A Project/Dissertation Review-1 Report

on

An Algorithm for Finding the Minimum Cost of Storing and Regenerating Datasets in Multiple Clouds

Submitted in partial fulfillment of the

requirement for the award ofthe degree of

BTech CSE with specialisation in AI&ML

Under The Supervision of
Dr.A.Daniel
Assistant Professor

Submitted By
Aviral Sharma
18SCSE1180007

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA
December,2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"An Algorithm for Finding the Minimum Cost of Storing and Regenerating Datasets in Multiple Clouds "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021,** under the supervision of **Dr.A.Daniel**, **Assistant Professor, Department of Computer Science and Engineering**, Galgotias University,Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

AVIRAL SHARMA - 18SCSE1180007

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr.A.Daniel, Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **AVIRAL SHARMA - 18SCSE1180007** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

The proliferation of cloud computing allows users to flexibly store, re-compute or transfer large generated datasets with multiple cloud service providers. However, due to the pay-as-you-go model, the total cost of using cloud services depends on the consumption of storage, computation and bandwidth resources which are three key factors for the cost of IaaS-based cloud resources.

In order to reduce the total cost for data, given cloud service providers with different pricing models on their resources, users can flexibly choose a cloud service to store a generated dataset, or delete it and choose a cloud service to regenerate it whenever reused. In this project we propose a novel algorithm that can calculate the minimum cost for storing and regenerating datasets in clouds whether datasets should be stored or deleted, and furthermore where to store or to regenerate whenever they are reused.

In this project, a novel GT-CSB algorithm has been implemented that can find the best trade- off among computation, storage and bandwidth costs in clouds represented by the theoretical minimum cost strategy for storing application data among multiple cloud service providers.

This minimum cost also achieves the best trade-off among computation, storage and bandwidth costs in multiple clouds. Comprehensive analysis and rigid theorems guarantee the theoretical soundness of the paper, and general (random) simulations conducted with popular cloud service providers' pricing models demonstrate the excellent performance of our approach.

Table of Contents

List of Figures

Acronyms

We use X, Y, Z to denote the computation cost, storage cost and bandwidth cost of datasets respectively. Specifically, for a dataset di E DDG:

$$X_{d_i}^{c_j}$$

denotes the computation cost of regenerating dataset di from its direct predecessors in the DDG with cloud service provider cj, which is calculated as the multiplication of the generation time of di and the price of computation resources of cj;

$$Y_{d_i}^{c_j}$$

denotes the storage cost per time unit of storing dataset di with cloud service provider cj, which is calculated as the multiplication of the size of di and the price of storage resources of cj;

denotes the bandwidth cost of transferring dataset di from cloud service provider ck to cj, which is calculated as the sum of outbound cost of di from ck and inbound cost of di to cj.

$$Z_{d_i}^{c_k,c_j}$$

Terminology Used

Definition 1:

Regeneration strategy of a deleted dataset is the selection of cloud service providers where the dataset is regenerated from its stored provenance dataset(s).

Hence, the regeneration cost of a dataset is twofold

- the bandwidth cost of transferring its stored provenance dataset(s) and intermediate dataset(s) to the corresponding cloud services, and

- the computation cost of regenerating the dataset in these cloud services.

We denote the minimum regeneration cost of dataset di E DDG as minGenCost(di).

Definition 2:
Cost rate of a dataset is the average cost spent on this dataset per time unit in clouds.

$$CostR(d_i) = \begin{cases} minGenCost(d_i) * v_{d_i}, & //\, d_i \quad is \quad deleted \\ Y_{d_i}^{c_j}, & //\, d_i \quad is \quad stored \quad in \quad c_j \end{cases}$$

The Total Cost Rate of a DDG is the sum of cost rate of all the datasets in it, which is denoted as

$$TCR = \sum_{d_i \in DDG} CostR(d_i)$$

Definition 3:

Minimum cost benchmark of a DDG is the minimum total cost rate for storing and regenerating datasets in the DDG with the minimum cost strategy, which is denoted as

$$TCR_{\min} = \min\left(\sum_{d_i \in DDG} CostR(d_i)\right)$$

Technology used

1. Java
2. Swing
3. JDBC
4. XAMPP

Chapter 1

Introduction

1.1 Need for Cost Minimisation

These days applications are getting increasingly more data intensive. The size of the generated data is often gigabytes, terabytes or even petabytes. This generated data contains important results of computation, which may need to be stored for reuse. So, for deploying applications in cloud computing environment, cutting the cost of cloud- based data management is a big concern.

