A Machine Learning Approach for SQL Queries Response Time Estimation in the Cloud

Victor A. E. Farias, José G. R. Maia, Flávio R. C. Sousa
Leonardo O. Moreira, Gustavo A. C. Campos, Javam C. Machado

Universidade Federal do Ceará (UFC), Brazil
{gilvanm, sousa, leoomoreira, gsantos, javam}@ufc.br, victorfarias@lia.ufc.br

Abstract. Cloud computing provides on-demand services with pay-as-you-go model. In this sense, customers expect quality from providers where QoS control over DBMSs services demands decision making, usually based on performance aspects. It is essential that the system be able to assimilate the characteristics of the application in order to determine the execution time for its queries' execution. Machine learning techniques are a promising option in this context, since they offer a wide range of algorithms that build predictive models for this purpose based on the past observations of the demand and on the DBMS system behavior. In this work, we propose and evaluate the use of regression methods for modeling the SQL query execution time in the cloud. This process uses different techniques, such as bag-of-words, statistical filter and genetic algorithms. We use K-Fold Cross-Validation for measuring the quality of the models in each step. Experiments were carried out based on data from database systems in the cloud generated by the TPC-C benchmark.

Categories and Subject Descriptors: H.Information Systems [**H.m. Miscellaneous**]: Databases

Keywords: Machine Learning, QoS, Cloud Computing

1. INTRODUCTION

Cloud computing is an extremely successful paradigm of service-oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Scalability, elasticity, pay-per-use pricing, and economies of scale are the major reasons for the successful and widespread adoption of cloud infrastructures. Since the majority of cloud applications are data-driven, database management systems (DBMSs) powering these applications are critical components in the cloud software stack [Elmore et al. 2011].

Many companies expect cloud providers to guarantee quality of service (QoS) using service level agreement (SLAs) based on performance aspects. It is crucial that providers offer SLAs based on performance to their customers. Thus, it is important that quality is applied to the DBMS to control the consumed time [Sousa et al. 2012]. Controlling the QoS in DBMSs often needs an estimation for time and resources required to perform a query. Thereby, knowledge about the execution time of an incoming query provides a valuable variable for decision making in the cloud.

In this work, we propose and evaluate the use of regression methods for modeling the SQL query execution time in the cloud. This process is carried out as follows. First, tokens from SQL statement samples are represented using a Bag-of-Words (BoW). Then model selection is performed along with parameter tuning in order to improve accuracy of the regression method. At this point, we perform feature selection using both a statistical filter and a genetic algorithm to reduce dimensionality of input vectors. Since each sample contains a myriad of features from BoW, K-Fold Cross-Validation is used for measuring model quality all along this process. Experiments were carried out in order to evaluate our approach based on data from database systems in the cloud generated by the benchmark TPC-C [TPC 2013]. To the best of our knowledge, this is the first paper to formulate and explore the problem of how to work the bag-of-word of SQL queries with the goal of estimating response time for queries.

This paper is organized as follows: Section 2 explains theoretical aspects of this paper. The proposed approach is detailed in Section 3. The implementation and evaluation of our solution is described in Section 4. Section 5 surveys related work and, finally, Section 6 presents the conclusions.

