SLA Definition for Multi-Cloud Queries

Damien T Wojtowicz, Shaoyi Yin, Franck Morvan

SLA Definition for Multi-Cloud Queries

Damien T. Wojtowicz
damien.wojtowicz@irit.fr
IRIT – Université Toulouse III

Shaoyi Yin
shaoyi.yin@irit.fr
IRIT – Université Toulouse III

Franck Morvan
morvan@irit.fr
IRIT – Université Toulouse III

ABSTRACT

Public data availability in cloud-hosted databases raises interests in systems providing multi-cloud querying capabilities. Since data access in this context induces monetary costs, we suggest a method to compute SLAs for multi-cloud queries. It consists in decomposing queries into a directed graph of maximal sub-queries per provider, and finding a financially cheapest execution plan. This method yields SLAs with monetary costs lower than a download-all approach, granted that inter-provider data transfers are minimised.

KEYWORDS

Database-as-a-Service, Multi-Cloud, Service-Level Agreement.

1 INTRODUCTION

Cross-analysis of datasets is of key importance to researchers, as the accumulation of data in fields such as astronomy and biology dramatically increases the opportunities for new discoveries. Various data sharing methods are available. File provision, either through a web interface or in the cloud[1], is popular and historically ingrained. Sharing can also be achieved by offering public read access to a database, which may be hosted in the cloud following the Database-as-a-Service (DBaaS) model[2]. In this case, users can leverage the services of public cloud providers in order to execute their queries. Alternatively, users can download datasets and run queries on self-administrated databases, implying sufficient skills and resources.

Unfortunately, providers do not offer out-of-the-box multi-cloud querying capabilities, leaving space for systems orchestrating query execution across multiple public databases in a DBaaS fashion. Since access to the providers' datasets is billed, such a service shall let users control their expenditures by means of Service-Level Agreement (SLA). Those contracts are known to play a key role in query optimisation [8], hence the importance of well-defined SLAs. In the context of DBaaS, a single performance objective cannot be retained for all queries due to different complexities. In addition, even for the same query, different tenants may have different performance expectations which are partially influenced by their budget (i.e. minimising response time may not be the goal). These challenges should be taken into account in the SLA definition process.

In Section 2, we briefly review literature about multi-cloud database systems. In Section 3, we propose a graph-based SLA definition method for multi-cloud relational queries. Its results are analysed in Section 4. We conclude in Section 5 by mentioning perspectives.

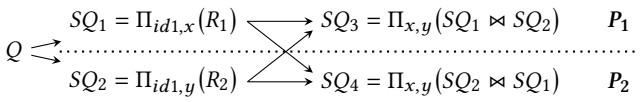

Figure 1: DAG and minimal sub-queries $SQ_\star$ for query Q. Two execution plans, both involving data transfer between providers P_1 and P_2, are possible with SQ_3 and SQ_4.

2 RELATED WORK

Work on multi-cloud database services has mostly focused on federated databases. Indeed, several query processing systems involving different cloud providers exist. To name a few, SCOPE [5] is the base of a collaborative document editing tool, MetaStorage [2] operates as a key-value storage system and SHAMC [6] acts as a relational DBMS. Despite their seemingly differences, in terms of context, objectives and data models, they share the same approach.

Indeed, they all own their data, seeking an overall system optimisation, using for example data placement or replication strategies, with respect to an objective (e.g. response time, availability, financial cost) rather than solely focusing on optimising data access. These systems are based on the Infrastructure-as-a-Service (IaaS) model, and their SLAs encompass broader services than querying.

Research has also been carried on multi-objective cost models for multi-cloud queries [3], aiming at adding a monetary aspect to usual cost models. In this case, query execution is performed on a set of cloud-hosted virtual machines leveraging IaaS capabilities. Those cost models are best suited to scale up or down resources, and are therefore not suitable to help defining SLAs for DBaaS-based multi-cloud systems.

