresponding $\emptyset$ will be loaded into $rec[]$ at **Line 13**. At the same time, the corresponding current number of replicas $NR_{current}$ will be recorded and updated at **Line 14**. Otherwise, if $CE \leq eva[]$, $\emptyset$ will be changed to the next step at **Line 16**. $\emptyset$ will be increased by following its step at **Line 16** until all cases of $\emptyset$ completed from 0 to $Num(D)$. Then $eva[]$, $rec[]$ and $NR_{current}$ will be loaded at **Line 19**. The $\emptyset$ value in $rec[]$ will be marked as the recommended value at **Line 20**. Finally,

the recommended value of $\emptyset$ from $rec[]$ and the number of replicas from $NR_{current}$ will be returned at **Line 21** and **Line 22**. The time complexity of Algorithm 4.2 is O(n).

Algorithm 4.2: Replica Creation and Recommendation Value Analysis Algorithm

Input: $DC, D, CSP,$

Output: The recommended value of $\emptyset$, the number of replicas $NR_{current}$

1. Initialization { Create $rec[]$ and $eva[]$
2. Set $Sizeofrec[] = 1$ and $Sizeofeva[] = 1$
3. Empty $rec[]$ and set $eva[0] = 0$ }
4. **for** $\emptyset = 0, \emptyset \leq Num(D)$, $\emptyset++$
5. Do Algorithm 4.3
6. Create new replicas for all eligible data
7. Place all replicas to corresponding relevant task locations
8. Count the number of replicas $NR_{current}$
9. Calculate $TC_{current}$ and $TC_{Initial}$
10. Calculate CE
11. **if** $CE > eva[]$
12. Replace CE into $eva[]$
13. Load current $\emptyset$ into $rec[]$
14. Record and update $NR_{current}$
15. **else**
16. $\emptyset++$
17. **end if**
18. **end for**
19. Load $eva[]$, $rec[]$ and $NR_{current}$
20. Mark $rec[]$ as the recommended value
21. Return the recommended value of $\emptyset$ from $rec[]$
22. Return the number of replicas $NR_{current}$

Algorithm 4.3 aims to identify the eligible data for replica creation, which will be iteratively executed at **Line 3** in Algorithm 1 until all eligible data are returned. In Algorithm 4.3, the locations of all data are located at **Line 1**. Then AF_{total} and AF_{avg} will be calculated by collecting the access frequency of each data in D at **Line 2**.

Then the eligible data will be identified for each data $d \in D$ from **Line 3** to **Line 19**. All data dependencies for the data d are calculated at **Line 4**. The access frequency of the data d will also be collected at **Line 4**. Then the location of d will be loaded at **Line 5**. W-DCD(d) and B-DCD(d) will be calculated by the function $DCD(dc, d)$ at **Line 6** based on the location of d. Then the function $DepCompare(d)$ will be used to compare between W-DCD(d) and B-DCD(d) at **Line 7**. The data d will be added into HDD if B-DCD(d) > W-DCD(d) from **Line 8** to **Line 10**. The data access frequency will also be compared by the function $AFCompare(d)$ at **Line 11**. The data d will be added into HAD if $AF(d) > \emptyset * AT_{avg}$ from **Line 12** to **Line 14**. After that if the data d belongs to both HDD and HAD, it will be marked as an eligible data for replica creation at **Line 16**. All eligible data for replica creation will be returned to the output at **Line 19**. The time complexity of Algorithm 4.3 is O($log_2 n$).

Algorithm 4.3: Eligible Data Identification for Replica Creation

Input: $DC, D, CSP, \emptyset$

Output: All eligible data

1. Locate the location of all data
2. Calculate AF_{total} and AF_{avg}
3. **for** each d in D
4. Calculate all data dependencies for d and collect $AF(d)$
5. Load dc in DC where d located
6. Calculate W-DCD(d) and B-DCD(d) by function $DCD(dc, d)$
7. $DepCompare(d)$
8. **while** B-DCD(d) > W-DCD(d)
9. Add d to HDD

10. **end while**

11. $AFCompare(d)$

12. **while** $AF(d) > \emptyset * AT_{avg}$

13.. Add d to HAD

14. **end while**

15. **if** $d \in \{\text{HDD} \cap \text{HAD}\}$

16. Label d as an eligible data for replica creation

17. **end if**

18. **end for**

19. Return all eligible data

4.2.6 Simulations

4.2.6.1 Simulation settings

Three scientific workflows are performed in different sizes, namely 25 nodes Montage workflow, 30 nodes CyberShake workflow, and 30 nodes LIGO Inspiral workflow, to simulate the effectiveness of the proposed cloud map oriented and cost efficiency driven replica creation strategy. These three types of scientific workflows are referenced and adjusted from [10].

To evaluate the performance of the proposed cloud map oriented and cost efficiency driven replica creation strategy, two simulations are conducted on CloudSim. The first simulation aims to test the total cost performance with and without the proposed cloud map oriented and cost efficiency driven replica creation strategy in all three types of scientific workflows. The second simulation aims to identify the recommended value of the access frequency threshold $\emptyset$ in order to achieve the optimal cost reduction per replica performance for the three types of scientific workflows, respectively.

The data items of the Montage workflow are shown in Table 4.3. The applied Montage

workflow has 18 data and 25 tasks. The data items of the CyberShake workflow are shown in Table 4.4. The applied CyberShake workflow has 5 data and 30 tasks. The data items of the LIGO Inspiral workflow are shown in Table 4.5. The applied CyberShake workflow has 8 data and 30 tasks.

Table 4.3 The data items of the Montage workflow

Data number	**Access frequency**	**Data size**
d_1	1	0.29
d_2	45	4000
d_3	45	4000
d_4	45	4000
d_5	45	4000
d_6	45	4000
d_7	107	0.26
d_8	107	270
d_9	1	7.2
d_{10}	1	2.3
d_{11}	1	2.8
d_{12}	1	21
d_{13}	1	12
d_{14}	1	7.2
d_{15}	1	165430
d_{16}	1	165430
d_{17}	1	6600
d_{18}	1	320

Table 4.4 The data items of the CyberShake workflow

Data number	Access frequency	Data size
d_1	90	220
d_2	572	5500
d_3	574	0.3
d_4	200	2000
d_5	1	2100

Table 4.5 The data items of the LIGO Inspiral workflow

Data number	Data access frequency	Data size
d_1	42	800
d_2	84	150
d_3	42	8600
d_4	14	230
d_5	79	300
d_6	14	320
d_7	35	940
d_8	42	1200

The pricing model of four real cloud service providers including Amazon, Microsoft, AT&T, and Google are applied, as shown in Table 4.6. The data storage cost rate and the data transfer cost rate are included in the pricing model of these four real cloud service providers.

Besides, the cost model proposed in Section 4.1 is followed as abovementioned. Apart from that, the data storage time interval for each data is set to 1 for the cost calculation convenience in order to make the data storage time interval consistent in each cloud

service provider.

Table 4.6 The pricing model of the cloud service providers

Cloud service provider	**Storage service**	**Storage price (per data unit)**
Amazon	Amazon S3	0.025
Microsoft	Microsoft Azure	0.034
AT&T	AT&T Cloud Storage	0.040
Google	Google Cloud Storage	0.026
Data transfer cost	0.070 per data unit	

4.2.6.2 Simulation results

In Simulation 1, four comparative scenarios on all three scientific workflows are tested. As shown in Figure 4.7, it is evident that the proposed cloud map oriented and cost efficiency driven replica creation strategy can significantly decrease the total cost of all three scientific workflows in Scenario 1 in comparison with all other three comparative scenarios. The four comparative scenarios are listed as follows.

- Scenario 1: The proposed replica creation strategy applied.
- Scenario 2: No replication strategy applied.
- Scenario 3: Only data dependency constraint applied.
- Scenario 4: Only data access times constraint applied.

The proposed cloud map oriented and cost efficiency driven replica creation strategy in Scenario 1 has 94.12%, 99.10%, and 69.91% total cost reduction on the Montage workflow, the CyberShake workflow, and the LIGO Inspiral workflow, respectively, in comparison with the Montage workflow, the CyberShake workflow, and the LIGO Inspiral workflow under Scenario 2. Besides, the proposed cloud map oriented and cost efficiency driven replica creation strategy in Scenario 1 has 40.11% and 92.49% total

cost reduction on the Montage workflow and the CyberShake workflow, respectively, in comparison with the Montage workflow and the CyberShake workflow under Scenario 3. The proposed cloud map oriented and cost efficiency driven replica creation strategy in Scenario 1 achieves almost equal total cost in comparison with the LIGO Inspiral workflow under Scenario 3. Apart from that, the proposed cloud map oriented and cost efficiency driven replica creation strategy in Scenario 1 has 31.41%, 92.80%, and 67.32% total cost reduction on the Montage workflow, the CyberShake workflow, and the LIGO Inspiral workflow, respectively, in comparison with the Montage workflow, the CyberShake workflow, and the LIGO Inspiral workflow under Scenario 4.

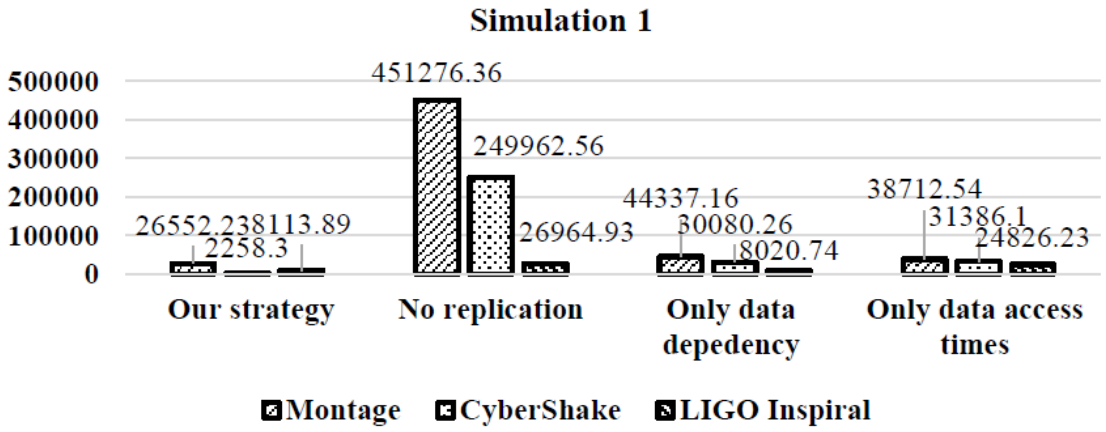

Figure 4.7 The result of simulation 1

In Simulation 2, the access frequency threshold Ø is dynamically changed by a self-defined increment 0.001 in order to view the impact on the number of replicas and the total cost. The simulation 2 results are shown in Figure 4.8, Figure 4.9, and Figure 4.10. The values of Ø, which corresponds to the change points of the cost reduction per replica, are shown in these figures. As shown in Figure 4.8, there is a clear fluctuation on the total cost and the number of replicas when the value of Ø dynamically increases from 0 to 18 in the Montage workflow. It is recommended that the cost reduction per replica CE remains at a maximum level when Ø stays at 2.3 in the Montage workflow.

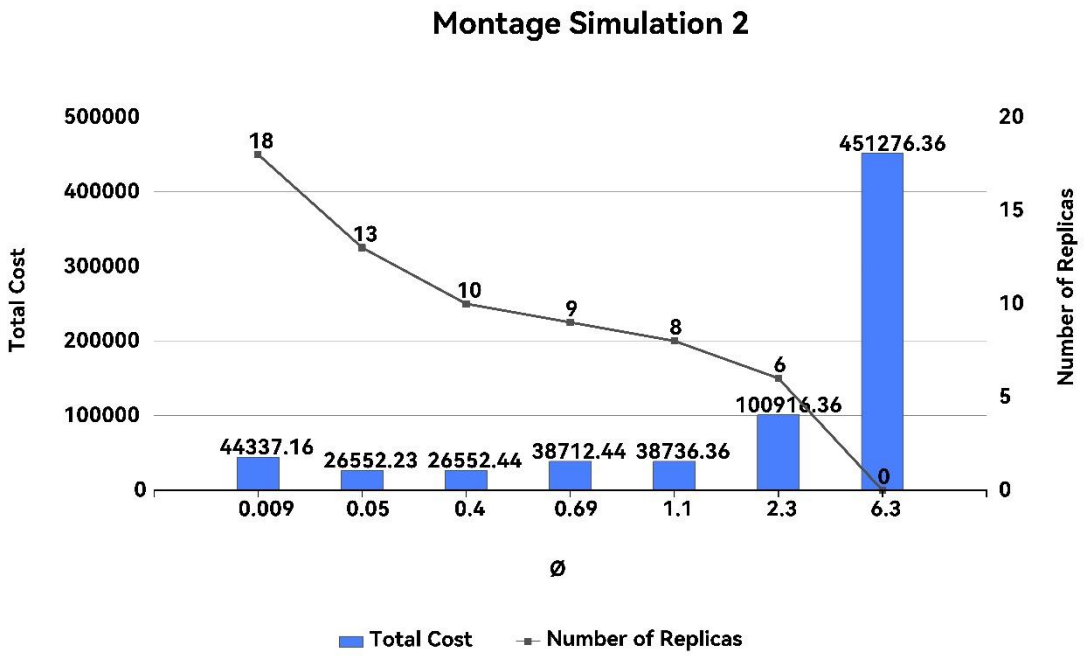

Figure 4.8 The result of the Montage workflow in Simulation 2

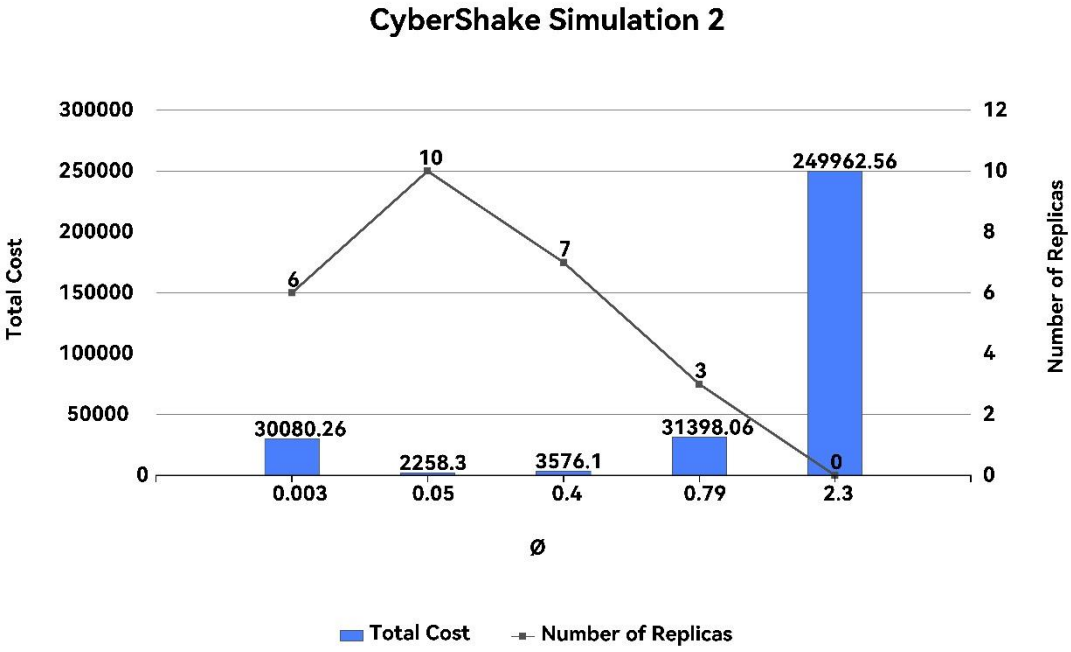

Figure 4.9 The result of the CyberShake workflow in Simulation 2

Similarly, the recommended value of Ø for the CyberShake workflow and the LIGO Inspiral workflow can be identified in Figure 4.9 and Figure 4.10, respectively. The value of Ø dynamically increases from 0 to 5 in the CyberShake workflow. The value of

Ø dynamically increases from 0 to 8 in the LIGO Inspiral workflow. It is recommended that the cost reduction per replica CE exists at a maximum level when Ø stays at 0.79 in the CyberShake workflow and when Ø stays at 0.95 in the LIGO Inspiral workflow.

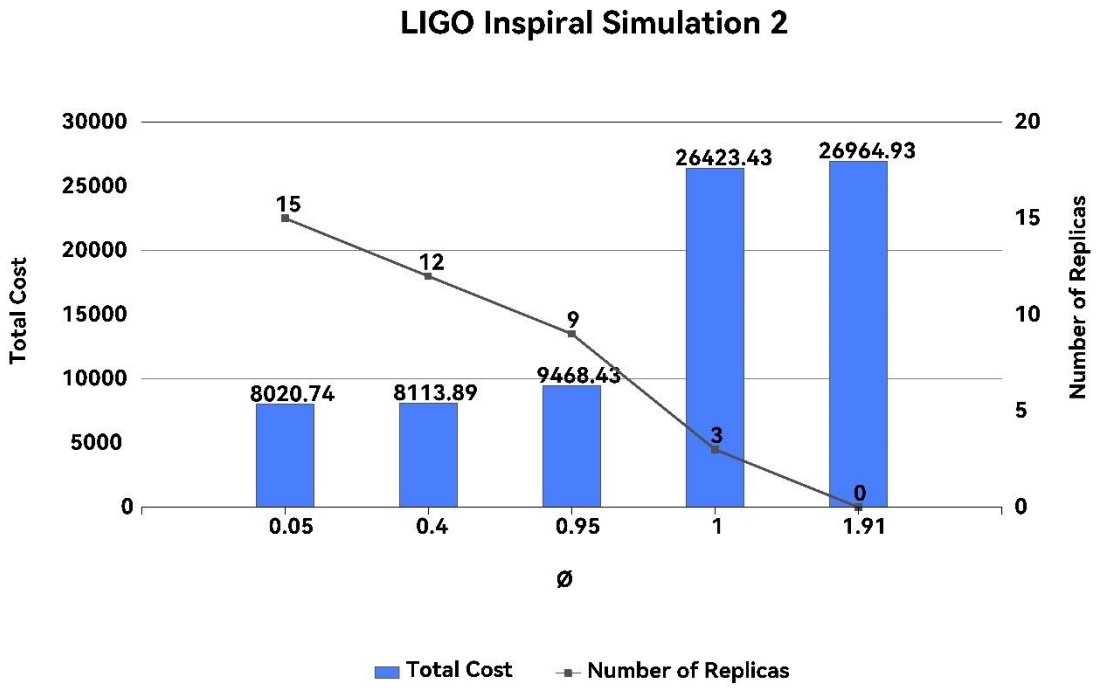

Figure 4.10 The result of the LIGO Inspiral workflow in Simulation 2

4.3 Network Performance Based Replica Selection

As described in the last two replica creation strategies in Section 4.1 and 4.2, replica creation is a significant process to create multiple data copies at multiple data centres. By applying the replica creation strategy, the cloud performance can be improved, as proved in Section 4.1 and Section 4.2. According to the literature in Chapter 2, the cloud performance can be further improved by applying the replica selection strategy. The replica selection strategy can guide the tasks to access the optimal data replica.

Although the existing research has made significant progress in the replica selection strategies, there are still some research gaps to be filled. Most of the current replica selection strategies focus on how to select a data replica to access without consideration

of the impacts among multiple concurrent-running cloud application instances under limited network capability. In particular, some of the current research might not be suitable to address the scenario when the increased number of cloud application instances and data access needs result in overloading in certain parts of the network. Thus, those replica selection strategies might not be able to perform well in the cloud environment with heavy workloads. Besides, most of the current data replication strategies model the network performance metrics in an isolated way. It may lose the comprehensiveness of evaluating the overall network performance.

To address the problems mentioned above, a network performance oriented replica selection strategy (NPRS) is proposed to avoid the potential network overloading problems. It also aims to increase the number of concurrent-running cloud application instances at the same time. Three common network performance metrics are applied to measure the network performance to generate the optimal replica selection solution. The replica selection decision for a single cloud application instance is made in the context of multiple concurrent-running cloud application instances with consideration of their access needs to different data replicas and their impacts on the network resource. The final replica selection decision may be one of the following:

- Find the best replica for a data access request to access
- Recreate a new replica for the required replica to access

4.3.1 System modelling

In this NPRS strategy, three network performance metrics are used to evaluate the overall network performance in the cloud environment and further support the replica selection decision making. Network latency, available network resource, and error rate are considered as three major network performance metrics. The network latency depends on a variety of factors including the data transmission speed of the network

path, the nature of the transmission medium, the physical distance between two locations, the size of the transferred data, the number of other data transmission requests being handled concurrently, etc. It is usually measured as either one-way delay or round-trip delay. The round-trip delay is commonly quoted by network managers for the reason that it can be measured from a single point. Ping value has been widely used to measure the round-trip delay. To simplify the problem in this research, the network latency of a data centre $dc \in DC$, $NL(dc)$, is modelled as a constant value.

The bandwidth is adopted to represent the network resource in this research because it is one of the most common network performance metrics to measure the network resource in the past literature. The bandwidth consumption of a specific data centre $dc \in DC$, $BC(dc)$, can be calculated using the equation in Eq. 4.13, where J^{dc} is the set of tasks accessing this data centre dc, $Size(j^{dc})$ is the size of the data that is requested by a task $j^{dc} \in J^{dc}$, and $Len(j^{dc})$ denotes the task execution duration of the task j^{dc}.

$$BC(dc) = \sum_{j^{dc} \in J^{dc}} \frac{Size(j^{dc})}{Len(j^{dc})} \quad (4.13)$$

Similarly, the bandwidth consumption of a specific data $d \in D$ being accessed, $BC(d)$, can be calculated as in Eq. 4.14, where $ACLen(d)$ denotes the maximum time length of the data d being accessed by its relevant tasks.

$$BC(d) = \frac{Size(d)}{ACLen(d)} \quad (4.14)$$

Then the available bandwidth of the data centre dc, $AB(dc)$, refers to the difference between the maximum bandwidth of this data centre $maxB(dc)$ and the current bandwidth consumption in this data centre $BC(dc)$, which can be presented as in Eq. 4.15.

$$AB(dc) = maxB(dc) - BC(dc) \quad (4.15)$$

The error rate is also a significant parameter to evaluate the network performance

because a network with a lower error rate is always greater to use than a network with a higher error rate. The error rate of the data centre $dc \in DC$, $ER(dc)$, refers to the ratio of the total number of transmitted data units in error to the total number of transmitted data units, which can be represented as in Eq. 4.16.

$$ER(dc) = \frac{Total\ number\ of\ transmitted\ data\ units\ in\ error}{Total\ number\ of\ transmitted\ data\ units} \tag{4.16}$$

4.3.2 Network performance based replica selection (NPRS) strategy

4.3.2.1 NPRS replica selection strategy

The proposed network performance based replica selection (NPRS) strategy is an evaluation method to the overall cloud network performance by applying three network performance metrics mentioned in Section 4.3.1 to select the best replica site to access. The *Min-Max* normalisation method is applied in the proposed NPRS strategy to develop a comprehensive evaluation among different network performance metrics.

Three weighted parameters are developed to configure the network performance metrics. W_{AB}^{dc} denotes the weight of the available bandwidth metric of the data centre dc, W_{NL}^{dc} denotes the weight of the network latency metric of the data centre dc, and W_{ER}^{dc} denotes the weight of the error rate metric of the data centre dc. The final weight of this data centre dc, $FW(dc)$, can be formulated in Eq. 4.17, where NC_{AB}^{dc} denotes the normalisation component of the available bandwidth metric of the data centre dc, NC_{NL}^{dc} denotes the normalisation component of the network latency metric of the data centre dc, and NC_{ER}^{dc} denotes the normalisation component of the error rate metric of the data centre dc. For a request to access a data that has replicas at multiple data centres, the data centre with the maximum final weight value will be selected as the optimal data access route. Tie-breaking is done randomly.

$$\begin{cases} FW(dc) = W_{AB}^{dc} * NC_{AB}^{dc} + W_{NL}^{dc} * NC_{NL}^{dc} + W_{ER}^{dc} * NC_{ER}^{dc} \\ W_{AB}^{dc} + W_{NL}^{dc} + W_{ER}^{dc} = 1 \end{cases} \tag{4.17}$$

All of these three network performance metrics have big impact on network performance. However, different network performance metrics should be treated in different ways depending on their own nature. The available bandwidth metric with the highest value should be the best case while the network latency metric and the error rate metric with the highest value should be the worst case. Hence, the normalisation processes of three network performance metrics can be formulated for a specific data centre $dc_z \in DC$ as in Eq. 4.18, Eq. 4.19, and Eq. 4.20, respectively.

$$NC_{AB}^{dc_z} = \frac{AB(dc_z) - min\{AB(dc)\}}{max\{AB(dc)\} - min\{AB(dc)\}}; dc \in DC \quad (4.18)$$

$$NC_{NL}^{dc_z} = \frac{max\{NL(dc)\} - NL(dc_z)}{max\{NL(dc)\} - min\{NL(dc)\}}; dc \in DC \quad (4.19)$$

$$NC_{ER}^{dc_z} = \frac{max\{ER(dc)\} - ER(dc_z)}{max\{ER(dc)\} - min\{ER(dc)\}}; dc \in DC \quad (4.20)$$

4.3.2.2 Replica re-creation mechanism

In this research, the initial replica creation and placement is assumed that it is already completed by applying the same rule to the proposed replica creation strategies in Section 4.1 or Section 4.2.

The proposed replica re-creation mechanism in the NPRS strategy will be initiated when the loss of replica availability occurs due to network overloading issues. A new replica of the required data will be re-created by considering the resource load at the data centres. Firstly, the required data to be replicated should be FlexD as mentioned in Section 4.1. Then a set of eligible data centres which meet the resource requirement of the required data are identified. The eligible data centres with sufficient resources will be sorted based on their current resource in descending order. The eligible data centre with the largest available resource will be chosen to create the new replica by copying a new required replica from the nearest replica-ready data centre. Tie-breaking is done randomly.

4.3.2.3 NPRS algorithm

The NPRS algorithm is a nested algorithm that contains two algorithms, the NPRS replica selection algorithm and the NPRS replica re-creation algorithm. The NPRS replica selection algorithm and the NPRS replica re-creation algorithm collaboratively determine the optimal data access route for each data request. The NPRS replica selection algorithm is shown in Algorithm 4.4. **Line 2** maps all required data in D to the data requests from the task j in J, and those required data will be added into an array for listing the required data $rd[]$. Each element in $rd[]$ will be tried to identify its optimal data access route $OptRoute$ for the task j from **Line 3** to **Line 23**. For each element $rd[v]$ in $rd[]$, all replica-ready data centres are mapped to $rd[v]$ and add into a new array for the replica-ready data centre list $rr[]$ at **Line 4**. For each element $rr[u]$ in $rr[]$, if the available bandwidth of $rr[u]$ satisfies the condition in **Line 6**, $FW(rr[u])$ will be calculated in **Line 7** under Eq. 4.17. Then it will be added into a new array $fw[]$ for listing the final weight including different weights of the elements in $rr[]$ at **Line 8**. Otherwise, the algorithm will move to the next element in $rr[]$ at **Line 10**. The capacity of $fw[]$ will be checked at **Line 13**. If the capacity of $fw[]$ is empty, then the NPRS replica re-creation algorithm will be initiated at **Line 14**. Otherwise, the new array $fw[]$ will be sorted by the Reverse QuickSort algorithm at **Line 16** and then $fw[0]$ will be mapped to its corresponding value $fw(rr[w])$ at **Line 17**. Then the optimal replica selection route $OptRoute$ will be generated from **Line 18** to **Line 19** to guide the optimal data access route for the task j. After that, $fw[]$ and $rr[]$ will be emptied at **Line 20** and then the algorithm will be move to the next element in $rd[]$ at **Line 21** until the replica selection solution of all elements in $rd[]$ is founded. The replica selection solution for all tasks in J can be worked out by iteratively run Algorithm 4.4 from **Line 1** to **Line 24**. The time complexity of Algorithm 4.4 is O(n^2).

Algorithm 4.4: NPRS Replica Selection

Input: D, DC, J

Output: Optimal data access route $OptRoute$

1. **for** each j in J
2. Map all required data to j and add all required data into $rd[]$
3. **for** $rd[v]$ in $RD[]$, $v = 0$, $v \leq Sizeof(rd[])$
4. Map the replica-ready locations to $rd[v]$ and add into $rr[]$
5. **for** $rr[u]$ in $rr[]$, $u = 0$, $u \leq Sizeof(rr[])$
6. **if** $AB(rr[u]) \geq BC(rd[v])$
7. $FW(rr[u])$ under Eq. 4.17
8. Add $FW(rr[u]) \rightarrow fw[]$
9. **else**
10. $u++$
11. **end if**
12. **end for**
13. **if** $fw[] = NULL$
14. Do Algorithm 4.5
15. **else**
16. Reverse QuickSort $fw[]$
17. Map $fw[0] \rightarrow FW(rr[w])$
18. Load $rd[v]$, $rr[w]$ and j
19. Return $OptRoute = \{rd[v], rr[w], j\}$
20. Empty $fw[]$ and $rr[]$
21. $v++$
22. **end if**
23. **end for**
24. **end for**

The proposed Algorithm 4.5 is used to identify a suitable data centre to re-create a new data replica for the required data because of the availability loss of the required data. In Algorithm 4.5, **Line 1** identifies the qualified data centres where do not have the required data replica and add them into a new array $qual[]$. Then, from **Line 2** to **Line**

9, the qualified data centres in $qual[]$ will be checked their available bandwidth situations in comparison with the bandwidth consumption of the input data $rd[v]$ from Algorithm 4.4. If the resource utilisation condition is satisfied at **Line 3**, then the qualified data centre will be added into a new array $elig[]$ for collecting the eligible data centres for replica re-creation. Otherwise, the next qualified data centre in $qual[]$ will be checked.

After the checking of the resource consumption condition, the capacity of the array $elig[]$ will be further checked from **Line 10** to **Line 25**. If the capacity of $elig[]$ is empty, then the array $qual[]$ will be emptied and the algorithm will back to Line 3 in Algorithm 4.4 to process the next element in $rd[]$. Otherwise, the available bandwidth value will be mapped to each element in $elig[]$ at **Line 14**. Then it will be added to a new array $ab[]$ for storing the available bandwidth information of the eligible data centres in $elig[]$ at **Line 15**. The new array $ab[]$ will be sorted by the Reverse QuickSort algorithm at **Line 17** and then $ab[0]$ will be mapped to its corresponding value $elig[r]$ in $elig[]$ at **Line 18.** Then the optimal replica re-creation route $OptRoute$ will be generated from **Line 19** to **Line 21** to make the replica re-creation to enable the data access for the task j. After that, $qual[], elig[]$, and $ab[]$ will be emptied at **Line 22** and $fw[]$ and $rr[]$ will also be emptied in Algorithm 4.4, at **Line 23** in Algorithm 4.5. At the same time, the algorithm will be move to the next element in $rd[]$ at Line 3 in Algorithm 4.4 at **Line 24**. The time complexity of Algorithm 4.5 is O(n).

Algorithm 4.5: NPRS Replica Re-Creation

Input: $rr[], rd[v]$

Output: Optimal data access route $OptRoute$

1. Remove $rr[]$ from DC and add the rest DC into $qual[]$
2. **for** $qual[c]$, $c = 0$, $c \leq Sizeof(qual[])$ **do**
3. **if** $BC(rd[v]) \leq AB(qual[c])$

4. Add $qual[c] \rightarrow elig[]$
5. $c++$
6. **else**
7. $c++$
8. **end if**
9. **end for**
10. **if** $elig[] = NULL$
11. Empty $qual[]$ and $v++$ at Line 3 in Algorithm 4.4
12. **else**
13. **for** each element in $elig[]$ **do**
14. Map $AB(elig[]) \rightarrow elig[]$
15. Add $AB(elig[]) \rightarrow ab[]$
16. **end for**
17. Reverse QuickSort $ab[]$
18. Map $ab[0] \rightarrow elig[r]$
19. Load $rd[v]$ and $elig[r]$
20. Load j from Algorithm 4.4
21. Return $OptRoute = \{rd[v], elig[r], j\}$
22. Empty $qual[], elig[]$, and $ab[]$
23. Empty $fw[]$ and $rr[]$ in Algorithm 4.4
24. $v++$ at Line 3 in Algorithm 4.4
25. **end if**

4.3.3 Simulations

To evaluate the effectiveness of the proposed NPRS strategy, three simulations under three different scenarios are performed on OMNeT++ 5.4.1. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators [41][86].

The comparison between the proposed NPRS strategy and the least response time replica selection algorithm [42] is performed in all three simulations. The Montage scientific workflow and the LEAD Mesoscale Meteorology workflow are applied as the input cloud application instances.

In Simulation 1 and Simulation 2, a total of 25 workflow instances in a single workflow type are executed. While in Simulation 3, 25 instances of each workflow type are executed. All of the data settings of the Montage scientific workflow and the LEAD Mesoscale Meteorology workflow are referenced and adjusted from [10] and [43].

Table 4.7 The network latency of each data centre

Data Centre	**Network Latency (*ms*)**
dc_1	40
dc_2	100
dc_3	25
dc_4	150
dc_5	200
dc_6	250
dc_7	50

A multi-cloud environment is constructed, including 3 different cloud service providers with a total of 7 data centres. The traditional three-replica placement strategy has been applied to this simulation environment. The network bandwidth in each data centre is set to 100 Gbps with a 100 Gigabit Ethernet network connection. The network latency of each data centre is shown in Table 4.7.

Several assumptions are made for the following simulations. Firstly, all cloud application instances are requested by a single user to keep a consistent view of the network latency in all three simulations. Secondly, the network performance evaluation metrics are assumed to be collectable in the network. Thirdly, W_{ER} is set to 0 because the network is assumed to be performed well in this simulation environment, which aims to simplify the problem. The network with a certain error rate will also be tested in the following chapters.

4.3.5.1 Simulation 1 – Synchronous instance input

In Simulation 1, the Montage workflow instances are synchronously input into the simulation environment. W_{NL} and W_{AB} are randomly set to 50% and 50%, respectively. Then the proposed NPRS algorithm is compared to the least response time replica selection algorithm to test the network bandwidth changes and the number of concurrent-running workflow instances.