Every cloud service provider has a different price of computation, storage and bandwidth resources. Cloud users have multiple options to adapt to the enormous generated application data because different cloud service providers unlimited storage and processing power on demand. Users can just pay for the storage cost by storing all data in the cloud, and alternatively, they can delete some data to save the storage cost and pay for the computation cost to regenerate them whenever they are reused. Users can also migrate to cheaper CSP to store or regenerate data with paying for the bandwidth cost to transfer data. This leads to a tradeoff among computation, storage and bandwidth in cloud where different storage and regeneration techniques lead to a different total cost for storing the generated application data. Users need to have knowledge of costs in cloud in order to have the benefits of cloud computing.

In this project, a novel GT-CSB algorithm has been implemented that can find the best trade- off among computation, storage and bandwidth costs in clouds represented by the theoretical minimum cost strategy for storing and regenerating application data among multiple cloud service providers.

The proposed GT-CSB algorithm that we are working on can automatically calculate the minimum cost storage and regeneration strategy for the application data in multiple clouds.

1.2 Purpose

1. To calculate the minimum cost for storing and regenerating datasets in clouds.

2. To delete used less datasets and regenerate them whenever required.

3. To achieve the best trade off among computation, storage and bandwidth costs in multiple clouds.

Chapter 2

Literature Review

2.1 Different approaches for the problem

In the research area of resource management, much work has been done about resource negotiation (K. Deng, J. Song, K. Ren, D. Yuan, and J. Chen), replica placement (W. Li, Y. Yang, J. Chen, and D. Yuan) and multi-tenancy in clouds, by taking advantage of different types of resources. Our work also investigates differ- ent types of resources, but we mainly focus on the trade-offs among them. Another important foundation for our work is the research on data provenance. More specifically, Osterweil et al. (L. J. Osterweil, L. A. Clarke, A. M. Ellison, R. Podorozhny) present how to generate a provenance based data derivation graph for execution of a workflow. Foster et al. (I. Foster, J. Vockler, M. Wilde, and Z. Yong) propose the concept of virtual data in the Chimera sys- tem, which enables the automatic regeneration of data when needed. Recently, research on data provenance in cloud com- puting systems has also appeared (K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer). Based on the above research, in this paper, we utilise a Data Dependency Graph (DDG) which is based on data provenance. DDG depicts the generation relationships of all the generated data in clouds, with which we can manage where the data are stored or how to regenerate them. Since graph theory is a useful tool in computer science, we create a transitive graph based on DDG and propose an algorithm that can find the best trade- off among computation, storage and bandwidth in clouds.

Plenty of research has been done with regard to the trade- off between computation and storage. The Nectar system is designed for automatic management of data and computation in data centres, where obsolete data are deleted and re- generated whenever reused in order to improve resource utilisation. In, Deelman et al. present that storing some popular intermediate data can save the cost in comparison to al- ways regenerating them from the input data. In, Adams et al. propose a model to represent the trade-off of computation cost and storage cost. In, the authors propose the CTT-SP algorithm that can find the best trade-off between computation and storage in the cloud, based on which a highly cost- effective and practical strategy is developed for storing datasets with one cloud service provider. However, the above work did not consider the bandwidth cost into the trade-off model.

As the trade-off among different costs is an important issue in the cloud, some research has already embarked on this is- sue to a certain extent. In, Joe-Wong et al. investigate computation, storage and bandwidth resources allocation in order to achieve a trade-off between fairness and efficiency. In, Baliga et al. investigate the trade-off among computation, storage and bandwidth at the infrastructure level of cloud systems, where reducing energy consumption is the main research goal. In, Chen et al. further analyse the impacts of computation, storage and bandwidth on the total energy consumption by investigating different types of applications in the cloud. In, Agarwala et al. transform application data to certain formats and store them with different cloud services in order to reduce storage cost in the cloud, but data dependency and the option of data regeneration are not considered in their work.