2. BACKGROUND

- BoW is a widely used simple model for text processing. Given a set of text samples, BoW first extracts each word (token) from all samples, thus building a "dictionary". Hence, a specific text is represented as a "token histogram", i.e., by a vector whose each component corresponds to the frequency of a given token in the dictionary for that text;
- Genetic algorithm is an optimization technique for finding approximate solutions inspired by the evolutionary theory. Such technique is typically used when the search space is very large, perhaps infinite. Evolution theory simulation is carried out over a population of individuals based on natural selection, inheritance, crossover and mutation operations. In this context, each individual represents a solution for the problem that is evaluated using a known fitness function;
- K-fold cross-validation (K-fold CV) provides a measure of accuracy for a particular model from a dataset. The generalization capacity of a model is evaluated by partitioning the dataset into k mutually exclusive sets with equal sizes. Thereby, each partition is used as test set and the other k-1 partitions are used as training set. A model is trained and evaluated for each of these k settings. Finally, the mean of errors found in each setting is used as the accuracy measure;
- For regression tasks, we used three supervised learning algorithms: (1) Gradient Boosting Regression (GBR) [Friedman 2002] [Friedman 2000] builds a set of weak learners, one by one, and each predictor is built using the residual of the last predictor built, and this error is optimized by using gradient descent algorithm on the differentiable loss function; (2) Random Forest Regression (RFR) [Breiman 2001] [Breiman 1998], on its turn, builds simple regression trees over random subsets of the dataset in order to generate a more complex predictor. Thus, the RFR analyses the contribution of each simple predictor to compute the final result; and (3) Support Vector Regression (ϵ-SVR) [Vapnik 1998] in an adaptation of the Support Vector Machine method, which was developed as a binary classifier, for dealing with regression based on the ϵ-loss function foundation. It makes an ϵ-tube around the training samples so that the samples within the ϵ-tube are not counted as errors, while samples outside of the ϵ-tube become support vectors that will be used for test;
- F-test is a statistical test, in which f-distribution is used. In this context, f-distribution is used as the null distribution, assuming the null hypothesis is true, i.e., there is no relationship between two measured phenomena. Disproving such hypothesis implies that a given model is well suited for explaining a dataset. The null hypothesis is that two independent normal variances are equal, and the observed sums of some appropriately selected squares are then examined to see whether their ratio is significantly incompatible with this null hypothesis. The p-value is the probability of rejecting the null hypothesis when it is true.

3. OUR APPROACH

The proposed approach builds on top of different techniques to address the predictive query time execution problem before its execution. This problem can be treated as a machine learning problem, in particular, as a regression problem, since the phenomenon to be estimated can be described as a continuous variable. Regression models have the ability to model a target function by means of the application of an algorithm on a training set. In our context, such model is able to estimate the execution time of a query that has not yet been submitted to the system. Regression models demand the application of a feature extraction algorithm, to generate feature vectors from an SQL query.

In this work, we focused on analyzing only the text of the SQL query. Such approach can generate less accurate models, but in general, the process of obtaining the vectors is less costly than an approach using data from the catalogs of the DBMS, because there is an additional cost of I/O. Thus, SQL statements issued to the DBMS and their corresponding execution times are used for building feature vectors comprising the dataset. An overview of our approach is depicted in Figure 1. The Feature Extraction (A) phase consists in extracting feature vectors from a mass of SQL data. For this purpose, we used the technique of vector representation BoW.

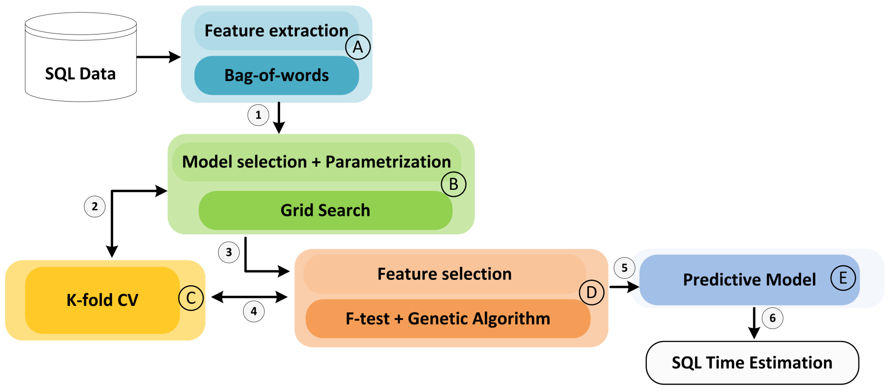

Fig. 1. Our Approach

Several regression methods were tested in our experiments, but we will focus only in those that had the best performance, which were: SVR, RFR and GBR. These methods have a large number of parameters. In addition, the change of parameters causes uncontrolled, non-intuitive effects to the model. To circumvent this problem, we use the technique of grid search to find a parameter configuration for each method, thus optimizing its accuracy. This technique consists of setting manually a set of parameters configuration, and exhaustively training a model for each of them and evaluating this model with a performance metric.