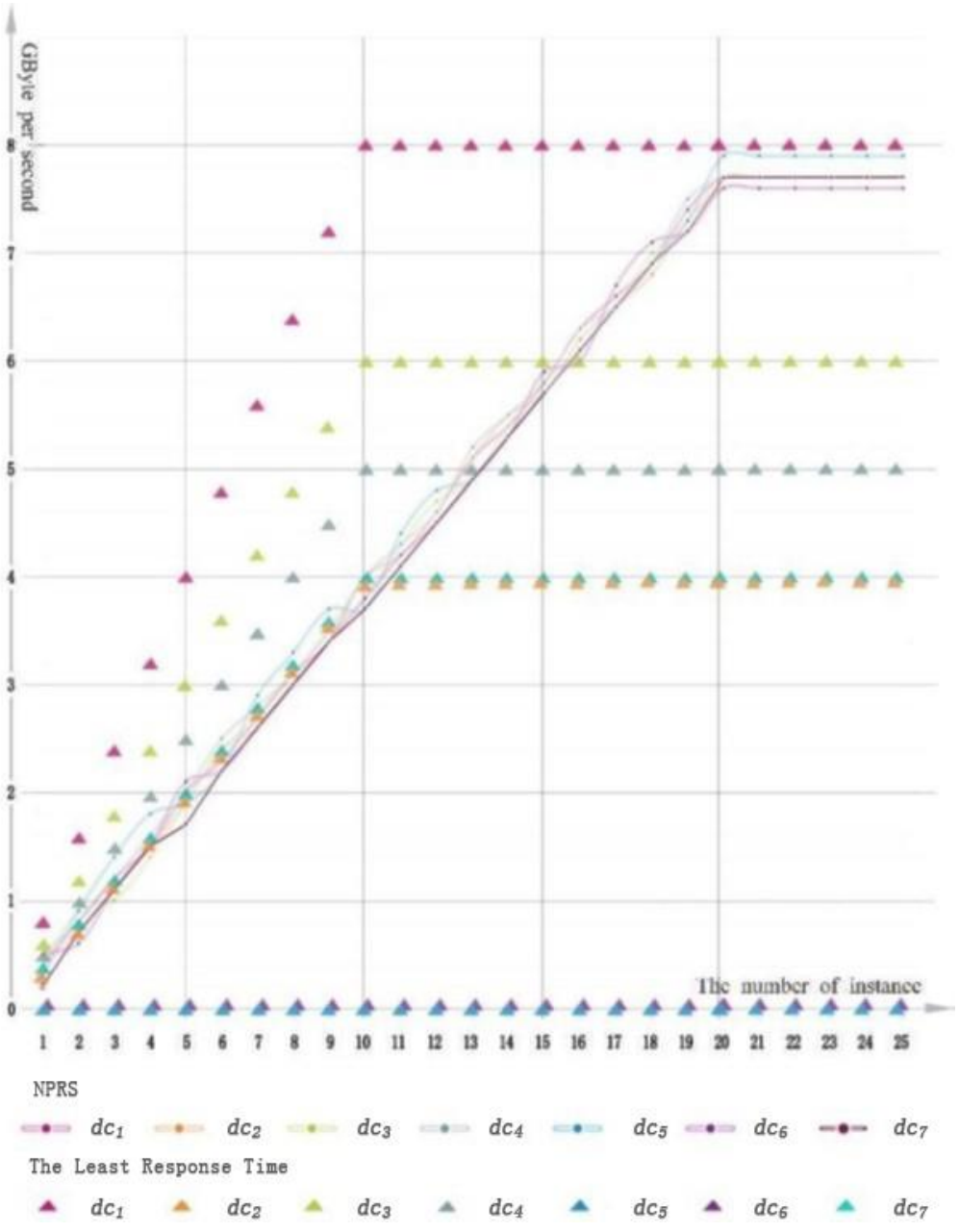

Figure 4.11 Simulation result 1 – Synchronous instance input

The result of Simulation 1 is shown in Figure 4.11. Along with the synchronous

instance input, the least response time replica selection algorithm has a sharp bandwidth utilisation increase in some data centres in the simulation environment when the number of concurrent-running workflow instances is before 10. Then those data centres will encounter the network overloading problem when the number of concurrent-running workflow instances reaches 10. The number of concurrent-running workflow instances peaks at 10 by applying the least response time algorithm.

Differently, the proposed NPRS strategy has a milder bandwidth utilisation increase before 20 concurrent-running workflow instances and then peaks the number of concurrent-running workflow instances at 20. Therefore, it is evident that the proposed NPRS strategy can significantly increase the number of concurrent-running workflow instances and balance the network utilisation when the workflow instances are synchronously input.

4.3.5.2 Simulation 2 – Asynchronous instance input

In Simulation 2, the Montage scientific workflow instances are asynchronously input into the simulation environment for testing the proposed NPRS replica selection strategy and the least response time replica selection algorithm. The instances will be input into the simulation environment one by one. In Simulation 2, W_{NL} and W_{AB} are randomly set to 10% and 90%, respectively.

Along with the asynchronous instance input, the least response time replica selection algorithm still has a sharp bandwidth utilisation increase in some data centres when the number of concurrent-running workflow instances is before 10. Then those data centres will encounter the network overloading problem when the number of concurrent-running workflow instances reaches 10. The number of concurrent-running workflow instances still peaks at 10 by applying the least response time algorithm. Differently, the proposed NPRS strategy has a milder bandwidth utilisation increase before 14

concurrent-running workflow instances and then still peaks the number of concurrent-running workflow instances at 14. Therefore, it is clear that the proposed NPRS strategy can still increase the number of concurrent-running workflow instances and balance the network utilisation when the workflow instances are asynchronously input.

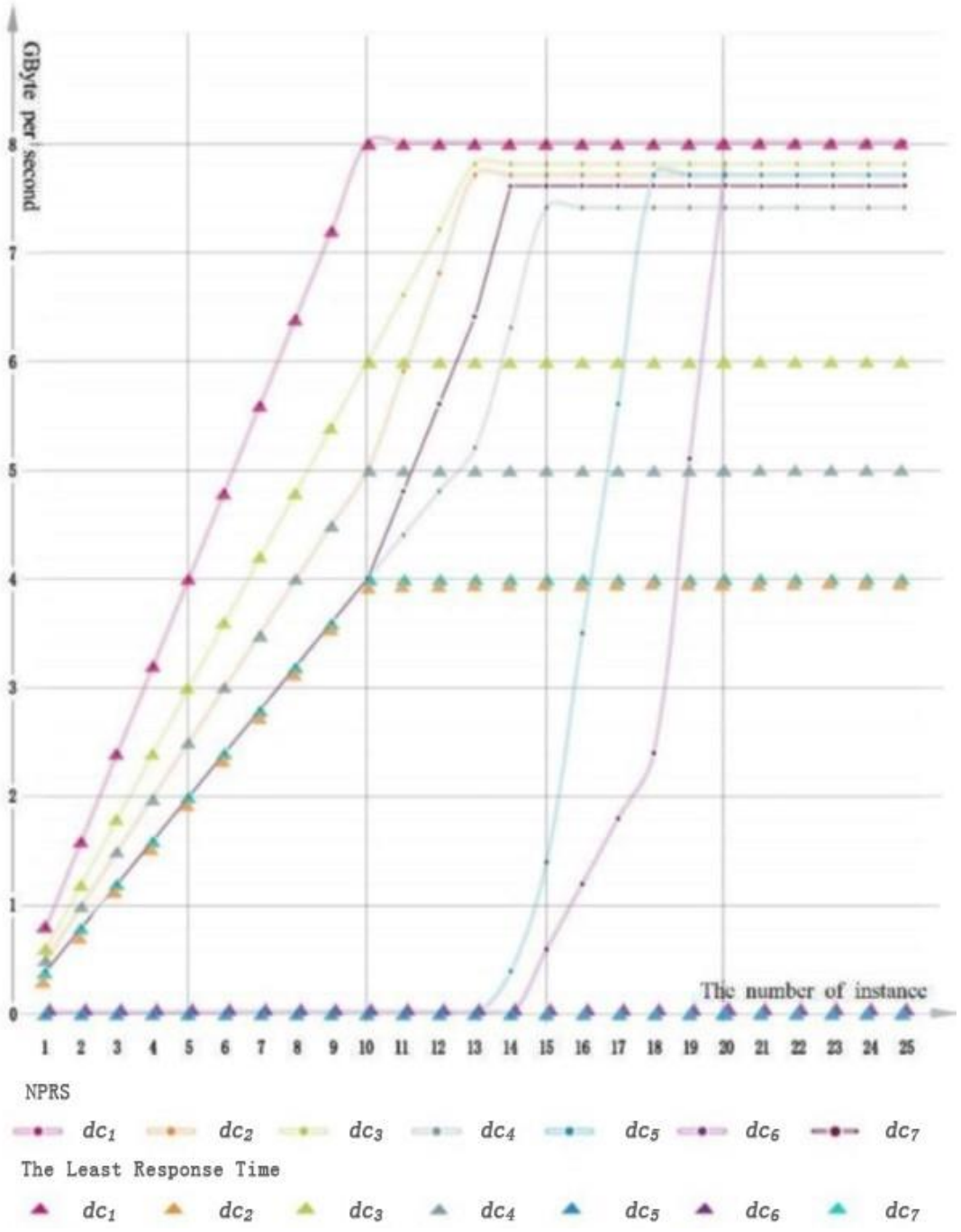

Figure 4.12 Simulation result 2 – Asynchronous instance input

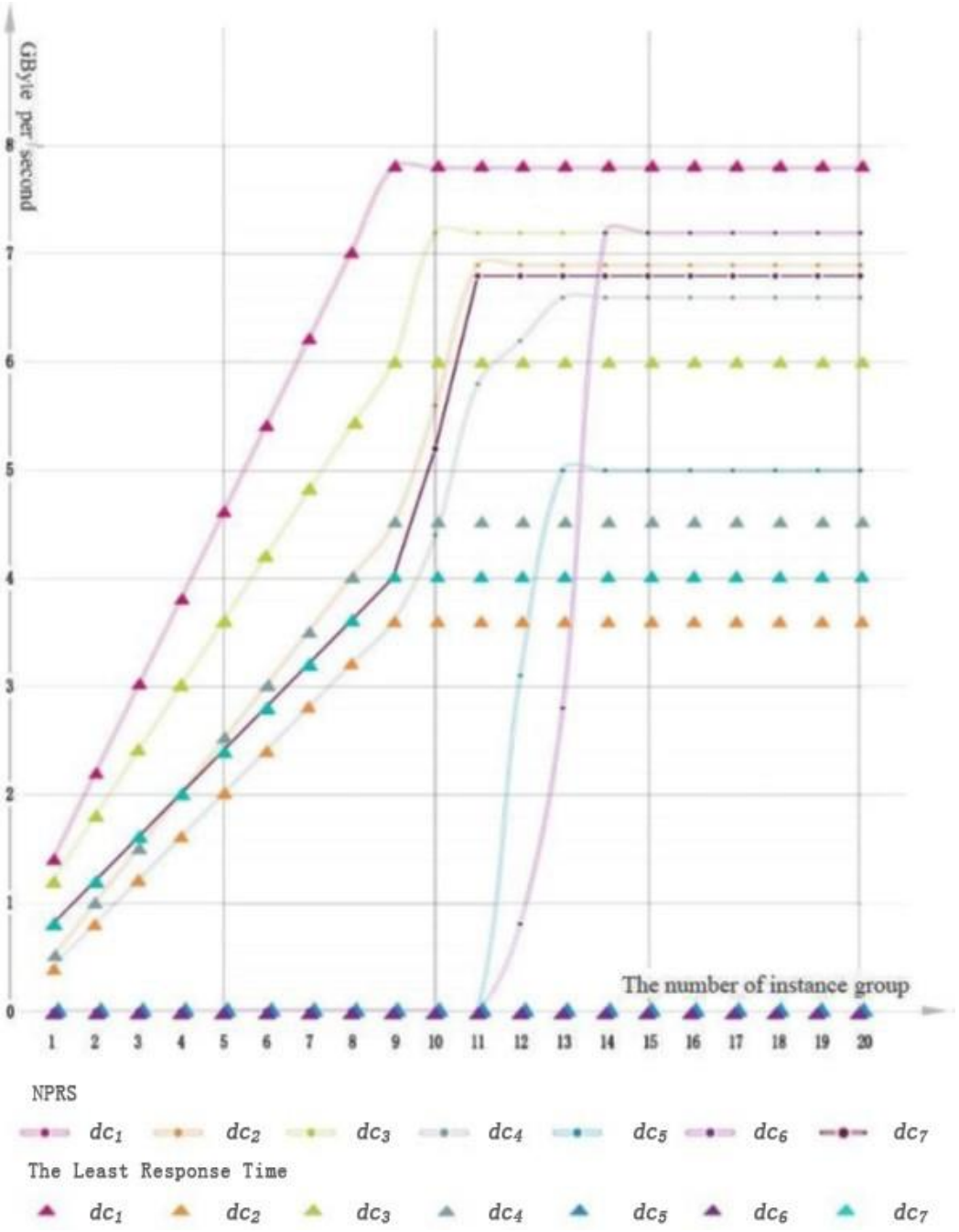

Figure 4.13 Simulation result 3 – Iterative input with instance group

4.3.5.3 Simulation 3 – Iterative input with instance group

In Simulation 3, the workflow instance group including both the Montage workflow instances and the LEAD Mesoscale Meteorology workflow instances are iteratively input for testing the proposed NPRS strategy and the least response time replica selection algorithm. The workflow instance group is input into the simulation environment one group by one group. Each workflow instance group contains one

Montage workflow instance and one LEAD workflow instance. The parameter setting in Simulation 3 is same to Simulation 1. The result of Simulation 3 is shown in Figure 4.13.

Along with the iterative instance input of the workflow instance group, the least response time algorithm peaks the number of concurrent-running workflow instance groups at 9. The proposed NPRS strategy peaks the number of concurrent-running workflow instances when the number of concurrent-running workflow instance groups reaches 13. Therefore, the proposed NPRS strategy can also significantly increase the number of concurrent-running workflow instance groups. At the same time, the proposed NPRS strategy still has better and more balanced bandwidth usage in comparison with the least response time replica selection algorithm, as shown in Figure 4.13.

4.4 Summary

In Chapter 4, two replica creation strategies and one replica selection strategy are proposed for creating, placing, and selecting the data replicas. The replica placement rule is included in the first two replica creation strategies. In Section 4.1, the first replica creation strategy is developed to reduce the total cost by considering both the data dependency and the access frequency when making the replica creation decision. A data classification method is introduced to classify the flexible data into two new data types, free-flexible data and constrained-flexible data. The free-flexible data can identify the flexible data which can be freely replicated to a specific data centre, while the constrained-flexible data can identify the flexible data which cannot be replicated to a specific data centre. A replica creation algorithm is proposed to address nine different scenarios for each pair of data. The total cost reduction is achieved by applying the proposed replica creation strategy in Section 4.1.

In Section 4.2, the second replica creation strategy is proposed to achieve the optimal cost reduction per replica by identifying a recommended access frequency threshold value. The data dependency and the access frequency are followed to use as two constraint parameters to constrain the replica creation. The data dependency is categorised into Within-DataCentre Data Dependency and Between-DataCentre Data Dependency to analyse the local data relationship and the remote data relationship, respectively. An eligible data candidate pool is introduced to identify the highly-dependent and hot-access data. The proposed replica creation strategy can obtain a recommended value of the access frequency threshold parameter to achieve the optimal cost reduction per replica.

In Section 4.3, a NPRS strategy is proposed for increasing the number of concurrent-running workflow instances and balancing the resource load. The network performance based replica selection method is developed by jointly considering different network performance metrics. Different network performance metrics are treated in different ways in the proposed replica selection method. A nested replica selection algorithm is introduced to handle both the normal case and the limited case of the cloud network. The proposed NPRS strategy in Section 4.3 achieves a greater number of concurrent-running workflow instances and more balanced network resource load in comparison with the least response time replica selection algorithm.

Chapter 5 Reactive Fault Tolerance for Independent Tasks

As stated in Chapter 2, the data replication strategies can enable a replica-applied cloud environment to protect the cloud environment against the upcoming faults. Normally, the task is operated on the primary data in the cloud environment. By adopting the proposed data replication strategies in Chapter 4, multiple data replicas can be created into multiple data centres by the replica creation strategies. Besides, a replica selection strategy can be developed for guiding the data access. In case that the primary data becomes inaccessible, the task execution can remain continuous by accessing one of the required replicas if the replica-applied cloud environment is deployed beforehand. The replica-applied cloud environment is widely adopted to achieve cloud robustness in many famous cloud environments such as Google Cloud and Amazon S3. However, it is not sufficient for improving the overall cloud performance when encountering faults. The reactive fault tolerance strategies are still needed to achieve better overall cloud performance if the fault already occurred.

Therefore, two reactive fault tolerance strategies are proposed for the independent tasks in the replication-applied cloud environment. The task resubmission technique and the task migration technique are introduced into these reactive fault tolerance strategies by integrating the proposed replica selection method proposed in Section 4.3.

5.1 Utility-Based Fault Tolerance for Independent Tasks

As explained in earlier chapters, most of the contemporary fault tolerance strategies paid insufficient attention to both the network performance and the attributes of affected tasks. When the tasks at the faulty data centre are resubmitted to other data centres, the impacts to the overall cloud performance have been largely overlooked. The task resubmission operations and the task migration operations may deplete the resources of

other data centres [102]. In addition, some tasks may not be able to catch the deadline even if they have been rescued to access one of the required data replicas in the replica-applied cloud environment under fault scenarios. This may result in cloud resiliency decrease, user dissatisfaction, reputation damage, future profit reduction, and economic compensation. Therefore, both the resource load of the cloud data centres and the task attributes are all significant factors to cloud resiliency. It is desirable to develop a reactive fault tolerance strategy to the replica-applied cloud environment, which fully considers both the resource load of the cloud data centres and the attributes of the affected tasks.

A utility-based reactive fault tolerance (UBFT) strategy is proposed for more efficient independent task rescue at the faulty data centre. The common network performance metrics and the task attributes are jointly considered as the major constraint parameters in this strategy. A utility function is developed to prioritise the tasks to be rescued, in other words, to be resubmitted. For each independent task rescue operation, the network performance at each replica-ready data centre is evaluated to find the optimal task resubmission route so that the task can be migrated out of the faulty data centre to access the required data replica. By doing so, this strategy aims to achieve better cloud resiliency in terms of task resilience ratio, task rescue utility, and task operation profit. The simulation results show that the proposed UBFT strategy has better task resilience ratio, task rescue utility, and task operation profit than the other three comparative HDFS, RR, and JSQ strategies.

5.1.1 System modelling

5.1.1.1 Definitions

The following definitions are defined in this reactive fault tolerance strategy. Cloud resiliency refers to the ability to rescue the tasks when a fault occurs at a data centre.

Cloud resiliency is commonly measured in terms of task resilience ratio (TRR). TRR refers to the ratio of the tasks successfully rescued from the faulty data centre to the total number of tasks to be rescued at the faulty data centre. The TRR for a faulty data centre can be demonstrated as follows in Eq. 5.1.

$$\text{Task Resilience Ratio} = \frac{Total\ number\ of\ rescued\ jobs}{Total\ number\ of\ jobs\ to\ be\ rescued} \tag{5.1}$$

The task utility refers to the modelled value of the tasks. The task rescue utility is the sum of the task utilities of those tasks which have been rescued from the faulty data centre. The task rescue utility is also used to measure cloud resiliency.

The task operation profit is directly proportional to revenue, and it is also inversely proportional to cost. The task operation profit refers to the subtracting result between the revenue and the cost. The cloud service providers always prefer to alleviate the task operation profit decrease as much as possible, at least at an acceptable level, after the fault occurred. The task operation profit is also used to measure cloud resiliency from an economic perspective.

5.1.1.2 Task urgency and task operation profit model

Each task $j \in J$ is associated with a hard deadline $DEAD(j)$. In this research, the task deadline is defined as a specific point on the timeline. If such a requirement is not specified, the task has an infinite deadline. This research only considers the task with a definite deadline because the task with an infinite deadline does not suffer from the negative influences of the fault and can be resumed when the faulty data centre is fully recovered from the fault.

There are two common task resubmission or migration scenarios for the independent tasks in the cloud environment when handling faults. The task at the faulty data centre might be re-executed if the task is resubmitted or migrated to another cloud data centre, or maybe the task rescued from the faulty data centre will be resumed to complete from

the most recent state after resubmitting or migrating to another cloud data centre if the checkpointing/restarting technique applied. In this research, all tasks are assumed to be re-executed if the task is resubmitted or migrated out of its initial location. To ensure the quality of service in this research, all resubmitted or migrated tasks should satisfy their own deadline requirement. Otherwise, the resubmission or the migration will be deterred.

Each task j also has a task execution duration $Len(j)$ which is determined by the nature of the task j. Besides, the past processing time in its original execution location $PA(j)$ should be considered if the task j has been selected to migrate or redirect out of its initial location. $PA(j)$ equals to 0 if the task has not been executed in its initial location. In addition, the internodal communication delay $IC(j)$ is another factor to be considered because the extra time will be generated when the task j is migrated across multiple network nodes. Furthermore, the input scheduling delay $IS(j)$ is the extra time generated by scheduling the execution of the task j.

Most importantly, the task urgency (UR) is defined as the time buffer of the task. The higher the task urgency value is, the more time buffer the task has. The task urgency is formulated as in Eq. 5.2, where $UR(j)$ is the task urgency value of the task j.

$$UR(j) = DEAD(j) - (Len(j) + PA(j) + IC(j) + IS(j)) \quad (5.2)$$

Each task $j \in J$ is also associated with the value of its task operation profit, $PRO(j)$, which is the subtracting result between the revenue and the cost of the task j.

5.1.1.3 Task utility and task rescue utility

The utility function is often used to compare the objects with multiple requirements and attributes. Generally, a data centre prefers to rescue as many tasks as possible to fit their deadline requirements. In this case, task urgency is a significant parameter to be considered for the task priority assignment when handling the tasks at the faulty data

centre. At the same time, cloud service providers always try to maximise their profit. In this case, the tasks that bring more profits to the data centre should have higher importance. The task utility value is proposed to prioritise the tasks by jointly considering the task urgency and the task operation profit.

For the task $j \in J$, the general expression of the task utility $U(j)$ is shown in Eq. 5.3 and should satisfy the condition in Eq. 5.4, where $U_{UR}(j)$ and $U_{PRO}(j)$ denote the utility value of the task urgency and the task operation profit of the task j, respectively. W_{UR} and W_{PRO} denote the corresponding weight of the task urgency and the task operation profit, respectively.

$$U(j) = W_{UR} * U_{UR}(j) + W_{PRO} * U_{PRO}(j) \tag{5.3}$$

$$W_{UR} + W_{PRO} = 1 \tag{5.4}$$

For a specific task $j_m^{dc} \in J^{dc}$ at the faulty location dc, the utility value of the task urgency of this task, $U_{UR}(j_m^{dc})$, is calculated as follows in Eq. 5.5.

$$U_{UR}(j_m^{dc}) = \frac{max\left(UR(j^{dc})\right) - UR(j_m^{dc})}{max\left(UR(j^{dc})\right) - min\left(UR(j^{dc})\right)}; j^{dc} \in J^{dc} \tag{5.5}$$

For a specific task $j_m^{dc} \in J^{dc}$ at the faulty location dc, the utility value of the task operation profit of this task, $U_{PRO}(j_m^{dc})$, is calculated as follows in Eq. 5.6.

$$U_{PRO}(j_m^{dc}) = \frac{PRO(j_m^{dc}) - min\left(PRO(j^{dc})\right)}{max\left(PRO(j^{dc})\right) - min\left(PRO(j^{dc})\right)}; j^{dc} \in J^{dc} \tag{5.6}$$

In this strategy, one of the optimisation objectives is the task rescue utility. The task rescue utility of a faulty data centre dc, $TRU(dc)$, can be calculated in Eq. 5.7, where ϑ is a variable parameter to judge the task rescue situation. If the task is rescued from the faulty data centre, ϑ will be 1, otherwise 0.

$$TRU(dc) = \sum_{j_m^{dc} \in J^{dc}} \vartheta * U(j_m^{dc}) \tag{5.7}$$

5.1.1.3 Replica selection method

The replica selection schema aims to guide the optimal replica-ready data centre

selection to access the required data replicas by evaluating the resource load at each replica-ready data centre. This strategy adopts the replica selection method proposed in Section 4.3 to find the optimal data access route for the task resubmission and migration operations.

5.1.2 Utility-based fault tolerance (UBFT) strategy and algorithms

Put simply, the proposed utility-based fault tolerance (UBFT) strategy tries to rescue the tasks from the faulty data centre and resubmit them to the backup replica-ready data centre. The task rescue process not only considers the resource load of accessing backup replicas but also strives to satisfy the deadline constraints. To achieve this goal, the UBFT algorithm uses two functions, task resubmission function Resubmission() and task migration function Migration(), to generate fault handling solutions under different scenarios for each task at the faulty data centre.

The UBFT algorithm applies the utility-based ranking method to calculate the task priority for the task resubmission or migration operations. The task utility should be treated in different ways depending on the fault circumstances in different data centres. The task with lower task utility has higher migration priority at the backup data centre while the task with higher task utility has higher migration priority at the faulty data centre. The proposed UBFT algorithm is shown in Algorithm 5.1.

Firstly, the tasks at the faulty data centre will be ranked in a descending order based on their task utility and then add into rank list $ranklist[]$ at **Line 1**. Then the fault handling solution $FTResult$ will be worked out for each element in the $ranklist[]$ by calling the task resubmission function Resubmission() in Function 5.1 at **Line 3**. The input parameter of the task resubmission function Resubmission() is the task in the $ranklist[]$ to be rescued. The $FTResult$ contains a set of data centre information including the task resubmission destination dc_{res} and the task migration destination dc_{mig}. The $FTResult$

will guide the fault handling processes for each task at the faulty data centre from **Line 8** to **Line 12** until all tasks in the $ranklist[]$ are addressed. The time complexity of Algorithm 5.1 is O(n^3).

Algorithm 5.1: UBFT Algorithm

Input: Resource situations at each data centre, J, task utility, fault location

Output: Fault handling solution

1. Quicksort j^{dc} in J^{dc} based on the task utility and add into $ranklist[]$
2. **for** $ranklist[v]$ in $ranklist[]$, $v = 0$, $v \leq sizeof(ranklist[])$ **do**
3. Do Function 5.1
4. Load $FTResult = \{dc_{res}, dc_{mig}\}$
5. Do $FTResult$ {
6. Resubmit $ranklist[v]$ to dc_{res}
7. Migrate $mov[r]$ to dc_{mig} }
8. $v++$
9. **end for**
10. End Algorithm 5.1

The task resubmission function is called at **Line 3** in Algorithm 5.1. The task resubmission function is shown in Function 5.1. When the task submission function is called, the backup replica-ready data centres will be mapped to the input task at **Line 4**. A comparison between the bandwidth consumption of the input task and the available bandwidth of the backup replica-ready data centres is created to find out the optimal task resubmission route from **Line 2** to **Line 22**.

In case that all backup data centres do not have sufficient resource capacity to accommodate a task rescued from the faulty data centre, the task migration function Migration() in Function 5.2 will be initiated at **Line 17** in Function 5.1. The task migration function Migration() in Function 5.2 aims to migrate a current-running task out of a replica-ready backup data centre to release some resources for accommodating a task rescued from the faulty data centre. It is a one-stop nested function for addressing

the limited resource case, in order to avoid the task rescue failures as much as possible.

Function 5.1: Resubmission Function - Resubmission()

Input: Resource situations at each data centre, J, $ranklist[v]$

Output: $FTResult$

1. Initialization {
2. Empty $resdes[]$
3. Set $Sizeof(resdes[]) = 1$ }
4. Map the replica-ready data centres to $ranklist[v]$ and add into $rr[]$
5. **for** each element $rr[u]$ in $rr[]$ **do**
6. Compare $BC(ranklist[v])$ with $AB(rr[u])$
7. **if** $BC(ranklist[v]) \leq AB(rr[u])$
8. $FW(rr[u])$
9. **if** $FW(rr[u]) > FW(resdes[])$
10. Update $rr[u]$ into $resdes[]$
11. Go to Line 19
12. **else**
13. Remain $resdes[]$
14. Go to Line 19
15. **end if**
16. **else**
17. Do Function 5.2
18. **end if**
19. Map $resdes[] \rightarrow dc_{res}$
20. Return $\{dc_{res}, null\} \rightarrow FTResult$
21. Empty $rr[]$
22. **end for**

The task migration function Migration() in Function 5.2 contains a series of operations to release an existing task out of a replica-ready backup data centre to accommodate the task rescued from the faulty data centre. Firstly, **Line 1** collects a set of current-running tasks in $J^{rr[u]}$ on $rr[u]$. A bandwidth utilisation comparison between the bandwidth consumption of the input task and the sum of the bandwidth consumption of $j^{rr[u]}$ and

the available bandwidth in its backup replica-ready data centres will be conducted at **Line 3**. A new group of migratable tasks will be further created in the array $mov[]$ at **Line 5**. A Quicksort algorithm will be applied on $mov[]$ to re-order the task in $mov[]$ in an ascending order based on the task utility at **Line 7**. A comparison between the bandwidth consumption of the migratable task in $mov[]$ and the available bandwidth of its backup replica-ready data centres is conducted at **Line 10** to identify the eligible replica-ready data centres for releasing the migratable task in $mov[]$. Then based on the proposed replica selection method in Section 4.3, the optimal task resubmission route for rescuing the task from the faulty data centre and the optimal task migration route for the migratable task in $rr[u]$ will be finalized from **Line 11** to **Line 26**.

Function 5.2: Migration Function – Migration()

Input: Resource situations at each data centre, $rr[u]$, J, $ranklist[v]$, task utility
Output: $FTResult$

1. Collect the set of current-running tasks $J^{rr[u]}$ in $rr[u]$
2. **for** each $j^{rr[u]}$ in $J^{rr[u]}$ **do**
3. Compare $BC(ranklist[v])$ with $AB(rr[u]) + BC(j^{rr[u]})$
4. **if** $BC(ranklist[v]) \leq AB(rr[u]) + BC(j^{rr[u]})$ and $U(j^{rr[u]}) < U(ranklist[v])$
5. Add $j^{rr[u]}$ into $mov[]$
6. **end if**
7. Quicksort $mov[]$ based on the task utility
8. **for** $mov[r]$ in $mov[]$, $r = 0$, $r \leq Sizeof(mov[])$ **do**
9. Map each replica-ready data centre to $mov[r]$ and add into $migrr[]$
10. Add $migrr[]$ where $BC(mov[r]) \leq AB(migrr[])$ into $eligmig[]$
11. **if** $eligmig[] = null$
12. Return $\{null, null\} \rightarrow FTResult$
13. $r++$
14. **else**
15. **for** $eligmig[c]$ in $eligmig[]$, $c = 0$, $c \leq Sizeof(eligmig[])$ **do**
16. $FW(eligmig[c])$

17. **end for**
18. Load the maximum FW value of $eligmig[w]$ in $eligmig[]$
19. Map $eligmig[w] \rightarrow dc_{mig}$
20. Load $rr[u] \rightarrow dc_{res}$ and $mov[r]$
21. Go to Line 25
22. **end if**
23. **end for**
24. **end for**
25. Return $\{dc_{res}, dc_{mig}\} \rightarrow FTResult$
26. End Function 5.2

5.1.3 Simulation results

To evaluate the effectiveness and efficiency of the proposed UBFT strategy, three simulations are performed on OMNeT++ 5.4.1. A cloud environment was implemented including 5 data centres with 250 circuits of 100 Gbps optical-fibre network integrated at each data centre site. The major parameters of each data centre are shown in Table 5.1. The following settings are applied in all simulations.

- To avoid the fluctuation of uncertain internodal communication delay and input scheduling delay, both internodal communication delay and input scheduling delay are set to 5*ms*.
- To avoid the fluctuation of the uncertain network latency between different users and different data centres, a single user is applied to assign multiple tasks to different data centres. Therefore, the network latency can be regarded as stable between the user and different data centres, as shown in Table 5.1.
- A fault is set to occur at 10*ms* system running time in dc_2, which leads to the closing down of dc_2.
- The task deadline and the task execution duration are randomly set in the range of 0*ms* to 1000*ms*.
- The required data size of each task is randomly selected in the range of 0*GB* to

5GB.

- Each data has 3 replicas that are randomly placed in 5 data centres, one for primary accessed replica and two for backup replicas.

Table 5.1 The major parameters of each data centre

Data Centre	**Maximum Bandwidth (*Gbps*)**	**Network Latency(*ms*)**	**Error Rates**
dc_1	25000	20	0.1%
dc_2	25000	60	0.2%
dc_3	25000	40	0.5%
dc_4	25000	60	0.1%
dc_5	25000	100	0.4%

The proposed UBFT strategy is compared with the typical HDFS robustness strategy applied in the HDFS system, the RR strategy [42] applied in SQL server 2016, and the JSQ strategy applied in Cisco Local Director, IBM Network Dispatcher, and Microsoft SharePoint [28][33][111].

The cloud resiliency of these three strategies is evaluated in terms of task resilience ratio, task rescue utility, and task operation profit. The utility weights are changed in different simulations under the equivalent scenario, the urgency highly-weighted scenario, and the profit highly-weighted scenario to test the effectiveness of the proposed UBFT strategy. To simplify the problem, the proposed UBFT strategy, the typical HDFS robustness strategy, the RR strategy and the JSQ strategy are assumed to implement under a single-fault scenario in all simulations.

5.1.4.1 Simulation 1 – Equivalent utility weights

In Simulation 1, both the utility weights of the task urgency W_{UR} and the task operation profit W_{PRO} are set to 0.5 for evaluating an equivalent utility weight scenario between

the task urgency and the task operation profit. The simulation 1 results of the task resilience ratio, the task rescue utility, and the task operation profit are shown in Figure 5.1, Figure 5.2, and Figure 5.3, respectively.