3 SLA DEFINITION METHOD

In this paper, we focus on the performance objective for a query and its relation to the monetary costs. Therefore, we model the SLA for a query as a couple $SLA =< C, RT >$, with C the monetary cost of a query and RT its response time. This paper is exemplified by the query $Q = \Pi_{x,y}(R_1 \bowtie R_2)$, assuming relations $R_1(\underline{id1}, x)$ and $R_2(\underline{id2}, \#id1, y)$ respectively hosted on providers P_1 and P_2 (see fares[3] in Table 1a). For the sake of simplicity, network bandwidth in our example is supposed to be constant at 1.0 GB/s.

Our method is a three-step procedure, starting by decomposing the input query, then generating SLAs for sub-queries and ending by aggregating the latter.

[1] See examples of public datasets at Amazon (https://registry.opendata.aws/) or Google (https://cloud.google.com/public-datasets)
[2] See examples at Google (https://cloud.google.com/bigquery/public-data).

[3] Inspired from BigQuery's pricing policy (https://cloud.google.com/bigquery/pricing)

3.1 Query decomposition

Inspired by query decomposition techniques [3, 7], we suggest that multi-cloud queries can be modelled as a directed acyclic graph (DAG), where vertices are maximal sub-queries involving a minimal set of providers. Query Q can be decomposed as depicted in Figure 1. SQ_1 and SQ_2 are maximal sub-queries involving a single provider, respectively P_1 and $P2$. SQ_3 and SQ_4 involve data from P_1 and P_2 respectively executed on P_1 and P_2.

This decomposition creates two plans, one ending with SQ_3 and the other with SQ_4. It is worth noticing that both are similar to execution plans in distributed databases. In this context, the best plans usually minimise network transfers [9].

3.2 Mitigation of provider miscalculations

For each sub-query $SQ_\star$, a SLA is generated as well as an estimation of the output relation's size S (see Table 1d). Those might not reflect the actual sub-queries' output relation size due to internal cardinality estimation errors [4]. We suggest to keep track of those estimation errors by computing a supposed error ratio for each query submitted to the provider as $r = \frac{S^{(Real)}}{S^{(SLA)}}$.

Those ratios are then used to compute $\bar{r}$, a sliding average of r on the last n queries exemplified in Table 1f. While being sensitive to the complexity of the last queries, using a sliding average let our system take into account changes of the provider's estimator.

3.3 SLA components aggregation

We use separate methods for each component of the SLA, using corrected values (see Table 1e). Monetary cost C is the sum of all sub-queries costs and all transfers costs (see the latter in Table 1b). Response time RT is the sum of the RT-maximal path in the DAG, taking into account cross-providers transfer time.

Due to space limitation, we cannot put the cost formulas in the paper. We only show the results for illustration purposes. Table 1c shows the costing for each plan. At the end of the procedure, the SLA that will be presented to the user, in the context of our example query Q, is $< 0.184\,\$, 7.287\,s >$ stemming from SQ_4.

4 RESULT ANALYSIS

Figure 2 depicts the breakdown of each scenario costs. The less expansive one minimises inter-provider transfers, which is unsurprising given the similarity of our setting with distributed systems. Moreover, there is no significant differences between SQ_3 and SQ_4 in storage and processing costs.

SQ_4 is 1.8 times cheaper with an acceptable performance degradation (43%) compared to the download-and-process scenario, thus a multi-cloud query appears to be more competitive financially-wise than the download-all approach.

5 CONCLUSION

Our SLA computation method is a first step towards a middleware enabling multi-cloud querying in a DBaaS fashion. We showed that optimal multi-cloud query execution plans should minimise inter-providers transfers in order to limit costs.

Next steps will involve setting up a SLA-constrained execution engine. Given that sub-queries' output relation size is an estimation, the SLA-based optimal plan might not actually be the best. Hence, methods mitigating estimation errors, such as mobile-agent-based models [1] implementing reinforcement learning, appear to be relevant for the execution of multi-cloud queries.