In order to tune the parameters, the Model Selection phase (B) receives such data set (1) and performs a grid search over the most relevant, method-specific parameters. Each trained model is then evaluated using K-fold CV (2) in order to estimate its associated error. For K-fold CV method (C), the designated error metric is the mean absolute error from the expected value to the estimated target values of each partition. Experiments performed with BoW vectors generated from SQL queries resulted in very high number of components. Training a regression model with such dimensionality can be computationally unfeasible and, moreover, there may exist features that do not explain the phenomenon which may mislead the regression method producing inaccurate predictive models. Therefore a process of dimensionality reduction is applied, by selecting a subset of features that best describe the phenomenon.

The Feature Selection (D) uses the models with tuned parameters received from the Model Selection (3). And it is done in two stages: First, a statistical filter is applied. Let X be the set of all features in the dataset, the p-value score is computed for every features in relation to the target feature, thus iteratively we generate sets of features $V_1, V_2, ...$ in 3 steps:(I) We start the iteration with set of features V_1 containing only the feature with the highest p-value and $i = 0$; (II) $i = i + 1$, generate a new set V_i such that $V_i = V_{i-1} \cup u$ where u and is the feature with highest p-value in $V_{i-1} - X$; (III) Measure the error for the newly trained model with this new configuration using K-fold CV; (IV) if $i \leq 50$ and if the error associated for model with features V_i has decreased compared with the model with features V_i, stop. Otherwise, go to (II). At the end, we select the model that obtained the lowest error; Second stage, using genetic algorithm, let $X = x_1, x_2, \ldots, x_k$ be the features selected by statistical filter. Each individual in a population represents a subset of X, thus an individual I in the population is given by a binary string S of size k. Thereby, x_i belongs to the subset of X represented by I if and only if $S_i = 1$ and otherwise if $S_i = 0$, $\forall i \quad 1 \leq i \leq k$. The fitness function designated to measure the quality of each individual is the K-fold CV (4). Therefore, this step outputs the best model generated (5). This way, it is produced an accurate and not computationally expensive composed predictive model (E), which can predict the response time for a unseen SQL query that has not yet been executed in the system.

4. EXPERIMENTS

4.1 Environment

We deploy a prototype of our approach and conduct experiments on a virtual machine on OpenNebula infrastructure at Federal University of Ceara [OpenNebula 2013]. This machine has a 1.0 GHz Xeon processor, 1.7 GB memory and 160 GB disk capacity. We use the Ubuntu 12.04 operating system and MySQL 5.5. According to [OLTPBenchmark 2013], the following database was generated based on TPC-C with 400 MB. Regarding the regression methods, we used libsvm [Chang and Lin 2011] for Support Vector Regression, Scikit-learn [Pedregosa and et al. 2011] for Random Forest Regression and Gradient Boosting Regression and Pyevolve [Perone 2009] for the genetic algorithm implementation.

The SQL data was generated using this environment and a large amount of SQL queries was generated. Therefore, for purposes of experiments, we generated four data dets. Three of them are composed by the last 2000 executed queries in the benchmark of a single type - Select, Insert or Update - and the fourth is composed only by Delete queries, and it has only 400 queries due to the low occurrence of Delete queries in TPC-C.

4.2 Experimental Results

The full experiment was executed for each data set (Select, Insert, Update and Delete), using each regression method (SVR, RFR and GBR) as a model provider for k-fold CV. The tests were made using 4-fold CV. Firstly, the feature extraction step is executed on each dataset. The four sets of vectors are directly given as input for Model Selection step. The grid search is performed over each pair of Regression Method and dataset, the Mean absolute error (MAE) obtained and the number of features (#) is presented in Table I.