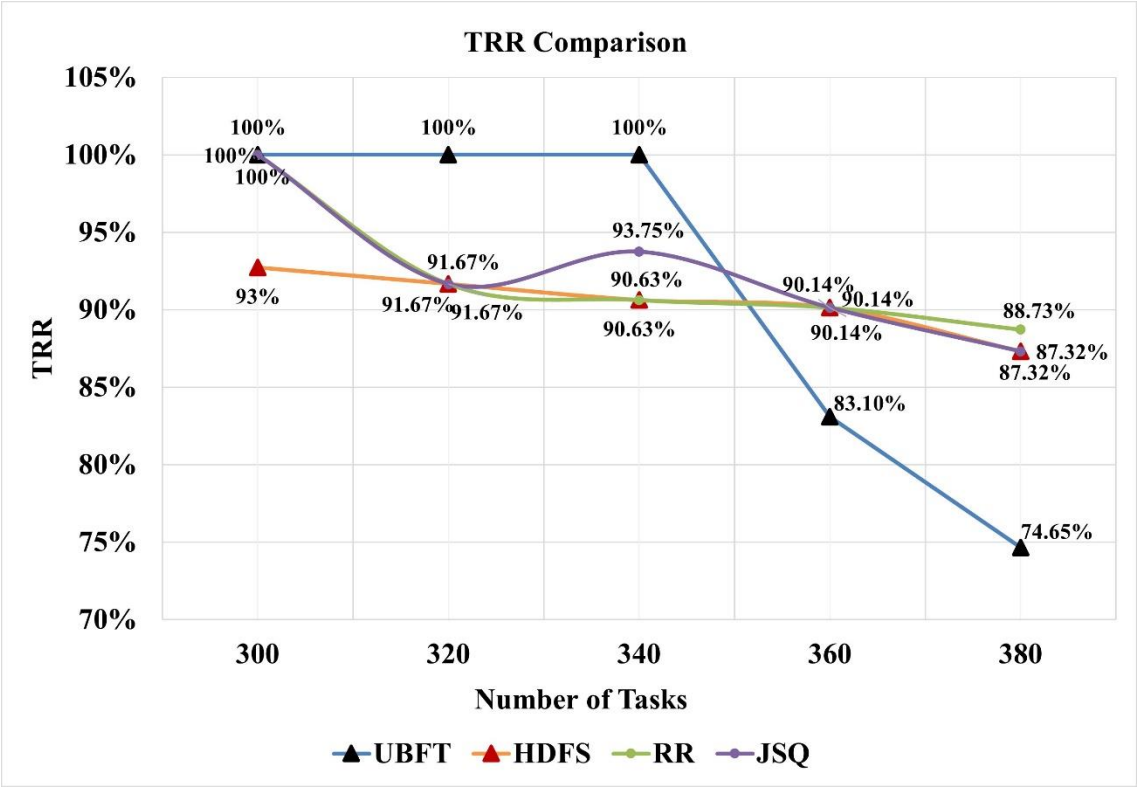

Figure 5.1 The TRR comparison of Simulation 1

In Figure 5.1, the proposed UBFT strategy has better task resilience ratio than the other three comparative strategies when the resource is sufficient to support the task execution. In contrast, when the resource becomes more and more limited, the proposed UBFT strategy aims to migrate the lower-utility tasks at the backup data centre to release resources for the higher-utility tasks to be rescued from the faulty data centre. By adopting this operation under limited resource cases, some lower-utility tasks might be sacrificed. This leads to a decrease in cloud resiliency when the resource becomes insufficient. For example, as shown in Figure 5.1, the proposed UBFT strategy has better TRR when the number of tasks is equal to or less than 340. However, the proposed UBFT strategy encounters a TRR decrease when the number of tasks is more

than 340.

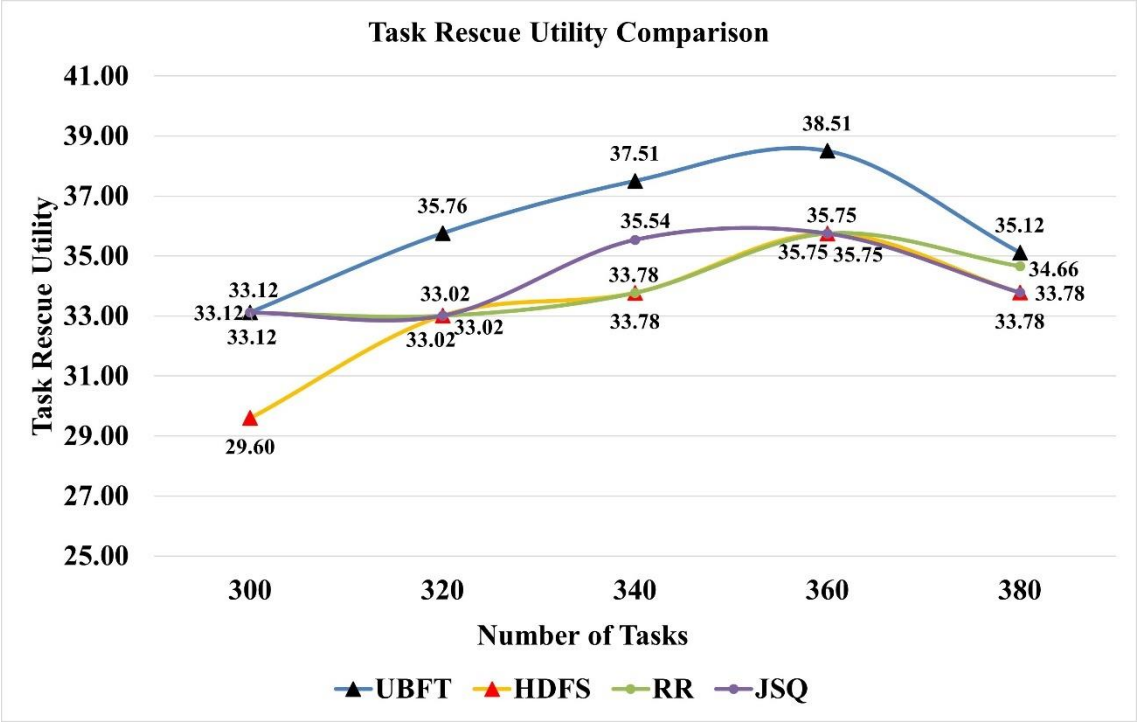

Figure 5.2 The task rescue utility comparison of Simulation 1

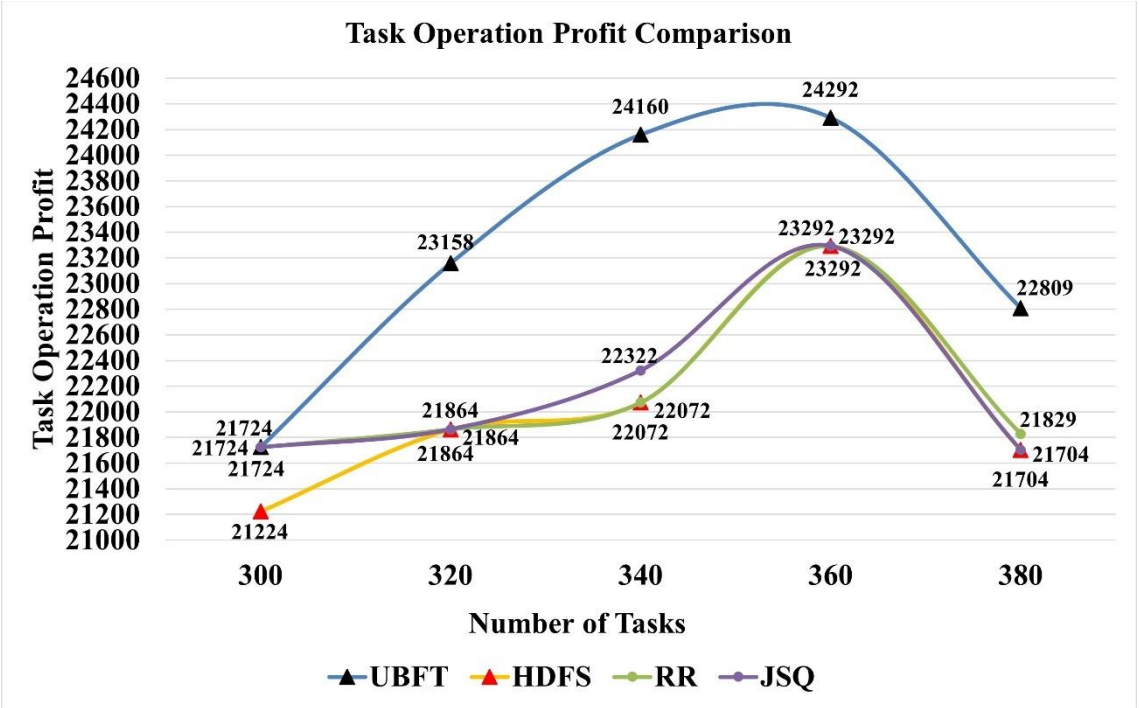

Figure 5.3 The task operation profit comparison of Simulation 1

In Figure 5.2 and Figure 5.3, it is evident that the proposed UBFT strategy is better than the other three comparative strategies in terms of task rescue utility and task operation

profit. When the number of tasks is equal to or less than 340 tasks, the proposed UBFT strategy achieves higher task rescue utility by a maximum of 11.89% increase and higher task operation profit by a maximum of 9.46% increase than the other three comparative strategies. When the number of tasks is more than 340, the proposed UBFT strategy still achieves higher task rescue utility by a maximum of 11.04% increase and higher task operation profit by a maximum of 5.09% increase than the other three comparative strategies.

5.1.4.2 Simulation 2 – Utility weights with urgency highly-weighted

In Simulation 2, the utility weight of the task urgency W_{UR} is increased to 0.67 and decrease the utility weight of the task operation profit W_{PRO} to 0.33 for evaluating the urgency highly-weighted scenario between the task urgency and the task operation profit. The simulation 2 results of the task resilience ratio, the task rescue utility, and the task operation profit are shown in Figure 5.4, Figure 5.5, and Figure 5.6, respectively.

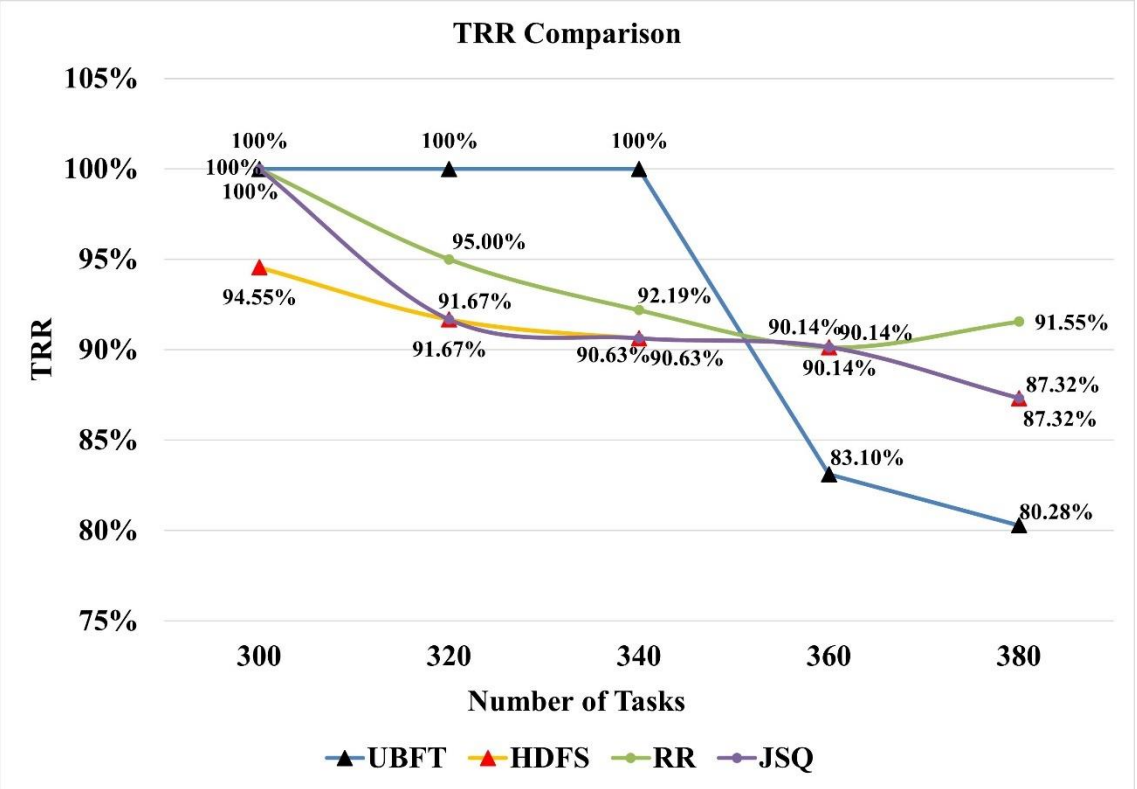

Figure 5.4 The TRR comparison of Simulation 2

In Figure 5.4, the proposed UBFT strategy remains higher TRR in comparison with the other three comparative strategies when the number of tasks is equal to or less than 340. However, the proposed UBFT strategy still encounters a mild degree of TRR decrease due to the same reason in Simulation 1 when the number of tasks is more than 340.

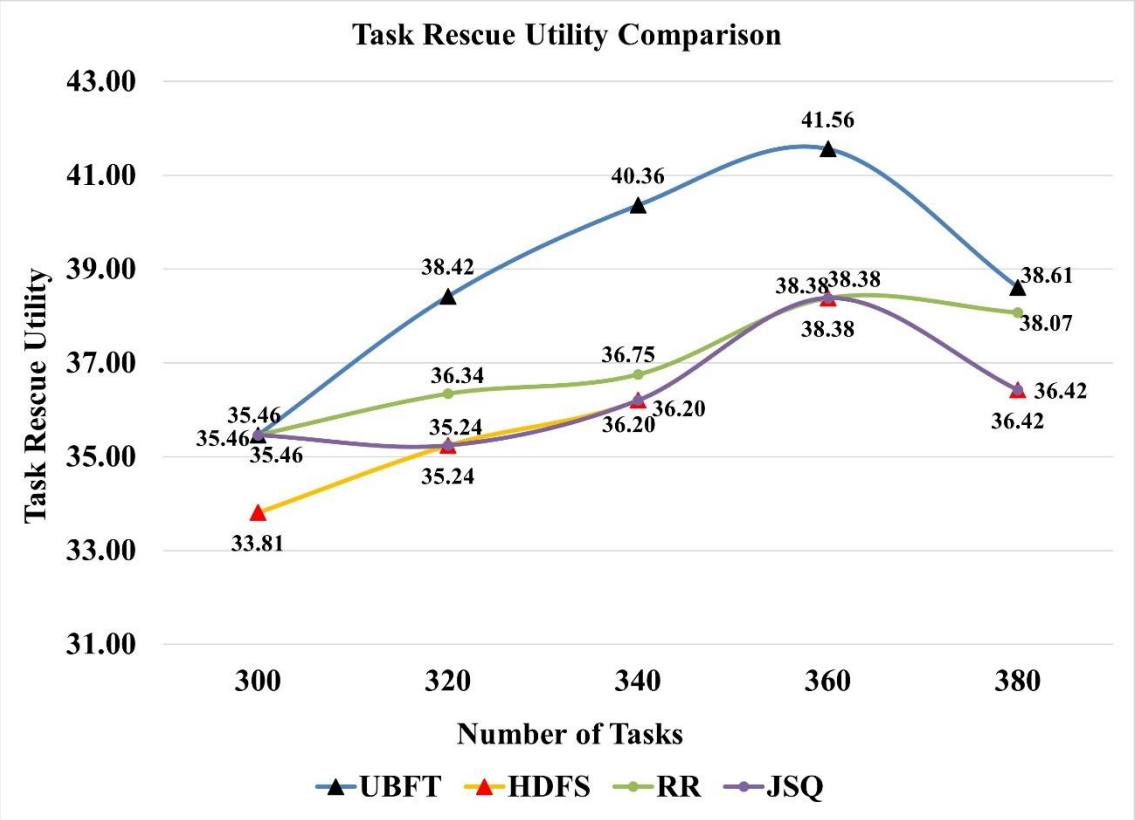

Figure 5.5 The task rescue utility comparison of Simulation 2

In Figure 5.5 and Figure 5.6, the proposed UBFT strategy remains the same trend as in Simulation 1. When the number of tasks is equal to or less than 340 tasks, the proposed UBFT strategy achieves higher task rescue utility by a maximum of 11.49% task rescue utility increase and a maximum of 9.46% task operation profit increase than the other three comparative strategies. When the number of tasks is more than 340, the proposed UBFT strategy still achieves higher task rescue utility by a maximum of 8.29% increase and higher task operation profit by a maximum of 4.29% increase than the other three comparative strategies.

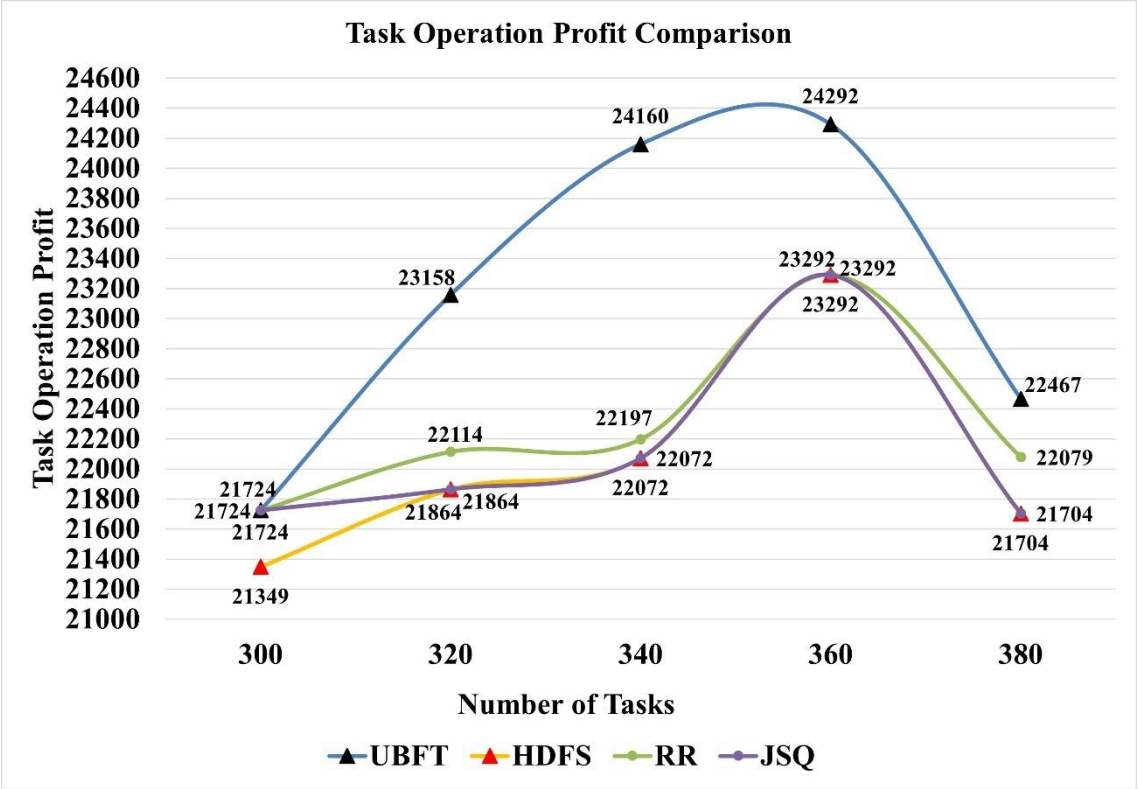

Figure 5.6 The task operation profit comparison of Simulation 2

5.1.4.3 Simulation 3 – Utility weights with profit highly-weighted

In Simulation 3, the weight of the task urgency W_{UR} is decreased to 0.33 and increase the weight of the task operation profit W_{PRO} to 0.67 for evaluating the profit highly-weighted scenario between the task urgency and the task operation profit. The Simulation 3 results are shown in Figure 5.7, Figure 5.8, and Figure 5.9.

In Figure 5.7, the proposed UBFT strategy still keeps higher TRR in comparison with the other three comparative strategies when the number of tasks is equal to or less than 340. However, the proposed UBFT strategy again experiences a TRR decrease due to the same reason in Simulation 1 when the number of tasks is more than 340.

In Figure 5.8 and Figure 5.9, the proposed UBFT strategy still maintains higher task rescue utility and higher task operation profit in comparison with the other three comparative strategies. The proposed UBFT strategy still achieves a maximum of 9.36% task rescue utility increase and a maximum of 8.84% task operation profit

increase than the other three comparative strategies.

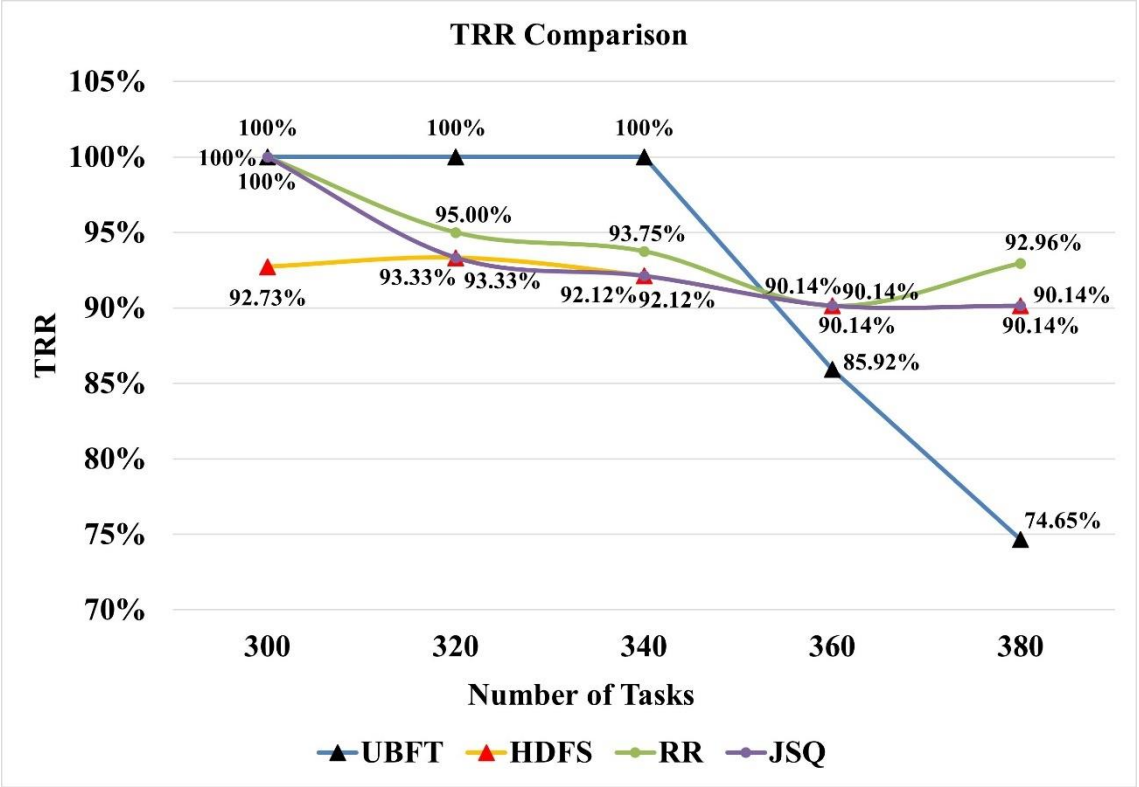

Figure 5.7 The TRR comparison of Simulation 3

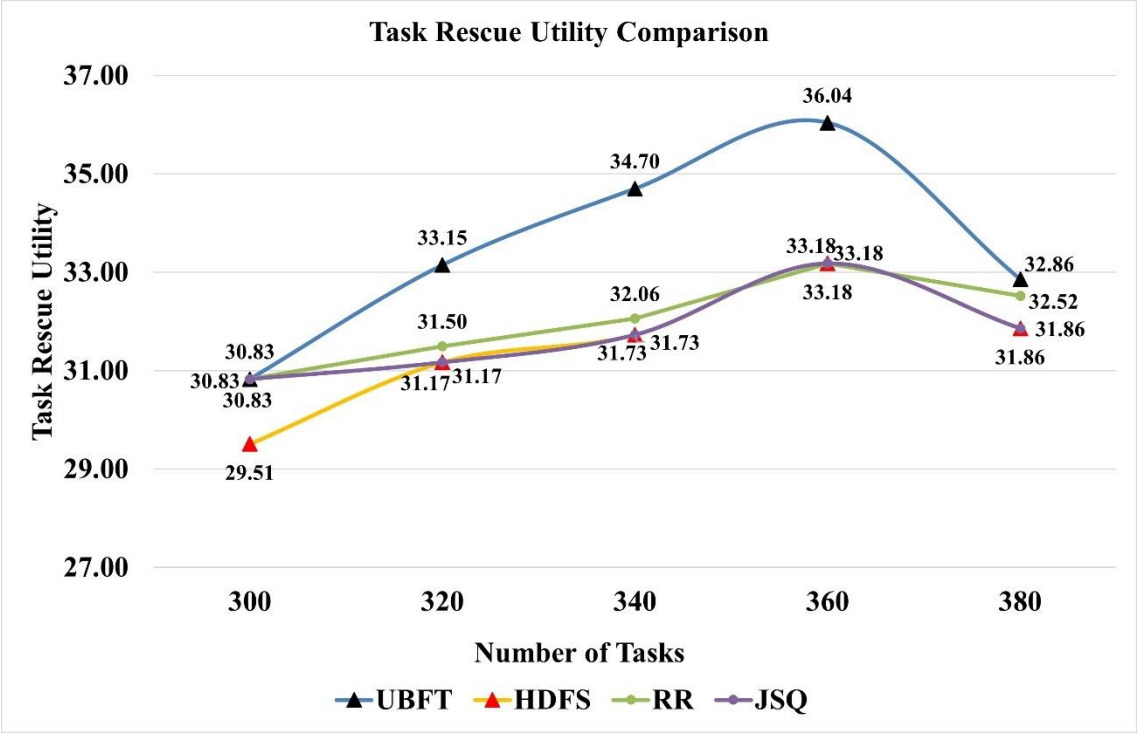

Figure 5.8 The task rescue utility comparison of Simulation 3

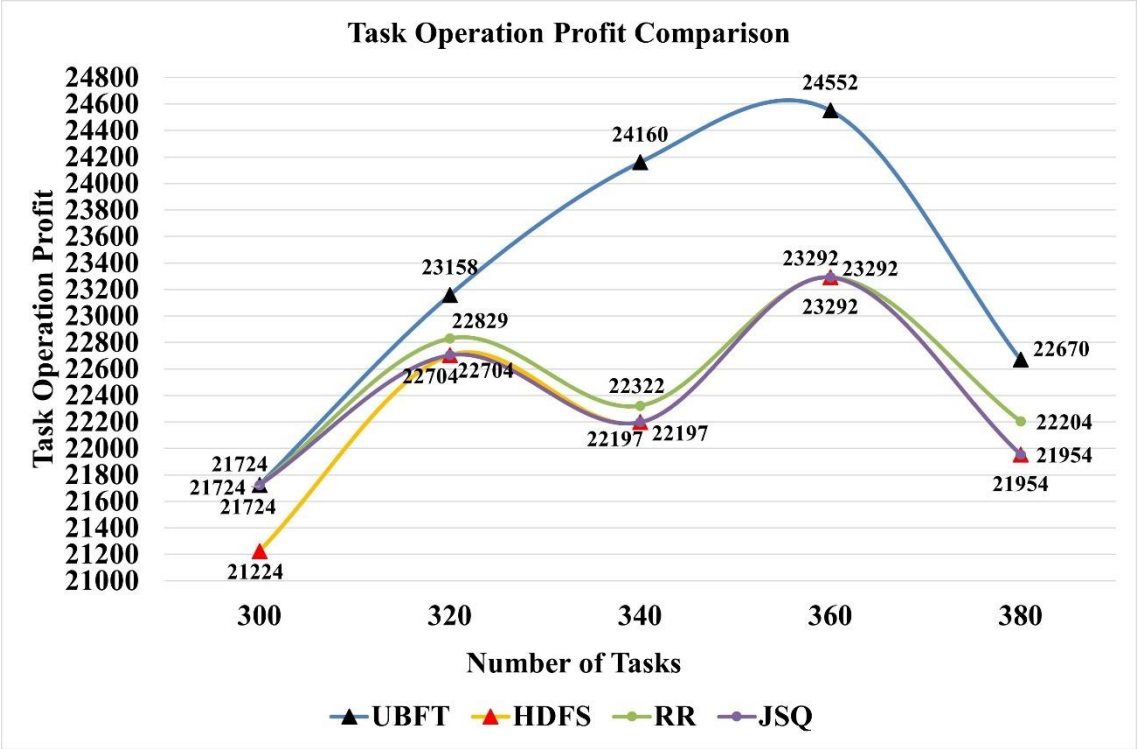

Figure 5.9 The task operation profit comparison of Simulation 3

5.2 Timeline-Oriented Fault Tolerance for Independent Tasks

The fault tolerance strategy is a significant way to enable the capability of a cloud data centre to keep performing its current-running and intended tasks in the presence of faults as the last strategy did in Section 5.1. The task resubmission and the task migration are two of the reactive fault tolerance techniques which are applied in Section 5.1. The core method of these two reactive fault tolerance techniques is the task scheduling method. Particularly, the HEFT series strategies are one of the most significant series of task scheduling strategies published from 2002 to date. Although the HEFT series strategies were proposed over the past decade, selecting the first available server might not be the optimal solution when handling faults [11][94][96]. It may cause unnecessary deadline contention and resource contention between the task with high priority and the task with low priority. As a result, the cloud resiliency might not be optimal with many low-priority tasks unsaved. Apart from that, selecting the first

available server may cause a temporary and dramatic resource load increase at some specific time points on the timeline, which leads to the performance bottleneck in the cloud data centres.

Therefore, a timeline-oriented reactive fault tolerance (TOFT) strategy for independent task rescue is proposed to achieve better cloud resiliency and load balancing performance. The proposed TOFT strategy further considers the timeline scenarios at each cloud data centre upon the proposed strategy in Section 5.1. The following questions are addressed in this TOFT fault tolerance strategy.

- How to handle the task rescue priority to ensure a better cloud resiliency?
- How to select the optimal eligible time slot for the rescued tasks to avoid the resource wastefulness and improve the load balancing performance?
- How to further improve the cloud resiliency when some tasks cannot be rescued directly?

A two-dimensional task parsing system is deployed to identify the eligible time slots for the independent tasks in the cloud environment. Then a three-dimensional priority assignment system is developed to prioritise the independent tasks in the cloud environment. To handle different cases, two sub-algorithms are applied in the proposed dynamic TOFT task rescheduling algorithm. The simulation results show that the proposed TOFT strategy has better cloud resiliency and load balancing performance than the HEFT series strategies. Besides, the proposed strategy can also fit both the single-fault scenario and the multi-fault scenario when handling faults.

5.2.1 System modelling

In general, each task $j \in J$ is associated with $R(j), DEAD(j)$ and $PRO(j)$, which present the resource requirement, the task deadline and the task operation profit of the task j, respectively. Each task j has a fixed task execution duration $Len(j)$. The task urgency

value of the task j, $UR(j)$, refers to the time buffer between current time point and its deadline $DEAD(j)$. Same to Section 5.1, the task deadline is defined as a specific point in time without the consideration of the task with an infinite deadline. Only the task with a definite deadline need to be rescued, as tasks with infinite deadline can be rescheduled when the faulty data centre is fully recovered from the fault. In this research, a task that is completed beyond its deadline is meaningless. All tasks in this research are independent, which means there are no task dependencies among the tasks. Additionally, like in other common strategies [20], this strategy assumes that all tasks are required to be restarted.

In this research, cloud resiliency is one of the optimisation objectives. The cloud resiliency for a faulty data centre can be calculated as same as the method used in Section 5.1 in terms of TRR.

5.2.2 Task parsing system

A two-dimensional task parsing system is developed to identify the eligible time slots for the tasks in the cloud environment. Firstly, a timeline exists at each data centre. The timeline range cannot be infinite because the tasks with the infinite deadline are not considered in this strategy. Therefore, the timeline range refers to $[T_0, T_{Late}]$, where T_0 denotes the current time point and T_{Late} denotes the latest deadline time point of the tasks in J.

The time slot is defined as a series of continuous time points. The available resource at each time point is the most significant factor for the further reception of the rescheduled tasks from the faulty data centre. Therefore, the timeline is parsed at each data centre site in a two-dimensional vector space. The x axis is the discrete time points ranged from $[T_0, T_{Late}]$ and the y axis is the available resource. Thus, the line in this space represents the available resource over time. In this strategy, it is called a resource line.

Each task can be parsed into this two-dimensional vector space as a rectangle. The height of the rectangle represents the resource requirement of the task and the length of the rectangle corresponds to the task execution duration. The rectangle will horizontally move from T_0 to T_{Late}. An eligible time slot for a task starts from a time point when the rectangle starts to stand completely below the resource line and ends at a time point when the rectangle starts to stand above the resource line. A function $Count(ET(j^{dc}))$ is deployed to count the number of eligible time slots of the task j at the data centre dc.

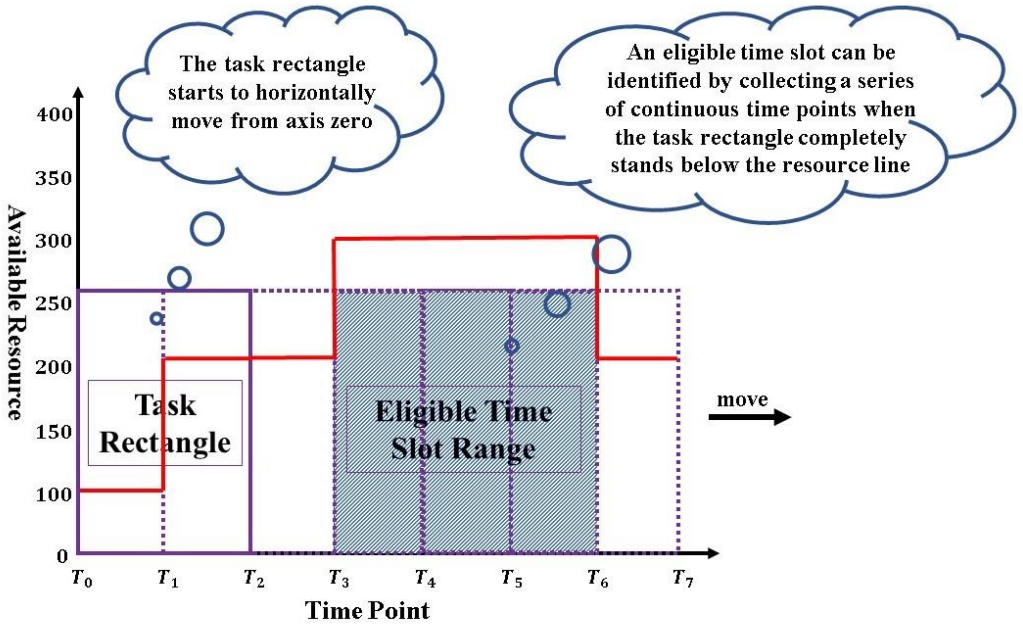

Figure 5.10 The example of the eligible time slot identification

An example of the proposed task parsing system is shown in Figure 5.10, if T_{Late} is T_7 and the rest available resource values at each time point from T_0 to T_7 are {100, 150, 200, 350, 350, 350, 150, 150} resource units and if the task needs 250 resource units, then the eligible time slot identification processes will be done as shown in Figure 5.10. The available resource from T_0 to T_2 cannot meet the resource requirement of this task rectangle. When the task rectangle reaches T_3 and T_4, the task rectangle completely stands under the resource line (red line in Figure 5.10). Therefore, the range from T_3 to T_5 and T_4 to T_6 will be recognized first as eligible time slot because the task rectangle

width is 2 time unit. As T_3 to T_6 are continuous time points, thus a final range of this eligible time slot can be identified between T_3 and T_6, as dash area shown.

