Performance Evaluation of Cloud-based RDBMS through a Cloud Scripting Language

Andrea S. Charão[1], Guilherme F. Hoffmann[1], Luiz A. Steffenel[2], Manuele K. Pinheiro[3] and Benhur de O. Stein[1]
[1]*Universidade Federal de Santa Maria, Santa Maria, Brazil*
[2]*CReSTIC Laboratory, Université de Reims Champagne-Ardenne, Reims, France*
[3]*CRI Laboratory, Université Paris 1 Panthéon-Sorbonne, Paris, France*

Keywords: Relational Database Management Systems, Cloud Computing, Performance Evaluation, Benchmarking.

Abstract: Cloud computing has brought new opportunities, but also new concerns, for developing enterprise information systems. In this work, we investigated the performance of two cloud-based relational database services, accessing them via scripts which also execute on a cloud platform, using Google Apps Script technology. Preliminary results show little differences between the services in their trial versions, considering limitations imposed by the Google platform.

1 INTRODUCTION

Cloud computing has proven to be an advantageous alternative for both businesses and end users, providing previously unimaginable services at affordable or even free subscription plans. Engineers and developers of enterprise information systems can count on various types of cloud services, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) (Mell and Grance, 2011).

In the market landscape of cloud computing, a recent trend is Database as a Service (DBaaS), which moves database storage and management to the cloud (Hacigumus et al., 2002; Mahmood, 2014). Applications that reside or not in the cloud can benefit from this model, which inherits advantages of other cloud services, such as availability, scalability and ease of management. Another recent trend is cloud scripting, a programming solution to automate tasks in the cloud. This little-known alternative allows to build entire information systems by coupling existing cloud services.

Google is a representative example of a company offering cloud services to both the general public (email, calendar, storage, etc.) and to developers. A useful resource for developers is Google Apps Script (GAS) (Google Inc., 2009; Ferreira, 2014), a cloud scripting solution to automate access to cloud services. This programming tool takes advantage of Google's Platform as a Service (PaaS) layer, enabling to build applications that run on Google's infrastructure and use its libraries, services, and tools. In this model, developers do not need to manage or control any cloud infrastructure: they only deal with developing applications and performing environment settings (Mell and Grance, 2011). Among its features, Google Apps Script offers the ability to access relational cloud databases (Google Inc., 2017a).

These new possibilities of cloud data management and programming bring, along with the benefits, some questions to be investigated. For example, cloud services often associate performance attributes (requisitions per second, maximum execution time, etc.) with pricing models (free, pay-as-you-go, etc.) (Li et al., 2010). Thus, knowing the performance of services becomes important because, by ignoring it, a developer can generate more expenses or even make a cloud execution unfeasible (Iosup et al., 2011).

In this paper, we investigated the access times for two different third-party cloud databases using Google Apps Script. This can provide subsidies for developers interested in these services. This type of performance data was not found in the literature, reinforcing the motivation for this work.

Indeed, we are focusing in a scenario where the

Charão, A., Hoffmann, G., Steffenel, L., Pinheiro, M. and Stein, B.
Performance Evaluation of Cloud-based RDBMS through a Cloud Scripting Language.
DOI: 10.5220/0006350803320337
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 332-337
ISBN: 978-989-758-247-9

application and the data services do not reside in the same provider. Several factors may conduct the splitting of services among different providers. Most of times, dependency issues prevent the complete migration from one provider to the other, obliging companies to integrate services hosted by different cloud providers.

2 BACKGROUND

2.1 Cloud Databases

As several cloud database providers exist and this preliminary work has no intention to perform a broad comparison between these services, we decided to concentrate in two relational database services: the Amazon Relational Database Service (RDS), a pioneer service launched by Amazon in 2009, and Morpheus, a recent database service offered by Morpheus Data since 2014. Our primary criteria to choose these services was related to their different launch dates, which could be an indicator of maturity of the service.

