LABAREDA: A Predictive and Elastic Load Balancing Service for Cloud-Replicated Databases

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

Federal University of Ceará, Fortaleza, Brazil
{sergio.marinho, leonardo.moreira}@lsbd.ufc.br, emanuel@virtual.ufc.br, {serafim.costa, 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 among replicas. Due to the distribution of database transactions among replicas, load balancing techniques improve the computational resources utilization. However, several solutions use the current state of the database service to make decisions for the distribution of transactions. This article proposes a predictive and elastic load balancing service for replicated cloud databases. Experiments carried out showed that the use of prediction models can help to predict possible SLA violations in time series that represent workloads of cloud-replicated databases.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous; H.3 [**Information Storage and Retrieval**]: Miscellaneous; I.7 [**Document and Text Processing**]: Miscellaneous

Keywords: Load Balancing, Cloud-Replicated Databases, Performance

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 rapidly be 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. Each physical machine can have different hardware and software configurations, with variations in capacity in terms of CPU, memory and disk storage. 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]. In the present research, we focus on service latency, or on response time, as the desired SLA guarantee.

Many cloud applications are data-driven, hence Database Management Systems (DBMSs) are potential candidates for cloud deployment [Moreira et al. 2014]. The expansion of cloud databases is due to the success that DBMSs, especially the relational model, have in modeling a wide variety of applications, from different contexts [Agrawal et al. 2011]. 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]. [Agrawal et al. 2011] discuss that an important point for the use of cloud DBMSs is that this should be readily scalable to suit user demands which, unlike web-servers and application servers, is not so trivial.

Some solutions [Xiong et al. 2011] [Sakr and Liu 2012] [Sousa and Machado 2012] [Moon et al. 2013] [Fetai and Schuldt 2013] [Pippal et al. 2015] [Fetai et al. 2017] that utilize database replication use load balancing strategies to distribute workloads among the replicas. However, a small amount of solutions found in our research use elasticity combined with predictive aspects in order 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 models in load balancing solutions can help predicting possible SLA violations presented in time series that represent workloads of cloud-replicated databases.* From this hypothesis, the main objective is to design a predictive and elastic load balancing service for replicated cloud databases. In order to achieve that goal, the following tasks were established and executed: (i) analyze load balancing techniques in replicated databases that can be used in computational clouds; (ii) study techniques used to forecast workload in databases; (iii) explore elasticity strategies that can be used within replicated cloud databases; and (iv) evaluate whether the prediction models can help to predict possible SLA violations presented in time series that represent workloads of cloud-replicated databases.

This article presents LABAREDA, a predictive and elastic load balancing service for cloud-replicated databases. This article presents a significant extension of work [Marinho et al. 2017]. The new contributions mentioned in this article are as follows:

—We present a new system architecture to support predictive and elastic load balancing service for cloud-replicated databases.

—We present a strategy to elasticity for cloud-replicated databases based on ARIMA and EMA prediction models.

—We present a full prototype implementation and experimental evaluation of our end-to-end system. Our approach presents accuracy for workload prediction in cloud-replicated databases environment.

Problem Definition: In this work, we consider a set of DBMSs and virtual machines (VMs) in the cloud. Each DBMSs is composed of set of replicas and each VM has one or more replicas. The target workload is given by the rate of transactions. The SLA (i.e. response time) gives the objectives of each DBMSs. The main problem is how to load balance the current workload among replicas in order to maintain the SLA based on predictive models.

Organization: The remainder of this article is organized as follows. In section 2, we present and discuss existing load balancing approaches for cloud computing environments. Then, in section 3, we introduce and detail our approach. Evaluation and results' discussion of the proposed approach are explained in section 4. Finally, section 5 concludes the article.

2. RELATED WORK

This section presents and discusses related work that provide solutions for load balancing in replicated cloud databases. The systematic review of the present work is based on the model presented by [Coutinho et al. 2015]. The papers in this section were found using the following keywords: "load balancing", "database", "cloud" and "replication". These keywords were used to find papers in the following sources of scientific artifacts: ACM Digital Library[1] and IEEE Xplore Digital Library[2].

[1]https://dl.acm.org/
[2]http://ieeexplore.ieee.org/Xplore/home.jsp

Besides these two scientific artifacts, papers published in the Brazilian Symposium on Databases (SBBD)[3] and Journal of Information and Data Management (JIDM)[4] were considered.

The criteria created to select papers was: (i) works used in replicated cloud databases; (ii) works that adopted, in some way, load balancing among replicas; and (iii) works that used the relational model as a persistence model. The preference was given to works that implemented the following aspects in their strategies: elasticity and forecast. However, works that do not present these two aspects were not disregarded.

Curino et al. (2010) presented Schism, a partitioning and replication approach to distributed databases, based on workload recognition, to minimize the number of distributed transactions, since distributed transactions are expensive in OLTP. Schism is graph based and explores machine learning techniques, such as decision tree, to find the best partitioning strategy. The Schism entries are: a database, a training workload, and the number of partitions you want. The output is the partitioning and replication strategy that balances the size of partitions and minimizes the overall cost of running the workload. Although it improves computational resources utilization, the strategy is not focused on reducing SLA violations and does not even mention the use of forecasting techniques, as in LABAREDA. Another important difference between both works is that ours clearly addresses aspects of adding or removing VMs according to demand.