5.2.3 Timeline-oriented fault tolerance (TOFT) strategy

The proposed timeline-oriented fault tolerance (TOFT) strategy has three phases, task prioritising phase, replica selection phase, and eligible time slot selection phase. It is an independent task rescheduling strategy for a bounded number of data centres when faults occur. In the case of a single-fault scenario, the proposed TOFT strategy can be applied by the faulty data centre in one time to rescue the tasks at the faulty data centre. While in the case of multi-fault scenarios, the proposed TOFT strategy should be separately applied in each faulty data centre.

- Task prioritising phase

This phase distributes the task rescue priority to the task at the faulty data centre. The task rescue priority list will preserve an ascending processing order based on the task rescue priority until no more tasks can be allocated. Tie-breaking is done randomly. In Section 5.1, a utility-based task prioritising method was proposed to prioritise the tasks by only considering the utility difference among tasks. Differently, in this TOFT strategy, a three-dimensional evaluation method is proposed to evaluate two significant task attributes used in Section 5.1 and the number of eligible time slots together for a comprehensive evaluation of the task priority from different domains.

A three-dimensional priority assignment system is developed to assign the task rescue priority by jointly taking the task urgency, the task operation profit, and the number of eligible time slots of the task into account. The task j can be parsed into a cuboid in a three-dimensional vector space as shown in Figure 5.11, where the cuboid length a denotes the task urgency value $UR(j)$ on the y axis, the cuboid height c represents the reciprocal of the task operation profit, $\frac{1}{PRO(j)}$, on the z axis and the cuboid width b

denotes the number of eligible time slots of the task j on the x axis.

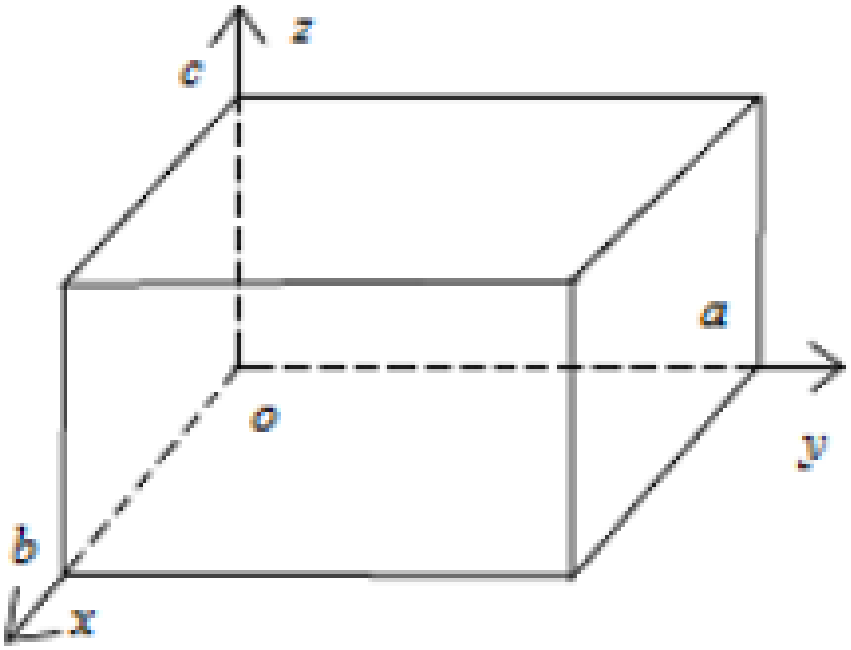

Figure 5.11 Task prioritising cuboid

According to the parsing method above, the volumes among cuboids will be compared. The smaller volume the cuboid has, the more urgent, the more profitable, and the more processing difficulty the task has. Hence, the cuboid with the smaller volume has a higher priority. The task allocation priority list is created based on the volume value of each task cuboid.

- Replica selection phase

The proposed TOFT strategy has a performance-oriented replica selection policy that adopts the replica selection method proposed in Section 4.3 to select the optimal replica to access. Tie-breaking is done randomly.

- Eligible time slot selection phase

The proposed eligible time slot selection method aims to select the optimal eligible time slot for the received tasks on the timeline at each working-proper data centre. The scenario-based allocation is applied for the normal cases (Algorithm 5.3) and the

limited resource or the insufficient time slot length cases (Algorithm 5.4), respectively. Tie-breaking is done randomly.

Both the time slot length similarity and the corresponding time slot resource situations are considered in this research to accommodate the task at its optimal eligible time slot. The eligible time slot with the more similar time slot length similarity to the task execution duration is more suitable to accommodate the task with less wastefulness in the time slot space. The higher the minimum available resource in the eligible time slot achieves the less possibility of the load spike problem.

Let $ET(j^{dc})$ denotes a set of eligible time slots for the task j at the data centre dc. Then $et(j^{dc})_p \in ET(j^{dc})$ is the pth eligible time slot in $ET(j^{dc})$. Let $LS(et(j^{dc})_p)$ denotes the time slot length similarity of $et(j^{dc})_p$, where $LS(et(j^{dc})_p)$ equals to $Len(et(j^{dc})_p)$ - $Len(j^{dc})$. Let $MR(et(j^{dc})_p)$ denotes the minimum available resource value of $et(j^{dc})_p$. Then the *Min-Max* normalisation method is applied in the timeline allocation method to identify the optimal eligible time slot by obtaining the maximum ranking value from Eq. 5.8, where W_{LS} and W_{MR} denote the weight of the time slot length similarity and the minimum available resource, respectively. The sum of W_{LS} and W_{MR} is 1. The optimal eligible time slot of a task j will be marked as $OE(j)$.

$$\begin{cases} rank\left(LS\left(et(j^{dc})_p\right)\right) = \frac{max\left(LS\left(ET(j^{dc})\right)\right) - LS\left(et(j^{dc})_p\right)}{max\left(LS\left(ET(j^{dc})\right)\right) - min\left(LS\left(ET(j^{dc})\right)\right)} \\ rank\left(MR\left(et(j^{dc})_p\right)\right) = \frac{MR\left(et(j^{dc})_p\right) - min\left(MR\left(ET(j^{dc})\right)\right)}{max\left(MR\left(ET(j^{dc})\right)\right) - min\left(MR\left(ET(j^{dc})\right)\right)} \\ rank\left(et(j^{dc})_p\right) = W_{LS} * rank\left(LS\left(et(j^{dc})_p\right)\right) + W_{MR} * rank\left(MR\left(et(j^{dc})_p\right)\right) \end{cases} \quad (5.8)$$

To implement the three phases discussed above, a dynamic task rescheduling algorithm is proposed in Algorithm 5.2. Algorithm 5.2 firstly sets the timeline at each data centre in the cloud environment at **Line 2** and initialises the task parsing vector space and the priority assignment system from **Line 3** to **Line 4**. Then the tasks at the faulty data

centre are collected into the faulty task list $fj[]$ at **Line 5**. The tasks in$fj[]$ will be prioritised at **Line 6**. The faulty task list $fj[]$ will be sorted by the QuickSort algorithm based on the task priority of each task in $fj[]$ at **Line 7**. Then the task rescheduling solution will be worked out for each task in the faulty task list $fj[]$ from **Line 8** to **Line 23**. The optimal replica-ready data centre is selected by following the proposed replica selection strategy in Section 4.3 at **Line 9**. Then the task will be scheduled to the optimal replica-ready data centre at **Line 10**. The task will be parsed to identify the eligible time slots on the timeline of the optimal replica-ready data centre at **Line 11**. The number of the eligible time slots will be counted at **Line 12**.

Two different scenarios are treated in Algorithm 5.2. Algorithm 5.3 is initiated at **Line 14** to handle the case under normal circumstances if the number of eligible time slots is not equal to 0 at **Line 13**. Then the fault handling solution will be implemented at **Line 15** for the normal circumstances. Algorithm 5.4 initiated at **Line 17** to handle the limited resource or the insufficient time slot length cases if the number of eligible time slots is equal to 0 at **Line 16**. Then the fault handling solution will be implemented from **Line 18** to **Line 21** for the limited resource or the insufficient time slot length cases. The time complexity of Algorithm 5.2 is O(n).

Algorithm 5.2: Dynamic Task Rescheduling Algorithm

Input: DC, J, fault location

Output: Task rescheduling solution

1. Initialization {
2. Set timeline
3. Create the two dimensional task parsing vector space
4. Create the three dimensional priority assignment system }
5. Load the tasks at the fault location and add into $fj[]$
6. Prioritise the tasks in $fj[]$
7. QuickSort $fj[]$ based on the task priority

8. **for** $fj[v]$ in $fj[]$, $v = 0$, $v \leq Sizeof(fj[])$ - 1 **do**
9. Select the optimal replica-ready data centre in DC and add into $rr[]$
10. Schedule $fj[v]$ to $rr[]$
11. Parse $fj[v]$ and the timeline at $rr[]$
12. $Count(ET(fj[v])^{rr[]})$
13. **if** $Count(ET(fj[v])^{rr[]}) > 0$
14. Do Algorithm 5.3
15. Move $fj[v] \rightarrow OE(fj[v])$
16. **else**
17. Do Algorithm 5.4
18. Load "Optimal Migratable Task" and "Migration Destination"
19. Migrate "Optimal Migratable Task" $\rightarrow T_{Begin}$ of "Migration Destination"
20. Record the original location of "Optimal Migratable Task" as $OE(fj[v])$
21. Move $fj[v] \rightarrow OE(fj[v])$
22. **end if**
23. **end for**

Algorithm 5.3: Optimal Eligible Time Slot Selection

Input: $fj[v]$, $rr[]$

Output: Optimal eligible time slot $OE(fj[v]])$

1. Insert the task $fj[v]$ from Line 14 in Algorithm 5.2
2. Calculate $rank(ET(fj[v])^{rr[]})$ under Eq. 5.8
3. Generate $OE(fj[v)$
4. Load the beginning time point T_{Begin} of $OE(fj[v])$
5. Allocate $fj[v]$ at T_{Begin}
6. Update resource line for $OE(fj[v])$
7. $v++$ at Line 8 in Algorithm 5.2

Algorithm 5.3 is used to generate the optimal eligible time slot for the input task from Algorithm 5.2. In Algorithm 5.3, the task $fj[v]$ is inserted from Line 14 in Algorithm 5.2 at **Line 1**. Then the optimal eligible time slot is identified for the inserted task $fj[v]$ from **Line 2** to **Line 7**. The ranking value for the eligible time slots will be calculated for the inserted task under Eq. 5.8 at **Line 2**. After that, the optimal eligible time slot for

the inserted task will be generated at **Line 3**. The optimal eligible time slot will be loaded to find its beginning time point T_{Begin} at **Line 4**. The inserted task should be allocated to the beginning time point of the optimal eligible time slot at **Line 5**. The resource consumption of the inserted task should be updated to the resource line in the task parsing vector space at **Line 6**. Finally, the next task in $fj[]$ will be addressed by increasing one order number at **Line 7**. In this strategy, the rescheduled task is commonly allocated at the first time point (the beginning time point) in the optimal eligible time slot because the "as early as possible" principle is insisted for all task completeness. For tie-breaking eligible time slots, the task is placed at the earliest available time slot as well. The time complexity of Algorithm 5.3 is O(1).

By applying Algorithm 5.3, the proposed TOFT strategy can rescue the tasks that already have eligible time slots. The tasks which are left unsaved are known as residual tasks because of the rescue failures due to the insufficient resource or the insufficient number of eligible time slots case. A residual task allocation is developed in Algorithm 5.4 by using the one-stop task migration technique to make a concession mechanism for better cloud resiliency.

In Algorithm 5.4, the current rest time slots are identified in the optimal data centre in $rr[]$ at **Line 1**, which meets the resource requirement of the input task $fj[v]$. Those identified time slots will be added to the probable eligible time slot list $pts[]$. The capacity of $pts[]$ will be checked at **Line 2**. If $pts[]$ is empty, the optimal data centre will be re-selected from **Line 3** to **Line 5**. Otherwise, the current-running tasks in $rr[]$ will be collected and added into the current-running task list $crj[]$ at **Line 7**. Each task in $crj[]$ will be processed to find the probable-release tasks by comparing between the resource requirement of the input task $fj[v]$ and the task in $crj[]$ at **Line 9**. Then the probable-release tasks will be identified and collected into the probable-release task list

$prj[]$ at **Line 11**. After that, the capacity of $prj[]$ will be checked at **Line 16**. If $prj[]$ is empty, the input task rescue will be failed at Line 17 and the order number at Line 8 in Algorithm 5.2 will be increased one. Otherwise, the element in $prj[]$ which are discrete to $pts[]$ will be removed at **Line 19**.

After the removal operations at **Line 19**, the input task rescue will be failed at **Line 21** if $prj[]$ is empty and the order number at Line 8 in Algorithm 5.2 will be increased one. Otherwise, the probable release tasks in $prj[]$ will be processed at **Line 26**. The algorithm will try to release the probable-release task in $prj[]$ at **Line 27**.

Then the after-release task completeness to this probable-release task in $prj[]$ will be confirmed at **Line 28**. If the probable-release task in $prj[]$ can be completed in time after the release, the after-release time slot length will be evaluated at **Line 29** for further testing its feasibility to accommodate the input task $fj[v]$ at **Line 30**. If the after-release time slot length is feasible to accommodate the input task $fj[v]$, the probable-release task in $prj[]$ will be added into the migratable task list $mig[]$ at **Line 32**. Otherwise, the order number of $prj[]$ will be increased one at **Line 34**. If the probable-release task in $prj[]$ cannot be completed in time after the release, the order number in $prj[]$ will be increased one at **Line 37**.

The migratable task list $mig[]$ will be sorted by the QuickSort algorithm based on the task execution duration at **Line 40**. Then the migratable task list $mov[]$ will start to be processed from **Line 41** to **Line 54**. An alternative eligible time slot will be identified for the current processing task in $mov[]$ at **Line 42**. If an alternative eligible time slot can be founded, then the current-processing task in$mov[]$ will be labelled as “Optimal Migratable Task” and this alternative eligible time slot will be labelled as “Migration Destination” at **Line 44** and **Line 45**, respectively. Then the “Optimal Migratable Task” and the “Migration Destination” will be returned to the output at **Line 46** and the order

number at Line 8 in Algorithm 5.2 will increase one. An alternative eligible time slot cannot be identified for the current-processing task in $mov[]$, the order number in $mov[]$ will increase one to test the next element in $mov[]$ at **Line 49**. If the bottom of $mov[]$ is reached, the input task rescue will be failed at **Line 51** and the order number at Line 8 in Algorithm 5.2 will increase one. The time complexity of Algorithm 5.4 is O(n).

Algorithm 5.4: Residual Task Allocation

Input: $fj[v]$, $rr[]$

Output: "Optimal Migratable Task" and "Migration Destination"

1. Identify the current rest time slots in $rr[]$ which meets $R(fj[v])$ and add into $pts[]$
2. **if** $pts[] = NULL$
3. Remove $rr[]$ from DC
4. Empty $rr[]$
5. Back to Line 9 in Algorithm 5.2
6. **else**
7. Collect the current-running tasks in $rr[]$ and add into $crj[]$
8. **for** each $crj[w]$ in $crj[]$, $w = 0$, $w \leq Sizeof(crj[]) - 1$ **do**
9. Compare $R(fj[v])$ with $R(crj[w])$
10. **if** $R(crj[w]) > R(fj[v])$
11. Add $crj[w]$ into $prj[]$
12. **else**
13. $w++$
14. **end if**
15. **end for**
16. **if** $prj[] = NULL$
17. $v++$ at Line 8 in Algorithm 5.2
18. **else**
19. Remove the element in $prj[]$ which are discrete to $pts[]$
20. **if** $prj[] = NULL$
21. $v++$ at Line 8 in Algorithm 5.2
22. **else**

23. Go to Line 26
24. **end if**
25. **end if**
26. **for** $prj[c]$ in $prj[]$, $c = 0$, $c \leq Sizeof(prj[]) - 1$ **do**
27. Try to release $prj[c]$
28. Evaluate after-release task completeness to $prj[c]$
29. **if** Line 28 = $True$
29. Evaluate the after-release time slot length
30. Test the after-release time slot length to accommodate $fj[v]$
31. **if** Line 30 = $TRUE$
32. Add $prj[c]$ into $mov[]$
33. **else**
34. $c++$
35. **end if**
36. **else**
37. $c++$
38. **end if**
39. **end for**
40. QuickSort $mov[]$ based on the task execution duration
41. **for** $mov[s]$ in $mov[]$, $s = 0$, $s \leq Sizeof(mov[]) - 1$ **do**
42. Find an alternative eligible time slot to $mov[s]$
43. **if** Line 42 = $TRUE$
44. Label $mov[s]$ as "Optimal Migratable Task"
45. Label the alternative eligible time slot as "Migration Destination"
46. Return "Optimal Migratable Task" and "Migration Destination"
47. $v++$ at Line 8 in Algorithm 5.2
48. **else**
49. $s++$
50. **if** the bottom of $mov[]$ is reached
51. $v++$ at Line 8 in Algorithm 5.2
52. **end if**
53. **end if**
54. **end for**

55. **end if**

56. End Algorithm 5.4

5.2.4 Simulation results

To evaluate the performance of the proposed strategy, three simulations are performed on OMNeT++ 5.4.1. The following assumptions are made in the simulations:

- Traditional three-replicas strategy is deployed. The replica placement policy is to put one replica on the local data centre, another two replicas are placed into two different remote data centres.
- The latency among data centres is insignificant.
- All data centres are interconnected. The data can be freely exchanged among data centres.
- Bandwidth is set as the consumed resource.
- W_{LS} and W_{MR} are set to 0.5 to simplify the problem.

Three types of real-world workflows are implemented in the simulations, such as Montage scientific workflow, LIGO Inspiral Analysis workflow, and SIPHT program. Each scientific workflow instance is compressed into a task package as an independent task. The details of these workflows are adjusted and referenced from [10].

The cloud resiliency is measured in all three simulations and the resource load situation is tested specifically in Simulation 2. The performance of the proposed TOFT strategy is compared to the average performance of the HEFT series strategies.

5.2.4.1 Simulation 1 – Multiple types of tasks with different deadlines

A cloud environment of 4 data centres with 6 circuits of 100 Gbps optical-fibre network integrated at each data centre site is set up in Simulation 1. The maximum bandwidth at each data centre is 600 Gbps. The disaster occurs at T_0 in the data centre dc_1.

In Simulation 1, the task input rule is set as follows. 200 tasks are input per input round.

- A random number of two types of tasks out of a total of 200 tasks is input per

input round when the resource is sufficient.

- Only feasible input combinations can be input per input round to the cloud environment when the resource is insufficient.

The result of the cloud resiliency in Simulation 1 is shown in Figure 5.12. It is evident that the proposed TOFT strategy has better cloud resiliency than the HEFT series strategies. As the number of tasks increases from 400 to 1400, the proposed TOFT strategy continues to rescue 100% of the faulty tasks. The HEFT series strategies fail to rescue 100% faulty tasks when the number of tasks exceeds 600. The cloud resiliency of the proposed TOFT strategy drops to 74.67% at 1600 tasks due to resource limitations and insufficient eligible time slots. However, the proposed TOFT strategy still remains greater cloud resiliency than the HEFT series strategies at 1600 tasks.

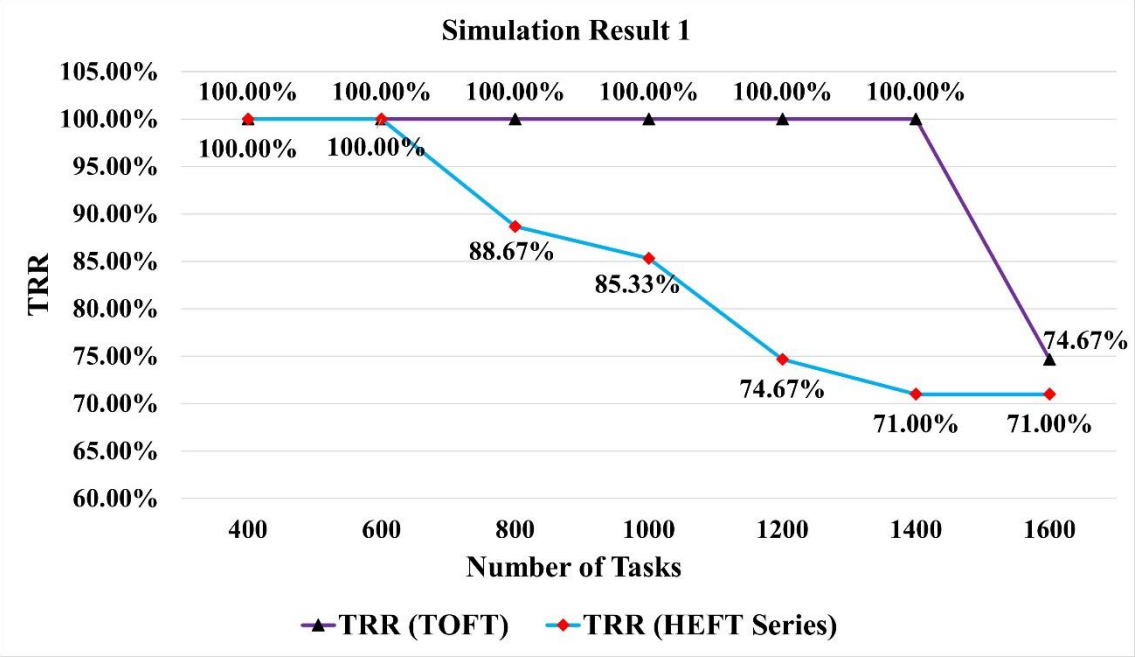

Figure 5.12 The TRR result of Simulation 1

5.2.4.2 Simulation 2 – Expanded cloud scale and load testing

In Simulation 2, not only the cloud resiliency but also the load balancing performance are evaluated under an expanded cloud scale. Normally, the great load balancing performance helps the cloud service providers avoid traffic spikes and degraded

performance. A cloud environment of 4 data centres with 60 circuits of 100 Gbps optical-fibre network integrated at each data centre is developed in Simulation 2. The maximum bandwidth at each data centre is 6000 Gbps. The disaster occurs at T_0 in the data centre dc_1.

In Simulation 2, the task input rule is set as follows. 1000 tasks are input per input round.

- A random number of random types of tasks out of a total of 1000 tasks is input per input round when the resource is sufficient.
- Only feasible input combinations can be input per input round to the cloud environment when the resource is insufficient.

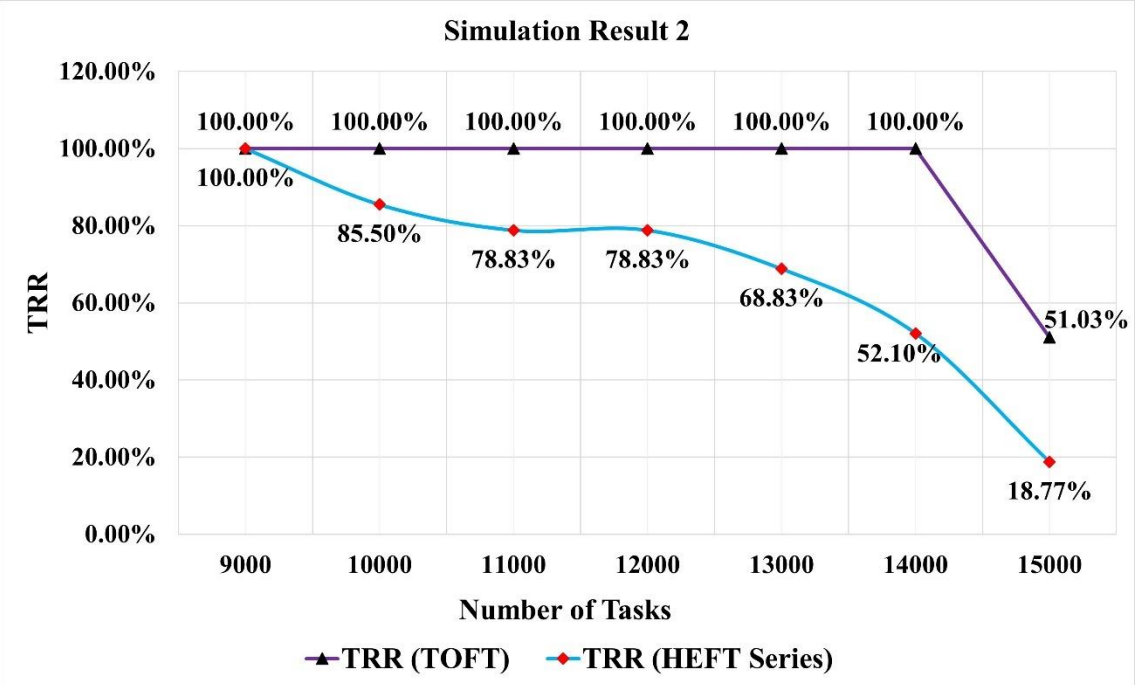

Figure 5.13 The TRR result of Simulation 2

The result of the cloud resiliency in Simulation 2 is shown in Figure 5.13. It is also evident that the proposed TOFT strategy has better cloud resiliency than the HEFT series strategies when the cloud scale expands. As the number of tasks increases from 9000 to 14000, the proposed TOFT strategy continues to keep 100% cloud resiliency.

The HEFT series strategies fail to rescue all tasks when the number of tasks exceeds 9000. The cloud resiliency of the proposed TOFT strategy drops to 51.03% at 15000 tasks because of the same reason in Simulation 1. However, the proposed TOFT strategy still keeps higher cloud resiliency than that of the HEFT series strategies at 15000 tasks.

In Simulation 2, the load situations are also tested at each time point for all proper-working data centres in the simulation environment. The resource load situations are shown in Figure 5.14, Figure 5.15, and Figure 5.16.

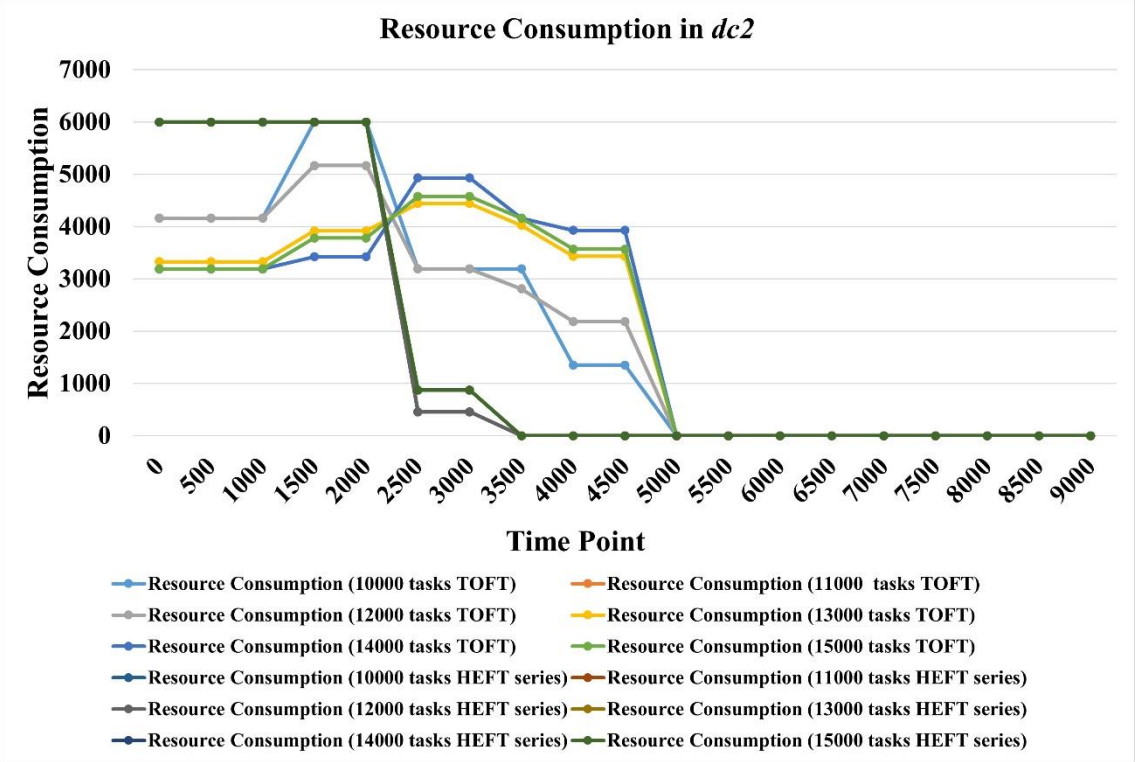

Figure 5.14 Resource load in dc_2

The HEFT series strategies remain a peak load between T_0 and T_{2500} in dc_2 and dc_4, and then have a sharp load decrease. They leave a long-time idle load after T_{2500} in dc_2 and dc_4 and make a crowd load before that time point.

However, the proposed TOFT strategy significantly reduces the load before T_{2500} in dc_2 and dc_4, and balances the load to the suitable time points at all three proper-working

data centres. Although the short-time peak load still exists, the proposed TOFT strategy is clearly better than the HEFT series strategies in terms of load balancing.

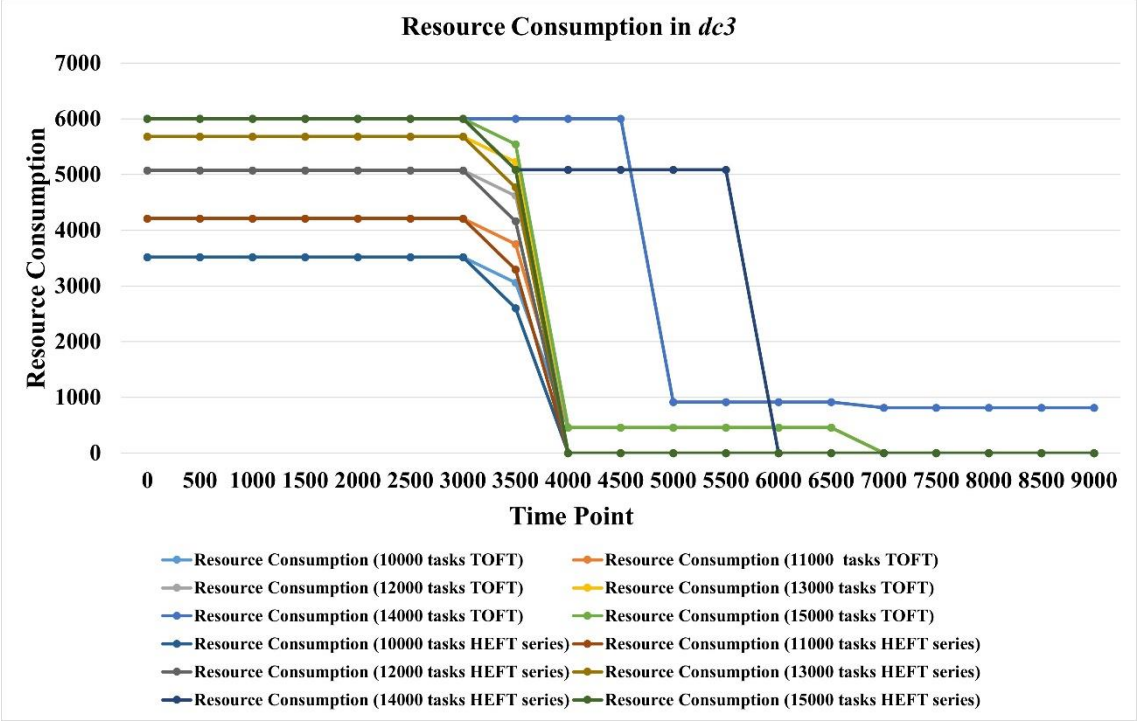

Figure 5.15 Resource load in dc_3

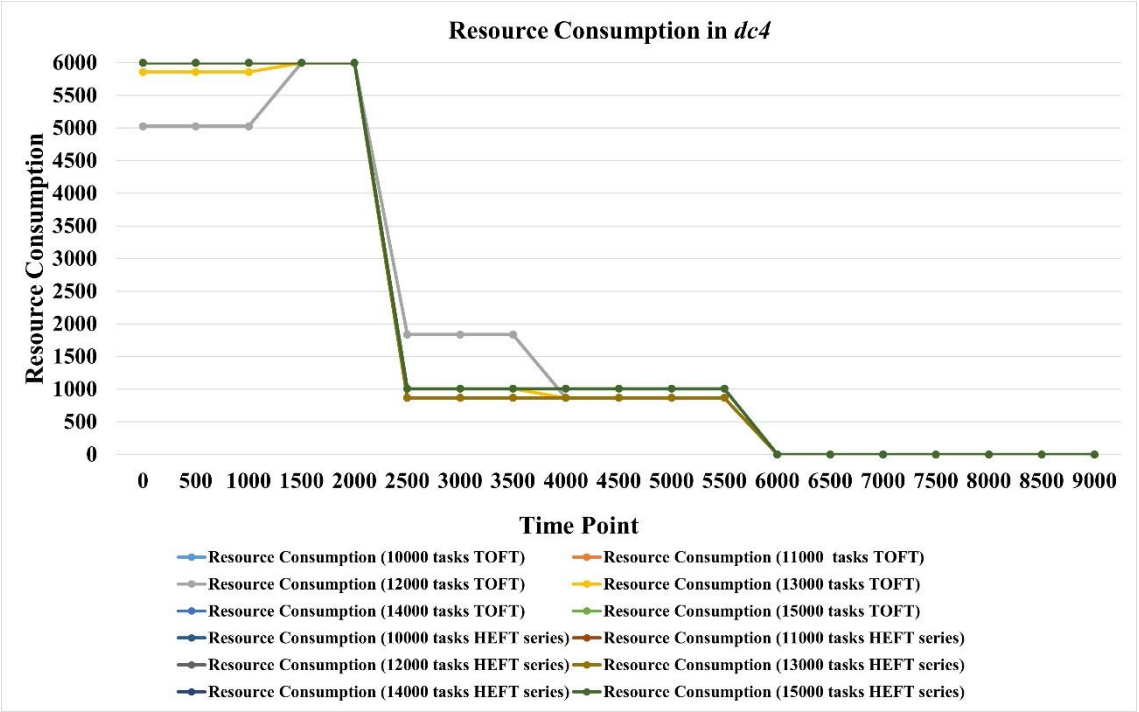

Figure 5.16 Resource load in dc_4

5.2.4.3 Simulation 3 – Multi-fault scenario

In Simulation 3, the multi-fault scenario is tested to evaluate the adaptability of the proposed TOFT strategy because the faults cannot be predicted and have high randomness. The multi-fault scenario testing is necessary and indispensable when evaluating the effectiveness and adaptability of the fault tolerance strategies [103]. In Simulation 3, multiple faults are set to generate at random time points to test the adaptability of the proposed TOFT strategy. As a result of the random fault occurrence, the first disaster occurs at T_0 in dc_1 and another fault occurs at T_{1000} in dc_4. The number of tasks is fixed at 1000 and the cloud resiliency is evaluated with the same cloud environment in Simulation 1.