Both services can be accessed through the JDBC (Java Database Connectivity) service offered by GAS. It is worth noting that Google also offers database services, both relational (Cloud SQL) and NoSQL (Bigtable and Datastore). However, as stated before, our aim was to evaluate the performance of disjoint services and using both cloud scripting and cloud databases from Google would bias the results.

2.1.1 Amazon Web Services - Amazon RDS for MySQL

Amazon Web Services is a cloud computing pioneer offering a comprehensive set of global computing, storage, database, analytics, application, and deployments services. As part of this suite, Amazon RDS (Relational Database Service) is a service that facilitates the installation, operation and scheduling of a MySQL, Oracle, Microsoft SQL Server or PostgreSQL database in the cloud. It provides scalable capacity and manages database administration tasks. In its free program, it offers 750 hours of database micro-instances per month.

2.1.2 Morpheus Cloud Database

Morpheus is a PaaS that started its services in 2015, offered by the Morpheus Data. Its flagship product is an online database service that allows users to create, deploy and host instances of four different types of database classes through public, private and hybrid clouds. It offers high reliability and availability, with each instance being deployed for free with a full replica, and with automated daily backups. The four database classes are:

- MongoDB for non-structured data and documents;
- MySQL for traditional RDBMS (Relational Database Management System);
- Redis for in-memory data management;
- ElasticSearch for distributed search.

2.2 Cloud Scripting Languages

Cloud scripting is an approach to exploit cloud-based services in a programmatic way. This can be used to perform large scale distributed computations (Murray and Hand, 2010; Dzik et al., 2013) or simply to coordinate multiple services from cloud providers.

GAS is a prominent example of cloud scripting solution. It comprises a JavaScript-based programming language that allows you to build web applications and integrate Google Apps and third-party services. These scripts are written to an editor directly from the browser and run on Google's servers.

Google Apps Script is a relatively new technology, launched in 2009. It is in constant development, with new features and features being implemented frequently. Table 1 summarizes the services accessible through GAS to the present date. Some of them are scheduled to be deprecated soon, as they are been replaced. Note, in this table, the JDBC service, which provides access to RDBMS through GAS.

Although GAS brings advantages such as portability, flexibility and unconcern of the developer in configuring the environment, it brings with it a time limitation that each script has to execute, which is currently set to **six minutes** (Google Inc., 2017b).

3 EXPERIMENTS

In order to carry out this work, the development was divided in two stages: preparation of the environment (databases and connection with GAS) and coding of the test scripts in GAS with operations on the databases.

3.1 Environment Setup

To start the work, we created free accounts in both services (Amazon RDS and Morpheus). The config-

Table 1: Google Services.

Google Apps Services	Advanced Google Services	Script Services
Calendar	Admin SDK	Base
Contacts	AdSense	Cache
Document	Analytics	Charts
Domain(Turned off)	Apps Activity	Content
Drive	BigQuery	HTML
Forms	Calendar	JDBC
Gmail	Classroom	Lock
Groups	Drive	Mail
Language	DoubleCLick Campaigns	Optimization
Maps	Fusion Tables	Properties
Sites	Gmail	Script
Spreadsheet	Google+	UI (Turned off)
	Google+ Domains	URL Fetch
	Mirror	Utilities
	Prediction	XML
	Shopping Content	
	Tasks	
	URL Shortener	
	Youtube	

urations provided in each service for this type of account are shown below:

- Amazon RDS: db.t2.micro instance with one virtual CPU, one ECU (Engine Control Unit), 615 MB memory, without EBS (Elastic Block Store) optimization, low performance network.
- Morpheus: Storage space of 5 GB and 1 GB of RAM.

The connection of the script to the databases is done through the JDBC Service Apps Script. This feature offers 10 different IP address ranges that the platform can use to access the databases. These addresses serve to configure access permissions on the cloud database.

The two databases use slightly different forms of configuring access permissions. Morpheus works with an Access Control List (ACL), and Amazon RDS creates the database within a Virtual Private Cloud (VPC), and VPC has a security group, which is responsible for controlling which IP addresses will be released to connect to the database.