Xiong et al. (2011) developed SmartSLA, a system made to intelligently manage resources in a database system in a shared cloud. The tool has two components: the system modeling module and the resource allocation decision module. SmartSLA uses three machine learning techniques to decide how to dynamically adjust resource allocation: Linear Regression, Regression Tree and Boosting. Tests performed with TPC-W obtained very expressive results. This approach uses the master-slave model, which has the advantage of being simpler to maintain than the multi-master model. However, according to [Kemme et al. 2010], in a scenario where there are many write operations and the only master is becoming a bottleneck for recordings, the greater complexity of the multi-master becomes worthwhile, since more replicas that perform write operations are required. Considering this scenario, the multi-master model was adopted in our research.

Sakr and Liu (2012) presented a adaptative framework for dynamically provisioning replicas to ensure SLA compliance in relational databases. Their strategy is also concerned with scaling in when there are many underutilized resources. The SLA metric used is transaction execution time. The decisions are based in action rules, which are defined using a XML dialect, for example: add a replica if the average SLA violation percentage for transactions T1 and T2 exceeds 10% for a continuous period of more than 8 minutes. Although it has been shown to reduce the number of violations, the paper does not discuss forecasting to fit the SLA requirements. In our strategy, the use of prediction techniques is a central point for decision-making to maintain, remove or add replicas.

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 order to avoid SLA violations, RepliC adjusts the amount of replicas available according to the workload, as elastic aspects are used. Frequently, values obtained under monitoring are used to decide whether replicas should be provisioned, released or whether the resources are adequate for that demand. To divide the workload among the replicas, RepliC uses load balancing implemented as a circular queue (round-robin), distributing the transactions evenly across the existing replicas. Unlike our strategy, RepliC does not take advantage of workload forecasting to create replicas in advance and avoid future predictable SLA violations.

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. The load balancing strategy implemented

[3]http://sbbd.org.br/
[4]https://seer.ufmg.br/index.php/jidm/index

by SWAT directs the workload to the replica that has greater computational resources availability at a given time. Despite better management and efficient use of existing resources in the cloud, SWAT does not discuss aspects of elasticity. Therefore, SLA violations may happen when the available replicas are being used to the maximum. For under extreme conditions, it is likely that the less stressed replica will still be overloaded, failing to avoid SLA violations. LABAREDA proposes to solve that by exploring aspects of elasticity, providing more replicas when necessary.

Fetai and Schuldt (2013) proposed SO-1SR, a predictive load balancing approach for cloud-replicated databases using multi-master model. Two models of prediction were used: the *Exponential Moving Average* (EMA), to predict each replica's load to the next period and publish it to others. Linear Regression was used to predict the response time according to the type of workload. The evaluation indicates that SO-1SR is able to efficiently use existing resources and optimize the execution of transactions in an adaptive and dynamic manner. However, the paper does not discuss the use of load balancing to avoid SLA violations, one of the main points of our work. Another difference between the works is that although SO-1SR is designed for dynamic environments, it does not address aspects of elasticity.

Pippal et al. (2015) developed a strategy based on partial replication, using the read-one-write-all model. That model proposes that the write requests are forwarded to all replicas and the read requests are only forwarded for the server that has the less available computational resources. CPU usage was considered for the requests distribution decision and experiments were made using MySQL Server. Furthermore, it is necessary for the Database Administrator to configure which tables should be replicated. Unlike LABAREDA, the strategy does not use forecasting techniques to predict SLA violations, as there is no discussion of contract compliance and elasticity to ensure this. Instead, the proposed strategy builds on the present workload of the system to balance it.

Fetai, Stiemer and Schuldt (2017) developed QuAD, an adaptive quorum protocol for providing strong data consistency to a fully replicated database system with homogeneous nodes. One possible event that causes quorum readjustment is changes in node properties, such as workload. From the monitored data, the prediction model used to predict the changes of workload is the *Exponential Moving Average* (EMA). Hence, the quorum will be changed in order to optimize computational costs. Other events that trigger quorum readjustments are addition of new nodes and nodes failures. The experiments were conducted using TPC-C Benchmark in the Amazon Elastic Computing Cloud (EC2) environment. However, despite being dynamic, the main focus of the QuAD is not compliance with a maximum acceptable response time, nor elasticity, which contrasts with our work.

Marinho et al. (2017) proposed a predictive load balancing service for replicated cloud databases with the multi-tenant model of shared DBMS. The research hypothesis is that predicting workload can avoid SLA violations. The model used is the *AutoRegressive Integrated Moving Average* (ARIMA), which proved effective in forecast the future workload. The indicators used for that conclusion were *root-mean-square error* (RMSE) and *Mean Absolute Percentage Error* (MAPE), which allowed to check the proximity between the real series and the forecasted series. However, aspects of elasticity are not discussed and there is no use of forecasting techniques to promote addition of replicas. Thus, this article aims to extend that, making use of forecast models to make decisions to add or remove replicas.