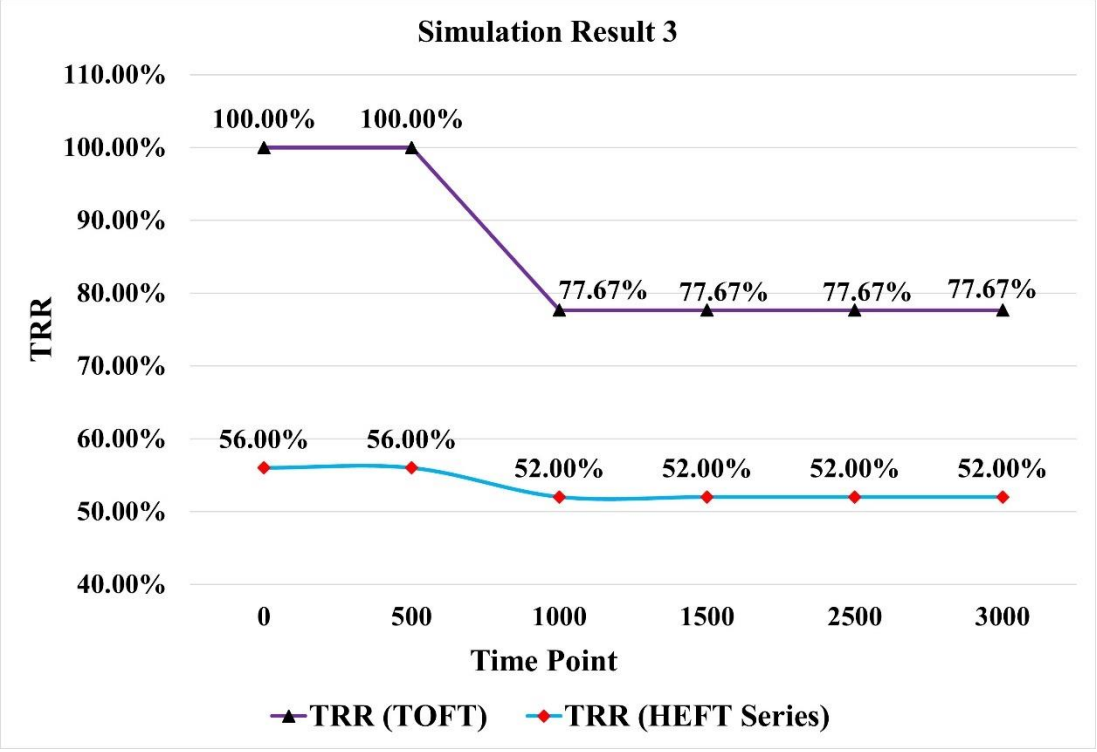

Figure 5.17 The cloud resiliency result of Simulation 3

The result of the cloud resiliency in Simulation 3 is shown in Figure 5.17. It is evident that the proposed TOFT strategy still has better cloud resiliency than the HEFT series strategies when multiple faults occur. The proposed TOFT strategy remains 100% cloud resiliency between T_0 and T_{1000} after the first fault occurs, while the HEFT series

strategies only achieve 56% cloud resiliency at that time period. As time goes on, the proposed TOFT strategy keeps continuous higher cloud resiliency than the HEFT series strategies after the second fault occurs at T_{1000}, although the proposed TOFT strategy experiences a cloud resiliency drop from 100% to 77.67%.

5.3 Summary

In Chapter 5, two reactive fault tolerance strategies are proposed for the independent tasks in the cloud environment. The task resubmission and the task migration are two reactive fault tolerance techniques applied in these two strategies. The proposed reactive fault tolerance strategy in Section 5.1 not only considers the resource load of accessing backup replicas but also strives to satisfy the deadline constraints. A utility-based task priority assignment system is developed to assign the task priority to each task by jointly considering the task operation profit and the task urgency. A utility-based fault tolerance algorithm is proposed to select appropriate data centres to accommodate the tasks rescued from the faulty data centre. The proposed utility-based fault tolerance strategy in Section 5.1 increases the cloud resiliency performance in terms of task resilience ratio, task rescue utility, and task operation profit in comparison with the typical HDFS robustness strategy, the RR strategy, and the JSQ strategy.

The proposed timeline-oriented fault tolerance strategy in Section 5.2 aims to avoid the degradation of both cloud resiliency and load balancing performance caused by selecting the first available server when doing the timeline allocation for the independent tasks rescued from the faulty data centre. A two-dimensional task parsing system is developed to identify the eligible time slots on the timeline by parsing the task into a rectangle based on its task execution duration and resource requirement. A novel three-dimensional priority assignment system is introduced to assign the task rescue priority to the tasks at the faulty data centre by evaluating the task urgency, the task

operation profit, and the number of eligible time slots of the task. A timeline allocation method is proposed to identify the optimal eligible time slot for the task rescued from the faulty data centre by considering the time slot length similarity and the corresponding time slot resource situations. A dynamic task rescheduling algorithm is developed to avoid timeline wastefulness and to improve cloud resiliency. The proposed timeline-oriented fault tolerance strategy in Section 5.2 achieves better cloud resiliency and load balancing performance in comparison with the HEFT series strategies.

Chapter 6 Reactive Fault Tolerance for Workflows

As mentioned in Chapter 2, independent tasks and dependent tasks should be treated in different ways because the fault tolerance for dependent tasks is more complex than the fault tolerance for independent tasks. In this chapter, the proposed strategy focuses on the reactive fault tolerance for the workflow applications in the replication-applied cloud environment because the workflows always have vast dependent tasks. The dependent tasks must be prioritised by inclusively considering the task dependencies among the tasks within the same workflow instance if a task cannot be initiated until all its preceding tasks are completed. The cloud performance will be largely degraded if a fault tolerance plan lacks the consideration of the task dependencies among workflow tasks. Hence, the proposed reactive fault tolerance strategies in Chapter 5 for the independent tasks in the replica-applied cloud environment are not suitable to apply to the dependent tasks because of a lack of task dependency analysis. It is important to develop a method to analyse the task dependencies when rescuing the workflow applications at the faulty data centre.

Besides, as mentioned in both Chapter 2 and also proved in Chapter 5, the idea behind the HEFT series, i.e., selecting the first available server, may not be the optimal solution when rescuing the tasks because of the resource contention problems. Particularly, the HEFT-T strategy focuses on the internal dependencies among the workflow tasks by applying an upward rank method. The external contention among different workflow instances and the entire workflow topology are hardly considered. For example, in the Montage workflow in Figure 6.1, all mProjectPP tasks in the workflow have the same priority when the HEFT-T strategy is applied. However, as the last mProjectPP task has more outbound tasks, compared with other mProjectPP tasks, it may be unreasonable to assign them the same priority value because this task will influence more tasks in the

context of the workflow topology. Therefore, the workflow topology should be further analysed to assign more accurate task priority when prioritising the workflow tasks.

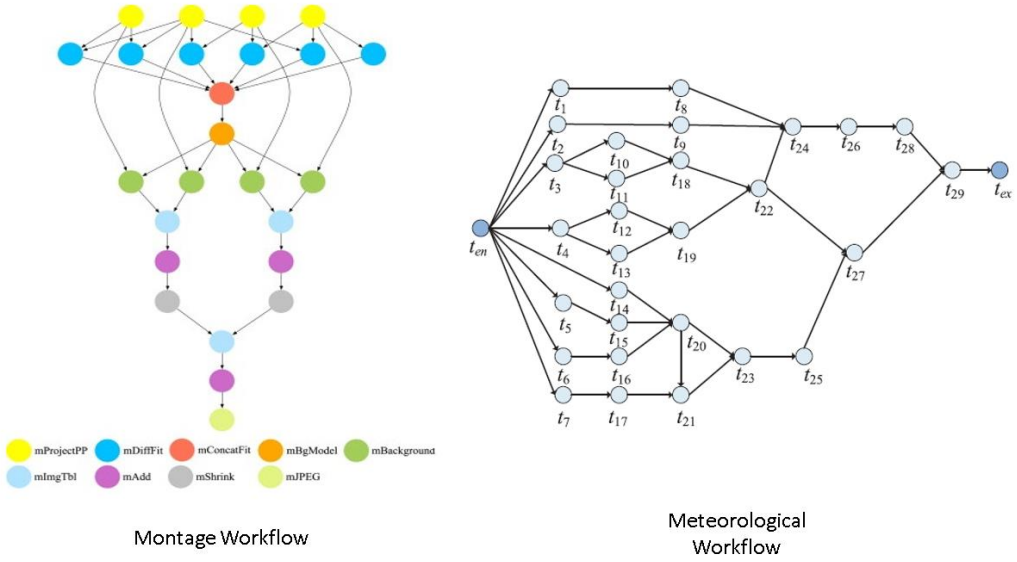

Figure 6.1 The example of the Montage workflow and the Meteorological workflow [10][130]

In rescuing workflow tasks, a common strategy is to migrate the workflow tasks to a proper-working data centre where a required replica is stored. In its implementation, it is important to take two significant parameters into consideration, the task deadline and the task execution duration [124]. As mentioned in Chapter 5, completing a task beyond its deadline is meaningless. However, in some real cases, the fault tolerance strategies often aim to complete the task with respect to its deadline requirement as much as possible. The fault tolerance strategies may not be able to rescue all tasks in each workflow instance, which means some workflow instances may still fail. For those failed workflow instances, they may still be required to be completed after the cloud data centre is fully recovered from the outage. Thus, it is assumed that the more tasks saved within an incomplete workflow instance, the better business continuity the fault tolerance strategy has. Nevertheless, the influence on business continuity is hardly

considered in most of the contemporary fault tolerance strategies. Hence, it is significant to evaluate the influence on business continuity when developing a fault tolerance strategy.

To address the above issues, a PageRank based fault tolerance (PRFT) strategy is proposed for the workflow rescue in the replica-applied cloud environment in this chapter. This strategy focuses on the workflow task rescue when handling faults based on the attributes of the task, the timeline scenario at each proper-working data centre, and the overall cloud performance. Firstly, a priority assignment system is developed based on the modified PageRank algorithm to prioritise the workflow tasks. Then a *Min-Max* normalisation method is applied for the replica selection method and the timeline allocation method. The replica selection method is based on the evaluation of the network performance at the replica-ready data centres by considering the common network performance metrics. The timeline allocation method is based on the evaluation of the time slot length similarity and the minimum available resource value in the time slot. A dynamic PageRank-constrained task scheduling algorithm is proposed to generate the task rescheduling solution for the tasks at the faulty data centre. The simulation results show that the proposed PRFT strategy can achieve better task resilience ratio, workflow resilience ratio, and workflow continuity ratio in comparison with the HEFT-T strategy, in both the traditional three-replica data replication environment and the image backup data replication environment.

6.1 System Modelling

This strategy focuses on the cloud-based workflow execution in which a number of workflow instances are deployed to cloud data centres. As mentioned in Chapter 3, the cloud environment may have x workflow instances running concurrently. A workflow instance G consists of multiple dependent tasks, which can be presented by a Directed

Acyclic Graph (DAG), $G = (N, E)$, where N denotes a set of nodes and E denotes a set of edges among nodes. $edge(Nod_p, Nod_q)$ denotes the edge between Nod_p and Nod_q. Each node $Nod_p \in N$ represents a task in the workflow and each $edge(Nod_p, Nod_q)$ represents the control dependency between Nod_p and Nod_q. For example, the DAGs of the Montage scientific workflow and the Meteorological workflow are shown in Figure 6.1. In a DAG, the node is known as an entry task Nod_{entry} if it has no predecessors, while the node is known as an exit task Nod_{exit} if it has no successors. This strategy assumes that a node cannot be initiated until all of its predecessors have been completed [81].

Each workflow instance G has an attribute associated with its deadline, $DEAD(G)$, which is a specific time point in the timeline. This deadline is known as a hard deadline as it cannot be negotiated. If the deadline is not specified for the workflow instance G, its hard deadline $DEAD(G)$ is regarded as infinite. When the faults occur at a data centre, only the workflow instance with a definite deadline need to be rescued, as the workflow instance with an infinite deadline can be rescheduled when the faulty data centre is fully recovered from the fault. Each node $Nod_p \in N$ is also associated with a fixed task execution duration $Len(Nod_p)$. To simplify the problem, this strategy assumes that a task will be re-executed if it is migrated out of its original location.

Each task $Nod_p \in N$ in the workflow application G has its own soft deadline. The soft task deadline of each task in the workflow can be calculated by reversely engineering from the exit task Nod_{exit}. For example, if $\{Nod_0, Nod_1, \ldots, Nod_{q-1}\}$ are tandem nodes in a workflow instance G, the soft deadline of Nod_{q-1} can be the time range from the time point of $T_0 + \sum_{p=0}^{q-2} Len(Nod_p)$ to the time point of $DEAD(G) - Len(Nod_{q-1})$. Then the soft deadline of Nod_{q-2} can be the time range from the time

point of $T_0 + \sum_{p=0}^{q-3} Len(Nod_p)$ to the time point of $DEAD(G) - Len(Nod_{q-1}) - Len(Nod_{q-2})$. By parity of reasoning, each soft task deadline can be calculated. In this strategy, the task parsing system is also applied, which is the same as the task parsing system proposed in Section 5.2.2.

The proposed PRFT strategy in this chapter focuses on the fault tolerance performance in terms of task resilience ratio (TRR), workflow resilience ratio (WRR), and workflow continuity ratio (WCR). The TRR refers to the ratio of the successfully rescued tasks from the faulty data centre to the total number of tasks to be rescued at the faulty data centre. The WRR refers to the ratio of the total number of rescued workflow instances out of the total number of workflow instances at the faulty data centre. The WCR refers to the number of tasks in a single workflow successfully rescued from the faulty data centre out of the total number of tasks in this workflow. The overall WCR will be evaluated by calculating the average value of the WCR in different workflow instances. The higher TRR is, the stronger task resilience performance is. The higher WRR is, the stronger workflow resilience performance is. The higher WCR is, the better potential business continuity can be achieved.

The TRR can be calculated using the same formula in Eq. 5.1 in Section 5.1. The WRR can be formulated as in Eq. 6.1. The WCR can be formulated as in Eq. 6.2.

$$\text{Workflow Resilience Ratio} = \frac{Total\ number\ of\ resuced\ workflows}{Total\ number\ of\ workflows} \tag{6.1}$$

$$\text{Workflow Continuity Ratio} = \frac{Total\ rescued\ tasks\ in\ a\ single\ workflow}{Total\ number\ of\ tasks\ in\ this\ workflow} \tag{6.2}$$

6.2 PageRank-Based Fault Tolerance (PRFT) Strategy

Originally, the PageRank algorithm is a link analysis algorithm to rank the web pages in the Google search engine results. It outputs a distribution probability to represent the likelihood that a user clicks on the links to other web pages. The PageRank value is

calculated using a mathematical algorithm based on the digraph of the web topology. The World Wide Web pages and the hyperlinks among those pages are represented as nodes and edges, respectively. Each element of a hyperlinked set of documents will be assigned a numerical weighting with the purpose of assigning its relative significance. Based on the characteristics of the PageRank algorithm, it may be applicable to any entity set with reciprocal quotations and references or any entities which can be parsed into a digraph. The web topology is similar to the workflow topology to some extent because most of the workflows, especially scientific workflows, can be parsed into a digraph. Therefore, the PageRank algorithm may be applicable to the workflow topology which can assign the numerical weighting to the nodes in the workflow digraph.

On the other hand, the PageRank algorithm lacks the relationship analysis among websites when ranking different websites. Therefore, the PageRank algorithm itself cannot be directly applied to the workflow topology. It should be modified to fit the workflow topology analysis. Additionally, as the PageRank algorithm only considers the topology structure to rank the websites, there is a lack of consideration of the workflow complexity and the workflow deadline when applying the PageRank algorithm to the workflow research. Therefore, the PageRank algorithm should be modified to generate more precise task rescue priority for further constraining the workflow rescue from the faulty data centre.

In this chapter, a PageRank based fault tolerance (PRFT) strategy is developed including the PageRank-based priority assignment system, the replica selection method, the timeline allocation method, and the PageRank-constrained task scheduling algorithm. Firstly, the PageRank-based priority assignment system is used to prioritise the tasks in the replica-applied cloud environment based on the modified PageRank

algorithm. Then the replica selection method aims to find the optimal replica-ready data centres for the tasks to be resubmitted or migrated. The timeline allocation method focuses on the allocation of the tasks to be rescued or migrated on the timeline of the target data centre. Lastly, the PageRank-constrained task scheduling algorithm generates the task scheduling solution for rescuing the tasks at the faulty data centre.

A PageRank-based priority assignment system is developed for workflow applications to assign the task rescue priority to each task in the workflow. Each node $Nod_p \in N$ has its own PageRank value $PR(Nod_p)$. The PageRank value for the node Nod_p, $PR(Nod_p)$, can be formulated as follows in Eq. 6.3, where $succ(Nod_p)$ denotes the set of successors of Nod_p, $L(Nod_q)$ represents the number of the outbound nodes of Nod_q, and ρ is the total number of nodes in the workflow. A damping factor δ is applied, which normally has a value of 0.85 in the PageRank algorithm, to handle the probability of the task termination. The purpose of applying this damping factor δ is to find out the probability that a task can be successfully executed at any given time with a successful inheritance to its outbound nodes. Correspondingly, $1 - \delta$ is the probability that a task is terminated.

$$PR(Nod_p) = \frac{1-\delta}{\rho} + \delta * \sum_{Nod_q \in succ(Nod_p)} \frac{PR(Nod_q)}{L(Nod_q)} \quad (6.3)$$

Although the workflows have a similar topology to the web topology, the relationship among workflow tasks is more complex than the relationship among websites. Each website is held independently in the web topology, while an intermediate task in the workflow should wait to start until all its preceding tasks are completed in the workflow as abovementioned. Therefore, the workflow tasks are given priority values and are sorted according to their upward rank values. The upward rank value of a task Nod_p, $Rank_u(Nod_p)$, can be calculated as in Eq. 6.4. If a task is an exit task, its upward rank

value is computed as $Rank_u(Nod_p) = PR(Nod_p)$.

$$Rank_u(Nod_p) = PR(Nod_p) + max_{Nod_q \in succ(Nod_p)}(PR(Nod_q) + Rank_u(Nod_q)) \quad (6.4)$$

In the proposed priority assignment system, the upward rank value is calculated according to the PR value. This means the rank value of a task's predecessor is always higher than that of the task itself. However, one workflow instance may be impacted by other workflow instances in the cloud environment when doing task resubmission or migration. Thus, two balancing coefficients are introduced to jointly balance the upward ranking values among different workflow instances in accordance with the hard deadline of the workflow and the workflow complexity. As mentioned in Chapter 3, x workflow instances $\{G_1, G_2 \dots, G_x\}$ are studied in this thesis. The balancing coefficient γ for a workflow instance $G \in \{G_1, G_2 \dots, G_x\}$ can be formulated in Eq. 6.5, where $\{UR(G_1), UR(G_2), \dots, UR(G_x)\}$ is a set of the urgency values of x workflow instances. The urgency of a workflow G is the time buffer between the fault occurrence time point and its hard deadline.

$$\gamma(G) = \frac{max(\{UR(G_1),UR(G_2),\dots,UR(G_x)\}) - UR(G)}{max(\{UR(G_1),UR(G_2),\dots,UR(G_x)\}) - min(\{UR(G_1),UR(G_2),\dots,UR(G_x)\})} \quad (6.5)$$

The balancing coefficient σ for a workflow instance $G \in \{G_1, G_2 \dots, G_x\}$ can be formulated in Eq. 6.6, where $CountNod(G)$ is a count function which counts the number of nodes in the workflow instance G, $CountEdge(G)$ is a count function which counts the number of edges in the workflow instance G, $Num(Nod)$ denotes the total number of nodes in the cloud environment and $Num(Edge)$ denotes the total number of edges in the cloud environment.

$$\sigma(G) = \frac{CountNod(G)}{Num(Nod)} * \frac{CountEdge(G)}{Num(Edge)} \quad (6.6)$$

As the total number of the workflow instances, the total number of nodes, and the total number of edges are constantly changing in the cloud environment, the two balancing

coefficients will be dynamically changed to influence the final upward rank value of a specific task in a workflow instance. The final upward rank value of Nod_p in G can be formulated in Eq. 6.7.

$$FinRank_u(Nod_p) = (\gamma(G) + \sigma(G)) * Rank_u(Nod_p), Nod_p \in G \quad (6.7)$$

A task priority list is created according to the final upward rank value in descending order. The first element of this list has the highest priority and will be rescued first when handling faults.

The replica selection schema aims to guide the best replica site to access by evaluating the replica site performance when handling tasks according to the task priority list. In this strategy, the replica selection method which is proposed in Section 4.3 is applied to evaluate the optimal data access route for resubmitting or migrating the tasks.

The optimal eligible time slot selection method proposed in Section 5.2 is also applied in this research, which fully considers the time slot length similarity and the corresponding time slot resource situation at the target proper-working data centre for the tasks to be rescued. The consideration of the time slot length similarity and the corresponding time slot resource situation aims to minimise the waste of resources in the time slots and avoid the resource contention problem.

6.3 PageRank-Constrained Task Scheduling Algorithm

A PageRank-constrained task scheduling algorithm is proposed in Algorithm 6.1 to rescue the dependent tasks at the faulty data centre when the faults already occurred. Algorithm 6.1 firstly initializes the timeline and the task parsing vector space at each data centre and loads the tasks at the faulty data centre into the faulty task list $ft[]$ from **Line 1** to **Line 4**. It also initializes the task prioritising process for the tasks in the replica-applied cloud environment based on Eq. 6.7 at **Line 5**. Then $ft[]$ will be sorted

based on the task priority in descending order by applying Reverse QuickSort algorithm at **Line 6**. The tasks in $ft[]$ start to be processed from **Line 7** by following the sorting order. The optimal replica-ready data centre will be identified by applying the proposed replica selection strategy in Section 4.3 and add into $rr[]$ at **Line 8**. After that, the number of eligible time slots for the selected task from $ft[]$ is calculated at **Line 9**. If at least one eligible time slot exists, Mechanism 6.1 will identify an optimal eligible time slot at **Line 11** and complete the task resubmission process for the selected task from $ft[]$ at **Line 12**. Otherwise, Mechanism 6.2 will produce a task migration solution for the selected task from $ft[]$ at **Line 14**. The time complexity of Algorithm 6.1 is O(n).

Algorithm 6.1: PageRank-Constrained Task Scheduling Algorithm

Input: J, fault location

Output: Task resubmission solution

1. Initialization {
2. Set timeline
3. Set up the task parsing vector space
4. Load the tasks at the faulty data centre and add them into $ft[]$
5. Prioritise the tasks using Eq. 6.7 }
6. Reverse QuickSort $ft[]$ based on the task priority
7. **for** $ft[v]$ in$ft[]$, $v = 0$, $v \leq Sizeof(ft[]) - 1$ **do**
8. Select the optimal replica-ready data centre and add into $rr[]$
9. $Count(ET((ft[v])^{rr[]}))$
10. **if** $Count(ET((ft[v])^{rr[]})) > 0$
11. Do Mechanism 6.1
12. Move $ft[v]$ to $OE(ft[v])$
13. **else**
14. Do Mechanism 6.2
15. **end if**
16. **end for**

Mechanism 6.1 is used to identify the optimal eligible time slot when the number of

eligible time slots is greater than 1 at the replica-ready data centres. In Mechanism 6.1, the ranking values of the eligible time slots in $rr[]$ will be calculated using Eq. 5.8 in Section 5.2 for the input task $ft[v]$ at **Line 1**. After that, the optimal eligible time slot of the task will be identified at **Line 2** by selecting the maximum ranking values of the eligible time slots in $rr[]$. The beginning time point T_{Begin} of the optimal eligible time slot will be loaded at **Line 3**. Then the input task $ft[v]$ should be allocated to the beginning time point of the optimal eligible time slot $OE(ft[v])$ at **Line 4**. The resource line in the task parsing vector space will be updated to reflect the resource consumption of the input task $ft[v]$ at **Line 5**. Then the order number at Line 7 in Algorithm 6.1 will increase one at **Line 6**.By applying Mechanism 6.1, the task at the faulty data centre which has the eligible time slots at the replica-ready data centres can be rescued.

Mechanism 6.1: Optimal Eligible Time Slot Selection

Input: $ft[v]$, $rr[]$

Output: Optimal eligible time slot $OE(ft[v])$

1. Calculate $rank(ET((ft[v])^{rr[]}))$ using Eq. 5.8 in Section 5.2

2. Generate $OE(ft[v])$ by selecting the maximum $rank(ET((ft[v])^{rr[]}))$

3. Load the beginning time point T_{Begin} of $OE(ft[v])$

4. Allocate $ft[v]$ at T_{Begin}

5. Update the resource line for $OE(ft[v])$

6. $v++$ at Line 7 in Algorithm 6.1

Mechanism 6.2 is used to generate the task migration solution for the input task $ft[v]$ with no eligible time slots at the optimal replica-ready data centre. The current running tasks at the optimal replica-ready data centre are added into $crj[]$ at **Line 1**. The current-running tasks in $crj[]$ starts to be processed at **Line 2**.

Then the final upward rank value is calculated and the soft task deadline is counted for the selected task in $crj[]$ at **Line 3**. The final upward rank values between the input task $ft[v]$ and the selected task in $crj[]$ are compared at **Line 4** and then the selected task in $crj[]$ are removed if they cannot satisfy the requirements of the upward rank value, the resource, the time slot length, and the soft task deadline, respectively, from **Line 5** to **Line 8**. The $Check(SoftD())$ function is used to evaluate the after-release end time point of the selected task in $crj[]$ in comparison with its soft deadline. If this function is satisfied, it means the task can be migrated and will not influence the workflow hard deadline. Otherwise, the task cannot be migrated.

After the processing of the initial current-running task list $crj[]$, $crj[]$ is sorted based on the final upward rank value by applying the QuickSort algorithm at **Line 10**. If $crj[]$ is empty, the input task $ft[v]$ will be failed at **Line 12** and the order number of $ft[]$ will increase one Line 7 in Algorithm 6.1. Otherwise, $crj[]$ will be re-processed from the first element in$crj[]$ from **Line 14** to **Line 30**. The eligible time slots of the selected task in $crj[]$ will be identified at **Line 15**. If the number of eligible time slots is not equal to 0, the optimal eligible time slot will be identified for the selected task in $crj[]$ at **Line 17**. Then the selected task in $crj[]$ can be released to its optimal eligible time slot at **Line 18**. The after-release time slot at the original location of the selected task in $crj[]$ will be re-organised at **Line 19**. Then the after-release end time point of the input task $ft[v]$ will be checked at **Line 20** by applying the $Check(SoftD())$ function. If the soft deadline of the input task $ft[v]$ is satisfied, the input task $ft[v]$ will be moved to the beginning time point of the reorganized after-release time slot at **Line 22**. Then the order number of $ft[]$ will increase one at Line 7 in Algorithm 6.1. If the soft deadline of the input task $ft[v]$ cannot be satisfied, then the order number of $crj[]$ will increase one at **Line 14**. If the number of eligible time slots of the selected task in $crj[]$ is equal to 0

when identifying the eligible time slots at **Line 15**, then the order number of $crj[]$ will increase one at **Line 14.**

Mechanism 6.2: PageRank-Constrained Residual Task Processing

Input: $ft[v]$, $rr[]$

Output: Task migration solution

1. Load the current-running tasks in $rr[]$ and add into $crj[]$
2. **for** $crj[w]$ in $crj[]$, $w = 0$, $w \leq Sizeof(crj[]) - 1$] **do**
3. Calculate $FinRank_u(crj[w])$ and $SoftD(crj[w])$
4. Compare $FinRank_u(ft[v])$ with $FinRank_u(crj[w])$
5. Remove $crj[w]$ from $crj[]$ if $FinRank_u(crj[w]) > FinRank_u(ft[v])$
6. Remove $crj[w]$ from $crj[]$ with insufficient resource release
7. Remove $crj[w]$ from $crj[]$ with insufficient time slot length release
8. Remove $crj[w]$ from $crj[]$ if $Check(SoftD(crj[w]))$ cannot be satisfied
9. **end for**
10. QuickSort $crj[]$ based on the final upward rank value
11. **if** $crj[]$ = *null*
12. $v++$ at Line 7 in Algorithm 6.1
13. **else**
14. **for** $crj[u]$ in $crj[]$, $u = 0$, $u \leq Sizeof(crj[]) - 1$] **do**
15. Identify $ET(crj[u])$
16. **if** $ET(crj[u])$!= $null$
17. Identify $OE(crj[u])$
18. Release $crj[w]$ to $OE(crj[u])$
19. Re-organise the after-release time slot at the original location of $crj[u]$
20. $Check(SoftD(ft[v]))$
21. **If** $Check(SoftD(ft[v]))$ is satisfied
22. Move $ft[v]$ to the T_{begin} of the reorganized after-release time slot
23. $v++$ at Line 7 in Algorithm 6.1
24. **else**
25. $u++$
26. **end if**

27. **else**
28. $u++$
29. **end if**
30. **end for**
31. **end if**

6.4 Simulations

To evaluate the performance of the proposed PRFT strategy, two simulations are performed on OMNeT++ 5.4.1. Two types of workflows are implemented in the simulations, the Montage scientific workflow referenced from [10] and the meteorological workflow referenced from [130]. The hard deadlines of the Montage workflow and the meteorological workflow are dynamically changed to evaluate the fault tolerance performance of the HEFT-T strategy and the proposed PRFT strategy, respectively. The fault tolerance performance is measured in terms of TRR, WRR, and WCR in all two simulations. The available bandwidth, the error rate, and the network latency are assumed to be three major network performance metrics in the replica selection stage. The values of these three network performance metrics are set randomly.

6.4.1 Simulation 1 – Single workflow type with image backup environment

A cloud environment of 4 data centres with 80 circuits of 100 Mbps fibre-optic network integrated at each data centre is set up in Simulation 1. The image backup data replication environment is applied in this simulation. The fault occurs at T_0 in dc_3. Only the meteorological workflow instances are applied in this simulation and they are randomly placed at 4 data centres. In this simulation, one group of 10 meteorological workflow instances, labelled Meteorological 1, is scheduled at T_0 and another group of 10 meteorological workflow instances, labelled Meteorological 2, is scheduled at $T_{13.60}$. The deadline of two groups of meteorological workflow instances is dynamically

changed to evaluate the TRR, the WRR, and the WCR. The simulation results of the TRR are shown in Figure 6.2 and Figure 6.3.

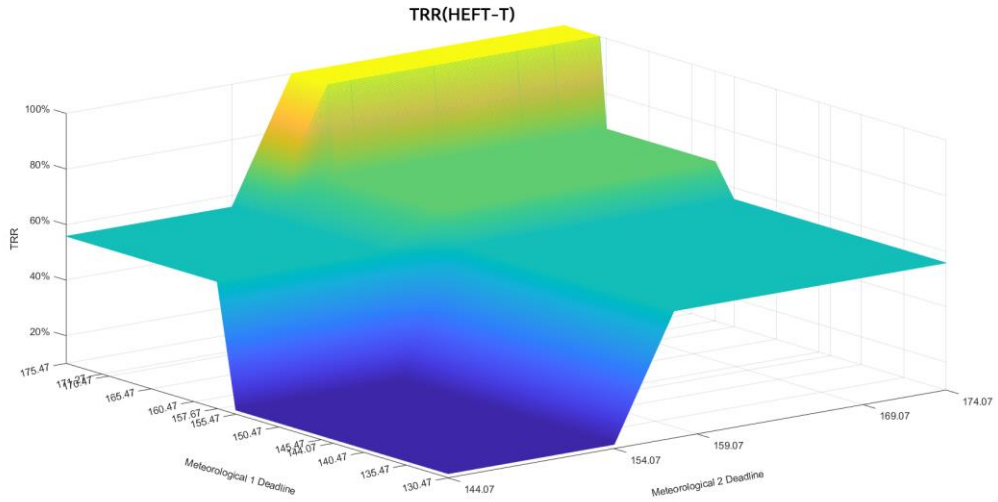

Figure 6.2 The TRR of the HEFT-T strategy

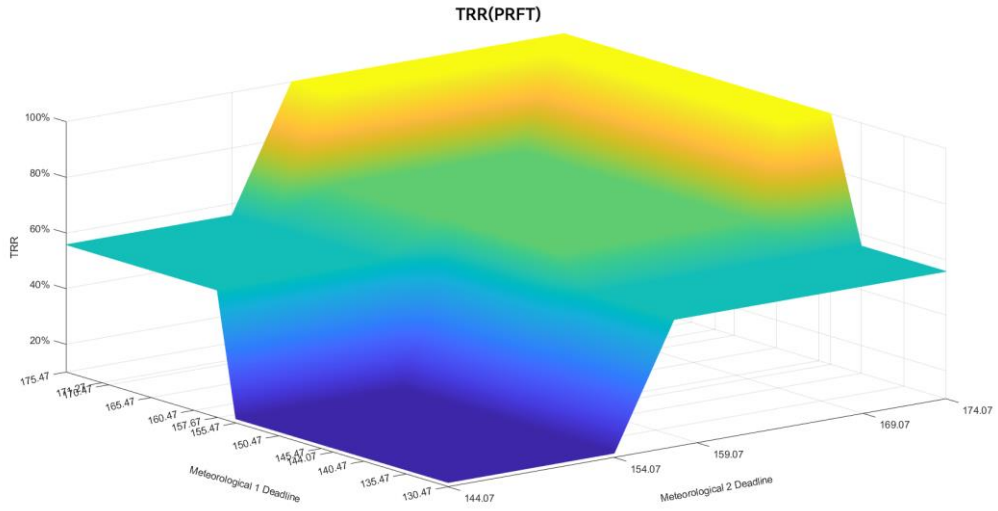

Figure 6.3 The TRR of the proposed PRFT strategy

The HEFT-T strategy keeps the TRR at 55.77% when the deadline of the Meteorological 1 group is in $[T_{130.47}, T_{157.67})$ and that of the Meteorological 2 group is in $[T_{157.67}, +\infty)$. When the deadline of the Meteorological 1 group is in $[T_{157.67}, T_{171.27})$ and that of the Meteorological 2 group is in $[T_{157.67}, +\infty)$, the TRR becomes 67.31%. The major difference between the proposed PRFT strategy and the HEFT-T strategy is

that the proposed PRFT strategy increases the TRR to 67.31% when the deadline of the Meteorological 1 group is in $[T_{144.07}, T_{157.67})$ and that of the Meteorological 2 group is in $[T_{157.67}, T_{171.27})$, and increases the TRR to 100% when the deadline of the Meteorological 1 group is in $[T_{144.07}, T_{171.27})$ and that of the Meteorological 2 group is in $[T_{171.27}, +\infty)$.

