An Advance Resource Reservation Approach in a Cloud Database Environment

Vinicius da S. Segalin[1], **Carina F. Dorneles**[1], **Mario A. R. Dantas**[1]

[1]Departamento de Estatistica e Informatica (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

vinicius.segalin@posgrad.ufsc.br,dorneles@inf.ufsc.br,mario.dantas@ufsc.br

Abstract. *A well-known challenge with long running time queries in database environments is how much time a query will take to execute. This prediction is relevant for several reasons. For instance, by knowing that a query will take longer to execute than desired, one resource reservation mechanism can be performed, which means reserving more resources in order to execute this query in a shorter time in a future request. In this research work, it is presented a proposal in which the use of an advance reservation mechanism in a cloud database environment, considering machine learning techniques, provides resource recommendation. The proposed model is presented, in addition to some experiments that evaluate benefits and the efficiency of this enhanced proposal.*

1. Introduction

Advance reservation can be defined as the assignment of the capacity of private resources in a restricted or limited fashion in previously defined time interval [MacLaren 2003, Gomes and Dantas 2014]. Therefore, in our scenario, this can be understood as the process where users can reserve computational resources to be used for a period of time. Usually, the resources reserved are CPU, memory, space in disk and bandwidth. Resource reservaftion aims to assure a user will have the requested resource at the requested time, allowing the user to have the expected performance in a defined time interval [Sulistio and Buyya 2004].

At the same time, cloud database services (database-as-a-service or DBaaS) manage data and automate operations of distributed database infrastructures. It has been seen a trend in the increase of the data size never seen before, as a consequence the more susceptible databases are to have a large amount of data [Assunção et al. 2015], and the more data can be translated to more data management complexity and usually longer the queries will take [Singhal and Nambiar 2016]. For this reason, in some cases it is important to have an estimation of the time in which a query will take to finish, such as the examples that are followed highlighted:

- A Database as a Service (DBaaS) provider requires to monitor constantly the workload so it can decide to accept (or even postpone) query executions in order to follow a Service Level Agreement (SLA). If a query takes longer than expected, the client will not have the results as promised and both would be harmed. However, if the provider knows this issue in advance, it could reschedule this query to a more appropriated period when it will be able to finish the requested execution without consequences.

- On online games, there are often events to attract new players or to benefit those who already play. In these events, that take place at a certain period of time, database requests increase substantially, demanding more from it. By having an estimate of how much it would increase and require from the database, the provider can, momentarily, improve its hardware capabilities in order to maintain the performance.

In such scenarios, a query execution time prediction would help to avoid undesirables problems, even if the prediction is not totally accurate, for cases when it is not very strict. Predicting a query execution time is not a simple task, though, since many variables can influence the query execution, such as concurrent and dynamic workloads, locks obtained by other queries and other processes keeping the machine busy. These and other reasons result in the same query having different execution times depending on the moment they are executed. As a result, to be able to provide a reasonable estimation, some works in literature usually ignore some of these variables due to their unpredictability.

Many approaches have been proposed to deal with these difficulties, such as analytical modeling [Wu et al. 2013a, Singhal and Nambiar 2016], statistical models [Ahmad et al. 2011] and machine learning [Ganapathi et al. 2009, Matsunaga and Fortes 2010, Gupta et al. 2008]. The latter, much more discussed in literature, has been chosen for the approach adopted to the research work described in this paper, since it is easier to implement, its accuracy is proven by previous works, and as observed in some works [Ganapathi et al. 2009, Wu et al. 2013b], the cost model by itself, provided by DBMSs, is not very useful and reliable to estimate query time execution. Our proposal aims to allow a user to reserve resources to utilize in the future and execute a query in a shorter time through an estimation of how long the query will take to be executed in different configurations (e.g., one machine with twice the memory of another).

In this paper, we present an approach to provide users of DBaaS an advance resource reservation mechanism, which allows them to enhance their databases hardware temporarily in order to improve performance. In addition, we use a prediction mechanism based on machine learning to give the user a resource recommendation regarding time and cost. To verify our approach, we have tested different machine learning algorithms and Docker [Merkel 2014] to simulate different machines. Experiments have been made using two different datasets: one with synthetic data and another with real data, and the DBMS used was PostgreSQL. Our experiments have presented interesting results for query running time prediction, with an accuracy, in average, above 80%. It is possible to conclude from our effort that we were able to provide a more accurate resource reservation by helping any user to choose the most suitable resource combination.