Table I lists the related works and highlights their characteristics. All the mentioned works use the relational model as a persistence model.

3. LABAREDA

LABAREDA represents our service approach and it is an acronym of e**L**astic and predictive lo**A**d **BA**lancing for cloud-**RE**plicated **DA**tabases. This section begins with the presentation of a system architecture for the load balancing service and details its respective components. Next, the prediction

Table I. Related works and their characteristics

Work	Replication Strategy	Forecast Techniques	Elasticity
Curino et al. (2010)	Partial	No	No
Xiong et al. (2011)	Total	Yes	Yes
Sakr and Liu (2012)	Total	No	Yes
Sousa and Machado (2012)	Total	No	Yes
Moon et al. (2013)	Total	No	No
Fetai and Schuldt (2013)	Total	Yes	No
Pippal et al. (2015)	Partial	No	No
Fetai, Stiemer and Schuldt (2017)	Total	Yes	No
Marinho et al. (2017)	Total	Yes	No
Our Approach	**Total**	**Yes**	**Yes**

models used in our solution are detailed and discussed. Finally, the elasticity aspects used for adding and removing database replicas are presented and highlighted.

3.1 System Architecture

In our previous work, the load balancing service was designed to be deployed in the QoSBDC [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) discussed some multi-tenant models for databases reinforcing that the shared multi-tenant DBMS model expresses the best relationship among provider resource usage, performance and security.

In this present work, a new system architecture was designed based on some components and workflow of QoSDBC system architecture. With that, our goal is to make our solution a load balancing service that is independent of system architecture. However, the following QoSDBC's characteristics were maintained: multi-tenant model of shared DBMS, database total replication and relational database persistence model. An overview of the new system architecture including its modules and components can be seen in Figure 1. Next, all modules and components of the system architecture are detailed.

The *Service Interface* provides a set of methods that enable client interaction with the databases that are in the cloud. In a simplified way, this component is similar to service APIs (arrow 1), and is used for the client to transparently communicate with a database replica. In addition, the *Service Interface* provides forms for deploying databases in the cloud and defines an SLA requirements. The *Scheduler Module* interprets *Service Interface* requests for processing in the cloud databases. It maintains database performance settings in the cloud. Also, it coordinates load balancing of *Service Interface* requests while maintaining predictability and elasticity characteristics with a focus on databases performance in the cloud. In summary, the *Scheduler Module* acts as a manager among the components of the entire architecture, working as bridge between the user and our approach.

The *Configuration Database* is a database that maintains the access, maintenance, and performance characteristics (i.e., performance-oriented SLA) settings of the databases being used by the load balancing service. All data held in *Configuration Database* is managed by *Service Interface* through *Scheduler Module* (arrows 1 and 2). The *Monitoring Module* provides an interface for querying and inserting monitored data from workloads and databases' states deployed in the cloud (arrow 3). The *Monitoring Database*, in turn, stores the data that is managed by the *Monitoring Module* (arrow 4). The components *Configuration Database* and *Monitoring Module* are important to produce data in order to perform, mainly, two aspects: verification of SLA compliance and history of data replication workloads for predictions.

