Application of Machine Learning Algorithms for the Query Performance Prediction

Mario MILICEVIC[1], Mirta BARANOVIC[2], Krunoslav ZUBRINIC[1]
[1]*University of Dubrovnik, 20000 Dubrovnik, Croatia*
[2]*University of Zagreb, Faculty of Electrical Engineering and Computing, 10000 Zagreb, Croatia*
mario.milicevic@unidu.hr

***Abstract*—This paper analyzes the relationship between the system load/throughput and the query response time in a real Online transaction processing (OLTP) system environment. Although OLTP systems are characterized by short transactions, which normally entail high availability and consistent short response times, the need for operational reporting may jeopardize these objectives. We suggest a new approach to performance prediction for concurrent database workloads, based on the system state vector which consists of 36 attributes. There is no bias to the importance of certain attributes, but the machine learning methods are used to determine which attributes better describe the behavior of the particular database server and how to model that system. During the learning phase, the system's profile is created using multiple reference queries, which are selected to represent frequent business processes. The possibility of the accurate response time prediction may be a foundation for automated decision-making for database (DB) query scheduling. Possible applications of the proposed method include adaptive resource allocation, quality of service (QoS) management or real-time dynamic query scheduling (e.g. estimation of the optimal moment for a complex query execution).**

***Index Terms*—machine learning, prediction algorithms, query processing, transaction databases.**

I. INTRODUCTION

The relationship between process response time and computer system throughput is intuitive, although not always easily modelled and simulated. The effectiveness of any applied algorithm depends on the unavoidable stochastic nature of the workload.

Several studies have shown that good correlations can be achieved between the observed system load and predicted process duration when corresponding values are averaged over longer periods of time (e.g. hourly). However, such (long-term) predictions have limited efficiency in the context of real-time system management.

In this paper we concentrate on the relationship between the SQL query response time and the database server's actual load/throughput in the real OLTP system environment. We consider the possibility of an accurate SQL (Structured Query Language) query response time prediction. This mechanism may be the foundation for adaptive resource allocation, quality of service (QoS) management or real-time dynamic query scheduling.

OLTP is characterized by a large number of short on-line transactions that generally require sub-second response times. In contrast, operational reporting queries, used to support day-to-day decisions, are relatively long-running in comparison to typical OLTP workloads. As a consequence of the resource contention and locks, the result might be a decreased transaction throughput. Initial research was presented in [1]. In the meantime, additional research has been conducted to further investigate the analyzed data mining methods and confidence intervals for the predictions made by the proposed algorithm. Additionally, the model was tested on another production DB server. Our experimental setup contains Informix and Oracle 10/11g database servers running under IBM AIX on IBM RISC architecture. The database schema contains more than 1000 tables and serves as a foundation for Enterprise Resource Planning (ERP) software for travel industry.

The aforementioned stochastic nature of the server workload definitely limits the accuracy of the query response time predictions. Still, there are several concepts that can be considered in order to increase the predictability of the events: existence of periodic workload patterns, user behaviour patterns, application granularity and operating system/database software inertia and hardware latency. The evaluation of historical workload data reveals that cyclical resource consumption is a regular phenomenon in many database server environments. For example, in [2] authors show the extraction and identification of load patterns, which can be used for optimization of static or dynamic allocation.

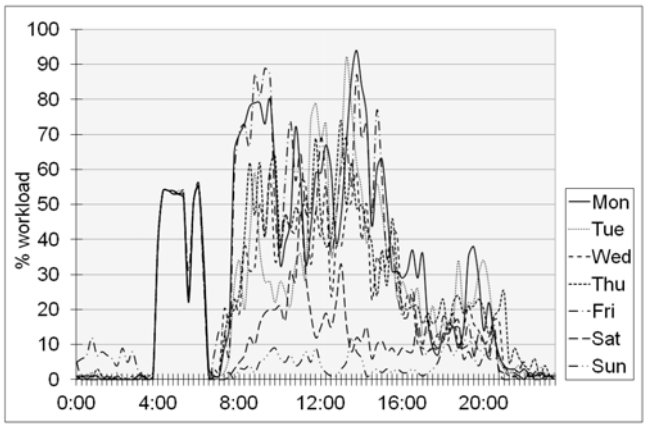

Figure 1. DB server CPU workloads during a typical week

Fig. 1 shows periodicity of the server workload during the week for the monitored ERP production database server. The number of users is noticeably reduced during the weekends. During the workdays, the server load depends on the work time of the employees and rhythm of the underlying business processes. Workload during the night is mainly the result of backup procedures and service jobs.