The paper is organized as follows. Section 2 discusses some related works. We present our approach and its objectives in Section 3. In Section 4, we describe our experimental environment and result cases from the utilization of our proposal approach. We conclude the paper in Section 5.

2. Related Work

In this section, we present related work in two main subject dimensions, those the query execution time prediction issue and resource reservation.

2.1. Query Execution Time Prediction

In past years, there has been significant work in predicting query execution time. Some works are developed using commercial database systems, such as HP NeoView [Ganapathi et al. 2009], IBM DB2 [Ahmad et al. 2011] and SQL Server [Lee et al. 2016]. The proposal described in [Ganapathi et al. 2009] uses machine learning to predict query execution time and other performance metrics, such as message count and disk I/O. In [Ahmad et al. 2011], the focus is to provide query completion time prediction by fitting statistical models to query samples based on machine learning techniques. [Lee et al. 2016] presents a new feature in Microsoft SQL Server 2016 to provide query progress estimation, giving a percentage of work done. Although this can be useful for the user to know how much progress has been done so far, this kind of approach does not predict how much time the query will take to finish, nor can it be done prior to the query execution.

Some works propose to predict query execution time on PostgreSQL. In [Wu et al. 2013b], the authors propose an alternative for machine learning, stating the cost model provided by the query optimizer can be used, if well calibrated, to predict query execution time, and other techniques, such as machine learning, are not strictly necessary. In [Wu et al. 2013a] the objective is to propose an analytical model to predict query execution time for dynamic concurrent workloads, which the authors state is competitive to machine-learning based approaches. Another analytical model proposed is presented in [Singhal and Nambiar 2016], which dynamically adapts itself to the structure of the query execution plans to deal with large data. A framework is proposed in [Duggan et al. 2014] using a combination of semantic information and empirical evaluation to build a model able to cope with concurrent and dynamic workloads.

The idea described in [Matsunaga and Fortes 2010] extends a classification tree algorithm [Gupta et al. 2008] and compares some other machine learning techniques to propose a new method to predict execution time, memory and disk requirements. Outside the relational database world, [Hasan and Gandon 2014] compares two machine learning algorithms to provide SPARQL query performance prediction, and [Farias et al. 2016] uses machine learning to predict mean response time (MRT) per second in NoSQL databases.

Considering all analyzed works about query execution time prediction, [Ganapathi et al. 2009] has been selected as the baseline for the approach presented in this paper. The reason is the good results the authors have achieved through machine learning, and the fact that they have used the TPC-DS benchmark for their experiments, which has allowed us to compare our results.

2.2. Resource Reservation

Resource reservation has been explored in [Funke et al. 2012], where the authors show how beneficial resource reservation may be in a cloud computing environment for both the consumer and the provider. In a similar approach, [He 2015] states nowadays cloud providers take the decision of provisioning resources based only on workload peaks, what usually results in excess of resource provisioning and the need to scale up/down drastically, causing high service costs. The author, therefore, proposes an elastic reservation mechanism to enhance the use of resources and the user's satisfaction. [Wang et al. 2015] proposes a new mechanism in cloud environments to allow users to reserve instances depending on their real need, offering more flexibility to reserve resources for any length and from any point in the future, and not only for predefined options as performed by cloud providers nowadays.

Resource reservation is not only applicable to cloud environments. For instance, [Gomes and Dantas 2014] proposes an approach for an opportunistic grid computing environment, where there are many personal computers in the grid sharing resources. In this type of environment, users share their idle resources, so in many situations there may be no available resource being shared, therefore reserving can be one way of increasing the chances of having the needed resource available for the future.

2.3. Summary

Our proposal is the first step forward to a solution of resource reservation that uses a method to predict the time the query will take to execute in order to make a resource recommendation to the user. [Funke et al. 2012, He 2015, Wang et al. 2015] assume the users know by advance the duration of their reservation, while [Gomes and Dantas 2014] calculates the average of CPU and memory usage to help the user to make the reservation choice. Moreover, no resource reservation approach has been proposed DBaaS.

To fill this gap, we have studied ways to provide query execution time prediction in order to use it in resource reservation. Section 2.1 has presented some different approaches that would allow us to perform the prediction, and as mentioned before, we have chosen machine learning to do so.