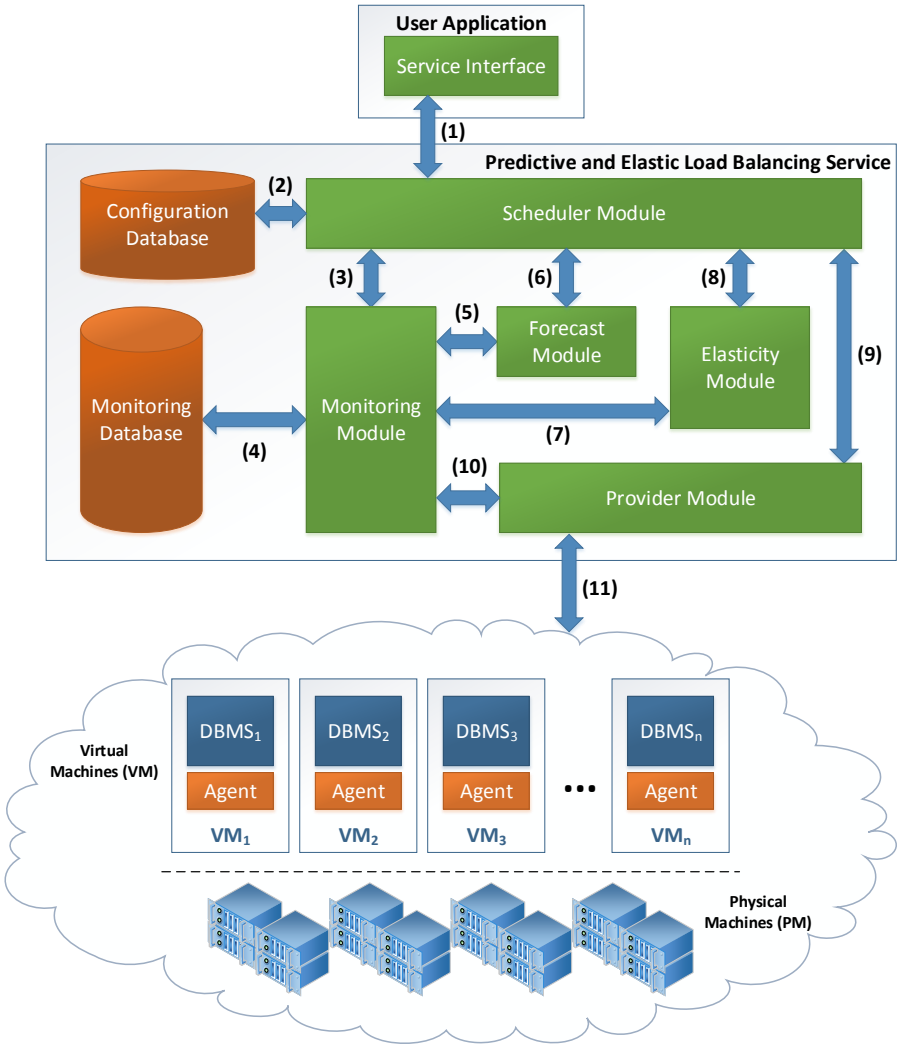


Fig. 1. System architecture

The *Elasticity Module* manages the elasticity metrics used to support the decision of provisioning or removing database replicas according to the workload and performance-oriented SLA of each database deployed in the cloud. The *Elasticity Module* uses information from the *Monitoring Module* (arrow 7) and *Configuration Database* (arrows 2 and 3), through the *Scheduler Module*, to decide provisioning or removing database replicas in the cloud. The *Forecast Module* maintains a set of prediction models that continuously produce predictive information about database replicas' workloads. The information produced by the *Forecast Module* is made available to the other modules through *Monitoring Module* (arrow 5). Briefly, *Forecast Module* component uses the monitoring data of the workload and SLA settings to make predictions. Finally, the *Forecast Module* provides the forecasts so that the *Scheduler Module*, through *Monitoring Module*, can make decisions of elasticity (arrow 8) or dispatch of the transactions between the existing replicas (arrow 11).

Agent is an embedded process in each VM that is part of the load balancing service. This process is responsible for collecting data from the execution of database transactions and also the VM's state (arrows 10 and 11). In addition, the *Agent* performs actions submitted by the *Scheduler Module*,

through the *Provider Module*, to maintain the VMs and replicas (arrows 9 and 11). Finally, the *Provider Module* encapsulates all interaction interfaces with the cloud provider. Examples of such interactions with the cloud provider are: add/remove virtual machines, add/remove database replicas, and manage accesses to the DBMSs. In addition, the *Provider Module* receives the monitoring data from the *Agent* and makes available in the *Monitoring Module* (arrows 10 and 11).

The *Predictive and Elastic Load Balancing Service* uses the monitored workload and transaction execution data as input to the prediction models. With the predictions results, our approach can verify which replica may violate the SLA or which replica is less overloaded in the future. Prediction models are run for all existing replicas, observing the monitored workload and transaction execution data. The replica with the shortest average response time of the transactions, in a future observation window, is considered the one that is less overloaded. If all existing replicas are overloaded, in other words, in all cases the SLA can be violated, a new replica is provisioned and that replica will receive the new transactions.

3.2 Prediction Models for Databases

Several works [Xiong et al. 2011] [Fetai and Schuldt 2013] [Moreira et al. 2014] [Fetai et al. 2017] have used prediction models to observe the future behavior of database workloads in order to fulfill some quality of service requirement. The use of predictive models can provide support in solving some of the challenges, such as allowing a database service to determine if certain features are required to meet certain SLAs and to adjust, according to model estimation, appropriate database configurations [Mozafari et al. 2013]. In this paper, we use the SLA metric of response time: The value of maximum time response expected for processing the input workload. LABAREDA uses an SLA associated with each database (e.g. $Wiki_{SLA} = 0.5$s).

In our *Forecast Module* the following prediction models were used: *Autoregressive Integrated Moving Average* (ARIMA) and *Exponential Moving Average* (EMA). ARIMA decomposes the data into: an automatic regressive (AR) process; in which there is a memory of past events; (ii) an integrated (I) process, which makes the data stationary and ergodic; and (iii) a moving average (MA) of the forecast errors. Normally, this model is denoted by $ARIMA(p, d, q)$, where p is the number of auto-regressive terms, d is the number of non-seasonal differences, and q is the number of terms of the moving average. Besides these three variables, there are others that can be seen in the mathematical definition presented in 1.