User behaviour patterns are quite unexplored, but they can be used as an important premise for an accurate

Digital Object Identifier 10.4316/AECE.2015.03005

characterization of the server workload. A business process can be observed as a sequence of logically related tasks, procedures and activities leading to the desirable outcome. That sequence also determines the order, content and complexity of the database transactions and queries. One of the consequences of the mentioned users' behaviour patterns is the fact that workload patterns can be also recognized and used to predict future resource requirements.

Operating system/database software inertia and hardware latency are probably a less important, but not irrelevant detail. Dynamic resource (re)allocation, process scheduling, context switching and similar actions often use complex algorithms, which cannot respond instantaneously.

II. Related work and background

Information system response time is usually defined as the number of seconds it takes from the moment users initiate an activity or process until the computer begins to present results on the display or printer [3]. There are still no industry standards for the estimation of acceptable application response time.

Fundamental considerations about response times in the context of Human-Computer Interaction (HCI) were discussed about thirty years ago, when Miller [4] introduced three important threshold levels of human attention.

The same limits were confirmed for the web-based application in [5]: 0.1 second is the limit for having the user feel that the system is reacting instantaneously, 1.0 second is the limit for the user's flow of thought to stay uninterrupted, and finally 10 seconds is the limit for keeping the user's attention focused on the dialogue. System indicators should provide information that the system is working on the user's problem. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Extensive work has been conducted in the past twenty years trying to predict computer performance. McNab et al. (1998) in [6] outlined the need for search engines to provide user feedback on the expected time for a query and described a scheme for learning a model of query time by observing sample queries. Most of the eleven attributes were static, as a result of the query structure, and only one attribute, the processor activity, was dynamic. In [7], authors applied a similar method for workload characterization and analysis. They concluded that combining background knowledge with performance data sources results in a more appropriate model, while capturing the domain expert knowledge remains an awkward and non-standard process.

P.Dinda and D.O'Hallaron [8] collected fine grain load measurements on a wide variety of servers for an extensive statistical analysis. The same authors in [9-10] used the mentioned data for a detailed evaluation of the performance of linear time series models (AR, MA, ARMA, ARIMA and ARFIMA) for predicting the five-second load average on a server. The main conclusion is that the load is consistently predictable to a very useful degree. Based on these results, P.Dinda in [11] described the interface and implementation of a real-time scheduling advisor based on the prediction of host load when targeted at compute-intensive tasks. It is designed to serve as a distributed interactive application for recommending assignments of the application's tasks to hosts. The goal of this advisor is not load-balancing or load sharing, but rather helping its client application meet deadlines and telling it when deadlines cannot be met.

In [12], the authors tried to predict the future load conditions of a resource by considering measures obtained from the load monitors of servers. These raw data is extremely variable, making it quite difficult to forecast the behaviour of future resource measures, and deduce a clear trend about the load behaviour of a resource. A two-step approach was proposed - the first aim was to get a representative view of the load trend (SMA, EMA, cubic spline) from measured raw data, and apply a load prediction algorithm based on linear regression for the load trends. The authors demonstrated in a multi-tier Web-based system that the proposed approach is suitable to support different decision systems even in highly variable contexts.

Xu et al. [13] studied predictive closed-loop control techniques for systems management. Three predictive algorithms (based on AR, ANOVA-AR and MP models) and one adaptive integral controller are evaluated in a case. The authors found that the predictive controller can deal with time-varying demands in a more proactive way once the demand pattern is learned and the prediction is accurate. However, it may also result in poor performance when the prediction error is big.

In [14], Vilalta et al. discussed three case studies in which predictive models were developed and used to deal with three types of potential failures in systems management: long-term prediction of performance variables (e.g., disk utilization), short-term prediction of abnormal behaviour (e.g., threshold violations), and short-term prediction of system events (e.g., router failure).

The prediction of query performance - with a somewhat different motivation - is an important issue in the field of the Information Retrieval (IR). For instance, the authors in [15] developed a method for predicting query performance by computing the relative entropy between a query language model and the corresponding collection language model. The resulting "clarity score" measures the coherence of the language usage in documents whose models are likely to generate the query.