3. Proposed Approach

As illustrated in Figure 1, our approach contribution can be understood in two dimensions: the offline processing and the online execution. During the offline processing, the model that will provide the predictions is prepared to be used on the online execution. First of all, we create the containers [Bernstein 2014], which will represent to the user the machines that can be hired. Afterward, the queries are executed inside each container to obtain the query plans with their execution time. The query plan works as the feature vector, while the execution time is the variable we want to predict. The algorithm, during the training, adjusts parameters in order to map the input features to the output variable. The queries executed inside the containers should be the same, so the containers are all trained with the same data and can achieve the same prediction performance. The queries can be acquired either by analyzing the database log or by generating new ones. As all the containers execute the same queries, this process of acquiring them needs to be performed only once. Finally, with the query plans and the execution times as input, the machine learning algorithm can be trained. This step should be performed by someone who knows the queries and the database schema, so this specialist can decide the best parameters and tune the model in order to get the best prediction possible. This entire part is done offline, since it may take hours, or even days, depending on the dataset, to execute all the training queries and to extract the data needed to train the models.

The second step is the online execution, which starts with the query executed by the user. This query is sent to the database, which returns only the query plan without the execution time, since it does not execute the query. The query plan is sent to the previously trained models, which will be able to output the prediction of the execution

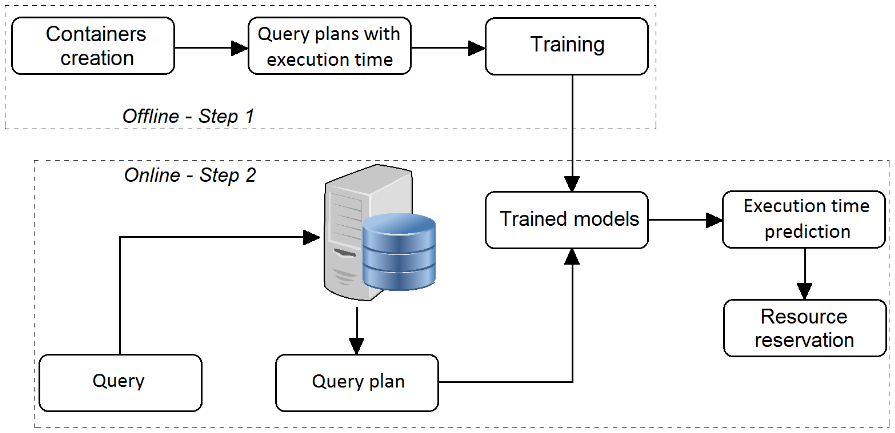


Figure 1. Resource reservation approach

time for each container. With these predictions, the user will be able to know which resource combination to choose according to the needs in terms of time and cost.

This model allows the prediction to be nearly instantaneous to the user, provided that the long processing task (i.e., to obtain the query plans and execution times and to train the machine learning models) is performed offline. The prediction, to the final user, is fast, since it will only use information available before query execution starts (i.e., the query plan produced by the query planner).

The final step of the approach - the resource reservation - depends on the choice the user has made. After receiving the resource combinations from the provider with time and cost for the entered query, the user must decide whether to stay with the same resources or migrate temporarily the database. By choosing the second option, the user selects the resource set with the best cost benefit ratio and chooses the moment the database migration should occur (for example, user reserves resources for 5 pm on the next day). The provider is then in charge of creating the container of choice, migrating the database to the new container at the moment chosen by the user, and migrating the database back to the original container after this period has ended. Figure 2 illustrates the entire online execution process.

Using this approach, the user receives a resource recommendation through the query execution time prediction. The cloud provider is then in charge of the choice of how it should be implemented, including the containers creation, machine learning process and database migration. An important consideration though, is that the machine learning training should be executed in a frequent manner as the data volume in the database grows. The reason is that query plans change according to how much data there is in the database, therefore the trained model needs to be up-to-date in order to provide a correct prediction. The training frequency varies according to the database utilization, thus it is the specialist duty to monitor the need to perform this procedure.