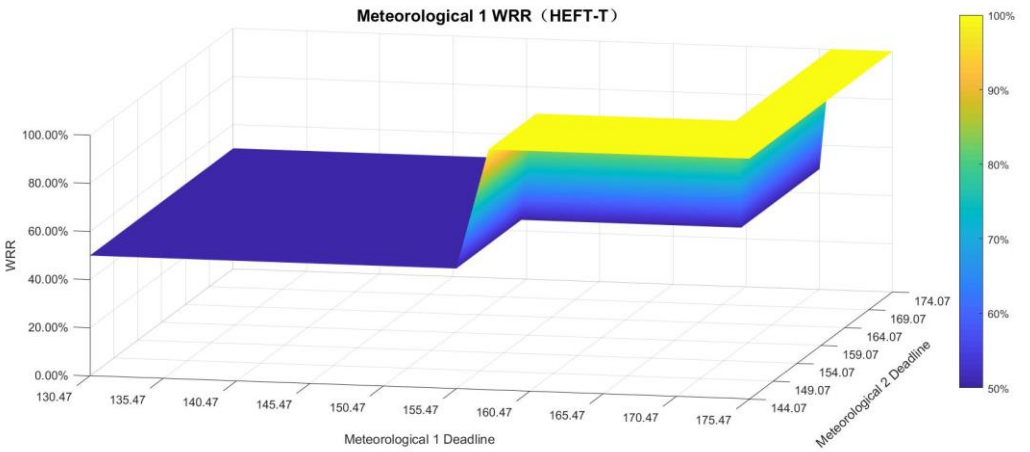

Figure 6.4 The WRR of Meteorological 1 (HEFT-T applied)

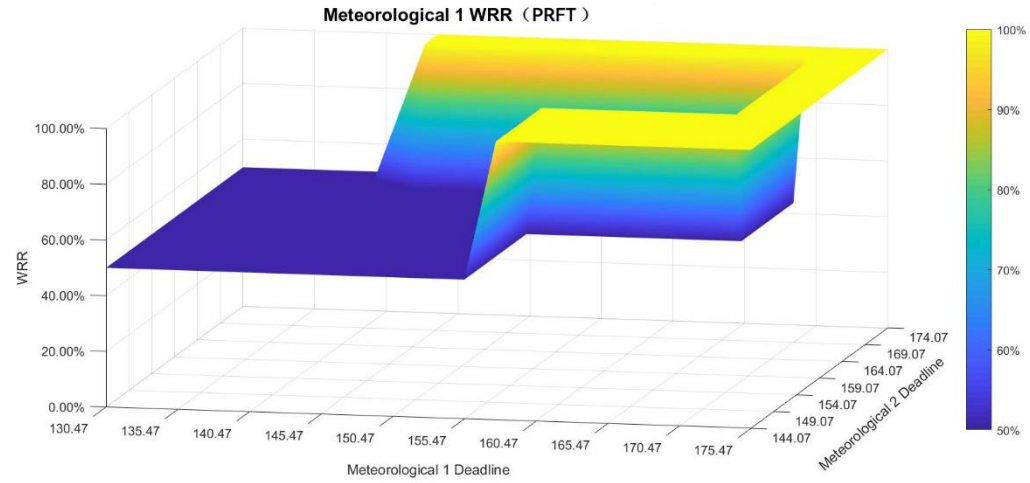

Figure 6.5 The WRR of Meteorological 1 (PRFT applied)

The simulation results of the WRR are shown in Figure 6.4 to Figure 6.7. It is evident that the proposed PRFT strategy can significantly improve the WRR in comparison with the HEFT-T strategy. Both the proposed PRFT strategy and the HEFT-T strategy keep

the same WRR trend in the Meteorological 2 group in Figure 6.6 and Figure 6.7.

The proposed PRFT strategy and the HEFT-T strategy achieve different WRR in the Meteorological 1 group. As shown in Figure 6.4 and Figure 6.5, the proposed PRFT strategy achieves better WRR performance when the deadline of the Meteorological 1 group is in $[T_{144.07}, T_{171.27})$ and that of the Meteorological 2 group is in $[T_{171.27}, +\infty)$. The WRR increases from 50% to 100%.

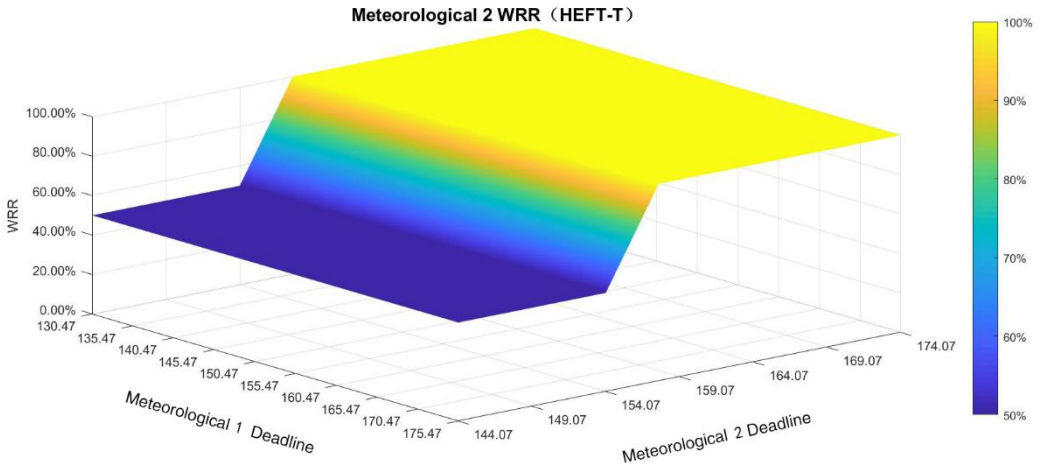

Figure 6.6 The WRR of Meteorological 2 (HEFT-T applied)

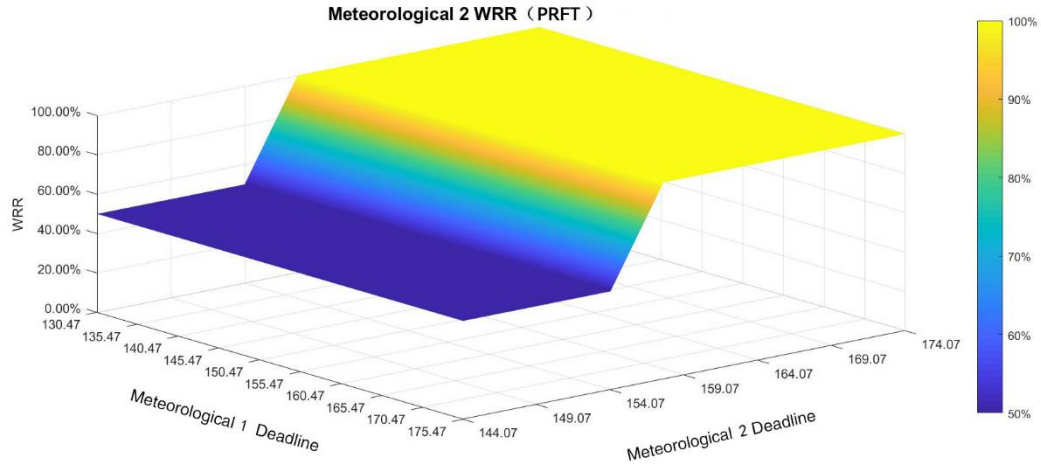

Figure 6.7 The WRR of Meteorological 2 (PRFT applied)

The simulation results of the WCR are shown in Figure 6.8 to Figure 6.11. It is clear that the proposed PRFT strategy can significantly improve the WCR in comparison with

the HEFT-T strategy. Firstly, the proposed PRFT strategy and the HEFT-T strategy achieve different WCR in the Meteorological 1 group. As shown in Figure 6.8, the WCR value stays at 9.70% when the deadline of Meteorological 1 group is in $[T_{130.47}, T_{157.67})$ and that of Meteorological 2 group is in $[T_{144.07}, +\infty)$ with the HEFT-T strategy applied. The WCR value increases to 14.18% when the deadline of the Meteorological 1 group is in $[T_{157.67}, T_{171.27})$ and that of the Meteorological 2 is in $[T_{157.67}, +\infty)$ with the HEFT-T strategy applied. The WCR value increases to 100% when the deadline of the Meteorological 1 group is in $[T_{157.67}, +\infty)$ and that of the Meteorological 2 group is in $[T_{144.07}, T_{157.67})$ or when the deadline of the Meteorological 1 group is in $[T_{171.27}, +\infty)$ and that of the Meteorological 2 group is in $[T_{157.67}, +\infty)$ with the HEFT-T strategy applied.

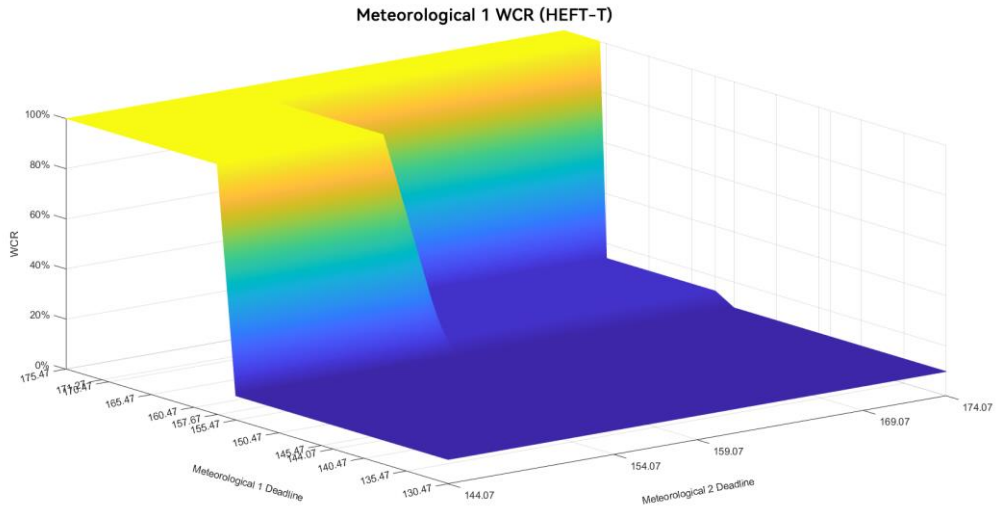

Figure 6.8 The WCR of Meteorological 1 (HEFT-T applied)

Different from the HEFT-T strategy, as shown in Figure 6.9, the proposed PRFT strategy increases the WCR when the deadline of the Meteorological 1 group is in $[T_{144.07}, T_{157.67})$ and that of the Meteorological 2 group is in $[T_{157.67}, T_{171.27})$. The WCR is also increased to 100% when the deadline of the Meteorological 1 group is in $[T_{144.07}, T_{171.27})$ and that of the Meteorological 2 group is in $[T_{171.27}, +\infty)$.

Besides, both the proposed PRFT strategy and the HEFT-T strategy keep the same trend in the Meteorological 2 group in Figure 6.10 and Figure 6.11. The WCR value stays at 9.70% when the deadline of the Meteorological 1 group is in $[T_{130.47}, +\infty)$ and that of the Meteorological 2 group is in $[T_{144.07}, T_{157.67})$. The WCR value will increase to 100% when the deadline of the Meteorological 1 group is in $[T_{130.47}, +\infty)$ and that of the Meteorological 2 group is in $[T_{157.67}, +\infty)$.

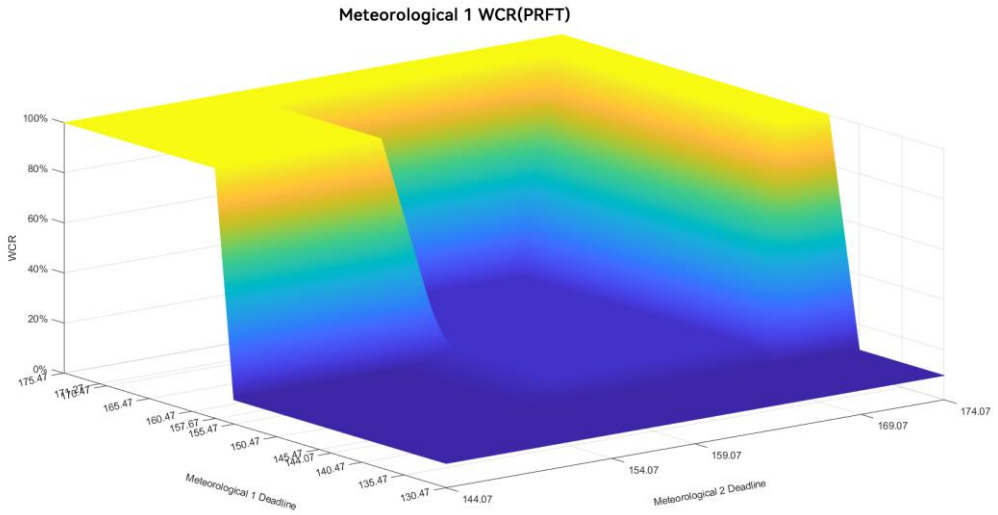

Figure 6.9 The WCR of Meteorological 1 (PRFT applied)

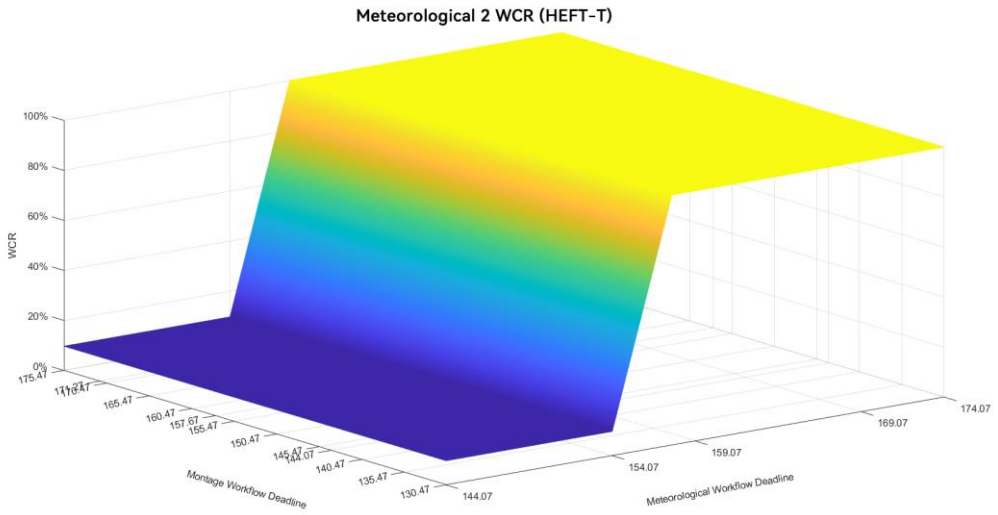

Figure 6.10 The WCR of Meteorological 2 (HEFT-T applied)

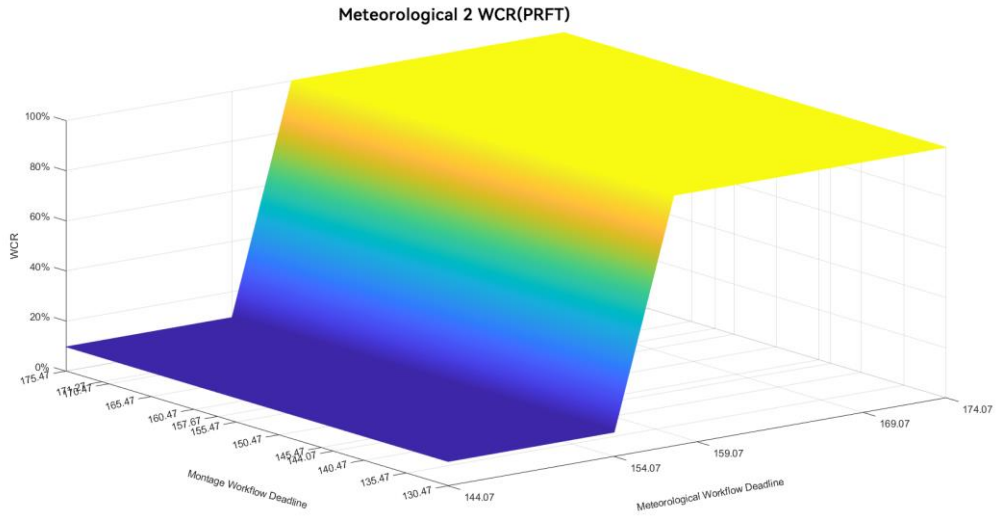

Figure 6.11 The WCR of Meteorological 2 (PRFT applied)

6.4.2 Simulation 2 – Multiple workflow types with mixed environment

A cloud environment of 4 data centres with 60 circuits of 100 Mbps fibre-optic network integrated at each data centre is set up in Simulation 2. The image backup data replication strategy and three-replicas data replication strategy are both applied to the simulation environment. The fault is set to occur at $T_{13.59}$ in dc_2. 10 meteorological workflow instances and 10 Montage workflow instances are applied in this simulation and they are randomly placed at 4 data centres. In this simulation, the Montage workflow instances are scheduled at T_0 and the meteorological workflow instances are scheduled at $T_{24.18}$. The deadline of two types of workflow instances is dynamically changed to evaluate the TRR, the WRR, and the WCR.

The simulation results of the TRR are shown in Figure 6.12 and Figure 6.13. The proposed PRFT strategy is still better than the HEFT-T strategy. The TRR increases from 53.19% to 100% with the proposed PRFT strategy when the deadline of Montage workflow instances is in $[T_{123.65}, T_{137.25})$ and that of meteorological workflow instances is in $[T_{168.25}, +\infty)$ while the TRR remains unchanged at 53.19% with the HEFT-T strategy in this deadline range.

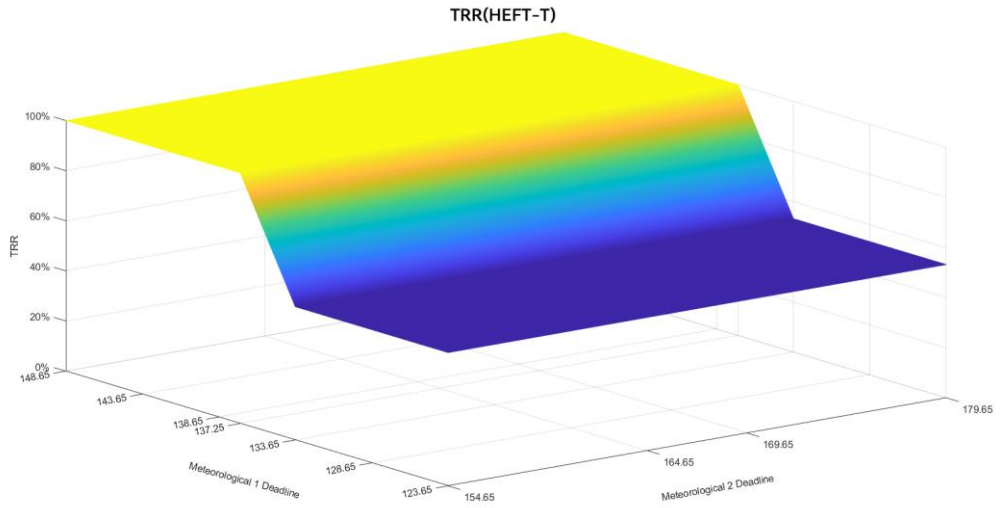

Figure 6.12 The TRR of the HEFT-T strategy

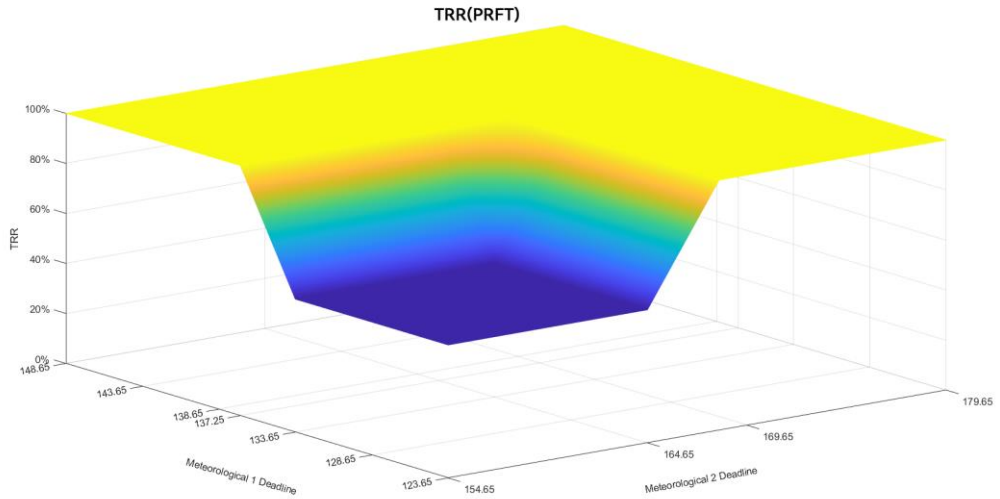

Figure 6.13 The TRR of the proposed PRFT strategy

The simulation results of the WRR are shown from Figure 6.14 to Figure 6.17. It is evident that the proposed PRFT strategy can achieve better WRR performance in comparison with the HEFT-T strategy. Firstly, different WRR performance is achieved in the Montage workflow instances as shown in Figure 6.14 and Figure 6.15. The proposed PRFT strategy achieves 100% WRR when the deadline of the Montage workflow instances is in $[T_{123.65}, T_{137.25})$ and that of the meteorological workflow instances is in $[T_{168.25}, +\infty)$, while the HEFT-T strategy only achieves 50% WRR in

this deadline range.

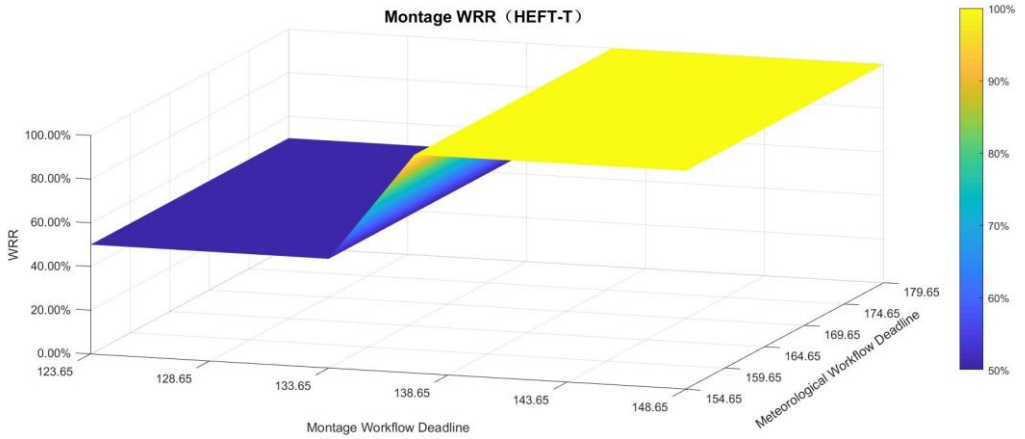

Figure 6.14 The WRR of the Montage workflow (HEFT-T applied)

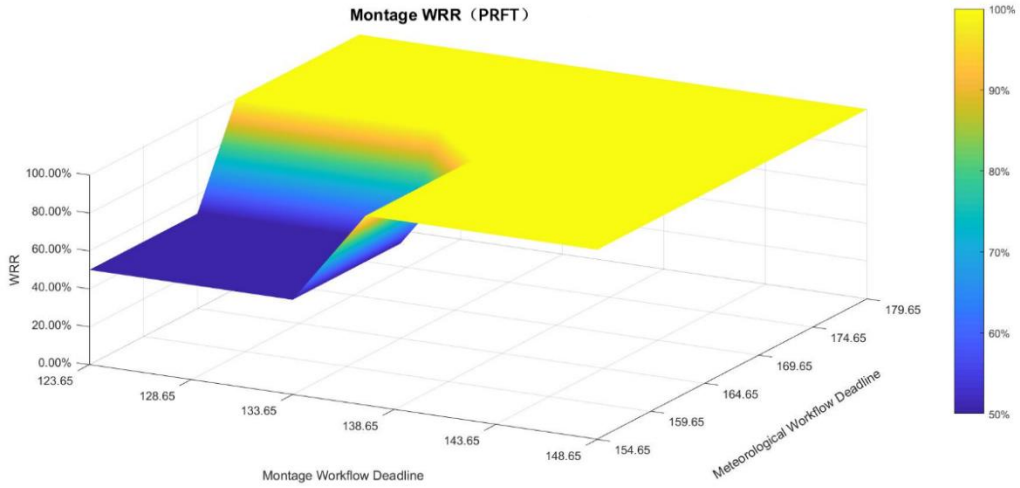

Figure 6.15 The WRR of the Montage workflow (PRFT applied)

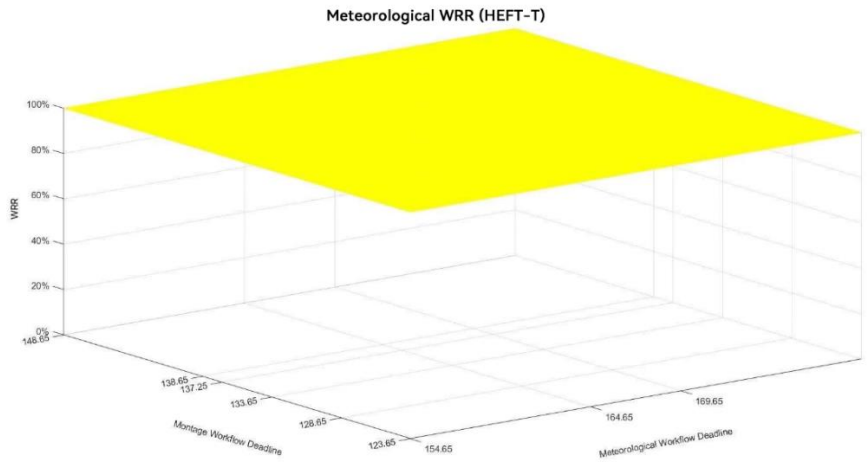

Figure 6.16 The WRR of the Meteorological workflow (HEFT-T applied)

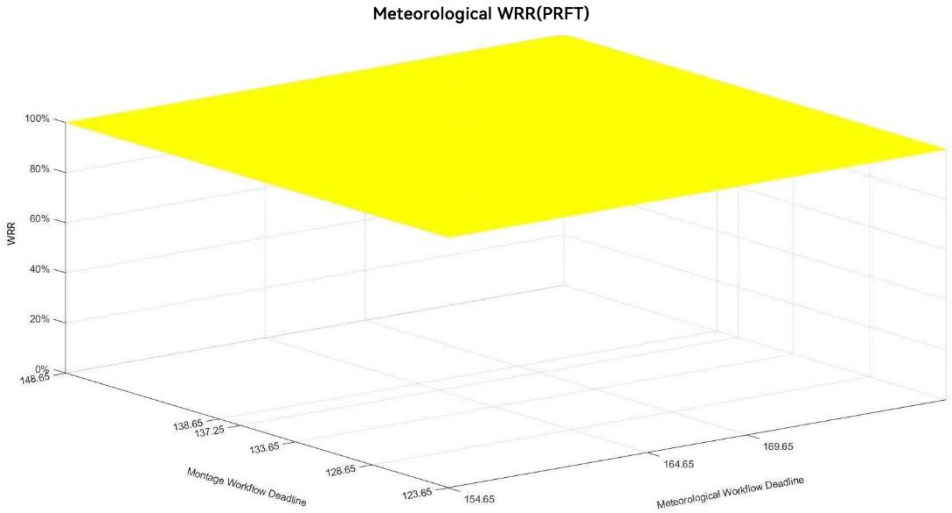

Figure 6.17 The WRR of the Meteorological workflow (PRFT applied)

The proposed PRFT strategy keeps the same WRR trend to the HEFT-T strategy on the meteorological workflow instances when the deadline of the Montage workflow instances is in $[T_{123.65}, +\infty)$ and that of the Meteorological workflow instances is in $[T_{154.65}, +\infty)$, as shown in Figure 6.16 and Figure 6.17.

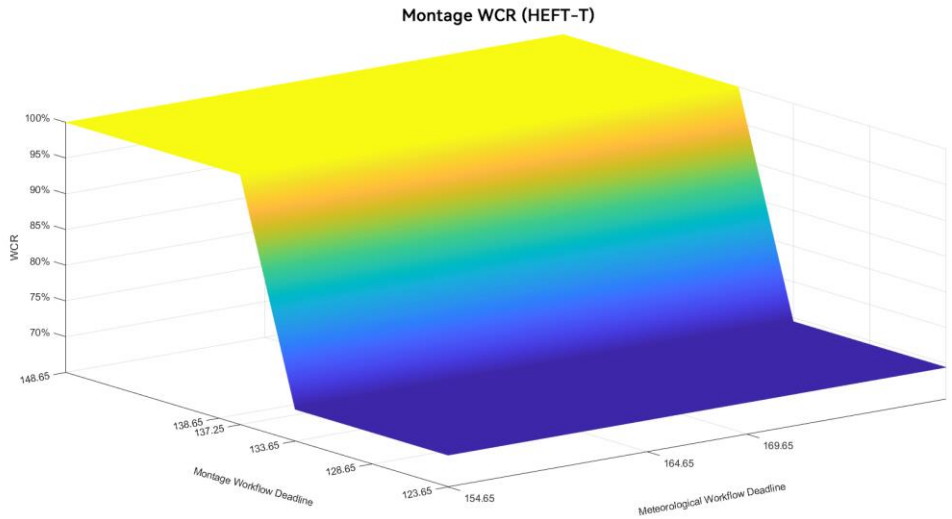

Figure 6.18 The WCR of the Montage workflow (HEFT-T applied)

Figure 6.19 The WCR of the Montage workflow (PRFT applied)

The simulation results of the WCR are shown from Figure 6.18 to Figure 6.21. It is evident that the proposed PRFT strategy can achieve better WCR in comparison with the HEFT-T strategy. Firstly, as shown in Figure 6.18 and Figure 6.19 for the Montage workflow instances, the proposed PRFT strategy achieves 100% WCR when the deadline of the Montage workflow instances is in $[T_{123.65}, T_{137.25})$ and that of the Meteorological workflow instances is in $[T_{168.25}, +\infty)$, while the HEFT-T strategy only achieves 69.46% WCR in this deadline range.

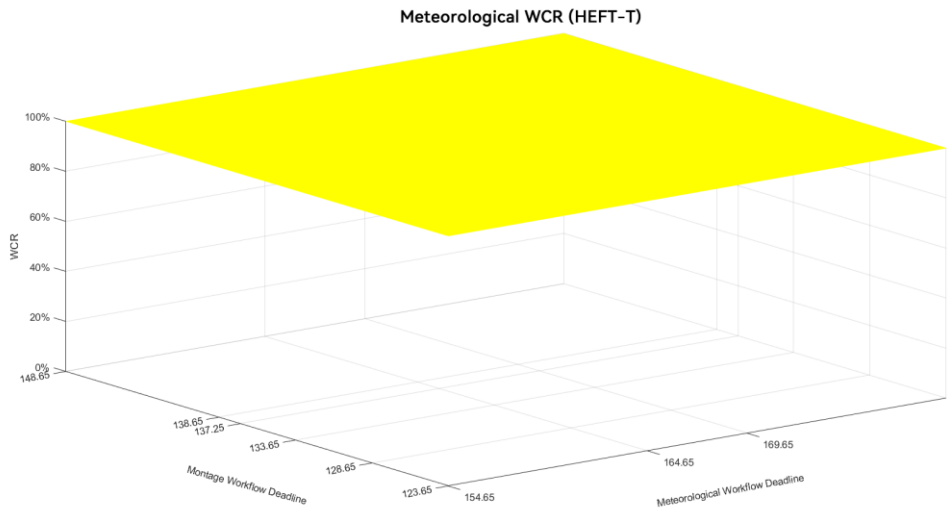

Figure 6.20 The WCR of the Meteorological workflow (HEFT-T applied)

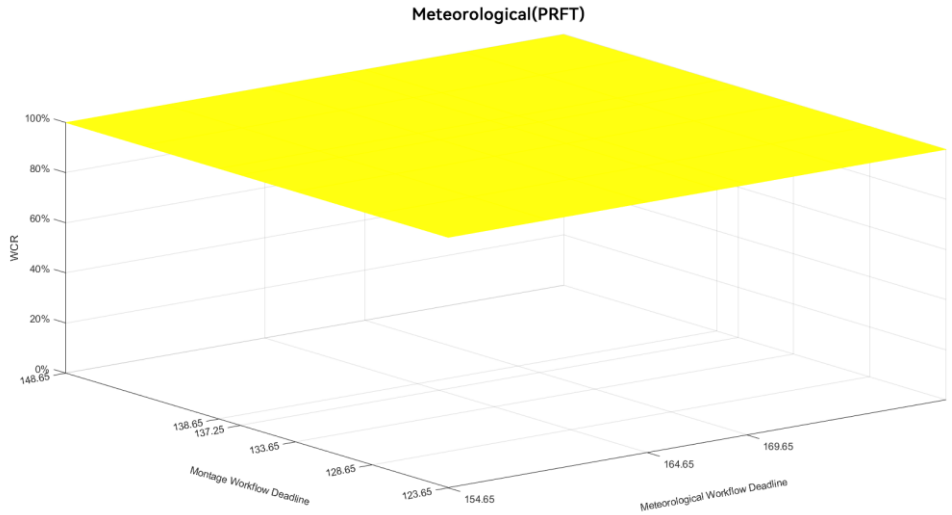

Figure 6.21 The WCR of the Meteorological workflow (PRFT applied)

Besides, the proposed PRFT strategy keeps the same WCR trend to the HEFT-T strategy on the Meteorological workflow instances when the deadline of the Montage workflow instances is in $[T_{123.65}, +\infty)$ and that of the Meteorological workflow instances is in $[T_{154.65}, +\infty)$, as shown in Figure 6.20 and Figure 6.21.

