Harnessing the Power of Multiple Cloud Service Providers: An Economical and SLA-Guaranteed Cloud Storage Service

Guoxin Liu and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631, USA
{guoxinl, shenh}@clemson.edu

***Abstract*—It is critical for a cloud service broker to minimize its payment cost to cloud service providers for serving the broker's customers while providing SLA-guaranteed services. In this paper, we propose an Economical and SLA-guaranteed cloud Storage Service (ES^3), which finds a data allocation and resource reservation schedule with cost minimization. ES^3 incorporates (1) a data allocation and reservation algorithm, which allocates each data item to a datacenter and determines the reservation amount on datacenters by leveraging all the pricing policies; (2) a genetic algorithm based data allocation adjustment approach, which makes data Get/Put rates stable in each datacenter to maximize the reservation benefit. Our trace-driven experiments show the superior performance of ES^3.**

I. Introduction

Cloud storage (e.g., Amazon S3, Microsoft Azure and Google Cloud Storage), as an emerging commercial service, is becoming increasingly popular. For a cloud customer's application, the data access latency is negatively proportional to the customer's income. In order to maximize profits, cloud customers must provide low data Get/Put latency and high availability to their clients while minimizing the total payment cost to the Cloud Service Providers (CSPs). Since different CSPs provide different storage service prices, customers tend to use services from different CSPs instead of a single CSP to minimize their payment cost (cost in short). However, the technical complexity of this task makes it non-trivial to customers, which calls for the assistance from a third-party organization. Under this circumstance, cloud service brokers [1] have emerged. A broker collects resource usage requirements from many customers, makes resource requests to multiple clouds, pays the CSPs for the actually consumed resources as a customer, and charges its customers as a CSP. It is critical for a broker to minimize its payment cost to the CSPs for serving the broker's customers while providing Service Level Agreement (SLA) guaranteed services. This enables the broker to make its prices lower than CSPs' prices, which reduces its customers' payment cost and hence attracts more businesses from the customers.

Datacenters in different areas of a CSP and datacenters of different CSPs in the same area offer different prices for resource usages including data Get/Put, Storage and Transfer. In addition, the Storage/Transfer pricing follows a tiered model, which supplies a cheaper unit price for storing/transferring a larger size of data. Further, the data transfer prices vary widely depending on whether the destination datacenter belongs to the same CSP or the same location of the source datacenter. Besides the pay-as-you-go pricing model, in which the consumer pays the CSPs based on resources actually used, CSPs also offer reservation pricing model [2], in which a consumer reserves its resource usage for a certain time in advance with much lower price. Thus, to minimize its payment, a broker must fully leverage these pricing policies. Specifically, given the data items and their requests from its customers, the broker needs to make a data allocation (including data storage and Get request allocation) and resource reservation schedule, which minimizes its payment cost to CSPs and provides SLA guarantee. This problem however is NP-hard [3].

To handle this problem, in this paper, we propose an Economical and SLA-guaranteed cloud Storage Service (ES^3) for brokers. As far as we known, this is the first work to find a geo-distributed data allocation schedule over multiple CSPs with cost minimization by fully leveraging all aforementioned pricing policies and SLA guarantee.

II. System Design of ES^3

A. Data Allocation and Resource Reservation

For each customer of ES^3, its data is constituted of many data items. A data item d_l's payment cost consists of Get, Put, Transfer and Storage cost denoted by $C_s^{d_l}$, $C_g^{d_l}$, $C_t^{d_l}$ and $C_p^{d_l}$. To identify the datacenter to store a given data item, the allocated datacenters should have enough Get/Put capacity and satisfy Get/Put SLA requirements. Among these qualified datacenters, we need to choose β datacenters that can reduce the cost as much as possible. For this purpose, we consider different pricing policies. First, storing the data in the datacenter that has the cheapest unit price for its dominant cost (e.g., Get, Put or Storage) can reduce the cost greatly. We consider data item d_l as Storage-intensive if $C_s^{d_l}$ dominates the total cost (e.g, $C_s^{d_l} \gg C_g^{d_l}$+$C_p^{d_l}$), and the Get/Put-intensive data items are defined similarly. Second, if the data is Storage-intensive, based on the tiered pricing policy, storing the data in the datacenter that results in the largest aggregate storage size $\mathbb{S}_{dp_j}$ can reduce the cost greatly. Third, if the data is Get/Put-intensive, in order

to minimize the reservation cost, the data should be stored in the datacenter with the lowest unit reservation price, and in order to maximize the reservation benefit, the data should be stored in the datacenter that has the maximum reservation benefit increment after allocation. Based on these three considerations, the datacenter candidates to store the data are further selected. After the data allocation, the reservation of Gets/Puts in each datacenter is determined according to [3].