A similar approach had been proposed in [16], where authors studied six predictors of query performance, which can be generated prior to the retrieval process without the use of relevance scores.

C.Hauff et al. [17] examined a number of newly applied methods for combining pre-retrieval query performance predictors in order to obtain a better prediction of the query's performance. They critically examined the current evaluation methodology and showed that the use of linear correlation coefficients does not provide an intuitive measure indicative of a method's quality. It can provide a misleading indication of performance, and overstate the performance of combined methods. Later, in [18], authors proposed a method for the evaluation of query performance prediction, which overcomes previous drawbacks by avoiding the use of correlation coefficients and transforming the performance prediction evaluation into a classification task. It is done by assuming that each topic belongs to a unique type, based on their retrieval performance.

In the database environment, N. Tomov et al. in [19]

described a study of different approximation techniques used to predict response times of database transactions, which were represented as patterns of resource consumptions and modelled with queuing networks. The analyzed transaction had a known resource consumption and execution strategy.

Similarly, analytic queuing models combined with hill climbing techniques were used in [20-21] to guide the search for the best combination of configuration parameters, predict performance and manage QoS control for e-commerce sites.

P. Martin et al. in [22] argued that the facilities available to manage local resources in current database management systems, which only allow relatively static resource allocations across all transactions, are not adequate to meet the performance requirements of an electronic commerce system. The authors also described a goal-oriented resource management system to support dynamic resource management.

In [23], authors introduced a framework which addresses the resource consumption problem by employing statistical models that provide resource and performance analysis and prediction for highly concurrent OLTP workloads. Models are built on a small amount of training data from standard log information collected during system operation. The authors argued that the mentioned models are capable of accurately measuring several performance metrics, including resource consumption on a per-transaction-type basis, resource bottlenecks, and throughput at different load levels.

The same authors [24] have implemented a similar concept in the cloud computing context, noting that current approaches to cloud computing suffer from their own advantages: infrastructure sharing and the decoupling of applications from providers of the service.

A. Mumtaz and I. Bowman presented in [25] an experimental study to highlight how the consolidated databases in multi-tenant settings share resources and compete with each other for these resources. The authors argued that individual database staging or workload profiling is not an adequate approach to understanding the performance of the consolidated system.

C. Gupta, A. Mehta and U. Dayal discussed - in several papers [26-27] - the problem of predicting the execution time of a query on a loaded data warehouse with a dynamically changing workload. They used a machine learning approach that takes the query plan, combines it with the observed load vector of the system and uses the new vector to predict the execution time of the query. Those predictions were expressed as time ranges.

In [28], the authors predicted multiple performance metrics of business intelligence (BI) queries using a statistical machine learning approach. They were able to predict multiple resource usage characteristics using only information available before the queries execute. These predictions can be used to calibrate optimizer cost estimates for a customer site at runtime, and also give a quantitative comparison of different query plans for the same query.

S. Krompass et al. in [29] gave an overview of workload management techniques already implemented in a database. They described OLTP and BI workloads and identified factors in workload generation and submission that impact their service level objectives. They looked at BI workload management and sketched points in a workload's life cycle at which management is applicable. They also presented a synergy matrix that characterizes the impact of running particular batch queries concurrently. Later, in [30], S. Krompass et al. focused on adaptively scheduling mixed data warehouse workloads that have multiple objectives. Authors used an experimental framework for testing policies to evaluate the extent to which prior approaches to adaptive workload scheduling addressed mixed workloads. Experiments demonstrated the difficulty of searching for solutions in the space of scheduling dynamic mixed workloads and discussed why prior approaches had not addressed certain scenarios.

M. Akdere et al. in [31] studied techniques for learning-based query performance prediction over analytical workloads. All prediction models are built using only static features (available prior to query execution) and the performance values obtained from the offline execution of the training workload. This paper does not address query performance prediction in the presence of concurrent query execution.

In [32], the authors showed that query interactions had a significant effect on database system performance, so it is important to take these interactions into account in database tuning. They advocated using an experimental sampling and modelling approach for capturing query interactions, and discussed some potential benefits of interaction awareness. Later, M. Ahmad et al. [33] presented an approach for predicting workload completion times that take into account the effect of interaction among concurrently running queries. This approach relies on experiment-driven performance modelling, and a workload simulator that uses the performance models to simulate the execution of a workload and thereby predict its completion time.