2.2 Research Gap

In this project, we propose the GT-CSB algorithm, which can find a Generic best Trade-off among Computation, Storage and Bandwidth (GT-CSB) in clouds. This generic best trade-off represents the minimum cost storage and regeneration strategy, in which data can be stored or regenerated with any cloud service providers.

2.3 Preliminaries

1.Datasets

The algorithm discussed in here works on datasets. When we say the word “dataset”, we refer to the data newly produced in the cloud while the applications are running. They are the intermediate or final computation results of the applications, which can be reused in the future. For these data, their storage can be decided by the system since they can be regenerated if their provenance is known.

2.Data Dependency Graph (DDG)

DDG is a directed acyclic graph (DAG) which is based on data provenance in applications. All the datasets once generated in the cloud, whether stored or deleted, their references are recorded in DDG. In other words, it depicts the generation relationships of datasets, with which the deleted datasets can be regenerated from their nearest existing preceding datasets. Every node in the graph denotes a dataset, and the whole DDG will be the input of the GT- CSB algorithm.

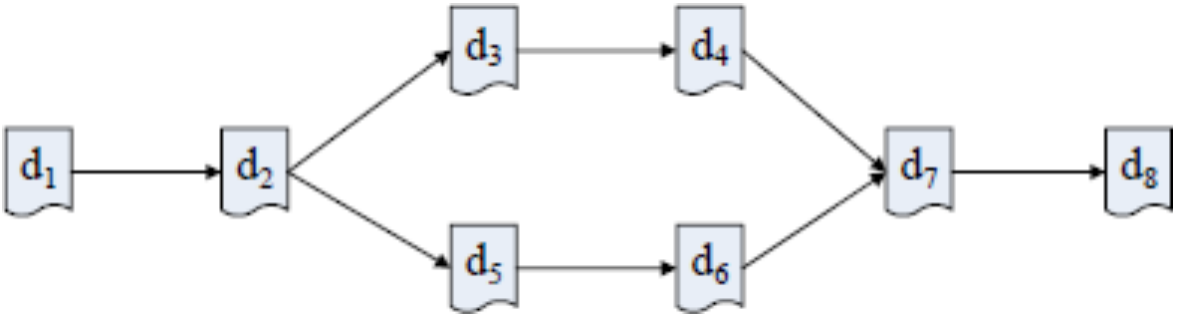


Fig. A simple Data Dependency Graph (DDG)

3.Datasets Storage and Regeneration Cost Model

“Cost = Computation + Storage + Bandwidth”
where the total cost of the dataset’s storage, Cost, is the sum of

- Computation, which is the total cost of computation resources used to regenerate datasets,

- Storage, which is the total cost of storage resources used to store the datasets, and

- Bandwidth, which is the total cost of bandwidth re-sources used for transferring datasets via the network.

Chapter 3

Architectural Diagram of the proposed system

3.1 Methodology/Modules and their description

GT-CSB Algorithm:

For the ease of illustration, we present the GT-CSB algorithm on linear DDG. Linear DDG means a DDG with no branches, where each dataset in the DDG only has one direct predecessor and successor except the first and last datasets.

Module 1:

Construct a Cost Transitive Graph (CTG) based on the DDG. First, for every dataset in the DDG, we create a set of vertices in the CTG representing services from different cloud service providers where datasets can be stored or regenerated. Then, we add edges to the CTG and guarantee that the paths in the CTG (from a start vertex to an end vertex)have one-to-one mapping to the storage strategies of datasets in the DDG.

Module 2:

Set weights to the edges in the CTG. We present how to calculate the weights of edges that guarantee that the length of every path in the CTG (from a start vertex to an end vertex) equals to the total cost rate of the corresponding storage strategy with the minimum cost regeneration strategy of the deleted datasets.

Module 3:

Find the shortest path in the CTG. We can use the well-known Dijkstra shortest path algorithm (or Dijkstra algorithm for short) to find the shortest path in the CTG, which infact represents both the minimum cost strategy for storing and regenerating datasets in the DDG with multiple cloud service providers, and the best trade-off among computation, storage and bandwidth costs in clouds.

