A Predictive Load Balancing Service for Cloud-Replicated Databases

Carlos S. S. Marinho[1,2], Emanuel F. Coutinho[1], José S. Costa Filho[2], Leonardo O. Moreira[1,2], Flávio R. C. Sousa[2], Javam C. Machado[2]

[1]Instituto Universidade Virtual (UFC Virtual)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Laboratório de Sistemas e Banco de Dados (LSBD)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

sergio.marinho@lsbd.ufc.br, emanuel@virtual.ufc.br,

{serafim.costa,leonardo.moreira,flavio.sousa,javam.machado}@lsbd.ufc.br

Abstract. *Cloud computing emerges as an alternative to promote quality of service for data-driven applications. Database Management Systems must be available to support the deployment of cloud applications resorting to databases. Many solutions use database replication as a strategy to increase availability and decentralize the workload of database transactions between replicas. By the distribution of database transactions between replicas, load balancing techniques improve the computational resources utilization. However, several solutions use the current state of the database service in making decisions for the distribution of transactions. This paper proposes a predictive load balancing service for replicated cloud databases.*

1. Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [Mell and Grance 2011]. The cloud computing infrastructure is typically composed of a large number of physical machines, connected through a network. On each physical machine there is a variable number of virtual machines (VMs), according to the hardware capacity available on the physical machine. In these VMs, services are executed and usually must attend to some Service Level Agreement (SLA) contracted by the users. SLA can be defined as an obligation where the cloud provider gives its users some guarantees of Quality of Service (QoS) levels such as: performance and availability [Moreira et al. 2012].

Many cloud applications are data-driven, and because of this, Database Management Systems (DBMSs) are potential candidates for cloud deployment [Moreira et al. 2014]. Other reasons are: (i) in general, the DBMS installations are complex and involve a large amount of data, causing high hardware and software costs [Moreira et al. 2012]; (ii) most of the time spent processing in data-driven applications is related to processing in the DBMS [Sousa et al. 2012]. Several solutions are being proposed to use cloud databases with aspects of quality of service [Sousa and Machado 2012] [Moreira et al. 2014]. Some solutions that utilize database replication use load balancing

strategies to distribute workloads among the replicas. However, the identified existing solutions do not use predictive aspects to observe the future impact of the workload in relation to the use of existing cloud resources and SLA compliance.

The hypothesis of this research is as follows: *The use of prediction in load balancing solutions for replicated databases can maximize resource utilization in computational clouds while taking into account aspects of SLA*. From this hypothesis, the main objective is to design and implement a predictive load balancing service for replicated cloud databases. In order to achieve the main objective, the following specific objectives have been established: (i) to analyze load balancing techniques in replicated databases that can be used in computational clouds; (ii) to study techniques used to forecast workload in databases; (iii) to design a load balancing service that uses workload forecasting techniques for replicated cloud databases; and (iv) to evaluate whether the forecast techniques can be used to observe performance behavior of replicated cloud databases.

2. Related Work

The following filtering criteria were used to collect some related work: (i) works used in replicated cloud databases; (ii) works that adopted, in some way, load balancing between replicas; and (iii) works that used the relational model as a model of persistence. Table 1 lists the related works and highlights their characteristics. Sousa and Machado (2012) developed RepliC, an approach for total replication of relational database in the cloud. This work considers the aspects of quality of service, elasticity and multi-tenant model of shared DBMS. In RepliC there is the adjustment of the number of replicas according to the workload to comply the SLA. To divide the workload between the replicas, RepliC uses load balancing implemented as a circular queue (round-robin), distributing the transactions evenly across the existing replicas.

Moon et al. (2013) proposed SWAT, a middleware for load balancing on replicated cloud databases. SWAT uses the same multi-tenant model adopted by RepliC that does the DBMS sharing to host the replicas in the VMs. The load balancing strategy implemented by SWAT directs the workload to the replica that has greater computational resources availability at a given time. Although it seeks to better management and efficient use of existing resources in the cloud, SWAT does not solve problems related to SLA violations when resources are scarce, because it does not employ provisioning or elasticity. Pippal et al. (2015) developed a strategy based on partial replication, using the read-one-write-all model. This model proposes that the write requests are forwarded to all replicas and the read requests are only for the server that has the less available computational resources. In this work, CPU usage was considered for the requests distribution decision and experiments were made using the MySQL Server. Farther, it is necessary for the Database Administrator to configure which tables should be replicated together.