Both databases use the IP address allocation method called CIDR (Classless Inter-Domain Routing), which stores an IPv4 or IPv6 network specification. The specification format used is composed of the IP address followed by the number of bits of the netmask.

3.2 Test Scripts

The scripts developed for the tests are based on the SysBench benchmark [1], a simple performance evaluation tool that allows you to quickly get an impression of the performance of systems. Part of this tool is dedicated to evaluating database performance and is used by authors who have investigated the performance of cloud databases (Schwartz et al., 2012). The original SysBench codes are written in the Lua programming language. For the tests in this work, codes have been rewritten using Google Apps Script.

All codes have as principle to do basic SQL operations in a database, searching for a random value and then performing the operation. The main goal is to measure the time taken to process a specific amount of database operations. The tests are divided into the following categories of database operations: *Inserts*, *Updates*, *Selects* and *Deletes*. The table used in the tests has five numeric/alphanumeric fields (see Table 2). Figure 1 present a code excerpt from the GAS test scripts. It performs a database connection, creates a table and executes a set of database insert operations.

Table 2: Table for tests (Sysbench).

id	Name	Gender	Age	DonutsEaten

Initially, measurements were taken to identify the maximum number of operations each database could

[1] https://github.com/akopytov/sysbench

```
function dbOperations() {
  Logger.log("Start");
  var age;
  var donuts;
  var num;

  var conn = Jdbc.getConnection('jdbc:mysql:// \
    162.252.108.127:13342/Sysbench', 'User', '*****');

  conn.createStatement().execute(' \
    CREATE TABLE Benchmark \
      (id INTEGER, \
       Name VARCHAR(50), \
       Gender VARCHAR(10), \
       Age INTEGER, \
       DonutsEaten INTEGER, PRIMARY KEY (id))');

  for(i=1; i<100; i++){
    age = Math.floor((Math.random() * 100) + 1);
    donuts = Math.floor((Math.random() * 1000) + 1);
    num = Math.floor((Math.random() * 3500) + 1);
    conn.createStatement().execute(' \
      INSERT INTO `Benchmark` \
        (`id`, `Name`,`Gender`,`Age`,`DonutsEaten`) \
      VALUES ('+i+', "Name'+i+'", "Female", '+age+', '+donuts+')');
  }
  Logger.log("End");
}
```

Figure 1: Code excerpt from the test scripts.

make within the maximum six-minute time period imposed by Google Apps Script. From this, in order for the two databases to successfully execute within the time limit, three test sets with different amounts of operations were performed for each type of access, in order to obtain the response times perceived by GAS. Times were measured using the `Logger.log` function, which records the time it started and finished running the script.

4 RESULTS

Table 3 shows the maximum number of operations that each database performed in the six-minute time limit imposed by GAS.

It can be seen in Table 3 that Amazon RDS performed a slightly higher number of operations than Morpheus, in the interval of six minutes. Considering the limits expressed in Table 3, we set the following number of operations for the tests of access times: 100, 1000 and 3500.

For each type of operation, we executed the scripts 10 times for each set of operations in each of the databases, then we calculated the mean and the standard deviation for each case. Tables 4, 5, 6, 7 present the results for each type of operation.

Observing the results obtained, it can be said that both databases presented similar performance. Access times were slightly lower for Amazon RDS in all cases, but the standard deviation was lower for Morpheus in some cases. In all cases, the coefficient of variation was below 25%, suggesting homogeneity in the set of measures.

5 FINAL REMARKS

In this work, we investigated the performance of two cloud databases, through codes written in Google Apps Script and executed in the Google cloud. The codes were adaptations of the SysBench benchmark, which performs basic SQL operations on databases.

The results indicated similarity of performance among the cloud databases, although they are instances with non-identical configurations. Note that although a cloud service (GAS) is used to perform the access to databases, response times are feasible for applications that perform few operations. It is also worth mentioning that the instances used in this work are of low processing power, because they are both trials and free.