The lag operator is represented by L, while ϕ is a polynomial associated with the auto-regressive operator of order p and θ represents the polynomial associated with the moving average operator of order q. Finally, ϵ_t is white noise, which is a discrete signal whose samples are a sequence of non-self-correlated random variables with zero mean and finite variance [Box and Jenkins 1976]. Studies carried out by [Santos et al. 2013], [Moreira et al. 2014] and [Marinho et al. 2017] showed that ARIMA 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.

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_t \quad (1)$$

EMA is a weighted average of values of n observations considered. The weights decrease exponentially, according to how recent the data is. Thus, the most recent data is of greater importance than the older values. As observed in the 2 equation, at each prediction, a new prediction interval is initiated and an old one is closed, that closed interval is denoted by p, and the next interval, which is desired for prediction, is denoted by $p+1$. The variable α which has a value between 0 and 1, is known as a *smoothing factor* and its choice affects the quality of the prediction. Thus, in EMA, forecasts

are recursively based on the past and need the predicted load for the last period [Fetai et al. 2017]. Using EMA, the first value is initialized to the arithmetic mean of the first n measures, which can be observed in the 3 equation. his model detects quickly the changes in workload behavior, making EMA a good model for rapid decision making [Andreolini and Casolari 2006].

$$EMA(load_{p+1}) = \alpha \cdot load_p + (1 - \alpha) \cdot EMA(load_p)) \tag{2}$$

$$EMA(load_{p+1}) = \frac{\sum\limits_{0 \leq i \leq n} load_i}{n} \tag{3}$$

According to [Andreolini and Casolari 2006], EMA seems to be more appropriated to run-time decision making than popular linear automatic regression models, such as ARIMA. Therefore, if the analyzed variables change behavior frequently, ARIMA needs frequent updating of its parameters, increasing the computational costs, which not occur using EMA. Hence, ARIMA is commonly used offline after examining all available data, or applied to workload with little variability and high self-correlation of load measurements.

3.3 Elastic Database Replication

Often infrastructure-based metrics are used to assess the elasticity of an environment. Some studies have proposed more generic metrics for measuring elasticity, as in [Coutinho et al. 2016a]. Other works proposed an architecture for the provision of cloud elasticity based on autonomic computing [Coutinho et al. 2016b]. Metrics related to the database or directly associated with the application in use can be considered for the evaluation of the elasticity. Such metrics can support the development of architectures that aggregate elasticity in the support of predictive solutions in database. In the context of replicated cloud databases, elasticity is achieved through self provisioning, which adds new replicas if the system can not handle the current workload or removes unnecessary replicas. In the present research, we focus on service latency, or on response time, as the desired SLA guarantee.

This work uses the elasticity model adopted in [Sousa et al. 2018], where the quality metric used for the SLA specification is the average response time of database transactions. In [Sousa et al. 2018], the forecast solution periodically observes and predicts the workload in the various replicas of the environment. Our approach continuously checks SLA compliance by monitoring it. If the monitored value is not in accordance to the defined SLA, more resources are added. On the other hand, if the SLA is satisfied over time, resources are removed. The time until violation and replication time is provided by the forecasting module.

In this work, replication aspects are handled by the MySQL Cluster [MySQL 2018]. This cluster focus on synchronous and multi-master replication. Each replica can accept write operations and data within each replica is synchronously replicated to at least one other data node. If a data node fails, then there is always at least one other data node storing the same information. Replication uses *one-copy serializability* as its model of correctness.

Multi-master replication allows data to be stored by a group of virtual machines, and updated by any machine of the group. The multi-master replication system is responsible for propagating the data modifications made by each machine to the rest of the group, and resolving any conflicts that might arise between concurrent changes made by different machine. All queries are executed locally on the node, and there is special handling only on commit. When the commit operation is issued, the transaction has to pass certification on all nodes. If it does not pass, you will receive error as the response for that operation. After that, the transaction is applied on the local node. More details can be obtained from [MySQL 2018].

4. EVALUATION

The main objective of the experimental evaluation is to verify if the prediction aspects in a load balancing service can reduce SLA violations through workload analysis represented in a time series. For this, we use a workload, presented in [Sousa et al. 2018] for multi-tenant cloud-replicated databases. Our experimental evaluation will be focused on the accuracy' analysis of the ARIMA and EMA models. The choice of these two models is justified by the fact that the related works point these two models as interesting for predictive analysis of workloads in replicated databases.

4.1 Environment

The OLTPBenchmark [Difallah et al. 2013] was used to create and generate database workloads in the experimental environment. OLTPBenchmark is a framework written in Java 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. MySQL Cluster 7.6 has been configured to manage replication and consistency aspects. VMs of type *t2.small* were used to deploy databases. To host the *coordination service* (service that receives a request and directs to a database replica), a VM of type *c4.2xlarge* was used. VMs of type *t2.small* were used to run the OLTPBenchmark in order to simulate a workload for each database. All VMs were created in the same availability zone (*us-west-2b*).

4.2 Evaluation Scenario

The experiment scenario aims to observe the prediction models' accuracy when applied to a cloud replicated databases workload. In this experiment, we used a *t2.small* VM to manage the databases. The 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 OLTPBenchmark'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 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.