Table 1. Main related works and their characteristics

Work	Replication Strategy	Persistence Model	Forecast Techniques
Sousa and Machado (2012)	Total	Relational	No
Moon et al. (2013)	Total	Relational	No
Pippal et al. (2015)	Partial	Relational	No
Our Approach	**Total**	**Relational**	**Yes**

3. Our Approach

The predictive load balancing service for replicated cloud databases was designed as a service to be deployed in the QoSDBC [Sousa et al. 2012] system architecture. QoSDBC was designed to provide a data-persistence solution in relational databases with the multi-tenant model of shared DBMS, covering aspects of data distribution and quality of service in computational clouds. In the multi-tenant model of shared DBMS, different application databases can share the same DBMS. Barker et al. (2012) discuss some multi-tenant models for databases reinforcing that the shared multi-tenant DBMS model expresses the best relationship between provider resource usage, performance and security. QoSDBC was thought to be a generic system architecture and it have the possibility of being instantiated for several quality strategies for SLA compliance in cloud databases. Works proposed by [Moreira et al. 2014] and [Sousa and Machado 2012] used QoSDBC to provide service quality in cloud databases, using, respectively, migration and replication strategies. An overview of the QoSDBC system architecture can be seen in Figure 1.

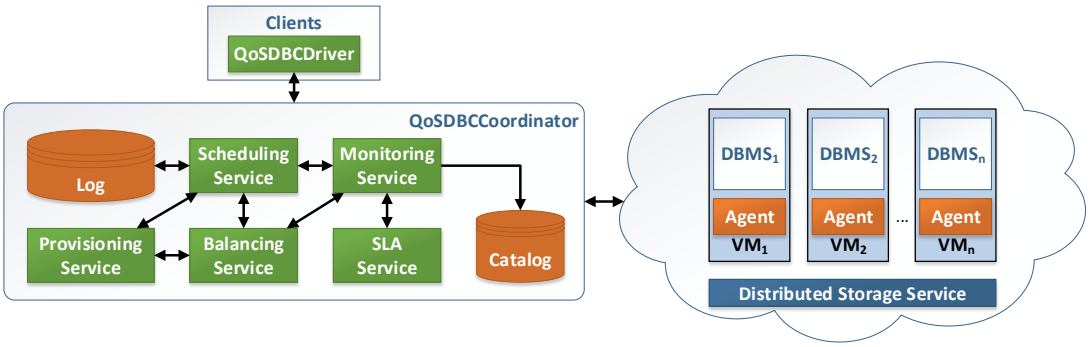

Figure 1. QoSDBC system architecture [Sousa et al. 2012]

The *QoSDBCDriver* is the component that provides access to the system, replacing the driver of a specific DBMS. It offers the same interface of communication with DBMS without the need of modifications in the DBMS. The *QoSDBCCoordinator* consists of a set of services that handle database management. The *Agent* is the component added in each VM, being responsible for collecting, monitoring and interacting with the VM and the DBMSs. The *Monitoring Service* manages the information collected by the *Agent*. This *Monitoring Service* aggregates information about processing, main and secondary memory of each VM, state of the DBMSs, workload summary, statistics etc. This information can be used to define the allocation of databases replicas in the DBMSs or to provision resources in order to guarantee the quality of service. All of this information is stored in the *Catalog* and continuously updated to fit the other services.

The *SLA Service* manages the information of the service level agreement defined between the user and the provider who has the QoSDBC installation. This information is related to SLA definitions, such as performance and consistency. This service provides parameters for the *Monitoring Service* to check the values set and adjust the other services. The *Provisioning Service* uses information from other services to provision VM resources. The *Scheduling Service* lists and selects the provisioned resources. For this, the scheduler manages the replicas, guaranteeing access to the DBMS during and after

the replication process. The *Log* stores all transactions their details that have been performed on the replicas of the system. Finally, the *Balancing Service* implements the strategy of distributing transactions between the replicas of the system. To do this, the *Scheduling Service* uses the *Balancing Service* to decide which replica to execute a transaction. Further details on the components and workflow of QoSDBC can be obtained in [Sousa et al. 2012].