3.2 Pseudo Code

Algorithm: GT-CSB

Input: A linear DDG $\{d_1, d_2 \dots d_n\}$;

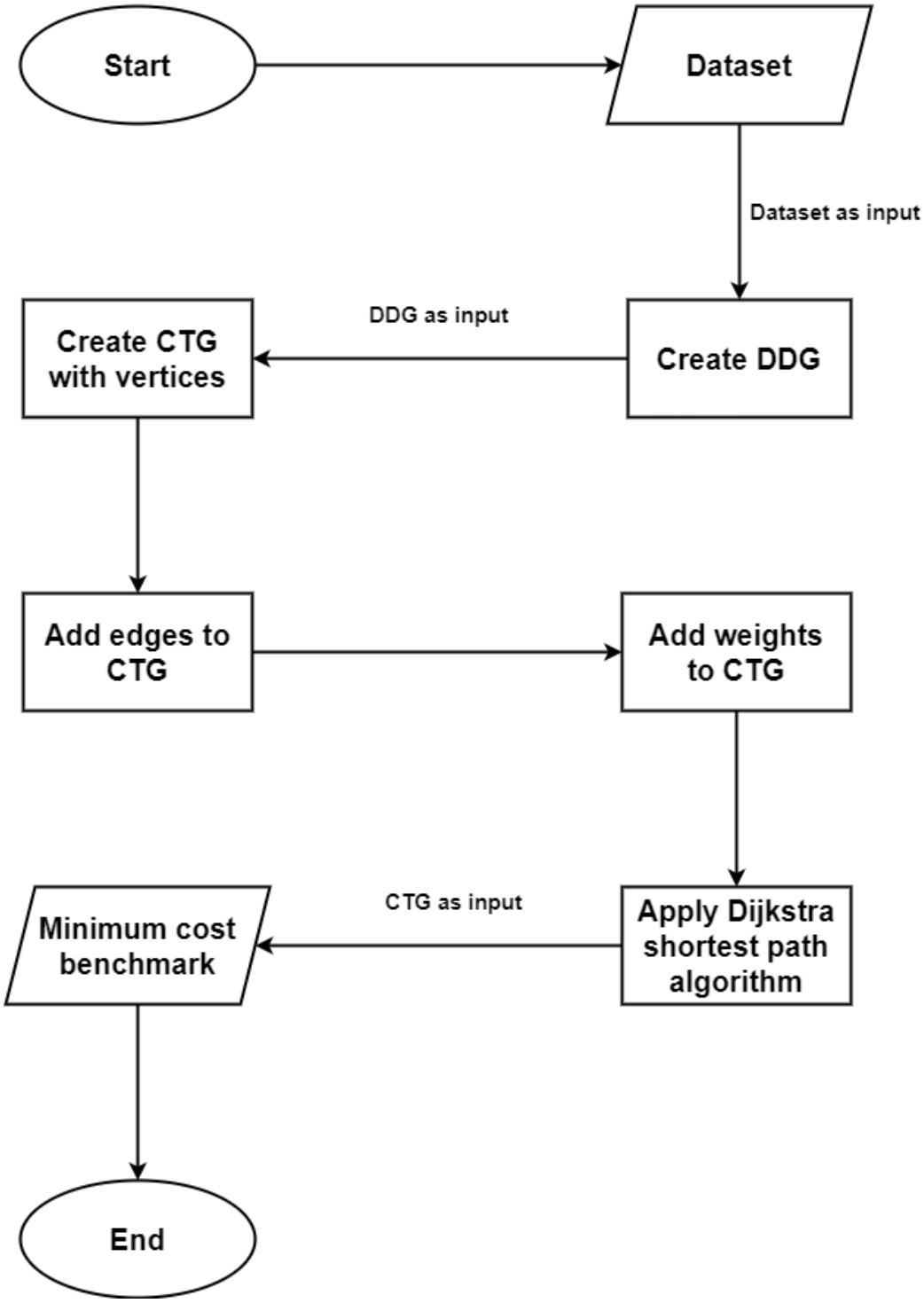
Cloud service providers $\{c_1, c_2 \dots c_m\}$;