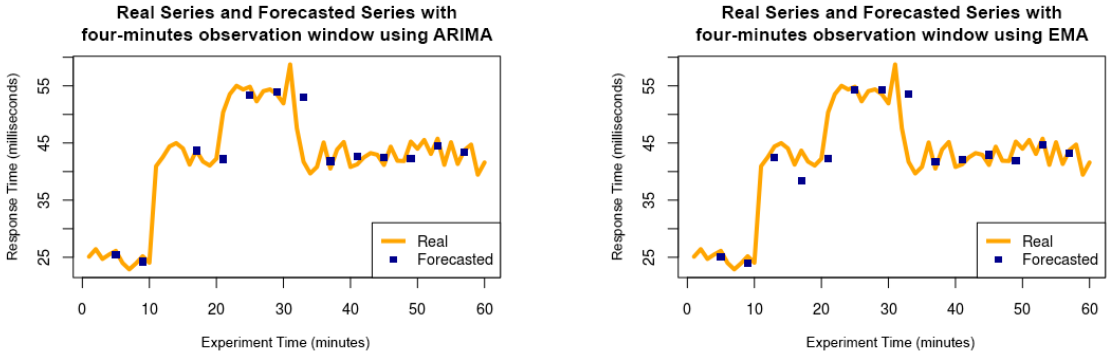
Through the *Monitoring Module*, the average response times of YCSB transactions were retrieved. These average response times were observed by the *Forecast Module* using the ARIMA and EMA models. We investigate which prediction models had better results when applied to workload monitoring data from cloud-replicated databases. For that, the error value, for each prediction model, will be compared between the forecasted time series and the real time series by means of the *root-mean-square error* (RMSE) and *Mean Absolute Percentage Error* (MAPE) functions.

The mathematical definitions of RMSE and MAPE can be visualized in functions 4 and 5 respectively, where yi denotes the value of the real output, $\hat{yi}$ is the expected output value and n is the number of observations in which a predicted value and a real value were obtained, so that they are used to measure the error

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{4}$$

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{y_i} \tag{5}$$

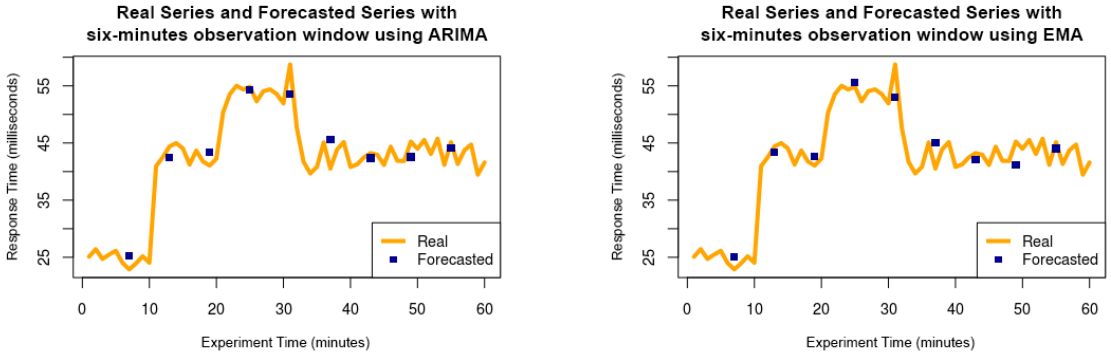
By using the RMSE and MAPE functions, a value can be reached which reveals the proximity of the real series with the forecasted series [Santos et al. 2013]. Lower values of RMSE and MAPE indicate superior prediction accuracy [Marinho et al. 2017]. Figures 2(a) and 2(b) demonstrate, respectively, the prediction results of ARIMA and EMA with a four minutes observation window.



(a) ARIMA (b) EMA

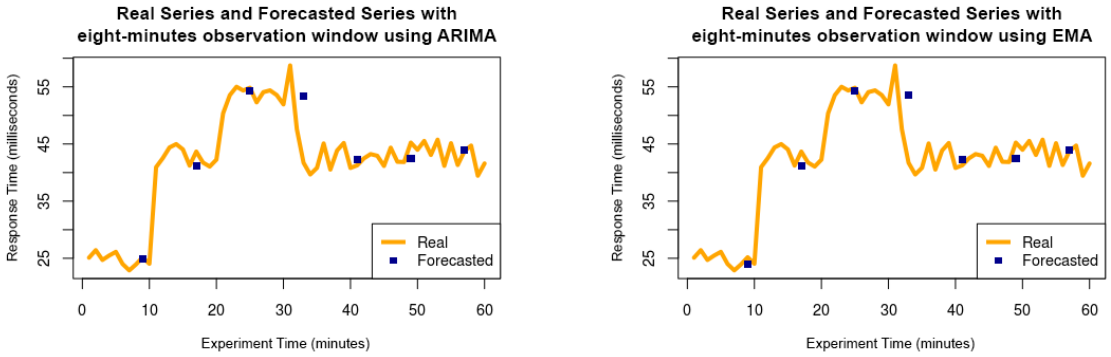
Fig. 2. Comparison between ARIMA and EMA for a four-minutes observation window

We increased the size of the prediction window to observe the impact of window size on the prediction accuracy with the models used. Figures 3(a) and 3(b) show, respectively, the prediction results of ARIMA and EMA with a six minutes observation window. Finally, Figures 4(a) and 4(b) illustrate, respectively, the prediction results of ARIMA and EMA with an eight minutes observation window.