In [34], authors presented a modelling approach to estimate the impact of concurrency on query performance for analytical workloads. Their solution relies on the analysis of query behaviour in isolation, pair-wise query interactions and sampling techniques to predict resource contention under various query mixes and concurrency levels.

Several papers considered the possibilities of dynamic QoS control in a grid computing context. In [35], authors presented time series prediction strategies that track recent trends by giving more weight to recent data. Likewise, in [36], the authors proposed an instance based learning technique to predict how long a job executes on the resource (application run time), and how long the job waits in a queue before starting (queue wait time) by mining of historical workloads data. W. Smith et al. in [37] presented a technique for predicting run times of parallel applications based upon the run times of similar applications executed in the past. Search techniques were used to determine application characteristics that yield the best definition of similarity for making predictions.

One can notice that numerous papers address the problem of the future workload conditions and corresponding resource demands, in different contexts, based on the historical data. However, only a few works have considered

numerous dynamic attributes and models that utilize instantaneous system status information. Likewise, the authors in virtually all these works use prior knowledge about the importance of individual attributes, instead of using machine learning methods for the selection of relevant features.

III. TRAINING THE PREDICTION MODELS

We propose a method that collects sufficient information to describe the system's state, which is then processed with machine learning algorithms to predict query duration (Fig. 2).

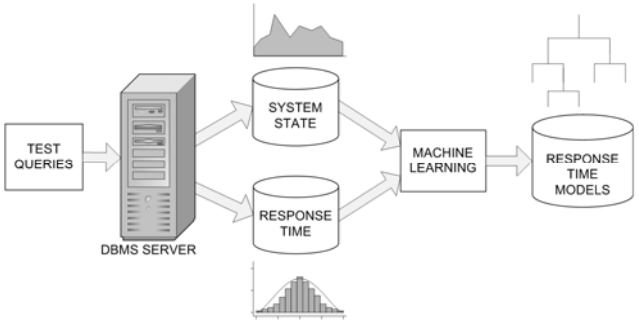

Figure 2. Learning phase

The goal of the Learning phase is to create a system's profile using multiple queries, which in principle could be (a) part of standard online transaction processing benchmark - e.g. TPC-C or TPC-E, or (b) reference queries of different complexity, selected to represent frequent (business) processes - based on knowledge of the nature of the system (hardware (HW), operating system (OS), applications, Database Management System (DBMS) etc.) and available statistics from archive logs.

Variant (a) has several advantages, as reproducibility, comparability, availability, but also it has significant drawbacks:

- within the database, it is usually necessary to define additional objects, as well as significant amounts of data;
- the impact of frequently accessed objects (tables) is neglected, which bypasses the analysis of "complications" such as locking, the effects of transaction isolation levels, index update, etc.

Since the main purpose of the system's profile, which is going to be generated, is to ensure the agreed level of QoS to the end-users of the system, the approach (b) should be preferred. The system is viewed as a "black box". Information such as the type and number of processors, operating frequency, memory, etc. has no direct impact on the final profile, for example as parameter values in the appropriate regression expression.

The distribution of response time for two (of nine) reference queries is shown in Fig. 3. Based on the available AIX/UNIX and DB tools and methods (e.g. sar, vmstat, iostat, onstat, etc.), it was possible to collect about 40 different OS and DB parameters. After having analyzed the capabilities of individual tools, as well as taking into account the dynamics of the system, one second is determined as the basic time period during which cumulative data is to be collected in order to represent the status of the system. Collected attributes describe CPU usage (CPU_user, CPU_sys, CPU_iowait, CPU_idle), memory and swap space usage (mem_fre, pg_pi, pg_po, faults_in, faults_sy, faults_cs), DB server shared-memory structures and disk subsystem (dskreads, pagreads, bufreads, rdcached, dskwrits, pagwrits, bufwrits, wrcached, bufwaits, lokwaits, lockreqs, deadlks, dltouts, lchwaits, ckpwaits nad compress) and disk subsystem throughput (Kbps_HD_DB, tps_HD_DB, Kb_rd_HD_DB, Kb_wr_HD_DB, tm_actLG, Kbps_HD_LG, tps_HD_LG, Kb_rd_HD_LG, Kb_wr_HD_LG).