The *Balancing Service* proposed in this paper uses a predictive strategy taking into account the monitoring data that is managed by the *Monitoring Service*. The prediction model adopted in this work was *AutoRegressive Integrated Moving Average* (ARIMA), since the studies carried out by [Santos et al. 2013] and [Moreira et al. 2014] showed that this forecast model has good prediction results of database workload in short prediction windows. Working with short-prediction windows helps adjust a highly dynamic environment, especially to avoid SLA violations. The *Balancing Service* requires that three parameters be configured: (i) the frequency that the prediction model should be trained and executed for each replica of the database schemas; (ii) the size of the prediction window that is returned for each replica of the database schemas; and (iii) the size of the monitoring data that must be entered into the ARIMA training step for each replica of the database schemas.

4. Preliminary Evaluation

The objective of the proposed evaluation is to verify whether the forecast techniques can be used to observe performance behavior of replicated cloud databases. For this, we will run RepliC [Sousa and Machado 2012] in a scenario with replication of cloud databases. The justifications to use RepliC are: (i) our service and RepliC's load balancing service can be implemented in the same system architecture, facilitating future comparisons between the works; (ii) both works use the same persistence model; and (iii) both works use the same replication strategy.

The OLTPBenchmark [Difallah et al. 2013] was used to create and generate database workloads in the experimental environment. OLTPBenchmark is a framework for evaluating the performance of different relational DBMSs against OLTP workload configurations. The framework has several benchmarks and with different data schemas, such as TPC-C, Twitter, YCSB, Wikipedia etc. OLTPBenchmark allows setting the time rate for submitting requests, setting the percentage of each type of transaction per experiment time, obtaining throughput information, average response time, and information on the use of operational system resources [Moreira et al. 2012]. Amazon EC2 was adopted as an environment for management of the VMs to execute the experiments. All the VMs used in the experiments have the Ubuntu Server 16.04 LTS. MySQL Server 5.7 with InnoDB engine and 128MB buffer was adopted as DBMS to manage the all databases. VMs of type *t2.small* were used to deploy databases. To host the *QoSDBCCoordinator*, a VM of type *c4.2xlarge* was used. VMs of type *t2.small* were used to run the OLTPBenchmark with *QoSDBCDriver* in order to simulate a workload for each database. All VMs were created in the same availability zone (*us-west-2b*).

The experiment scenario aims to observe the ARIMA performance when applied to a workload performed by RepliC. In this experiment, we used a *t2.small* VM to manage the databases. This VM has two databases of TPC-C, two databases of YCSB and

two databases of Wikipedia. Initially, all databases had 500MB of size. To simulate client connections, a total of 8 *t2.small* VMs were created, where each VM has 6 OLTP-Benchmark's instances. Each OLTPBenchmark's instance submits 50 connections to each database. Therefore, in total 300 connections were created for the DBMS in each *t2.small*. In total, 48 databases were created and evenly distributed in 8 *t2.small* VMs. The SLA metric used in the experiment was the average response time of the transactions and the SLA adopted for the YCSB was 60ms. The YCSB rate, per connection, was defined following the sequence: 250, 500, 750, 1000, 1000, 1000. Each transition in the sequence occurred every 10 minutes. The TPC-C and Wikipedia rates, per connection, were fixed in 10 and 100, respectively. For the use of ARIMA in the workload, one minute was inferred from every four minutes read from the RepliC monitoring system. Figure 2 shows the average response time of the workload performed in RepliC and the forecasted points inferred by ARIMA.