In general, it is noted that the greatest limitation comes from the GAS itself, when establishing that an

Table 3: Number of operations for each database in the 6-minutes limit.

Operation	Number	
	Amazon RDS	Morpheus
INSERT	4022	3763
SELECT	4271	4142
UPDATE	4022	3763
DELETE	4022	3730

Table 4: Execution times for INSERT operations.

INSERT				
# of Operations	Time (seconds)			
	Amazon RDS		Morpheus	
	Mean	Standard Deviation	Mean	Standard Deviation
100	9.65	0.01555634919	10.513	0.008485281374
1000	93.0389	2.5130575	92.9074	0.1704127343
3500	321.911	9.557255255	337.9359	0.9418662325

Table 5: Execution times for SELECT operations.

SELECT				
# of Operations	Time (seconds)			
	Amazon RDS		Morpheus	
	Mean	Standard Deviation	Mean	Standard Deviation
100	8.85745	0.2921058113	9.90745	0.02467802666
1000	86.91995	4.345807566	90.8749	0.7863027407
3500	291.8325	14.41012909	317.15945	0.1690692314

Table 6: Execution times for UPDATE operations.

UPDATE				
# of Operations	Time (seconds)			
	Amazon RDS		Morpheus	
	Mean	Standard Deviation	Mean	Standard Deviation
100	9.6315	0.4037579721	10.28395	0.1385222184
1000	91,56245	3,283167496	95.35045	3.625407178
3500	325.5305	18.83803176	340.2379	2.699733691

Table 7: Execution times for DELETE operations.

DELETE				
# of Operations	Time (seconds)			
	Amazon RDS		Morpheus	
	Mean	Standard Deviation	Mean	Standard Deviation
100	9.9109	0.04808326112	10.42595	0.05932625894
1000	90.68245	0.8153648294	93.3664	5.216326725
3500	306.7209	16.11284222	330.6804	17.72221726

execution can not exceed six minutes. In this sense, longer tasks can benefit from strategies that divide processing into multiple runs, as is done in some applications using GAS (Ferreira, 2014).

REFERENCES

Dzik, J., Palladinos, N., Rontogiannis, K., Tsarpalis, E., and Vathis, N. (2013). Mbrace: Cloud computing with monads. In *Proceedings of the Seventh Workshop on Programming Languages and Operating Systems*, PLOS '13, pages 7:1–7:6, New York, NY, USA. ACM.

Ferreira, J. (2014). *Google Apps Script, 2nd Edition – Web Application Development Essentials*. O'Reilly Media.

Google Inc. (2009). Google Apps Script. Available: https://developers.google.com/apps-script. Last accessed: March 2017.

Google Inc. (2017a). JDBC – Apps Script. Available: https://developers.google.com/apps-script/guides/jdbc. Last accessed: March 2017.

Google Inc. (2017b). Quotas for Google services. Available: https://developers.google.com/apps-script/guides/services/quotas. Last accessed: March 2017.

Hacigumus, H., Iyer, B., and Mehrotra, S. (2002). Providing database as a service. In *Proceedings 18th International Conference on Data Engineering*, pages 29–38.

Iosup, A., Yigitbasi, N., and Epema, D. (2011). On the performance variability of production cloud services. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 104–113.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). Cloudcmp: Comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 1–14, New York, NY, USA. ACM.

Mahmood, Z., editor (2014). *Cloud Computing: Challenges, Limitations and R&D Solutions*. Springer.

Mell, P. M. and Grance, T. (2011). The NIST definition of cloud computing. Technical report, Gaithersburg, MD, United States.

Murray, D. G. and Hand, S. (2010). Scripting the cloud with skywriting. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 12–12, Berkeley, CA, USA. USENIX Association.

Schwartz, B., Zaitsev, P., and Tkachenko, V. (2012). *High Performance MySQL, 3rd Edition*. O'Reilly.