6.5 Summary

In Chapter 6, a PageRank based fault tolerance (PRFT) strategy is proposed for rescuing dependent tasks. This strategy focuses on the workflow task rescue by considering the attributes of the task, the timeline scenario, and the cloud performance. A priority assignment system is developed based on the modified PageRank algorithm to prioritise the workflow tasks. A dynamic PageRank-constrained task scheduling algorithm is proposed to generate the task scheduling solution when rescuing the tasks from the faulty data centre. The simulation results show that the proposed PRFT strategy can achieve better task resilience ratio, workflow resilience ratio, and workflow continuity ratio in comparison with the HEFT-T strategy, in both the traditional three-replica data replication environment and the image backup data replication environment.

Chapter 7 Contribution Summary, Discussions and Limitations

7.1 Contribution Summary

The proposed six strategies work in the field of replica creation, replica selection, fault tolerance for independent tasks, and fault tolerance for dependent tasks, respectively. Although the proposed six strategies aim to solve different problems and achieve different optimisation objectives, they are inter-related strategies, which can be aligned together to achieve a management chain by following the proposed data replication and fault management framework. This section will discuss the contribution of each proposed strategy and the inter-relationship among these six proposed strategies. The contribution summary of the six proposed strategies is shown in Table A1.5 in Appendix 1.

7.1.1 Contribution summary

- In Section 4.1, a replica creation strategy is discussed to consider both external data attributes and internal data attributes when making the replica creation decision. A data classification method categorises the flexible data type into two new data types to identify whether the flexible data can be replicated to a specific data centre. The external data attribute (access frequency) and the internal data attribute (data dependency) are jointly considered to constrain the replica creation, as they have been independently proved many times as two of the most significant data attributes in the past literature. The total cost is reduced by applying the proposed replica creation strategy in Section 4.1, in comparison with the total cost scenario without applying the proposed strategy.
- In Section 4.2, in addition to considering the data dependency and the access

frequency, the cloud map is also considered when making replica creation decisions. The cloud map has essential impacts when doing data relationship analysis because each data centre is seen as an individual host entity in the cloud environment. The local data relationship and the remote data relationship should be analysed towards the data location. Two new data dependency types, Within-DataCentre Data Dependency and Between-DataCentre Data Dependency are defined to analyse the local data relationship and the remote data relationship, respectively. An eligible data candidate pool is developed by identifying the highly-dependent and hot-access data. A recommended access frequency threshold value will be worked out to enable the optimal cost reduction per replica.

- In Section 4.3, a replica selection strategy is developed to avoid the potential network overloading problems related to the increased number of concurrent-running cloud application instances and the accompanying heavy data access needs. Different network performance metrics are jointly evaluated in the replica selection process and they are treated in different ways because of their own nature. A nested replica selection algorithm is developed to guide the optimal data replica access under the resource-sufficient scenario or the resource-insufficient scenario. The proposed replica selection strategy achieves a greater number of concurrent-running cloud application instances and more balanced resource load in comparison with the least response time replica selection algorithm.

The proposed three data replication strategies in the three sections above can be aligned together to guide the replica creation, the replica placement, and the replica selection for creating a replica-applied cloud environment. This replica-applied cloud environment

not only achieves the benefits mentioned above but also protects the cloud environment against the upcoming faults. However, the reactive fault tolerance strategies can also further improve the cloud performance after the faults occurred. The reactive fault tolerance thus enters the research view.

- In Section 5.1, a reactive fault tolerance strategy is developed to rescue independent tasks for better cloud resiliency. The task resubmission and the task migration are two core reactive fault tolerance techniques used in Section 5.1. However, frequent task resubmission and task migration operations may cause the resource contention problem at the proper-working data centres. Besides, some of the tasks at the faulty data centre may still fail to catch their deadlines after the task resubmission or the task migration. Therefore, the proposed fault tolerance strategy in Section 5.1 not only considers the resource load of accessing backup replicas but also strives to satisfy the deadline constraints. The utility-based task priority assignment system is developed by jointly considering the task urgency and the task operation profit. Then a one-stop concession mechanism is applied to the proposed fault tolerance algorithm for selecting appropriate data centres to accommodate the task rescued from the faulty data centre. The proposed reactive fault tolerance strategy achieves better cloud resiliency in terms of task resilience ratio, task rescue utility, and task operation profit in comparison with the typical HDFS robustness strategy, the RR strategy, and the JSQ strategy.
- Section 5.2 further adds the timeline allocation to the reactive fault tolerance strategy proposed in Section 5.1. To identify the eligible time slots on the timeline for the tasks rescued from the faulty data centre, a two-dimensional task parsing system is developed by parsing the task into a rectangle based on its task

execution duration and resource requirement. A novel three-dimensional priority assignment system is introduced to assign the task rescue priority to the tasks at the faulty data centre by comprehensively evaluating the task urgency, the task operation profit, and the number of eligible time slots. A timeline allocation method is proposed to identify the optimal eligible time slot for the tasks rescued from the faulty data centre by considering the time slot length similarity and the corresponding time slot resource situations. A one-stop concession mechanism is also applied to the proposed dynamic task rescheduling algorithm for avoiding timeline wastefulness and achieving better cloud resiliency. The proposed reactive fault tolerance strategy in Section 5.2 achieves better cloud resiliency in terms of task resilience ratio and enables more balanced resource load.

The two reactive fault tolerance strategies in Section 5.1 and Section 5.2 are both for independent tasks. As discussed in Chapter 2, the independent tasks have different nature in comparison with the dependent tasks. Therefore, the two proposed reactive fault tolerance strategies in Section 5.1 and Section 5.2 might not be applicable to the dependent tasks. The specific reactive fault tolerance strategy for rescuing dependent tasks should be analysed.

- In Chapter 6, a reactive fault tolerance strategy is developed for rescuing the workflow applications because the workflows always contain a large number of dependent tasks. Firstly, the timeline allocation is still an important issue to the dependent tasks when doing the task resubmission or the task migration operations. As demonstrated in Section 5.2, selecting the first available server may not achieve the optimal cloud resiliency. Besides, the insufficient consideration of the resource contention and the deadline contention among the tasks in different concurrent-running workflow instances may disrupt cloud

resiliency. Apart from that, the workflow topology should be fully analysed because the workflow tasks must be prioritised by considering the task dependencies, as a workflow task cannot be initiated until all its preceding workflow tasks are completed. A PageRank-based priority assignment system is developed to fully analyse the workflow topology and address the resource contention and the deadline contention among the tasks in different concurrent-running workflow instances. By following the task priority assigned by the proposed PageRank-based priority assignment system, a dynamic PageRank-constrained task scheduling algorithm is developed to generate the fault handling solution for the tasks at the faulty data centre. The proposed reactive fault tolerance strategy in Chapter 6 can significantly increase the task resilience ratio, the workflow resilience ratio, and the workflow continuity ratio in comparison with the HEFT-T strategy, in both the traditional three-replica data replication environment and the image backup data replication environment.

7.1.2 Inter-relationship among the proposed strategies

The proposed six strategies are guided and developed by following the proposed data replication and fault management framework in Chapter 3. Each proposed strategy can be applied in a specific module in the proposed data replication and fault management framework. They can be aligned together to achieve a management chain for the cloud environment.

- The proposed replica creation strategies in Section 4.1 and 4.2 are two alternative replica creation strategies including the replica placement rules, to be applied into the replica creation module and the replica placement module in the replica agent. These two replica creation strategies can guide the creation and the placement of the data replicas to multiple cloud data centres. The replica

scheduling unit in the data centre scheduling module will create and place multiple data replicas to multiple cloud data centres by referencing the replica creation strategy applied in the replica creation module and the replica placement strategy applied in the replica placement module.

- The proposed replica selection strategy in Section 4.3 can be applied in the replica selection module in the replica agent. This replica selection strategy can guide the tasks to access the optimal required replicas. The task scheduling unit in the data centre scheduling module will control the replica selection processes by referencing the replica selection strategy applied in the replica selection module. The replica scheduling unit in the data centre scheduling module will control the replica re-creation process and re-create the required replica by referencing the replica creation strategy applied in the replica creation module.
- The proposed reactive fault tolerance strategy in Section 5.1, Section 5.2, and Section 6.1 can be applied in the fault handling guide unit in the fault management agent. The proposed reactive fault tolerance strategy in Section 5.1 and Section 5.2 are two alternative fault tolerance strategies for rescuing the independent tasks. The proposed reactive fault tolerance strategy in Section 6.1 is to rescue the dependent tasks. The task scheduling unit in the data centre scheduling module will reference the corresponding reactive fault tolerance strategies for different task types from the fault handling guide unit in the fault management agent.

7.2 Discussions

In this thesis, six strategies are proposed including two alternative replica creation strategies, one replica selection strategy, two alternative reactive fault tolerance strategies for independent tasks and one reactive fault tolerance strategy for dependent

tasks. The applicability of each proposed strategy will be discussed in this section.

- The applicability of the alternative replica creation strategies

The proposed two replica creation strategies are alternative. They cannot be simultaneously applied in a single data centre. The first replica creation strategy proposed in Section 4.1 considers the data dependency and the access frequency only to constrain the replica creation. It focuses on the data attribute analysis without consideration of any environmental information in the cloud environment. Differently, the second replica creation strategy proposed in Section 4.2 considers the cloud map in the data dependency analysis. Each data centre is recognized as an individual host entity in the cloud environment. The data dependency analysis is conducted towards the data location. The data dependency will be categorised into two new data dependency types to reveal the local data relationship and the remote data relationship.

The difference between these two alternative replica creation strategies highly distinguishes the applicability of these two alternative replica creation strategies in different cloud architectures. The first replica creation strategy proposed in Section 4.1 is more suitable to apply in the public cloud architecture because the computing resources of a data centre are always shared resources among the public cloud data centres. There is no boundary among those public cloud data centres.

Differently, the second replica creation strategy proposed in Section 4.2 is more applicable to the private cloud architecture. The private cloud architecture always requires higher customisation and stronger cloud security than the public cloud architecture. Therefore, a data in a private cloud data centre will encounter stronger policy constraints to share with other cloud data centres than a data in a public cloud data centre. Therefore, each private cloud data centre should be recognized as a strong individual host entity. Hence, the cloud map oriented replica creation is more suitable to

be applied by strongly considering the analysis of the local data relationship and the remote data relationship. This can enable more precise localization of the data dependency situations to the data in the private cloud data centres.

- The applicability of the replica selection strategy

The proposed replica selection strategy in Section 4.3 can be applied in the replica selection module to guide the optimal data to access. It can also be applied in three proposed fault tolerance strategies to guide the replica selection when rescuing the tasks from the faulty data centre. This replica selection strategy is applicable for any type of cloud architecture because it is a network performance oriented strategy by analysing the network performance metrics without any constraints to the cloud architectures. Besides, the proposed replica selection algorithm is fit to adapt and extend more network performance evaluation metrics. It should be noted that different evaluation metrics should still be treated in different ways when extending the replica selection algorithm.

- The applicability of the alternative fault tolerance strategies for independent tasks

The proposed two fault tolerance strategies for independent tasks are alternative. They also cannot be simultaneously applied in a single data centre. The proposed reactive fault tolerance strategy in Section 5.1 places emphasis on the utility-based priority assignment to prioritise the independent tasks. The task urgency and the task operation profit are two major task attributes to be considered in the utility calculation. The goal of the proposed strategy in Section 5.1 aims to achieve better task resilience ratio, task rescue utility, and task operation profit. Although the overall network performance at each replica-ready data centre is taken into account, it is used to identify the optimal replica-ready data centre only. The detailed task allocation on the timeline of the

optimal replica-ready data centre is not considered. Therefore, the proposed reactive fault tolerance strategy in Section 5.1 is more applicable to a cloud environment with low workloads because each data centre in such a cloud environment will be influenced by the task resubmission operations or the task migration operations to a small extent when handling faults. The proposed strategy in Section 5.1 can significantly achieve better task resilience ratio, task rescue utility, and task operation profit to the cloud environment with low workloads.

Differently, the proposed reactive fault tolerance strategy in Section 5.2 puts emphasis on the detailed task allocation on the timeline of the data centre. The timeline scenario is considered in both the task prioritising phase and the eligible time slot selection phase when handling faults. The consideration of the number of eligible time slots in the task prioritising phase can reveal the task processing difficulty to allocate in a specific data centre. The consideration of the time slot length similarity and the time slot resource situations can avoid the time slot wasteness and the resource contention problem. Therefore, the proposed reactive fault tolerance strategy in Section 5.2 is more applicable to a cloud environment with high workloads because each data centre in such a cloud environment will be largely impacted by the task resubmission operations or the task migration operations when handling faults. The proposed strategy in Section 5.2 can significantly improve the task resilience ratio while balancing the resource load to the cloud environment with high workloads.

- The applicability of the reactive fault tolerance strategy for dependent tasks

The proposed reactive fault tolerance strategy for dependent tasks in Chapter 6 develops a PageRank-based priority assignment method to assign the priority to the workflow tasks. This is the first time that the PageRank algorithm is applied in the fault tolerance research area. The PageRank algorithm is modified to achieve an applicable priority

assignment system for the dependent tasks in the workflow applications by integrating the task dependency analysis and the impact analysis among different workflow instances into the priority calculation process. It can be applied to all workflow types which a workflow task cannot be initiated until all its preceding tasks are completed.

7.3 Limitations

Although the research in this thesis achieves a lot of benefits to cloud performance, it still has some limitations. Five major limitations are listed as follows.

- Optimisation objective diversity

Multiple optimisation objectives have been achieved in this thesis. There are still many other optimisation objectives to be considered, such as energy consumption, response time, and makespan, etc.

- Replica placement simplification

In this thesis, the replica placement is simplified by adopting the traditional replica placement strategy, in which each replica will be placed to the locations of its relevant tasks. Many replica placement strategies have been proposed in the past literature. In some cases, the traditional replica placement strategy may increase the number of replicas and incur more extra storage costs. It may also have other negative influences in terms of energy consumption, data synchronization, and data deduplication, etc.

- Workflow type limitation

In this thesis, it is assumed that a workflow task can only be initiated after all its preceding tasks are completed. However, this might not always be the case in reality. Therefore, the proposed fault tolerance strategy for workflows in Chapter 6 may not be applicable to all workflow types in the real world.

- Lack of experiments

In this thesis, the simulations are conducted to evaluate the proposed six strategies. In the simulation results, it is evident that the proposed strategies can achieve better performance than the comparative strategies. Nevertheless, the simulations are different to the experiments because the simulations are always conducted in the virtual simulation environment. More experiments are still needed to prove the applicability of the proposed six strategies in the real world.

- The experiment threats

As mentioned above, the proposed six strategies in this thesis are evaluated based on simulations. Although the simulations are commonly used to evaluate the cloud-related research, the experiments are still required. However, there are also some technical and social obstacles to the implementation of the proposed six strategies in the real world. From the social perspective, the implementation of new management strategy in each cloud service provider should be progressive to keep the stable running of cloud services. Therefore, it may take a long time period to update the management rules. Besides, from the technical perspective, the current cloud control system in each cloud service provider may not be adaptable to implement the proposed algorithms. For example, it may not be able to create a three-dimensional vector space to prioritise the tasks. Apart from that, the proposed strategies may be not adaptable to the cloud environment with multiple cloud service providers in some special cases if the cloud service provider boundary is necessary.

- The applicability to the server level or the cloud service provider level

In this thesis, all of the six strategies are proposed for the cloud environment. As mentioned many times, each cloud data centre is recognized as an individual host entity in the cloud environment. The proposed six strategies can greatly work at the data centre level. However, the servers in the cloud data centres or the cloud service

providers in the cloud environment can also be defined as individual host entities. Therefore, the proposed six strategies in this thesis should be tested to prove their applicability at the server level or the cloud service provider level.

Chapter 8 Conclusions and Future Work

To conclude, data explosion becomes one of the major challenges to organizations all over the world. The cloud computing service offers a novel paradigm to alleviate massive data processing challenges based on its on-demand service model and distributed cloud architecture. As the number of users increases, the computing capability in a single data centre might restrict the overall cloud performance. At the same time, unexpected faults may occur in the cloud environment. Therefore, data replication is proposed to enable a strategical data access distribution to multiple cloud data centres to improve cloud performance. It can also achieve cloud robustness to avoid the negative influences of the upcoming faults. Furthermore, the replica-applied cloud environment still needs the reactive fault tolerance strategy to further improve the cloud performance after the faults occurred.

A data replication and fault management framework is firstly introduced to achieve a decentralised management to offer the flexibility, the adaptability, and the geo-diversity for the global collaborators in the cloud environment. This framework contains two types of platforms at the user side and the data centre side, respectively. Each type of platform contains multiple modules which are responsible for different management functionalities.

Six strategies have been proposed in this thesis, which include three data replication strategies and three fault tolerance strategies. Firstly, a replica creation strategy is proposed to reduce the total cost by jointly considering the data dependency and the access frequency. Secondly, a cloud map oriented replica creation strategy is proposed to achieve the optimal cost reduction per replica with the balancing between the total cost and the number of replicas. Thirdly, a network performance based replica selection strategy is proposed to avoid the potential network overloading problem and increase

the number of concurrent-running instances at the same time.

The data replication strategy as a data management approach is also widely adopted to create a replica-applied cloud environment to protect the cloud environment against the upcoming faults. The reactive fault tolerance strategies are also required to further improve the cloud performance by rescuing the tasks from the faulty data centres after the faults already occurred. A utility-based fault tolerance strategy is firstly proposed to rescue the independent tasks at the faulty data centre for achieving better cloud resiliency with respect to the resource load of accessing replicas and the task deadline. Secondly, a timeline-oriented fault tolerance strategy for rescuing the independent tasks is proposed to achieve better cloud resiliency and load balancing performance by taking the timeline allocation into consideration. Thirdly, a PageRank based fault tolerance strategy is proposed to rescue the workflow applications for improving the task resilience ratio, the workflow resilience ratio, and the workflow continuity ratio by applying the modified PageRank algorithm based priority assignment method. However, this thesis still has the following limitations.

- Optimisation objective diversity
- Replica placement simplification
- Workflow type limitation
- Lack of experiments
- The applicability to the server level or the cloud service provider level

In future works, the proposed strategies are planned to extend into different types of cloud architectures such as edge computing and mobile computing. The PageRank algorithm is also planned to extend into the replica placement research area for developing a PageRank-based replica placement strategy.

Bibliography

[1] Abdollahi Nami, A., & Rajabion, L. (2019). Data replication techniques in the mobile ad hoc networks. *International Journal of Pervasive Computing and Communications*, *15*(3/4), 174–198. https://doi.org/10.1108/IJPCC-06-2019-0051

[2] Agarwal, H., & Sharma, A. (2015). A comprehensive survey of Fault Tolerance tech niques in Cloud Computing. *2015 International Conference on Computing and Network Communications (CoCoNet)*, 408–413. https://doi.org/10.1109/CoCoNet.2015.7411218

[3] Almuttairi, R. M., Wankar, R., Negi, A., & Rao, C. R. (2010). Replica Selection in Data Grids Using Preconditioning of Decision Attributes by K-means Clustering (K-RS DG). *2010 Second Vaagdevi International Conference on Information Technology for R eal World Problems*, 18–23. https://doi.org/10.1109/VCON.2010.11

[4] Altiparmak, N., & Tosun, A. S. (2016). Multithreaded Maximum Flow Based Optim al Replica Selection Algorithm for Heterogeneous Storage Architectures. *IEEE Transac tions on Computers*, *65*(5), 1543–1557. https://doi.org/10.1109/TC.2015.2451620

[5] Arabnejad, H., & Barbosa, J. G. (2014). List Scheduling Algorithm for Heterogeneo us Systems by an Optimistic Cost Table. *IEEE Transactions on Parallel and Distribute d Systems*, *25*(3), 682–694. https://doi.org/10.1109/TPDS.2013.57

[6] Awad, A., Salem, R., Abdelkader, H., & Salam, M. A. (2021). A Novel Intelligent A pproach for Dynamic Data Replication in Cloud Environment. *IEEE Access*, *9*, 40240–4 0254. https://doi.org/10.1109/ACCESS.2021.3064917

[7] Aygun, B., Gunel Kilic, B., Arici, N., Cosar, A., & Tuncsiper, B. (2021). Applicatio n of binary PSO for public cloud resources allocation system of video on demand (VoD) services. *Applied Soft Computing*, *99*, 106870. https://doi.org/10.1016/j.asoc.2020.1068 70

[8] Bai, X., Jin, H., Liao, X., Shi, X., & Shao, Z. (2013). RTRM: A Response Time-Bas

ed Replica Management Strategy for Cloud Storage System. *Grid and Pervasive Computing*, 124–133. https://doi.org/10.1007/978-3-642-38027-3_13

[9] Baldwin, M., & Cromity, J. (2012). SaaS and Cloud Computing, the Rise of Compartmentalizing Users Online Via Subscription. *New Review of Information Networking*, *17*(2), 120–126. https://doi.org/10.1080/13614576.2012.724302

[10] Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Mei-Hui Su, & Vahi, K. (2008). Characterization of scientific workflows. *2008 Third Workshop on Workflows in Support of Large-Scale Science*, 1–10. https://doi.org/10.1109/WORKS.2008.4723958

[11] Bittencourt, L. F., Sakellariou, R., & Madeira, E. R. M. (2010). DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm. *2010 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 27–34. https://doi.org/10.1109/PDP.2010.56

[12] Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., & Zomaya, A. Y. (2015). Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, *18*(1), 385–402. https://doi.org/10.1007/s10586-014-0404-x

[13] Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., & Zomaya, A. Y. (2015). Models for efficient data replication in cloud computing datacenters. *2015 IEEE International Conference on Communications (ICC)*, 6056–6061. https://doi.org/10.1109/ICC.2015.7249287

[14] Bouzerzour, N. E. H., Ghazouani, S., & Slimani, Y. (2020). A survey on the service interoperability in cloud computing: Client-centric and provider-centric perspectives. *Software, Practice & Experience*, *50*(7), 1025–1060. https://doi.org/10.1002/spe.2794

[15] Cao, S., Deng, K., Ren, K., Li, X., Nie, T., & Song, J. (2019). A Deadline-Constrained Scheduling Algorithm for Scientific Workflows in Clouds. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th In*

ternational Conference on Smart City; IEEE 5th International Conference on Data Scie nce and Systems (HPCC/SmartCity/DSS), 98–105. https://doi.org/10.1109/HPCC/Smart City/DSS.2019.00029

[16] Cao, X., DeVries, B., Scripps, J., & Trefftz, C. (2020). Data Allocation and Replica tion in Data Center: Tradeoff and Solutions. *2020 IEEE International Conference on El ectro Information Technology (EIT)*, 239–244. https://doi.org/10.1109/EIT48999.2020.9 208247

[17] Challita, S., Paraiso, F., & Merle, P. (2017). Towards Formal-Based Semantic Inter operability in Multi-Clouds: The FCLOUDS Framework. *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 710–713. https://doi.org/10.1109/CLOUD. 2017.98

[18] Chang, R. S., & Chang, H. P. (2008). A dynamic data replication strategy using acc ess-weights in data grids. *The Journal of Supercomputing*, *45*(3), 277–295. https://doi.or g/10.1007/s11227-008-0172-6

[19] Chellouf, M., & Hamrouni, T. (2021). Popularity and correlation aware data replica tion strategy based on half-life concept and clustering in cloud system. *Concurrency an d Computation*, *33*(10). https://doi.org/10.1002/cpe.6159

[20] Chen, G., Guan, N., Huang, K., & Yi, W. (2020). Fault-tolerant real-time tasks sch eduling with dynamic fault handling. *Journal of Systems Architecture*, *102*, 101688. http s://doi.org/10.1016/j.sysarc.2019.101688

[21] Chen, T., Chuang, T.-T., & Nakatani, K. (2016). The perceived business benefit of cloud computing: an exploratory study. *Journal of International Technology and Inform ation Management*, *25*(4), 101–121.

[22] Crago, S., Dunn, K., Eads, P., Hochstein, L., Dong-In Kang, Mikyung Kang, Modi um, D., Singh, K., Jinwoo Suh, & Walters, J. P. (2011). Heterogeneous Cloud Computin

g. *2011 IEEE International Conference on Cluster Computing*, 378–385. https://doi.org/10.1109/CLUSTER.2011.49

[23] Cunha, D., Neves, P., & Sousa, P. (2014). PaaS manager: a platform-as-a-service aggregation framework. *Computer Science & Information Systems*, *11*(4), 1209-1228. https://doi.org/10.2298/CSIS130828028C

[24] Deng, K., Song, J., Ren, K., Yuan, D., & Chen, J. (2011). Graph-Cut Based Coscheduling Strategy Towards Efficient Execution of Scientific Workflows in Collaborative Cloud Environments. *2011 IEEE/ACM 12th International Conference on Grid Computing*, 34–41. https://doi.org/10.1109/Grid.2011.14

[25] Deng, S., Huang, L., Taheri, J., & Zomaya, A. Y. (2015). Computation Offloading for Service Workflow in Mobile Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, *26*(12), 3317–3329. https://doi.org/10.1109/TPDS.2014.2381640

[26] Devarajan, H., Kougkas, A., & Sun, X. H. (2020). HReplica: A Dynamic Data Replication Engine with Adaptive Compression for Multi-Tiered Storage. *2020 IEEE International Conference on Big Data (Big Data)*, 256–265. https://doi.org/10.1109/BigData50022.2020.9378167

[27] Ebadi, Y., & Jafari Navimipour, N. (2019). An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm. *Concurrency and Computation*, *31*(1), e4757. https://doi.org/10.1002/cpe.4757

[28] ElYamany, H. F., Mohamed, M. F., Grolinger, K., & Capretz, M. A. (2015). A generalized service replication process in distributed environments. In *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, 186-193. https://doi.org/10.5220/0005485201860193

[29] Faragardi, H. R., Saleh Sedghpour, M. R., Fazliahmadi, S., Fahringer, T., & Rasoul

i, N. (2020). GRP-HEFT: A Budget-Constrained Resource Provisioning Scheme for Wo rkflow Scheduling in IaaS Clouds. *IEEE Transactions on Parallel and Distributed Syste ms*, *31*(6), 1239–1254. https://doi.org/10.1109/TPDS.2019.2961098

[30] Gill, N. K., & Singh, S. (2014). Dynamic Cost-Aware Re-replication and Rebalanci ng Strategy in Cloud System. In *Proceedings of the 3rd International Conference on Fr ontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, 39–47. Spring er International Publishing. https://doi.org/10.1007/978-3-319-12012-6_5

[31] Gill, N. K., & Singh, S. (2016). A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. *Future Generation Computer Systems*, *65* , 10–32. https://doi.org/10.1016/j.future.2016.05.016

[32] Gupta, P., Seetharaman, A., & Raj, J. R. (2013). The usage and adoption of cloud c omputing by small and medium businesses. *International Journal of Information Mana gement*, *33*(5), 861–874. https://doi.org/10.1016/j.ijinfomgt.2013.07.001

[33] Gupta, V., Harchol Balter, M., Sigman, K., & Whitt, W. (2007). Analysis of join-th e-shortest-queue routing for web server farms. *Performance Evaluation*, *64*(9), 1062–10 81. https://doi.org/10.1016/j.peva.2007.06.012

[34] Gupta, V., Kaur, B. P., & Jangra, S. (2019). An efficient method for fault tolerance in cloud environment using encryption and classification. *Soft Computing (Berlin, Germ any)*, *23*(24), 13591–13602. https://doi.org/10.1007/s00500-019-03896-6

[35] Hasan, M., & Goraya, M. S. (2018). Fault tolerance in cloud computing environme nt: A systematic survey. *Computers in Industry*, *99*, 156–172. https://doi.org/10.1016/j.c ompind.2018.03.027

[36] He, L., Qian, Z., & Shang, F. (2020). A novel predicted replication strategy in clou d storage. *The Journal of Supercomputing*, *76*(7), 4838–4856. https://doi.org/10.1007/s1 1227-018-2647-4

[37] Jafari Navimipour, N., Rahmani, A. M., Habibizad Navin, A., & Hosseinzadeh, M. (2015). Expert Cloud: A Cloud-based framework to share the knowledge and skills of h uman resources. *Computers in Human Behavior*, *46*, 57–74. https://doi.org/10.1016/j.ch b.2015.01.001

[38] Janpet, J., & Yean-Fu Wen. (2013). Reliable and Available Data Replication Planni ng for Cloud Storage. *2013 IEEE 27th International Conference on Advanced Informati on Networking and Applications (AINA)*, 772–779. https://doi.org/10.1109/AINA.2013. 125

[39] Jayasinghe, M., Tari, Z., Zeephongsekul, P., & Zomaya, A. Y. (2011). Task assign ment in multiple server farms using preemptive migration and flow control. *Journal of Parallel and Distributed Computing*, *71*(12), 1608–1621. https://doi.org/10.1016/j.jpdc. 2011.07.001

[40] Jhawar, R., & Piuri, V. (2017). Fault tolerance and resilience in cloud computing e nvironments. In *Computer and information security handbook* (pp. 165-181). Morgan K aufmann. https://doi.org/10.1016/B978-0-12-803843-7.00009-0

[41] Jiang, J., Li, Y., Hong, S. H., Xu, A., & Wang, K. (2018). A time-sensitive network ing (TSN) simulation model based on OMNET++. In *2018 IEEE International Confere nce on Mechatronics and Automation (ICMA)*, 643-648. https://doi.org/10.1109/ICMA. 2018.8484302

[42] Jiang, W., Xie, H., Zhou, X., Fang, L., & Wang, J. (2017). Performance Analysis a nd Improvement of Replica Selection Algorithms for Key-Value Stores. *2017 IEEE 10t h International Conference on Cloud Computing (CLOUD)*, 786-789. https://doi.org/10. 1109/CLOUD.2017.115

[43] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., & Vahi, K. (2013). Characterizing and profiling scientific workflows. *Future Generation Computer System*

s, *29*(3), 682–692. https://doi.org/10.1016/j.future.2012.08.015

[44] Kaseb, M. R., Khafagy, M. H., Ali, I. A., & Saad, E. M. (2019). An improved tech nique for increasing availability in Big Data replication. *Future Generation Computer S ystems*, *91*, 493–505. https://doi.org/10.1016/j.future.2018.08.015

[45] Kaul, S., Sood, K., & Jain, A. (2017). Cloud Computing and its Emerging Need: A dvantages and Issues. *International Journal of Advanced Research in Computer Science* , *8*(3), 618-624.

[46] Khalajzadeh, H., Dong Yuan, Grundy, J., & Yun Yang. (2016). Improving Cloud-Based Online Social Network Data Placement and Replication. *2016 IEEE 9th Internati onal Conference on Cloud Computing (CLOUD)*, 678–685. https://doi.org/10.1109/CL OUD.2016.0095

[47] Khaldi, M., Rebbah, M., Meftah, B., & Smail, O. (2020). Fault tolerance for a scien tific workflow system in a Cloud computing environment. *International Journal of Com puters & Applications*, *42*(7), 705–714. https://doi.org/10.1080/1206212X.2019.164765

[48] Khelifa, A., Hamrouni, T., Mokadem, R., & Charrada, F. B. (2020). SLA-aware tas k scheduling and data replication for enhancing provider profit in clouds. *Procedia Com puter Science*, *176*, 3143–3152. https://doi.org/10.1016/j.procs.2020.09.174

[49] Kim, W., Lee, J. H., Hong, C., Han, C., Lee, H., & Jang, B. (2012). An innovative method for data and software integration in SaaS. *Computers & Mathematics with Appli cations (1987)*, *64*(5), 1252–1258. https://doi.org/10.1016/j.camwa.2012.03.069

[50] Ko, A. C. & Zaw, W. T. (2014). Fault Tolerant Erasure Coded Replication for HDF S Based Cloud Storage. *2014 IEEE Fourth International Conference on Big Data and C loud Computing*, 104–109. https://doi.org/10.1109/BDCloud.2014.69

[51] Kumar, P., & Kumar, R. (2019). Issues and Challenges of Load Balancing Techniq ues in Cloud Computing: A Survey. *ACM Computing Surveys*, *51*(6), 1–35. https://doi.o

rg/10.1145/3281010

[52] Laribi, I. & Didi, F. (2014). Studies and Analysis of Cloud Computing Solution. *Jo urnal of Computer Science and Control Systems*, *7*(2), 19–22.