4. Experimental Evaluation

In this section, we describe the experiments we have performed in order to empirically validate our approach. The goal of these experiments is to demonstrate that it is possible to

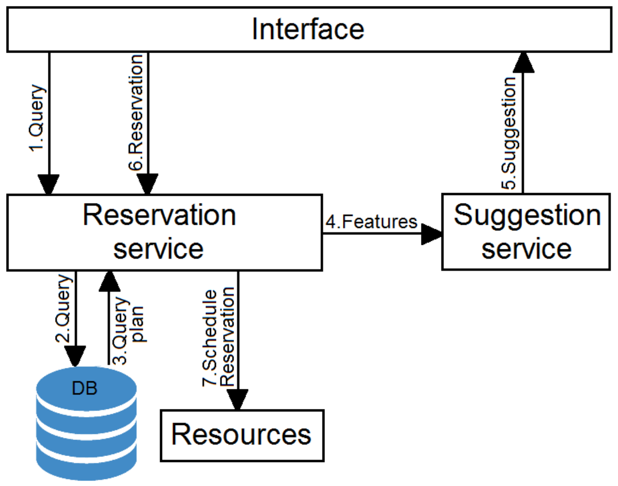


Figure 2. Resource reservation

provide suggestions for resource reservation by having a good execution time prediction, so the user could select the best resource combination, allowing the provider to reserve it. The section presents how our experiments environment has been configured, which machine learning techniques have been used, and which results we have reached on testing our approach.

4.1. Environment Settings, Datasets and Evaluation Metric

The experiments have been executed on a virtual machine provided by the cloud service at the Federal University of Santa Catarina (UFSC), which uses VMware VSphere ESXi 5.5, with an 8 core Intel i7 3.07 GHz processor and 32 GB of RAM. We have used PostgreSQL 9.5.4 on Ubuntu 16.04 to execute the queries and to obtain the query plans. We discuss and compare the results of query execution time prediction with a baseline from literature that we have considered to be most relevant, that is that presented in [Ganapathi et al. 2009].

The datasets used are mainly an open dataset from the Brazilian government[1], called *Prova Brasil DB*, and the benchmark TPC-DS [Nambiar and Poess 2006]. *Prova Brasil DB* has been used for the experiments of resource reservation, while TPC-DS has been used to compare our machine learning results to our baseline.

Since we did not have real user queries for any of the datasets, we have designed a simple algorithm to generate a workload, which aimed to provide us all kinds of representative queries that would give us distinct execution times. This script has generated different kinds of queries with single tables and using joins, so the combination variety would be large, providing us hundreds of different queries.

We have evaluated our models comparing the graphs of predicted time vs. actual time. These graphs show the precision the models have obtained by predicting several query execution times, therefore have helped us to identify which model had better fitted our data. Besides, we have predicted the execution time for random queries to confirm we could get good predictions with the model.

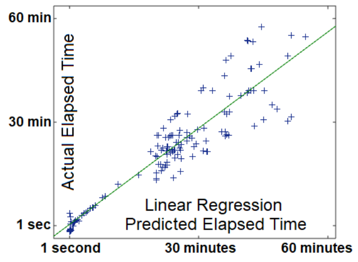
[1]http://dados.gov.br/dataset/microdados-prova-brasil

4.2. Query Execution Time Prediction

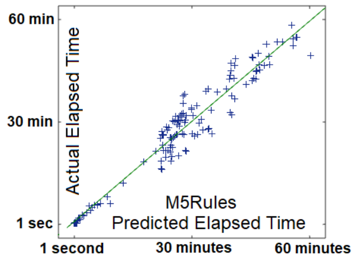
This section presents the results we have reached by comparing our query time prediction experiments with our baseline [Ganapathi et al. 2009]. In the same manner as performed by [Ganapathi et al. 2009], we have instantiated the TPC-DS benchmark at scale factor 1. We have then generated 1487 queries with the script mentioned before. Since it was difficult to have an estimate of the time the queries would take, specially because the same template could generate a query that takes only seconds and another one that would take hours (e.g., a cross join with 3 small tables and a cross join with 3 big tables), they were executed with the intention of having a wide time variation. In the end, the queries execution took from milliseconds to 63 minutes.

Figures 3a-3c show the prediction using our queries with Linear Regression [Kutner et al. 2004], M5Rules [Quinlan et al. 1992] and Multilayer Perceptron [Pal and Mitra 1992] respectively, while Figure 3d shows the prediction performed by [Ganapathi et al. 2009]. The graph in Figure 3a shows our data does not present a linear pattern, where a few short queries have had good predictions, while longer queries have presented bad results, with both very high or low predictions. Figure 3b indicates the algorithm M5Rules is better for this dataset than Linear Regression. For this algorithm, the data is closer to the perfect prediction line than for the first experiment, indicating better results, although still with many wrong predictions (the data more distant from the line).