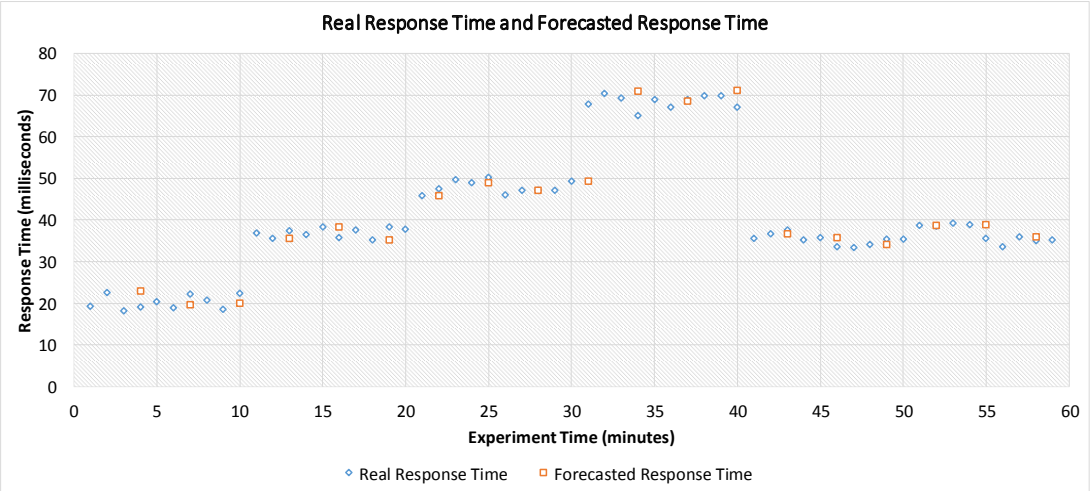

Figure 2. Real Response Time and Forecasted Response Time

After the time instant 30 minutes, RepliC detected the SLA violation for the YCSB. With this, a new *t2.small* VM was provisioned to provide a replica for the YCSB. After the availability of a new replica for the YCSB, there was a workload distribution between the replicas and the average response time decreased. From the graph shown in Figure 2 it can be seen that the forecasted points are close to the real points. Taking the points in the same timeline and plotting a time series with the real points and another one with the forecasted points, a comparison can be made to show present the accuracy level of the forecast with ARIMA. By using the *root-mean-square error* (RMSE) and *Mean Absolute Percentage Error* (MAPE) functions [Santos et al. 2013], a value can be reached which reveals the proximity of the real series with the forecasted series. Lower values of RMSE and MAPE indicate superior prediction accuracy. Calculating the values by means of the functions give the following results: RMSE = 4.925 and MAPE = 0.071. The values obtained by the RMSE and MAPE functions reinforce that ARIMA can be used in workloads for replicated databases in the cloud.

5. Conclusion

This paper presented a proposal for a predictive load balancing service for databases replicated in computational clouds. The proposed service is designed to be deployed on QoSDBC system architecture. In the design of the proposed service, the adoption of the ARIMA forecast model was considered because this model had good results for short prediction windows. Preliminary experiment was conducted and showed that ARIMA can be used to forecast workloads of replicated cloud databases. As future works we intend: (i) implement the prediction service designed in the QoSDBC system architecture; (ii) conduct deeper experiments, with a greater variety of workloads, and in several scenarios for replicating cloud databases; (iii) evaluate the performance of our approach with the use of other prediction models; and (iv) upgrade the QoSDBC system architecture to support the use of other multi-tenant and persistence models.

Acknowledgment

This research is a partial result of the project nº 455214/2014-0 supported by CNPq.

References

Barker, S., Chi, Y., Moon, H. J., Hacigümüş, H., and Shenoy, P. (2012). "cut me some slack": Latency-aware live migration for databases. In *EDBT '12*, pages 432–443, New York, NY, USA. ACM.

Difallah, D. E., Pavlo, A., Curino, C., and Cudré-Mauroux, P. (2013). Oltp-bench: An extensible testbed for benchmarking relational databases. *PVLDB*, 7(4):277–288.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. *National Institute of Standards and Technology (NIST)*.

Moon, H. J., Hacümüş, H., Chi, Y., and Hsiung, W.-P. (2013). Swat: A lightweight load balancing method for multitenant databases. In *EDBT '13*, pages 65–76, New York, NY, USA. ACM.

Moreira, L. O., Farias, V. A. E., Sousa, F. R. C., Santos, G. A. C., Maia, J. G. R., and Machado, J. C. (2014). Towards improvements on the quality of service for multi-tenant rdbms in the cloud. In *ICDE Workshops*, pages 162–169, Chicago, IL, USA.

Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2012). Analisando o desempenho de banco de dados multi-inquilino em nuvem. In *SBBD '12*, pages 161–168.

Pippal, S., Singh, S., Sachan, R. K., and Kushwaha, D. S. (2015). High availability of databases for cloud. In *INDIACom*, pages 1716–1722.

Santos, G. A. C., Maia, J. G. R., Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2013). Scale-space filtering for workload analysis and forecast. In *CLOUD '13*, pages 677–684. IEEE.

Sousa, F. R. C. and Machado, J. C. (2012). Towards elastic multi-tenant database replication with quality of service. In *UCC '12*, pages 168–175. IEEE.

Sousa, F. R. C., Moreira, L. O., Santos, G. A. C., and Machado, J. C. (2012). Quality of service for database in the cloud. In *CLOSER '12*, pages 595–601.