Output: The minimum cost benchmark

---

01. Create ver_{start}, ver_{end};
02. **for** (every dataset d_i in DDG) //Create vertices for CTG
03. Create $V_{d_i} = \{ver_{d_i}^{c_1}, ver_{d_i}^{c_2}, \dots, ver_{d_i}^{c_m}\}$;
04. Add vertices $V_{d_1} \cup V_{d_2} \dots \cup V_{d_n} \cup \{ver_{start}, ver_{end}\}$ to CTG;
05. **for** (every $ver_{d_i}^{c_s} \in$ CTG) //Create edges for CTG
06. **for** (every $ver_{d_{i'}}^{c_{s'}} \in \text{CTG} \wedge (d_i, d_{i'} \in \text{DDG} \wedge d_i \rightarrow d_{i'})$)
07. Create $e < ver_{d_i}^{c_s}, ver_{d_{i'}}^{c_{s'}} >$; //Create an edge
08. **for** (every $ver_{d_{i+1}}^{c_k} \in V_{d_{i+1}}$) //Calculate the edge weight
09. $ver_{d_{i+1}}^{c_k} = Z_{d_i}^{c_s, c_k} + X_{d_{i+1}}^{c_k}$;
10. $weight = \min_{h=1}^{m} \{ver_{d_{i+1}}^{c_h}\}$;
11. **for** (every $d_j \in \text{DDG} \wedge (d_{i+1} \rightarrow d_j \rightarrow d_{i'})$)
12. **for** (every $ver_{d_j}^{c_k} \in V_{d_j}$)
13. $ver_{d_j}^{c_k} = \min_{h=1}^{m} \left\{ver_{d_{j-1}}^{c_h} + Z_{d_{j-1}}^{c_h, c_k}\right\} + X_{d_j}^{c_k}$;
14. $weight = weight + \min_{h=1}^{m} \{ver_{d_j}^{c_h}\}$;
15. $weight = weight + Y_{d_{i'}}^{c_{s'}}$;
16. Set $e < ver_{d_i}^{c_s}, ver_{d_{i'}}^{c_{s'}} > = weight$; //Set edge weight
17. P = Dijkstra (ver_{start}, ver_{end}, CTG); //Find the shortest path
18. Return P; //The benchmark is the length of P

3.3 Flowchart of the Project



Chapter 4

Results and Discussion

4.1 Evaluation

As Amazon is a well-known and widely recognised cloud service provider, we conduct experiments on Amazon cloud using on-demand services for simulation. We implement the GT-CSB in the Java programming language and run it on the virtualised EC2 instance with the Amazon Linux Image to evaluate its cost effectiveness and efficiency. We choose the standard small instance (m1.small) to conduct the experiments, because it is the basic type of EC2 CPU instances, which has a stable performance of one ECU[8]. In the simulation, we use randomly generated DDG with datasets of random sizes, generation times and usage frequencies. We also use popular cloud service providers' pricing model.

In this section we evaluate the cost effectiveness of the minimum cost benchmark. We compare it with different representative storage strategies and demonstrate their cost differences to the minimum cost benchmark. We investigate the cost constitution in the mini- mum cost benchmark. We demonstrate the proportions of computation, storage and bandwidth costs in the mini- mum cost benchmark and the change of the proportion with different DDG inputs. We evaluate the efficiency of the GT-CSB algorithm. We demonstrate that the algorithm has a polynomial time complexity as both the number of datasets in the DDG and the number of cloud service providers grow.