	Select		**Insert**		**Update**		**Delete**	
Regression Methods	MAE	#	MAE	#	MAE	#	MAE	#
Support Vector Regression	73.62	968	14.96	5182	162.61	2021	6.56	34
Gradient Boosting Regression	74.21	968	14.90	5182	154.36	2021	6.54	34
Random Forest Regression	85.18	968	19.79	5182	167.31	2021	9.78	34

Table I. Model Selection (MAE in milliseconds)

Insert and Update queries add information on the Data Base, so they tend to have a wider variety of word, and thus, they have more words than the other types. Delete queries have so few features because of the number of delete queries in TPC-C and, in addition, the actions of the queries are always over the same table. Delete and Insert queries have a lower MAE because their response times are low and do not vary much. Differently from Select and Update queries, which can reach a response time of 6000 ms. In general, all the queries types have a large dimensionality. For this reason, we search for a smaller set of explanatory features. The Statistical Filter is then applied as we can in Table II.

Note that we can describe most of the events with few features, and with an acceptable trade-off between number of features and obtained MAE. Furthermore, in some cases the MAE has decreased. Now, we run the genetic algorithm with 30 individuals and 30 generations. At each generation, the best individual is preserved, and we maintain the same sets of features obtained in the previous

	Select		**Insert**		**Update**		**Delete**	
Regression Methods	MAE	#	MAE	#	MAE	#	MAE	#
Support Vector Regression	75.95	34	14.96	305	162.12	3	6.56	5
Gradient Boosting Regression	70.53	7	14.90	48	161.55	39	6.54	5
Random Forest Regression	84.34	51	24.93	20	171.63	22	11.56	4

Table II. Statistical Filter (MAE in milliseconds)

generation. We illustrate, with two cases, the convergence of the algorithm in Figure 2, showing the evolution of the MAE along the generations, we show the minimum, average and maximum in each generation.

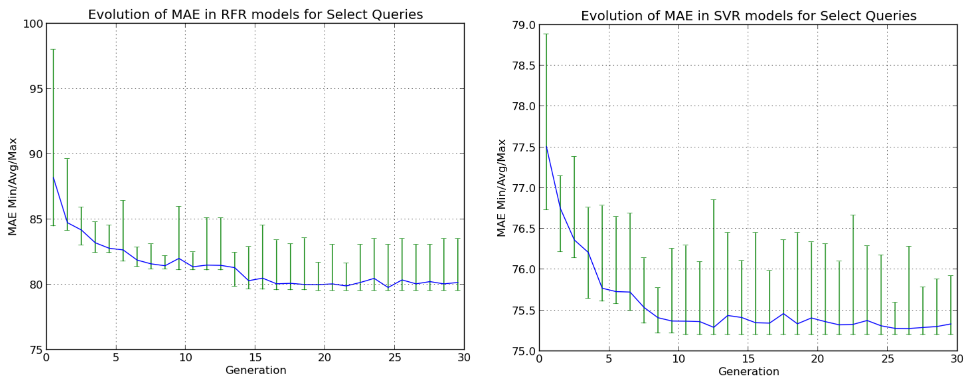

Fig. 2. Evolution of MAE in RFR and SVR models for Select Queries

	Select		**Insert**		**Update**		**Delete**	
Regression Methods	MAE	#	MAE	#	MAE	#	MAE	#
Support Vector Regression	75.23	24	14.96	133	162.12	3	6.56	5
Gradient Boosting Regression	70.53	7	14.90	22	160.59	13	6.54	5
Random Forest Regression	79.51	30	24.01	8	172.58	11	11.56	4

Table III. Genetic Algorithm (MAE in milliseconds)