B. GA-based Data Allocation Adjustment

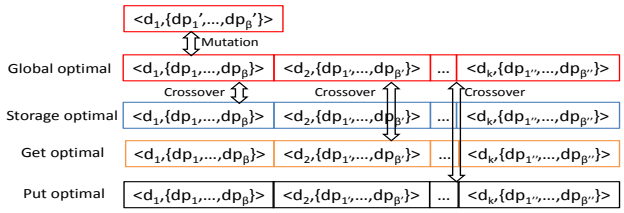


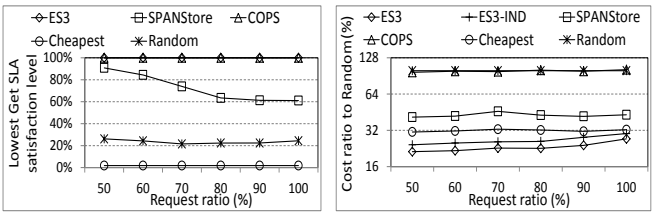
Figure 1: GA-based data allocation enhancement.

In the GA-based data allocation adjustment approach, as shown in Figure 1, a data allocation schedule is formed by $<d_l, \{dp_1, ..., dp_\beta\}>$ of each data item requested by a customer datacenter, where $\{dp_1, ..., dp_\beta\}$ is the set of datacenters that store d_l. This algorithm regards each data allocation schedule as a genome string. Using the algorithm in Section II-A, it generates data allocation schedules with the lowest total cost (named as global optimal schedule), and with the lowest Storage cost, lowest Get cost and lowest Put cost (named as partial optimal schedules) by assuming all data items as Storage-, Get- and Put-intensive data, respectively. To generate the children of the next generation, the global optimal schedule sequentially conducts crossover with each partial optimal schedule with crossover probability (Figure 1). In order not to be trapped into a sub-optimal result, the genome mutation occurs after the crossover.

III. PRELIMINARY RESULTS

We conducted trace-driven experiments on Clemson University's Palmetto Cluster [4]. We simulated 50 geographically distributed datacenters in 25 cloud storage regions. The distribution of the inter-datacenter Get/Put latency between any pair of cloud storage datacenters follows the real latency distribution as in [5]. The unit prices and reservation price for Storage, Get, Put and Transfer in each region follow the real prices of commercial clouds. We simulated ten times of the number of all customers listed in the major three commercial clouds. The Get deadline is 100ms [5], the percentage of latency guaranteed Gets is 90%. Also, the aggregate data size of a customer was randomly chosen from $[0.1TB, 1TB, 10TB]$.

Comparison methods. We compared ES^3 with the following systems. i) *COPS* [6]. It allocates requested data into a datacenter with the shortest latency. ii) *Cheapest*. It selects the datacenters with the cheapest cost in the pay-as-you-go manner. iii) *Random*. It randomly selects datacenters. iv) *SPANStore* [5]. It is a storage system over multiple CSPs' datacenters to minimize cost in the pay-as-you-go manner and support SLAs without considering capacities.



(a) Get SLA guarantee (b) Cost minimization

Figure 2: Performance of ES^3.

In this section, we varied each data item's Get rate from 50% to 100% (named as request ratio) of its actual Get rate in the trace. In order to evaluate the SLA guaranteed performance, we measured the lowest SLA satisfaction level of a customer among all customers. The Get SLA satisfaction level of a customer is calculated by the ratio between actual percentage of Gets satisfying the deadline requirement and the desired percentage. Figure 2(a) shows the lowest Get SLA satisfaction level of each system. From the figure, we can see that ES^3 can supply a Get SLA guaranteed service, since it considers both the Get SLA and capacity constraints while data allocation. Figure 2(b) shows the ratio of each system's cost to *Random*'s cost. We also tested a variant of ES^3, denoted by ES^3*-IND*, in which each customer individually uses ES^3 to allocate its data without aggregating their workload together. ES^3*-IND* generates a smaller cost than comparison methods, because it chooses the datacenter under SLA constraints that minimizes each customer's cost by considering all pricing policies. ES^3 generates the smallest cost, because it further aggregates workloads from all customers to get a cheaper Storage and Transfer unit price based on the tiered pricing model. The figures confirm that ES^3 generates the smallest payment cost in all systems and the effectiveness of considering tiered pricing model.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

REFERENCES

[1] D. Niu, C. Feng, and B. Li. A Theory of Cloud Bandwidth Pricing for Video-on-Demand Providers. In *Proc. of INFOCOM*, 2012.

[2] Amazon DynamoDB. http://aws.amazon.com/dynamodb/, [Accessed in Apr. 2015].

[3] G. Liu and H. Shen. Geographical Cloud Storage Service with SLA Guarantee over Multiple Cloud Providers. Technical report, Clemson University, 2014.

[4] Palmetto Cluster. http://citi.clemson.edu/palmetto/, [Accessed in Apr. 2015].

[5] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha. SPANStore: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services. In *SOSP*, 2013.

[6] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Dont Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS. In *Proc. of SOSP*, 2011.