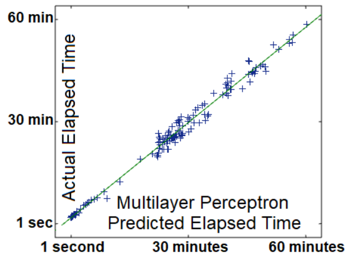
Multilayer Perceptron, represented in Figure 3c, has given the best results among the experimented algorithms. Data is closer to the diagonal line than to the other algorithms, presenting a similar pattern to the one in Figure 3d, our baseline.



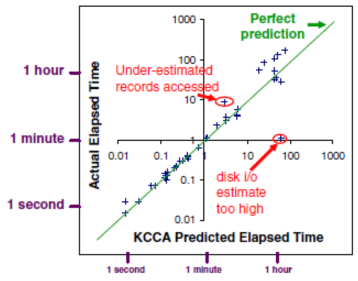
(a) Prediction using Linear Regression



(b) Prediction using M5Rules



(c) Prediction using Multilayer Perceptron



(d) Result from Ganapathi et al. 2009

Figure 3. Comparison of different algorithms and (d) the baseline

In addition to having presented a similar pattern to the approach we consider as our baseline, we randomly generated 10 more queries in order to see whether we could make good execution time predictions. Table 1 shows the results. It can be seen good predictions have been made for the long queries (>100 seconds) and bad predictions for the short queries (<60 seconds). This indicates this model is not suitable for predicting the execution time for short queries, although it looks accurate for long queries. As our approach uses query execution time prediction to provide the users a resource recommendation to execute long queries faster, bad predictions for short queries do not affect the proposed model. Thus these results encouraged us to use this algorithm to provide a good resource suggestion for advance reservation.

Table 1. predictions made on TPC-DS

#	Exec time (s)	Prediction	Prediction error
1	1029.30	1102.25	72.95 / 7.08%
2	625.33	679.33	54.00 / 08.63%
3	104.83	114.20	9.37 / 8.93%
4	7.98	56.79	48.81 / 611.74%
5	1.86	47.704	45.83 / 2452.38%
6	50.97	82.03	31.06 / 60.94%
7	210.21	233.80	23.59 / 11.22%
8	565.23	552.14	-13.08 / -2.31%
9	996.77	1036.41	39.64 / 3.97%
10	1163.05	1117.25	-45.801 / -3.94%

To assure our model would work and provide us good predictions, more experiments have been performed on two other datasets (a synthetic DVD rental database[2] and *Prova Brasil DB*). The results obtained have presented the same accuracy as the ones of this section, but due to the limited space, we do not present them in this paper.

4.3. Containers Setup

We have used Docker to simulate new machines with different configurations. Table 2 shows the containers created and their respective resources. All the containers have been created in the same virtual machine, therefore they all share the same Operating System and resources (we have only limited some of them for the experiments). We have stipulated a fictitious price to each of them as part of the experiments, so the user would have to consider the cost benefit ratio when choosing which configuration to reserve.

Table 2. Containers configuration

Container Name	Memory (MB)	PE (Processing Element)	Price Per hour
C1	1024	1	$0.05
C2	2048	1	$0.07
C3	2048	2	$0.08
C4	4096	4	$0.11

[2]http://www.postgresqltutorial.com/postgresql-sample-database/

4.4. Resource Reservation Experiments

In this section, we present the experiments performed to test resource reservation. We analyze two scenarios. The first one does not use our model, while the second one does. We then compare the scenarios to show the benefits in using resource reservation.

To simulate a real situation, the user has executed a query that was thought to be long[3] (for our scenario, longer than 1 hour):

```
SELECT *
FROM ts_item, ts_pesos, ts_quest_professor,
ts_resposta_aluno, ts_quest_diretor;
```

4.4.1. Without Resource Reservation

To test this scenario, we have used the database in a normal fashion. The container used for this test was C1. In this case, the user has executed the query and would take, according to our model prediction, 108 minutes to complete. Since the user did not use our proposed model, no prediction has been provided, thus the query was executed with no prior knowledge of how long it would take and took 103 minutes. In a fictitious situation, the user would have to wait for almost two hours in the expectation of the work to finish.