[53] Li, C., & Li, L. (2013). Efficient resource allocation for optimizing objectives of cl oud users, IaaS provider and SaaS provider in cloud environment. *The Journal of Super computing*, *65*(2), 866–885. https://doi.org/10.1007/s11227-013-0869-z

[54] Li, C., Wang, C., Tang, H., & Luo, Y. (2019). Scalable and dynamic replica consist ency maintenance for edge-cloud system. *Future Generation Computer Systems*, *101*, 5 90–604. https://doi.org/10.1016/j.future.2019.05.014

[55] Li, C., Zhang, Y., & Luo, Y. (2020). Adaptive Replica Creation and Selection Strat egies for Latency-Aware Application in Collaborative Edge-Cloud System. *Computer J ournal*, *63*(9), 1338–1354. https://doi.org/10.1093/comjnl/bxz070

[56] Li, W., Yang, Y., & Yuan, D. (2011). A Novel Cost-Effective Dynamic Data Repli cation Strategy for Reliability in Cloud Data Centres. *2011 IEEE Ninth International Co nference on Dependable, Autonomic and Secure Computing*, 496–502. https://doi.org/10 .1109/DASC.2011.95

[57] Li, W., Yang, Y., & Yuan, D. (2016). Ensuring Cloud Data Reliability with Minim um Replication by Proactive Replica Checking. *IEEE Transactions on Computers*, *65*(5), 1494–1506. https://doi.org/10.1109/TC.2015.2451644

[58] Limam, S., Mokadem, R., & Belalem, G. (2019). Data replication strategy with sati sfaction of availability, performance and tenant budget requirements. *Cluster Computin g*, *22*(4), 1199–1210. https://doi.org/10.1007/s10586-018-02899-6

[59] Lin, J. W., Chen, C. H., & Chang, J. M. (2013). QoS-Aware Data Replication for D ata-Intensive Applications in Cloud Computing Systems. *IEEE Transactions on Cloud Computing*, *1*(1), 101–115. https://doi.org/10.1109/TCC.2013.1

[60] Lin, Y., Chen, Y., Wang, G., & Deng, B. (2010). Rigel: A Scalable and Lightweigh t Replica Selection Service for Replicated Distributed File System. *2010 10th IEEE/AC M International Conference on Cluster, Cloud and Grid Computing*, 581–582. https://d oi.org/10.1109/CCGRID.2010.51

[61] Liu, G., Shen, H., & Chandler, H. (2016). Selective Data Replication for Online So cial Networks with Distributed Datacenters. *IEEE Transactions on Parallel and Distrib uted Systems*, *27*(8), 2377–2393. https://doi.org/10.1109/TPDS.2015.2485266

[62] Liu, J., Wang, S., Zhou, A., Kumar, S. A. P., Yang, F., & Buyya, R. (2018). Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability. *IEEE Trans actions on Cloud Computing*, *6*(4), 1191–1202. https://doi.org/10.1109/TCC.2016.2567 392

[63] Liu, L., Fan, Q., & Buyya, R. (2018). A Deadline-Constrained Multi-Objective Tas k Scheduling Algorithm in Mobile Cloud Environments. *IEEE Access*, *6*, 52982–52996. https://doi.org/10.1109/ACCESS.2018.2870915

[64] Liu, L., Yang, Y., Wang, H., Tan, Z. & Li, C. (2017). A group based genetic algorit hm data replica placement strategy for scientific workflow. *2017 IEEE/ACIS 16th Intern ational Conference on Computer and Information Science (ICIS)*, 459–464. https://doi.o rg/10.1109/ICIS.2017.7960036

[65] Long, S. Q., Zhao, Y. L., & Chen, W. (2014). MORM: A Multi-objective Optimize d Replication Management strategy for cloud storage cluster. *Journal of Systems Archite cture*, *60*(2), 234–244. https://doi.org/10.1016/j.sysarc.2013.11.012

[66] Lynn, T., Morrison, J. P., & Kenny, D. (2018). *Heterogeneity, High Performance C omputing, Self-Organization and the Cloud* (1st ed. 2018.). Springer International Publis hing. https://doi.org/10.1007/978-3-319-76038-4

[67] Mansouri, N. (2016). Adaptive data replication strategy in cloud computing for per

formance improvement. *Frontiers of Computer Science*, *10*(5), 925–935. https://doi.org/10.1007/s11704-016-5182-6

[68] Mansouri, N., & Javidi, M. M. (2018). A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers. *The Journal of Supercomputing*, *74*(10), 5349–5372. https://doi.org/10.1007/s11227-018-2427-1

[69] Mansouri, N., & Javidi, M. M. (2020). A review of data replication based on meta-heuristics approach in cloud computing and data grid. *Soft Computing (Berlin, Germany)*, *24*(19), 14503–14530. https://doi.org/10.1007/s00500-020-04802-1

[70] Mansouri, N., Javidi, M. M., & Zade, B. M. H. (2021). Hierarchical data replication strategy to improve performance in cloud computing. *Frontiers of Computer Science*, *15*(2), 1-17. https://doi.org/10.1007/s11704-019-9099-8

[71] Mansouri, Y., & Buyya, R. (2019). Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers. *Journal of Parallel and Distributed Computing*, *126*, 121–133. https://doi.org/10.1016/j.jpdc.2018.12.003

[72] Mansouri, Y., Toosi, A., & Buyya, R. (2018). Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions. *ACM Computing Surveys*, *50*(6), 1–51. https://doi.org/10.1145/3136623

[73] Manvi, S. S., & Krishna Shyam, G. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications*, *41*, 424–440. https://doi.org/10.1016/j.jnca.2013.10.004

[74] Marahatta, A., Wang, Y., Zhang, F., Sangaiah, A. K., Tyagi, S. K. S., & Liu, Z. (2019). Energy-Aware Fault-Tolerant Dynamic Task Scheduling Scheme for Virtualized Cloud Data Centers. *Mobile Networks and Applications*, *24*(3), 1063–1077. https://doi.org/10.1007/s11036-018-1062-7

[75] Maresova, P., Sobeslav, V., & Krejcar, O. (2017). Cost-benefit analysis - evaluatio

n model of cloud computing deployment for use in companies. *Applied Economics*, *49*(6), 521–533. https://doi.org/10.1080/00036846.2016.1200188

[76] Marinescu, D. C. (2013). *Cloud computing theory and practice* (1st ed.). Morgan K aufmann

[77] Matani, A., Naji, H. R., & Motallebi, H. (2020). A Fault-Tolerant Workflow Sched uling Algorithm for Grid with Near-Optimal Redundancy. *Journal of Grid Computing* , *18*(3), 377–394. https://doi.org/10.1007/s10723-020-09522-2

[78] Milani, B. A., & Navimipour, N. J. (2016). A comprehensive review of the data rep lication techniques in the cloud environments: Major trends and future directions. *Journ al of Network and Computer Applications*, *64*, 229–238. https://doi.org/10.1016/j.jnca.2 016.02.005

[79] Modisane, P., & Jokonya, O. (2021). Evaluating the benefits of Cloud Computing i n Small, Medium and Micro-sized Enterprises (SMMEs). *Procedia Computer Science* , *181*, 784–792. https://doi.org/10.1016/j.procs.2021.01.231

[80] Mokadem, R., & Hameurlain, A. (2020). A data replication strategy with tenant per formance and provider economic profit guarantees in Cloud data centers. *The Journal of Systems and Software*, *159*, 110447. https://doi.org/10.1016/j.jss.2019.110447

[81] Mousavi Nik, S. S., Naghibzadeh, M., & Sedaghat, Y. (2021). Task replication to i mprove the reliability of running workflows on the cloud. *Cluster Computing*, *24*(1), 34 3–359. https://doi.org/10.1007/s10586-020-03109-y

[82] Mseddi, A., Salahuddin, M. A., Zhani, M. F., Elbiaze, H., & Glitho, R. H. (2021). Efficient Replica Migration Scheme for Distributed Cloud Storage Systems. *IEEE Tran sactions on Cloud Computing*, *9*(1), 155–167. https://doi.org/10.1109/TCC.2018.28587 92

[83] Mukwevho, M. A., & Celik, T. (2021). Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems. *IEEE Transactions on Services Computing*, *14*(2) , 589–605. https://doi.org/10.1109/TSC.2018.2816644

[84] Nazari Cheraghlou, M., Khadem-Zadeh, A., & Haghparast, M. (2016). A survey of fault tolerance architecture in cloud computing. *Journal of Network and Computer Appli cations*, *61*, 81–92. https://doi.org/10.1016/j.jnca.2015.10.004

[85] Ouda, G. K., & Yas, Q. M. (2021). Design of Cloud Computing for Educational Ce nters Using Private Cloud Computing: A Case Study. *Journal of Physics. Conference Se ries*, *1804*(1), 12119–. https://doi.org/10.1088/1742-6596/1804/1/012119

[86] Oujezsky, V., & Horvath, T. (2016). Case study and comparison of SimPy 3 and O MNeT++ Simulation. *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, 15–19. https://doi.org/10.1109/TSP.2016.7760821

[87] Peniak, P. (2014). MODEL OF INFRASTRUCTURE PROVISIONING IN IAAS CLOUDS. *Annals of Faculty Engineering Hunedoara*, *12*(3), 189–194.

[88] Phaphoom, N., Wang, X., Samuel, S., Helmer, S., & Abrahamsson, P. (2015). A su rvey study on major technical barriers affecting the decision to adopt cloud services. *Th e Journal of Systems and Software*, *103*, 167–181. https://doi.org/10.1016/j.jss.2015.02. 002

[89] Prathiba, S., & Sowvarnica, S. (2017). Survey of failures and fault tolerance in clou d. *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*, 169–172. https://doi.org/10.1109/ICCCT2.2017.7972271

[90] Rajalakshmi, A., Vijayakumar, D., & Srinivasagan, K. G. (2014). An improved dyn amic data replica selection and placement in cloud. *2014 International Conference on R ecent Trends in Information Technology*, 1–6. https://doi.org/10.1109/ICRTIT.2014.699 6180

[91] Rasool, Q., Li, J., Oreku, G. S., Zhang, S., & Yang, D. (2008). A load balancing replica placement strategy in Data Grid. *2008 Third International Conference on Digital Information Management*, 751-756. http://doi.org/10.1109/ICDIM.2008.4746731

[92] Ray, B. K., Saha, A., Khatua, S., & Roy, S. (2020). Proactive Fault-Tolerance Technique to Enhance Reliability of Cloud Service in Cloud Federation Environment. *IEEE Transactions on Cloud Computing*, 1–1. https://doi.org/10.1109/TCC.2020.2968522

[93] Reinsel, D., Gantz, J., & Rydning, J. (2018). *The Digitization of the World*. http://book.itep.ru/depository/dig_economy/idc-seagate-dataage-whitepaper.pdf

[94] Samadi, Y., Zbakh, M., & Tadonki, C. (2018). E-HEFT: enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing. *2018 International Conference on High Performance Computing & Simulation (HPCS)*, 601-609. https://doi.org/10.1109/HPCS.2018.00100

[95] Sampaio, A. M., & Barbosa, J. G. (2018). A comparative cost analysis of fault-tolerance mechanisms for availability on the cloud. *Sustainable Computing Informatics and Systems*, *19*, 315–323. https://doi.org/10.1016/j.suscom.2017.11.006

[96] Sandokji, S., & Eassa, F. (2019). Dynamic Variant Rank HEFT Task Scheduling Algorithm Toward Exascle Computing. *Procedia Computer Science*, *163*, 482–493. https://doi.org/10.1016/j.procs.2019.12.131

[97] Sarvabhatla, M., Konda, S., Vorugunti, C. S., & Babu, M. M. N. (2017). A Dynamic and Energy Efficient Greedy Scheduling Algorithm for Cloud Data Centers. *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 47–52 https://doi.org/10.1109/CCEM.2017.9

[98] Schwarzkopf, M., Murray, D. G., & Hand, S. (2012). The seven deadly sins of cloud computing research. *4th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 12)*, 1.

[99] Setlur, A. R., Nirmala, S. J., Singh, H. S., & Khoriya, S. (2020). An efficient fault t olerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. *Journal of Parallel and Distributed Computing*, *136*, 14–28. https://doi.org/10.1016/j.jpdc.2019.09.004

[100] Shakkeera, L., & Tamilselvan, L. (2016). QoS and load balancing aware task sche duling framework for mobile cloud computing environment. *International Journal of W ireless and Mobile Computing*. *10*(4), 309–316. https://doi.org/10.1504/IJWMC.2016.078201

[101] Shorfuzzaman, M., Graham, P., & Eskicioglu, R. (2010). Adaptive popularity-dri ven replica placement in hierarchical data grids. *Journal of Supercomputing*, *51*(3), 374 –392. https://doi.org/10.1007/s11227-009-0371-9

[102] Shwe, T., & Aritsugi, M. (2018). PRTuner: Proactive-Reactive Re-Replication Tu ning in HDFS-based Cloud Data Center. *IEEE Cloud Computing*, *5*(6), 48–57. https://doi.org/10.1109/MCC.2018.064181120

[103] Sivagami, V., & Easwarakumar, K. (2019). An Improved Dynamic Fault Toleran t Management Algorithm during VM migration in Cloud Data Center. *Future Generatio n Computer Systems*, *98*, 35–43. https://doi.org/10.1016/j.future.2018.11.002

[104] Song, C., Kim, S. W., & Sohn, Y. (2020). Acceptance of public cloud storage serv ices in South Korea: A multi-group analysis. *International Journal of Information Mana gement*, *51*, 102035. https://doi.org/10.1016/j.ijinfomgt.2019.11.003

[105] Stergiou, C., Psannis, K. E., Kim, B. G., & Gupta, B. (2018). Secure integration o f IoT and Cloud Computing. *Future Generation Computer Systems*, *78*, 964–975. https://doi.org/10.1016/j.future.2016.11.031

[106] Subasi, O., Yalcin, G., Zyulkyarov, F., Unsal, O., & Labarta, J. (2017). Designing and Modelling Selective Replication for Fault-tolerant HPC Applications. *Proceedings*

of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing , 452–457. https://doi.org/10.1109/CCGRID.2017.40

[107] Sun, D. W., Chang, G. R., Gao, S., Jin L. Z., & Wang X. W. (2012). Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computin g Environments. *Journal of Computer Science and Technology*, *27*(2), 256–272. https:// doi.org/10.1007/s11390-012-1221-4

[108] Sun, S., Yao, W., Qiao, B., Zong, M., He, X., & Li, X. (2019). RRSD: A file repli cation method for ensuring data reliability and reducing storage consumption in a dyna mic Cloud-P2P environment. *Future Generation Computer Systems*, *100*, 844–858. http s://doi.org/10.1016/j.future.2019.05.054

[109] Sun, Z., Shen, J. & Yong, J. (2011). DeDu: Building a deduplication storage syste m over cloud computing. *Proceedings of the 2011 15th International Conference on Co mputer Supported Cooperative Work in Design (CSCWD)*, 348–355. https://doi.org/10.1 109/CSCWD.2011.5960097

[110] Sun, Z., Shen, J., & Yong, J. (2013). A novel approach to data deduplication over the engineering-oriented cloud systems. *Integrated Computer-Aided Engineering*, *20*(1), 45–57. https://doi.org/10.3233/ICA-120418

[111] Thorsen, S. (2015). *Replica selection in Apache Cassandra: Reducing the tail late ncy for reads using the C3 algorithm*. Digitala Vetenskapliga Arkivet. https://www.diva -portal.org/smash/record.jsf?pid=diva2%3A827372&dswid=7119

[112] TOLE. A. A. (2015). Cloud Computing and Business Intelligence. *Database Syste ms Journal*, *V*(4), 49–58.

[113] Tomas, L., Kokkinos, P., Anagnostopoulos, V., Feder, O., Kyriazis, D., Meth, K., Varvarigos, E., & Varvarigou, T. (2020). Disaster Recovery Layer for Distributed Open Stack Deployments. *IEEE Transactions on Cloud Computing*, *8*(1), 112–123. https://doi

.org/10.1109/TCC.2017.2745560

[114] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, *13*(3), 260–274. https://doi.org/10.1109/71.993206

[115] Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., & Bora, S. (2016). A Performance and Profit Oriented Data Replication Strategy for Cloud Systems. *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, 780–787. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0125

[116] Tudoran, R., Costan, A., & Antoniu, G. (2016). OverFlow: Multi-Site Aware Big Data Management for Scientific Workflows on Clouds. *IEEE Transactions on Cloud Computing*, *4*(1), 76–89. https://doi.org/10.1109/TCC.2015.2440254

[117] Vahi, K., Rynge, M., Juve, G., Mayani, R., & Deelman, E. (2013). Rethinking data management for big data scientific workflows. *2013 IEEE International Conference on Big Data*, 27–35. https://doi.org/10.1109/BigData.2013.6691724

[118] Vardhan, M., Goel, A., Verma, A., & Kushwaha, D. S. (2012). A Dynamic Fault Tolerant Threshold based Replication Mechanism in Distributed Environment. *Procedia Technology*, *6*, 188–195. https://doi.org/10.1016/j.protcy.2012.10.023

[119] Verma, A., & Kaushal, S. (2015). Cost-Time Efficient Scheduling Plan for Executing Workflows in the Cloud. *Journal of Grid Computing*, *13*(4), 495–506. https://doi.org/10.1007/s10723-015-9344-9

[120] Wan, C., Wang, C., & Pei, J. (2012). A QoS-awared scientific workflow scheduling schema in cloud computing. *2012 IEEE International Conference on Information Sci*

ence and Technology, 634–639. https://doi.org/10.1109/ICIST.2012.6221722

[121] Wang, C., Lu, Z., Wu, Z., Wu, J., & Huang, S. (2017). Optimizing Multi-Cloud C DN Deployment and Scheduling Strategies Using Big Data Analysis. *2017 IEEE Intern ational Conference on Services Computing (SCC)*, 273–280. https://doi.org/10.1109/SC C.2017.42

[122] Wang, M., Zhang, J., Dong, F. & Luo, J. (2014). Data Placement and Task Sched uling Optimization for Data Intensive Scientific Workflow in Multiple Data Centers En vironment. *2014 Second International Conference on Advanced Cloud and Big Data*, 77 –84. https://doi.org/10.1109/CBD.2014.19

[123] Wang, P., Gao, R. X., & Fan, Z. (2015). Cloud Computing for Cloud Manufacturi ng: Benefits and Limitations. *Journal of Manufacturing Science and Engineering*, *137*(4), 40901. https://doi.org/10.1115/1.4030209

[124] Wang, Z., Zheng, W., Chen, P., Ma, Y., Xia, Y., Liu, W., Li, X., & Guo, K. (2020). A Novel Coevolutionary Approach to Reliability Guaranteed Multi-Workflow Schedu ling upon Edge Computing Infrastructures. *Security and Communication Networks*, 202 0. https://doi.org/10.1155/2020/6697640

[125] Wei, Q., Veeravalli, B., Gong, B., Zeng, L., & Feng, D. (2010). CDRM: A Cost-E ffective Dynamic Replication Management Scheme for Cloud Storage Cluster. *2010 IE EE International Conference on Cluster Computing*, 188–196. https://doi.org/10.1109/C LUSTER.2010.24

[126] Welsh, T., & Benkhelifa, E. (2020). On Resilience in Cloud Computing: A Surve y of Techniques across the Cloud Domain. *ACM Computing Surveys*, *53*(3), 1–36. https: //doi.org/10.1145/3388922

[127] Wu, X. (2016). Data Sets Replicas Placements Strategy from Cost-Effective View in the Cloud. *Scientific Programming*, *2016*, 1–13. https://doi.org/10.1155/2016/14967

14

[128] Xia, Q., Xu, Z., Liang, W., Yu, S., Guo, S., & Zomaya, A. Y. (2019). Efficient Data Placement and Replication for QoS-Aware Approximate Query Evaluation of Big Data Analytics. *IEEE Transactions on Parallel and Distributed Systems*, *30*(12), 2677–2691. https://doi.org/10.1109/TPDS.2019.2921337

[129] Xu, H., Liu, W., Shu, G., & Li, J. (2016). Location-Aware Data Block Allocation Strategy for HDFS-Based Applications in the Cloud. *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 252–259. https://doi.org/10.1109/CLOUD.2016.0042

[130] Xu, X., Mo, R., Dai, F., Lin, W., Wan, S., & Dou, W. (2020). Dynamic Resource Provisioning with Fault Tolerance for Data-Intensive Meteorological Workflows in Cloud. *IEEE Transactions on Industrial Informatics*, *16*(9), 6172–6181. https://doi.org/10.1109/TII.2019.2959258

[131] Xue, M., Shen, J., & Guo, X. (2016). Two phase enhancing replica selection in cloud storage system. *2016 35th Chinese Control Conference (CCC)*, 5255–5260. https://doi.org/10.1109/ChiCC.2016.7554173

[132] Yang, J., Meng, Q., Wang, S., Li, D., Huang, T., & Dou, W. (2016). Energy-Aware Tasks Scheduling with Deadline-constrained in Clouds. *2016 International Conference on Advanced Cloud and Big Data (CBD)*, 116–121. https://doi.org/10.1109/CBD.2016.030

[133] Yao, G., Ding, Y., & Hao, K. (2017). Using Imbalance Characteristic for Fault-Tolerant Workflow Scheduling in Cloud Systems. *IEEE Transactions on Parallel and Distributed Systems*, *28*(12), 3671–3683. https://doi.org/10.1109/TPDS.2017.2687923

[134] Yao, G., Ren, Q., Li, X., Zhao, S., & Ruiz, R. (2020). A hybrid fault-tolerant scheduling for deadline-constrained tasks in Cloud systems. *IEEE Transactions on Services*

Computing, 1–1. https://doi.org/10.1109/TSC.2020.2992928

[135] Serhane, Y., Sekkaki, A., Benzidane, K., & Abid, M. (2020). Cost Effective Clou d Storage Interoperability Between Public Cloud Platforms. *International Journal of Co mmunication Networks and Information Security*, *12*(3), 440–449.

[136] Ye, Z., Li, S., & Zhou, J. (2014). A two-layer geo-cloud based dynamic replica cr eation strategy, *Applied Mathematics & Information Sciences*, *8*(1), 431-439. https://dx. doi.org/10.12785/amis/080154

[137] Yi, S., Andrzejak, A., & Kondo, D. (2012). Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances. *IEEE Transactions on Services Compu ting*, *5*(4), 512–524. https://doi.org/10.1109/TSC.2011.44

[138] Yuan, D., Cui, L., & Liu, X. (2014). Cloud Data Management for Scientific Work flows: Research Issues, Methodologies, and State-of-the-Art. *2014 10th International C onference on Semantics, Knowledge and Grids*, 21–28. https://doi.org/10.1109/SKG.20 14.37

[139] Yuan, D., Yang, Y., Liu, X., & Chen, J. (2010). A data placement strategy in scie ntific cloud workflows. *Future Generation Computer Systems*, *26*(8), 1200–1214. https: //doi.org/10.1016/j.future.2010.02.004

[140] Zhang, Q., Li, S., Li, Z., Xing, Y., Yang, Z., & Dai, Y. (2015). CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability. *IEEE Transactions on Cloud Computing*, *3*(3), 372–386. https://doi.org/10.1109/TCC.2015.2417534

[141] Zhao, J., Xiang, Y., Lan, T., Huang, H. H., & Subramaniam, S. (2017). Elastic Re liability Optimization Through Peer-to-Peer Checkpointing in Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems*, *28*(2), 491–502. https://doi.org/10.11 09/TPDS.2016.2571281

[142] Zhao, P., Sun, X., Shang, J., Lin, J., Dong, M., & Li, B. (2019). A Dynamic Conv

ergent Replica Selection Strategy Based on Cloud Storage. *2019 International Conferen ce on Artificial Intelligence and Advanced Manufacturing (AIAM)*, 473–478. https://doi. org/10.1109/AIAM48774.2019.00100

[143] Zhao, Y., Li, C., Li, L., & Zhang, P. (2017). Dynamic replica creation strategy bas ed on file heat and node load in hybrid cloud. *2017 19th International Conference on Ad vanced Communication Technology (ICACT)*, 213–220. https://doi.org/10.23919/ICAC T.2017.7890086

[144] Zheng, W., & Sakellariou, R. (2013). Budget-Deadline Constrained Workflow Pla nning for Admission Control. *Journal of Grid Computing*, *11*(4), 633–651. https://doi.or g/10.1007/s10723-013-9257-4

[145] Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R. N., Lyu, M. R., & Buyya, R. (2017). Cloud Service Reliability Enhancement via Virtual Machine Placeme nt Optimization. *IEEE Transactions on Services Computing*, *10*(6), 902–913. https://doi. org/10.1109/TSC.2016.2519898

[146] Zhou, A., Wang, S., Hsu, C. H., Sun, Q., & Yang, F. (2016). Task rescheduling op timization to minimize network resource consumption. *Multimedia Tools and Applicatio ns*, *75*(20), 12901–12917. https://doi.org/10.1007/s11042-015-2549-x

[147] Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L. T., & Liu, L. (2016). Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds. *IEEE Transactions on Parallel and Distributed Systems*, *27*(12), 35 01–3517. https://doi.org/10.1109/TPDS.2016.2543731

Appendices

Appendix 1 – Notations and Contribution Summary

The notations in the descriptions and equations of Chapter 4 are listed in Table A1.1.

Table A1.1 The notations in the descriptions and equations of Chapter 4

Symbols	Explanation
$J(d_i)$	The set of tasks which access the data d_i
$Dep(d_i, d_k)$	The data dependency of the data d_i to the data d_k
$Dep(d_k, d_i)$	The data dependency of the data d_k to the data d_i
$AF(d_i)$	The access frequency of the data d_i
$AT(d_i)$	The number of access times of the data d_i
$AI(d_i)$	The access time interval to the data d_i
ω	The data dependency threshold parameter
$\emptyset$	The access frequency threshold parameter
$Size(d_i)$	The data size of the data d_i
$ASS(dc)$	The available storage capacity in the data centre dc
$TC(d_i)$	The total cost of the data d_i
μ	The determinant variable for the data storage cost calculation
$DSC(d_i)$	The data storage cost of the data d_i
$DTC(d_i)$	The data transfer cost of the data d_i
$ST(d_i)^{dc}$	The data storage time interval of the data d_i stored in the data centre dc
$SP(dc)$	The data storage price of the data centre dc
α	The transfer cost ratio

β	The determinant variable for the data transfer cost calculation
TC	The overall total cost
$W\text{-}DCD(d_i)$	The Within-DataCentre Data Dependency of the data d_i
$B\text{-}DCD(d_i)$	The Between-DataCentre Data Dependency of the data d_i
$DCD(dc, d_i)$	A function to calculate $W\text{-}DCD(d_i)$ and $B\text{-}DCD(d_i)$ for the data d_i in the data centre dc
$DepCompare(d_i)$	A function to compare between $W\text{-}DCD(d_i)$ and $B\text{-}DCD(d_i)$ for the data d_i
$Num(D)$	The total amount of data in D
AF_{total}	The sum of the access frequency of all data
AF_{avg}	The average access frequency of all data
$AFCalculation()$	A function to calculate the value of AF_{total} and AF_{avg}
$AFCompare(d_i)$	A function to compare the value between $AF(d_i)$ and $\emptyset * AF_{avg}$
$TC_{initial}$	The total cost when there is no replica creation happened
$TC_{current}$	The current total cost value
$NR_{current}$	The current number of replicas
CE	An evaluation parameter to evaluate the cost efficiency in terms of cost reduction per replica
$NL(dc)$	The network latency of the data centre dc
$BC(dc)$	The bandwidth consumption of the data centre dc
J^{dc}	The set of tasks accessing the data centre dc
$Size(j^{dc})$	The size of the data requested by the task $j^{dc} \in J^{dc}$ at the data centre dc

$Len(j^{dc})$	The task execution duration of the task $j^{dc} \in J^{dc}$ at the data centre dc
$ACLen(d)$	The maximum time length of the data d being accessed by its relevant tasks
$AB(dc)$	The available bandwidth of the data centre dc
$maxB(dc)$	The maximum bandwidth of the data centre dc
$ER(dc)$	The error rate of the data centre dc
W_{AB}^{dc}	The weight of the available bandwidth metric of the data centre dc
W_{NL}^{dc}	The weight of the network latency metric of the data centre dc
W_{ER}^{dc}	The weight of the error rate metric of the data centre dc
$FW(dc)$	The final weight of the data centre dc
NC_{AB}^{dc}	The normalisation component of the available bandwidth metric of the data centre dc
NC_{NL}^{dc}	The normalisation component of the network latency metric of the data centre dc
NC_{ER}^{dc}	The normalisation component of the error rate metric of the data centre dc

The notations in the descriptions and equations of Chapter 5 are listed in Table A1.2.

Table A1.2 The notations in the descriptions and equations of Chapter 5

Symbols	**Explanation**
$DEAD(j)$	The task hard deadline of the task j
$Len(j)$	The task execution duration of the task j
$PA(j)$	The past processing time of the task j

$IC(j)$	The internodal communication delay of the task j
$IS(j)$	The input scheduling delay of the task j
$UR(j)$	The task urgency value of the task j
$PRO(j)$	The task operation profit of the task j
$U(j)$	The task utility of the task j
$U_{UR}(j)$	The utility value of the task urgency of the task j
$U_{PRO}(j)$	The utility value of the task operation profit of the task j
W_{UR}	The weight of the task urgency
W_{PRO}	The weight of the task operation profit
$TRU(dc)$	The task rescue utility of a faulty data centre dc
Resubmission()	The task resubmission function
Migration()	The task migration function
ϑ	A variable parameter to judge the task rescue situation
$R(j)$	The resource requirement of the task j
T_0	The current time point
T_{Late}	The latest deadline time point of the tasks in J
$Count(ET(j^{dc}))$	A function to count the number of eligible time slots of the task j at the data centre dc
$ET(j^{dc})$	A set of eligible time slots for the task j at the data centre dc
$et(j^{dc})_p$	The pth eligible time slot in $ET(j^{dc})$
$LS(et(j^{dc})_p)$	The time slot length similarity of $et(j^{dc})_p$
$Len(et(j^{dc})_p)$	The time slot length of $et(j^{dc})_p$
$MR(et(j^{dc})_p)$	The minimum available resource of $et(j^{dc})_p$

W_{LS}	The weight of the time slot length similarity
W_{MR}	The weight of the minimum available resource
$OE(j)$	The optimal eligible time slot of the task j
$rank\left(LS\left(et(j^{dc})_p\right)\right)$	The ranking value of the time slot length similarity of $et(j^{dc})_p$
$rank\left(MR\left(et(j^{dc})_p\right)\right)$	The ranking value of the minimum available resource of $et(j^{dc})_p$
$rank\left(et(j^{dc})_p\right)$	The ranking value of the pth eligible time slot in $ET(j^{dc})$

The notations in the descriptions and equations of Chapter 6 are listed in Table A1.3.

Table A1.3 The notations in the descriptions and equations of Chapter 6

Symbols	**Explanation**
Nod_{entry}	The entry task in the workflow
Nod_{exit}	The exit task in the workflow
$PR(Nod_p)$	The PageRank value for the node Nod_p
$succ(Nod_p)$	The set of successors of Nod_p
$L(Nod_q)$	The number of the outbound nodes of Nod_q
ρ	The total number of nodes in the workflow
δ	A damping factor to handle the probability of the task termination
$Rank_u(Nod_p)$	The upward rank value of a task Nod_p
$\gamma(G)$	The urgency balancing coefficient for a workflow instance G
$UR(G)$	The urgency value of the workflow application G
$\sigma(G)$	The complexity balancing coefficient for a workflow instance G

$CountNod(G)$	A count function to count the number of nodes in the workflow instance G
$CountEdge(G)$	A count function to count the number of edges in the workflow instance G
$Num(Nod)$	The total number of nodes in the cloud environment
$Num(edge)$	The total number of edges in the cloud environment
$FinRank_u(Nod_p)$	The final upward rank value of the node Nod_p

The notations in the pseudocodes are listed in Table A1.4.

Table A1.4 The notations in the pseudocodes

Symbols	**Explanation**
$tl[]$	The array for storing the location information
$dl[]$	The array for storing the location information
$rec[]$	The array for storing the recommended value $\emptyset$
$eva[]$	The array for storing the cost efficiency evaluation parameter CE
$OptRoute$	The optimal data access route including the required data information, the target data centre information and the relevant task information
$rd[]$	The array for listing the required data
$rr[]$	The array for storing the replica-ready data centre information
$fw[]$	The array for listing the final weight of the data centres in $rr[]$
$qual[]$	The array for storing the information of the qualified data centres

$elig[]$	The array for collecting the eligible data centres
$ab[]$	The array for storing the available bandwidth information of the data centres in $elig[]$
$ranklist[]$	The array for storing the ranking value of the tasks
$FTResult$	The fault handling solution including the task resubmission destination and task migration destination information
dc_{res}	The task resubmission destination
dc_{mig}	The task migration destination
$resdes[]$	The array for storing the destination information for task resubmission
$mov[]$	The array for storing a group of migratable tasks
$migrr[]$	The array for storing the replica-ready data centres for the migratable task
$eligmig[]$	The array for storing the eligible replica-ready data centres for migrating the migratable task
$fj[]$	The array for storing the tasks at the fault location
$pts[]$	The array for storing the probable time slots
$crj[]$	The array for storing the current-running tasks in $rr[]$
$prj[]$	The array for storing the probable-release tasks

The contribution summary of the six proposed strategies is shown in Table A1.5, including the context locations of the six proposed strategies, the research problems of the six proposed strategies, the novelty of the six proposed strategies and the optimisation objectives of the six proposed strategies.

Table A1.5 The contribution summary of the six proposed strategies

Context Location	Research Problems	Novelty	Optimisation Objectives
Section 4.1	• Lack of the joint consideration of external data attributes and internal data attributes when making the replica creation decision	• The data classification for constraining the replica creation • The joint consideration of the external data attribute (access frequency) and the internal data attribute (data dependency)	Total cost reduction
Section 4.2	• The insufficient consideration of the cloud map when analysing the data relationship	• The analysis of the data dependency inside data centre and outside data centre • Threshold-based eligible data candidate pool for replica creation • Recommended	The optimal cost reduction per replica

		value to achieve the optimal cost reduction per replica with balancing between the total cost and the number of replicas	
Section 4.3	• Network overloading problems because of increased number of concurrent-running instances and heavy data access needs • Lack of the consideration of the impacts among multiple concurrent-running instances under limited network	• *Min-Max* normalisation-based replica selection method with the joint consideration of different network performance measurements • Nested replica selection strategy with a replica re-creation mechanism to collaboratively guide the data	The number of concurrent-running instances increase; More balanced network load

	capability	access route	
Section 5.1	• Resource contention problems because of the task resubmission and task migration operations • Failure to meet the task deadline when rescuing tasks	• Joint consideration of the resource load capacity and the task attributes • Utility-based task priority assignment system • A concession mechanism for task allocation to appropriate data centres	Task resilience ratio increase; Task rescue utility increase; Task operation profit increase
Section 5.2	• Selecting the first available server to enable early task completion might not be the optimal solution in term of cloud resiliency when rescuing tasks • Selecting the first available server may cause a	• Two-dimensional task parsing system • Three-dimensional priority assignment system • A timeline allocation method with the joint consideration of the time slot length similarity	Task resilience ratio increase; More balanced resource load

	temporary and dramatic load increase when allocating tasks	and the minimum available resource at each time slot • A concession mechanism for task allocation on the timeline at each target data centre	
Chapter 6	• Selecting the first available server to enable early task completion might not be the optimal solution in term of cloud resiliency when rescuing tasks • Insufficient consideration of the resource contention and the deadline contention among the tasks in different	• PageRank-based priority assignment system • Dynamic PageRank-constrained task scheduling algorithm	Task resilience ratio increase; Workflow resilience ratio increase; Workflow continuity ratio increase

	concurrent-running workflow instances when rescuing tasks • Lack of the consideration of the entire workflow topology when prioritising tasks • Lack of the consideration of the influence on business continuity when developing fault tolerance strategy		