Table 1: Values used for SLA computations

	P_1	P_2
Export	0.060	0.075
Querying	0.006	0.008
Storage	0.003	0.002

(a) Providers' fares ($/GB)

Transfer	Time (s)	C ($)
$SQ_1 \rightarrow SQ_4$	1.102	0.066
$SQ_2 \rightarrow SQ_3$	3.050	0.229

(b) Transfer times and fares

Scenario	C ($)	RT (s)
SQ_3	0.353	10.762
SQ_4	0.184	7.287
Download-all	0.335	5.083

(c) Costs of all scenarios

$SQ\_$	C ($)	RT (s)	S (GB)
SQ_1	0.009	1.000	1.000
SQ_2	0.030	2.000	3.000
SQ_3	0.055	6.000	6.100
SQ_4	0.061	5.000	6.120

(d) Provider-generated SLA of each sub-query

$SQ\_$	C ($)	RT (s)	S (GB)
SQ_1	0.010	1.102	1.102
SQ_2	0.031	2.033	3.050
SQ_3	0.060	6.610	6.720
SQ_4	0.062	5.083	6.222

(e) Corrected SLAs

Query	1	2	3	4	5	6	$\bar{r}$
r for P_1	1.50	0.90	1.20	1.05	1.01	0.95	**1.102**
r for P_2	1.01	1.02	0.95	1.03	1.10	0.99	**1.017**

(f) Last SLA estimation error ratio for providers and $\bar{r}$ computed using the last $n = 6$ queries for each provider

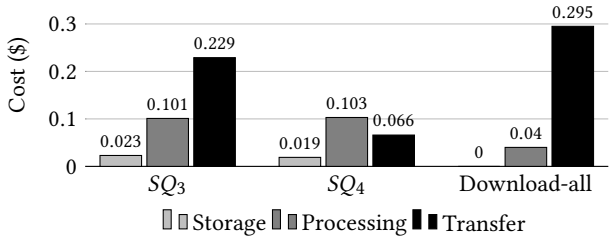

Figure 2: Breakdown of each scenario cost. Inter-providers transfers can tremendously increase the price of a scenario.

REFERENCES

[1] J.-P. Arcangeli, F. Morvan, A. Hameurlain, and F. Migeon. 2004. Mobile Agents Based Self-Adaptive Join for Wide-Area Distributed Query Processing. *Journal of Database Management* 15, 4 (2004), 25–44.
[2] D. Bermbach, M. Klems, S. Tai, and M. Menzel. 2011. MetaStorage: A Federated Cloud Storage System to Manage Consistency-Latency Tradeoffs. *IEEE CLOUD*, 452–459. Washington, USA.
[3] A. Gounaris, Z. Karampaglis, A. Naskos, and Y. Manolopoulos. 2014. A Bi-Objective Cost Model for Optimizing Database Queries in a Multi-Cloud Environment. *Journal of Innovation in Digital Ecosystems* 1, 1 (2014), 12–25.
[4] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 3, 9 (2015), 204–215.
[5] A. Rafique, D. van Landuyt, E. Truyen, V. Reniers, and W. Joosen. 2019. SCOPE: Self-Adaptive and Policy-Based Data Management Middleware for Federated Clouds. *Journal of Internet Services and Applications* 10, 1 (2019), 2.
[6] L Wang, Z Yang, and X Song. 2020. SHAMC: A Secure and Highly Available Database System in Multi-Cloud Environment. *FGCS* 105 (2020), 873–883.
[7] E. Wong and K. Youssefi. 1976. Decomposition — a Strategy for Query Processing. *ACM TODS* 1, 3 (1976), 223–241.
[8] S. Yin, A. Hameurlain, and F. Morvan. 2018. SLA Definition for Multi-Tenant DBMS and its Impact on Query Optimization. *IEEE TKDE* 30, 11 (2018), 2213–2226.
[9] M. T. Özsu and P. Valduriez. 2020. *Principles of Distributed Database Systems* (4 ed.). Springer International Publishing.