(a) ARIMA (b) EMA

Fig. 3. Comparison between ARIMA and EMA for a six-minutes observation window



(a) ARIMA (b) EMA

Fig. 4. Comparison between ARIMA and EMA for an eight-minutes observation window

Table II. Accuracy's analysis of the prediction models in the different observation windows

Observation window	ARIMA		EMA	
	MAPE	RMSE	MAPE	RMSE
3 minutes	7.17	4.23	6.48	3.48
4 minutes	5.60	4.05	6.78	4.43
5 minutes	9.41	6.15	10.95	6.54
6 minutes	5.87	2.92	5.81	3.00
7 minutes	4.9	3.92	4.42	3.90
8 minutes	6.41	4.67	6.96	4.71

In total we varied the observation window from 3 minutes to 8 minutes and calculated the RMSE and MAPE for all observation windows with the two prediction models. After the time instant around 25 minutes, the solution detected the need to provision new resources for the YCSB. With this, a new *t2.small* VM was provisioned to provide a replica for the YCSB. After the availability and synchronization of a new replica for the YCSB, there was a workload distribution among the replicas and the average response time decreased.

From the graphs shown in Figures 2, 3 and 4 it can be seen that the forecasted points are close to the real points. By plotting a time series of the actual points and another with the predicted points, it is possible to compare the accuracy level of the forecast with ARIMA and EMA. Table II details the accuracy's analysis results of the ARIMA and EMA models in the different observation windows of workload described in this experimental setup.

According to the results of MAPE and RMSE, it can be seen that both models have similar accuracy. For example, ARIMA obtained better results in the following observation windows: 4, 5 and 8 minutes. The EMA obtained better results in the following observation windows: 3, 6 and 7 minutes. Given these results, we could analyze the feasibility of testing other classes of prediction models in time series, for example: deep learning and genetic algorithms. Our experiments have shown that both prediction models have similar results. With that, we could verify if these classes of prediction models have better accuracy than the ARIMA and EMA.

It is also interesting to evaluate, in other experiments, the complexity of execution, training and parameterization of the prediction models. In our experiments we used the standard implementations and parametrizations of ARIMA and EMA. Table II shows that in 5-minute observation windows, there was a fall in accuracy for both prediction models. The combination of time series points, monitored for 5 minutes of observation, generates a prediction with less accuracy than the other combinations of points. To improve the accuracy of prediction models, a deeper study is needed to verify the best

window of observation for a given prediction model, taking into account the experimental environment and workload aspects.

5. CONCLUSION

This article presented LABAREDA, a predictive and elastic load balancing service for databases replicated in computational clouds. The following extensions of the work proposed in [Marinho et al. 2017] were highlighted: (i) a new system architecture to support the development of a predictive and elastic load balancing service for cloud-replicated databases; (ii) addition of a strategy to add elasticity characteristics for cloud-replicated databases; (iii) adoption of other prediction models in the load balancing decision; and (iv) new experiments to show the behavior of a load balancing service with prediction characteristics. According to the experiments, the ARIMA and EMA prediction models obtained similar accuracy for time series prediction.

As future work we intend to: (i) adopt other classes of prediction models, for example deep learning and genetic algorithms; (ii) evaluate the joint aspects of prediction and elasticity to avoid SLA violations; (iii) upgrade the system architecture to support the use of other multi-tenant and persistence models; (iv) conduct other experiments, with a greater variety of workloads, and in several scenarios for replicating cloud databases in order to evaluate our load balancing and replication techniques; and (v) study techniques that can help you choose the best window of observation for a given prediction model, taking into account the experimental environment and workload aspects. On the latest ambition, there are benchmarks that can give us a deeper insight into our strategy and performance, such as Model-based Database Stress Testing (MoDaST). This strategy uses a state transition model in a run-time DBMS, dynamically changing volumes and workload. Some of the possible states are: Thrashing, Stress and Steady. MoDaST also has prediction aspects, such as predicting database failures before entering the Thrashing state, indicating when this will occur [Meira et al. 2016].

Acknowledgment

This research is a partial result of the project number 455214/2014-0 supported by CNPq. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

Agrawal, D., El Abbadi, A., Das, S., and Elmore, A. J. Database scalability, elasticity, and autonomy in the cloud. In *Database Systems for Advanced Applications*, J. X. Yu, M. H. Kim, and R. Unland (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 2–15, 2011.

Andreolini, M. and Casolari, S. Load prediction models in web-based systems. In *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*. valuetools '06. ACM, New York, NY, USA, 2006.