In Table III we have the complete result of this step, which was not executed for all the cases, it was executed only for cases where there are more than 20 features. This model still performs poor predictions for Update queries, due to the their response time variations, and BoW cannot represent the complexity of the environment. But for the other types, we have a model with few features to quickly predict the response time for a query with an acceptable MAE. Besides this, given that it is a cloud environment, the aim is guarantee the QoS, so there is a comfortable gap for errors in prediction. For example, for a new query Select with SLA set at 800 ms, the response time provided by our approach will be between 730 ms and 870 ms. This information assists the provider in resource allocation and the elasticity, ensuring the QoS. A direct comparison with other works is not possible due to the different bechmarks employed in tests, as TPC-DS in [Ganapathi et al. 2009] and TPC-H in [Wu et al. 2013] and [Akdere et al. 2012]. Moreover, they used another error metric, the relative error.

5. RELATED WORK

A different approach using learning algorithms was proposed in [Akdere et al. 2012]. Their representations for the queries are based on the information provided by the query optimizer of the DBMS and by the cost estimation of each operator in the query execution plan. They developed modeling in different levels of granularity. They offer also a on-line approach to train model in the execution time with features captured from the newly received queries. The prediction of other metrics such as CPU, memory and disk usage was studied in [Ganapathi et al. 2009]. This approach also uses query optimizer information to extract feature for the queries. [Singhal 2012] presents a framework to estimate the query response time for database systems. This framework uses database concurrency model, CPU and, I/O to predict response time. [Wu et al. 2013] proposes a model to predict query execution time based on query plan using machine learning. These four approaches introduce an I/O overhead in the DBMS to produce features. In contrast, our approach uses only the text representation of queries to accomplish the prediction.

6. CONCLUSION

This work presented a machine learning approach for SQL queries response time estimation in the cloud. It combines different techniques to address each step of the predictive estimation time problem. We evaluated this work considering the cloud environment. According to the analysis of the obtained results, our approach uses few features to quickly predict the response time for a query with an acceptable error. As future work, we intend to conduct experiments with different benchmarks and make a deeper investigation of the reason for the poor perfomance of the model for Update queries. Other important issues to be addressed is to use a small set of information from the DBMS query optimizer or query execution plan to improve our approach. Finally, we intend to propose a new regression method specific to queries response time estimation.

REFERENCES

Akdere, M., Cetintemel, U., Riondato, M., Upfal, E., and Zdonik, S. Learning-based query performance modeling and prediction. In *ICDE '12*. pp. 390–401, 2012.

Breiman, L. *Arcing Classifiers*, 1998.

Breiman, L. Random forests. *Mach. Learn.* 45 (1): 5–32, 2001.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* vol. 2, pp. 27:1–27:27, 2011.

Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. Zephyr: live migration in shared nothing databases for elastic cloud platforms. In *SIGMOD '11*. pp. 301–312, 2011.

Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. In Annals of Statistics. *Annals of Statistics* vol. 29, pp. 1189–1232, 2000.

Friedman, J. H. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 2002.

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M., and Patterson, D. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *ICDE '09*. pp. 592–603, 2009.

OLTPBenchmark. *OLTPBenchmark*, 2013. `http://www.oltpbenchmark.com`.

OpenNebula. *OpenNebula - Open Source Data Center Virtualization*, 2013. `http://www.opennebula.org`.

Pedregosa, F. and et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* vol. 12, pp. 2825–2830, 2011.

Perone, C. S. Pyevolve: a python open-source framework for genetic algorithms. *SIGEVOlution* 4 (1): 12–20, 2009.

Singhal, R. A framework for predicting query response time. In *14th IEEE International Conference on High Performance Computing and Communication*. pp. 1137–1141, 2012.

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

TPC. *TPC-C*, 2013. `http://www.tpc.org/tpcc/`.

Vapnik, V. *Statistical learning theory.* Wiley, 1998.

Wu, W., Chi, Y., Zhu, S., Tatemura, J., Hacigumus, H., and Naughton, J. F. Predicting query execution time: Are optimizer cost models really unusable? *ICDE '13*, 2013.