4.4.2. With Resource Reservation

In this scenario, we have used our proposed approach. In order to do so, the cloud provider had to perform the steps presented in Section 3, which means it had created containers with different resources and trained, with the same queries and the same machine learning algorithm, one network for each container that would be able to provide query execution time prediction. For our experiment, we have created the four containers described in Section 4.4. Besides the different resources for each container, we have configured each PostgreSQL instance to be optimized according to the resources.

By using the resource reservation, we had an estimate of how long the query would take if we were to enhance our machine's capacities. To make the predictions we have used multilayer perceptron, which provided us the following values: 108 minutes for C1, 81 minutes for C2, 80 minutes for C3, and 87 minutes for C4.

As we can see, the improvement from C1 to C2 was significant, result of an extra GB of RAM added. On the other hand, adding an extra PE has not given apparent improvements to C3, probably to the fact that the query is executed by a single core, so the new one would be idle for this execution. Curiously, the prediction for C4 is about 7% worse than for C2, even with double the memory and PE. This could be only a prediction error, but we can conclude that it has not presented improvements because our queries did not demand more memory, so this extra GB was insignificant for our dataset.

The cloud provider, with the predictions made, can present to the user each prediction with the price it would take to execute the query. With this in mind, the user would

[3]This query is from the dataset *Prova Brasil DB*

see something similar to the information presented in Table 3. Note that the table does not show the configuration of each container. The only relevant information for the user is the time and cost it will take to execute the query. Any other information is not important for the user as the final client.

Table 3. Price/time to execute the query

Configuration	Estimated time (minutes)	Price
1	108	$0.10
2	81	$0.14
3	80	$0.16
4	87	$0.22

The information presented in Table 3, which is shown to the user, is considered to be a recommendation to help the user to choose a set of resources that are able to execute the input query faster than originally. The user can then select which resource combination to use or, if desired, to maintain the resources already hired. For our case, we have compared the cost we would have for each configuration. Considering it would take 108 minutes with C1, we would spend $0.10 to execute the query, according to Table 3, and $0.14 if the query takes 81 minutes in C2 (for this conclusion we have only considered full hours). Therefore we have chosen to migrate to C2 temporarily to execute the query and go back to C1 after its execution. To perform this, we have reserved C2 for 2 hours at the end of the day. The provider, after the user decides which configuration to use, should schedule the database migration from the original configuration to the configuration of choice, but only during the period agreed. After that, the database should migrate back to the original configuration.

For our experiment, thus, at the end of the day the database has been migrated to C2 during two hours, the sufficient time to execute the query. At the end, the query took 83 minutes to complete, 2 minutes later than expected. 37 minutes later, by the end of the reserved period, the database has been migrated back to the original configuration, and this transition was imperceptible, since container creation takes only a few seconds.

4.4.3. Considerations

This section has presented our approach experiments. We have presented two scenarios, in which the first the user has not used our approach and has taken 103 minutes to execute the query paying $0.10 for the two full hours. In the second scenario, the user has chosen to join our approach and executed the query 20 minutes faster paying $0.14. With our approach the user had to consider the cost benefit ratio, and has chosen to pay $0.04 more in order to execute the query 20 minutes faster.

5. Conclusions and Future Work

In this research work paper, we have presented an advance resource reservation approach for DBaaS, adopting machine learning to provide accurate resource recommendations. Through many experiments, we have obtained some very reasonable predictions with high accuracy for long-running queries. Using these predictions, our contribution proposal is able to present to any user an estimate of how long the input query would take to execute in different resource sets, therefore enabling a selection of one that could better match requirements from the user viewpoint in terms of time and cost. The approach proposal main goal allows users of DBaaS to execute a number of queries in a shorter time than they would at the resource set contracted. As a result, users have the option to pay a little more for better resources in a short predefined period of time, assuming the new resource set would result in a considerable execution time improvement.

We have executed experiments in both synthetic and real datasets, using only generated queries. For future experiments, we would like to be able to test the effectiveness of the technique in a real life situation. To do so, we should have a real workload with hundreds of queries, and also consider concurrency to simulate the everyday use. More experiments could be performed using bigger datasets to explore how the database size can influence, specially regarding the user experience. In addition, we have simulated a cloud environment with synthetic values. For future work, a real cloud provider is considered, providing more resource combinations and choosing the one with the best cost-benefit ratio automatically. An improvement that could be applied to our work is to further explore machine learning techniques or to use another approach in order to obtain more accurate query execution time predictions. For future research it is possible to conceive an improvement for the proposal with some features such as more complex queries and I/O analysis prediction similar to those presented in [Inacio et al. 2017].