Barker, S., Chi, Y., Moon, H. J., Hacigümüş, H., and Shenoy, P. "cut me some slack": Latency-aware live migration for databases. In *Proceedings of the 15th International Conference on Extending Database Technology (EDBT)*. ACM, New York, NY, USA, pp. 432–443, 2012.

Box, G. E. and Jenkins, G. M. *Time series analysis: forecasting and control, revised ed.* Holden-Day, 1976.

Coutinho, E. F., Rego, P. A., Gomes, D. G., and de Souza, J. N. Physics and microeconomics-based metrics for evaluating cloud computing elasticity. *J. Netw. Comput. Appl.* 63 (C): 159–172, Mar., 2016a.

Coutinho, E. F., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. An architecture for providing elasticity based on autonomic computing concepts. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. ACM, New York, NY, USA, pp. 412–419, 2016b.

Coutinho, E. F., Sousa, F. R. C., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. Elasticity in cloud computing: a survey. *Annales des Télécommunications* vol. 70, pp. 289–309, 2015.

Curino, C., Jones, E., Zhang, Y., and Madden, S. Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment* 3 (1-2): 48–57, 2010.

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

Fetai, I. and Schuldt, H. So-1sr: Towards a self-optimizing one-copy serializability protocol for data management in the cloud. In *Proceedings of the Fifth International Workshop on Cloud Data Management*. CloudDB '13. ACM, New York, NY, USA, pp. 11–18, 2013.

Fetai, I., Stiemer, A., and Schuldt, H. Quad: A quorum protocol for adaptive data management in the cloud. In *2017 IEEE International Conference on Big Data (Big Data)*. pp. 405–414, 2017.

Kemme, B., Jiménez-Peris, R., and Patiño-Martínez, M. *Database Replication*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

Marinho, C. S. S., Coutinho, E. F., Costa Filho, J. S., Moreira, L. O., Sousa, F. R. C., and Machado, J. C. A Predictive Load Balancing Service for Cloud-Replicated Databases. In *32nd Brazilian Symposium on Databases (SBBD)*. Brazilian Computer Society (SBC), Uberlândia, Minas Gerais, Brazil, pp. 210–215, 2017.

Meira, J. A., de Almeida, E. C., Kim, D., Filho, E. R. L., and Le Traon, Y. "overloaded!" — a model-based approach to database stress testing. In *Database and Expert Systems Applications*, S. Hartmann and H. Ma (Eds.). Springer International Publishing, Cham, pp. 207–222, 2016.

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

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

Moreira, L. O., Farias, V. A. E., Sousa, F. R. C., Santos, G. A. C., Maia, J. G. R., and Machado, J. C. Towards improvements on the quality of service for multi-tenant RDBMS in the cloud. In *30th International Conference on Data Engineering Workshops (ICDEW)*. IEEE Computer Society, Chicago, IL, USA, pp. 162–169, 2014.

Moreira, L. O., Sousa, F. R. C., and Machado, J. C. Analisando o desempenho de banco de dados multi-inquilino em nuvem. In *27th Brazilian Symposium on Databases (SBBD)*. Brazilian Computer Society (SBC), São Paulo, São Paulo, Brazil, pp. 161–168, 2012.

Mozafari, B., Curino, C., and Madden, S. Dbseer: Resource and performance prediction for building a next generation database cloud. In *CIDR*, 2013.

MySQL. Mysql cluster. `https://www.mysql.com/products/cluster/`, 2018.

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

Sakr, S. and Liu, A. Sla-based and consumer-centric dynamic provisioning for cloud databases. In *2012 IEEE Fifth International Conference on Cloud Computing*. pp. 360–367, 2012.

Santos, G. A. C., Maia, J. G. R., Moreira, L. O., Sousa, F. R. C., and Machado, J. C. Scale-Space Filtering for Workload Analysis and Forecast. In *6th International Conference on Cloud Computing (CLOUD)*. IEEE Computer Society, pp. 677–684, 2013.

Sousa, F. R. C. and Machado, J. C. Towards elastic multi-tenant database replication with quality of service. In *5th International Conference on Utility and Cloud Computing (UCC)*. IEEE, pp. 168–175, 2012.

Sousa, F. R. C., Moreira, L. O., Costa-Filho, J. S., and Machado, J. C. Predictive elastic replication for multi-tenant databases in the cloud. *Concurrency and Computation: Practice and Experience* 0 (0): e4437, 2018.

Sousa, F. R. C., Moreira, L. O., Santos, G. A. C., and Machado, J. C. Quality of service for database in the cloud. In *2nd International Conference on Cloud Computing and Services Science (CLOSER)*. Porto, Portugal, pp. 595–601, 2012.

Xiong, P., Chi, Y., Zhu, S., Moon, H. J., Pu, C., and Hacigumus, H. Intelligent management of virtualized resources for database systems in cloud environment. In *2011 IEEE 27th International Conference on Data Engineering*. pp. 87–98, 2011.