4.2 Cost Effectiveness Evaluation

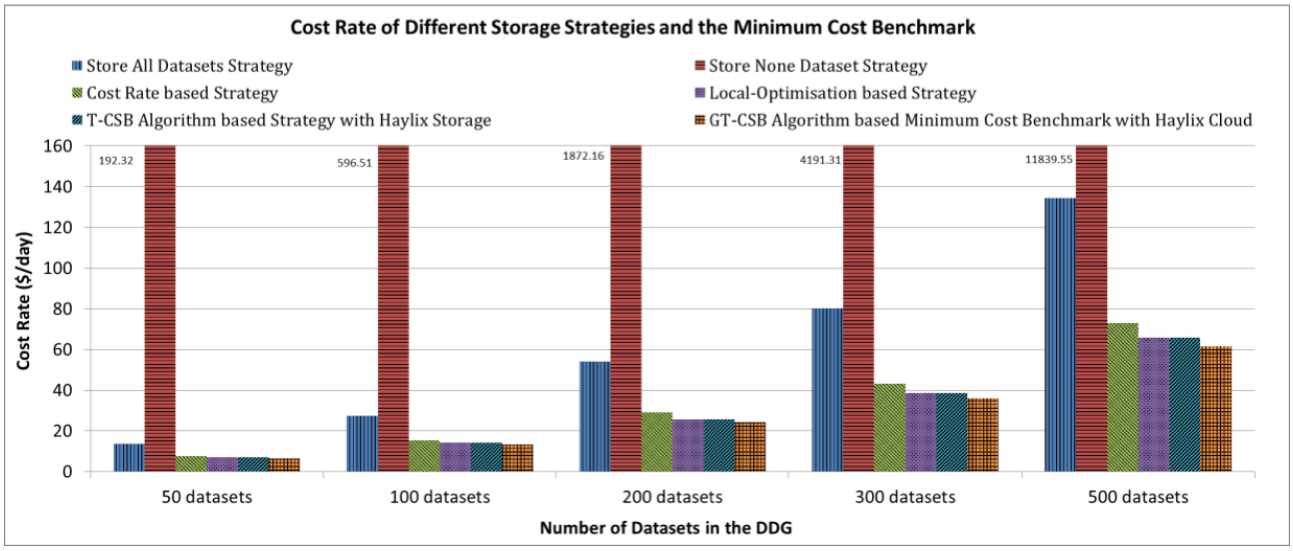
We evaluate the cost effectiveness of our minimum cost benchmark by comparing it with some representative storage strategies as follows.

- Store all datasets strategy, in which all generated da-
  tasets of the application are stored in the cloud. This strategy represents the common approach used in most applications in the cloud.

- Store none datasets strategy, in which all generated datasets of the application are deleted after being used. This strategy is often used in scientific applications that generate large and rarely used intermediate datasets.

- Cost rate based strategy reported in which we store datasets in the cloud by comparing their own generation cost rate and storage cost rate.

These strategies are designed for deploying applica- tions with one cloud service provider, which is assumed to be Amazon cloud, i.e. using EC2 service ($0.10 per CPU instance hour) for computation and S3 service ($0.15 per gigabyte per month) for storage. In the T-CSB algorithm based strategy, we assume that datasets can be transferred to Haylix cloud for storage.


Cost Rate of Different Storage Strategies and the Minimum Cost Benchmark
Store All Datasets Strategy
Store None Dataset Strategy
Cost Rate based Strategy
Local-Optimisation based Strategy
T-CSB Algorithm based Strategy with Haylix Storage
GT-CSB Algorithm based Minimum Cost Benchmark with Haylix Cloud
Cost Rate ($/day)
160
140
120
100
80
60
40
20
0
192.32
596.51
1872.16
4191.31
11839.55
50 datasets
100 datasets
200 datasets
300 datasets
500 datasets
Number of Datasets in the DDG

Conclusions and Future Scope

In this project, we have investigated the minimum cost strategy for storing and regenerating datasets based on the pay-as-you-go model with multiple cloud service providers. Towards achieving a generic best trade-off among computation, storage and bandwidth, we have designed the GT-CSB algorithm, which calculates the minimum cost benchmark for storing and regenerating datasets in clouds. We have presented the design of the algorithm in detail and rigid proof to guarantee the validity of minimum cost benchmark. Experimental results also demonstrated the excellent performance of the proposed approach.

In our current work, we assume that cloud service providers have unified prices for computation, storage and bandwidth resources. However, in the real world, the prices of cloud services can well be different according to different requirements and usages. Furthermore, extra cost might be caused by the "vender lock-in" issue among different cloud service providers, large number of requests from input/output (I/O) internsive applications, etc. In the future, we will incorporate more complex pricing models in our datasets storage and regeneration cost model.

In this paper, our focus is to solve the crucial problem of calculating the minimum cost for data storage and re-generation in multiple clouds. Hence the efficiency of the GT-CSB algorithm design has not been comprehensively investigated. In the future, we will re-design the algorithm by using Dynamic Programming techniques, which can further significantly reduce the algorithm complexity.