References

Ahmad, M., Duan, S., Aboulnaga, A., and Babu, S. (2011). Predicting completion times of batch query workloads using interaction-aware models and simulation. In *Proc. of the 14th Int. Conf. on Extending Database Technology*, pages 449–460. ACM.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., and Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79:3–15.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Duggan, J., Papaemmanouil, O., et al. (2014). Contender: A resource modeling approach for concurrent query performance prediction. In *EDBT*, pages 109–120.

Farias, V. A., Sousa, F. R., Maia, J. G., Gomes, J. P., and Machado, J. C. (2016). Machine learning approach for cloud nosql databases performance modeling. In *Cluster, Cloud and Grid Computing, 2016 16th IEEE/ACM Int. Symposium on*, pages 617–620. IEEE.

Funke, D., Brosig, F., and Faber, M. (2012). Towards truthful resource reservation in cloud computing. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on*, pages 253–262.

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M., and Patterson, D. (2009). Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *Data Engineering, 2009. IEEE 25th Int. Conf. on*, pages 592–603. IEEE.

Gomes, E. and Dantas, M. A. R. (2014). An advance reservation mechanism to enhance throughput in an opportunistic high performance computing environment. In *Network Computing and Applications, IEEE 13th Int. Symposium on*, pages 221–228. IEEE.

Gupta, C., Mehta, A., and Dayal, U. (2008). Pqr: Predicting query execution times for autonomous workload management. In *Autonomic Computing, 2008. ICAC'08. International Conference on*, pages 13–22. IEEE.

Hasan, R. and Gandon, F. (2014). A machine learning approach to sparql query performance prediction. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM Int. Joint Conferences on*, volume 1, pages 266–273. IEEE.

He, H. (2015). Virtual resource provision based on elastic reservation in cloud computing. *Int. J. Netw. Virtual Organ.*, 15(1):30–47.

Inacio, E. C., Barbetta, P. A., and Dantas, M. A. (2017). A statistical analysis of the performance variability of read/write operations on parallel file systems. *Procedia Computer Science*, 108:2393–2397.

Kutner, M. H. et al. (2004). *Applied linear regression models*. McGraw-Hill/Irwin.

Lee, K., König, A. C., Narasayya, V., Ding, B., et al. (2016). Operator and query progress estimation in microsoft sql server live query statistics. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1753–1764. ACM.

MacLaren, J. (2003). Advance reservations: State of the art. *Global Grid Forum*.

Matsunaga, A. and Fortes, J. A. (2010). On the use of machine learning to predict the time and resources consumed by applications. In *Proc. of the 10th IEEE/ACM Int. Conf. on Cluster, Cloud and Grid Computing*, pages 495–504. IEEE Computer Society.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2.

Nambiar, R. O. and Poess, M. (2006). The making of tpc-ds. In *Proc. of the 32nd Int. Conf. on Very large data bases*, pages 1049–1058. VLDB Endowment.

Pal, S. K. and Mitra, S. (1992). Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on neural networks*, 3(5):683–697.

Quinlan, J. R. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. Singapore.

Singhal, R. and Nambiar, M. (2016). Predicting sql query execution time for large data volume. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 378–385. ACM.

Sulistio, A. and Buyya, R. (2004). A grid simulation infrastructure supporting advance reservation. In *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, pages 9–11.

Wang, C., Ma, W., Qin, T., Chen, X., Hu, X., and Liu, T.-Y. (2015). Selling reserved instances in cloud computing. In *IJCAI*, pages 224–231.

Wu, W., Chi, Y., Hacígümüş, H., and Naughton, J. F. (2013a). Towards predicting query execution time for concurrent and dynamic database workloads. *Proceedings of the VLDB Endowment*, 6(10):925–936.

Wu, W., Chi, Y., Zhu, S., Tatemura, J., Hacigümüs, H., and Naughton, J. F. (2013b). Predicting query execution time: Are optimizer cost models really unusable? In *Data Engineering (ICDE), 2013 IEEE 29th Int. Conf. on*, pages 1081–1092. IEEE.