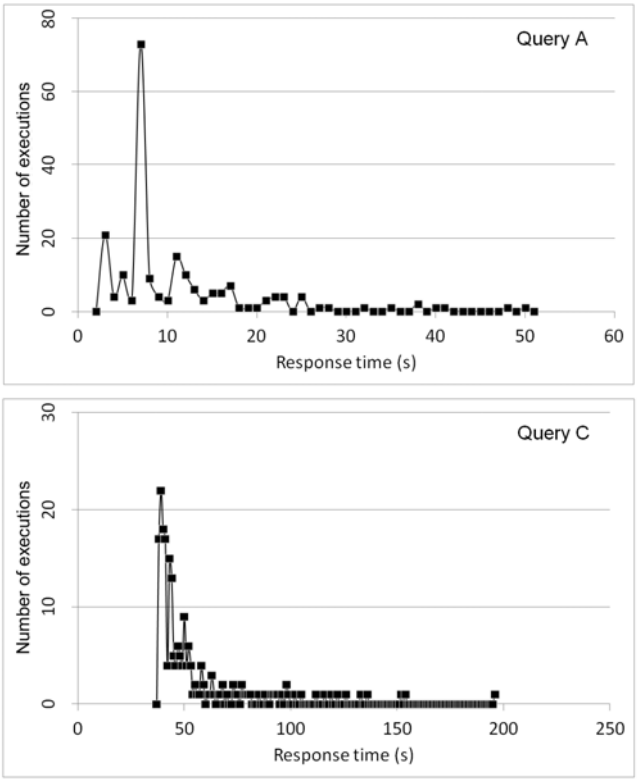

Figure 3. Distribution of response time for two reference queries

The system is generally described (1) by a state vector *Xt* of dimension *n* at time *t*:

$$X_t = (x_1, x_2, x_3, x_4, ..., x_n) \quad (1)$$

... where x_1, ..., x_n are cumulative values of individual features (n=36) within the one second interval [*t-1, t*].

The goal is to find, for each of the *r* reference queries, a function Ψr that expresses the response time (2), i.e. the duration of the query T_r as:

$$T_r = \psi_r(X_t) \quad (2)$$

... if the query is initiated at time t.

The actual structure of the function depends on the selected machine learning method.

Furthermore, the complexity of each query, reference or production, can be expressed by its cost *C*, which can be determined from the Cost-Based Optimizer (CBO). It attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Upon knowing the complexity cost C_r of the reference queries and functions Ψr, which define the relationships between the actual query response time T_r and system's state X_t during the query initiation, we can assume that it is reliable enough to estimate the response time T_p of a production query p - if its cost C_p and the system state $X_{t'}$ are known at the time of initiating the query.

TABLE I. COMPARISON OF MACHINE LEARNING ALGORITHMS FOR CONTINUOUS CLASSES

Algorithm	SQL query								
	A	B	C	D	E	F	G	H	I
M5P									
Corr.coeff.	0.887	0.873	0.890	0.825	0.820	0.804	0.901	0.836	0.897
Model interpretation	+++	+++	+++	+++	+++	+++	+++	+++	+++
Speed	+++	+++	+++	+++	+++	+++	+++	+++	+++
M5 Rules									
Corr.coeff.	0.860	0.870	0.861	0.811	0.802	0.801	0.906	0.835	0.869
Model interpretation	++++	++++	++++	++++	++++	++++	++++	++++	++++
Speed	++	++	++	++	++	++	++	++	++
REPTree									
Corr.coeff.	0.727	0.849	0.779	0.746	0.663	0.754	0.843	0.801	0.685
Model interpretation	+++	+++	+++	+++	+++	+++	+++	+++	+++
Speed	++++	++++	++++	++++	++++	++++	++++	++++	++++
IBk (k=3)									
Corr.coeff.	0.725	0.767	0.754	0.733	0.673	0.714	0.708	0.740	0.755
Model interpretation	0	0	0	0	0	0	0	0	0
Speed	++	++	++	++	++	++	++	++	++
Lin.Regr.									
Corr.coeff.	0.857	0.846	0.863	0.787	0.776	0.767	0.896	0.746	0.664
Model interpretation	++++	++++	++++	++++	++++	++++	++++	++++	++++
Speed	++++	++++	++++	++++	++++	++++	++++	++++	++++
Multilay.Perc.									
Corr.coeff.	0.683	0.801	0.836	0.683	0.545	0.538	0.854	0.570	0.830
Model interpretation	+	+	+	+	+	+	+	+	+
Speed	+	+	+	+	+	+	+	+	+
SVMreg									
Corr.coeff.	0.855	0.821	0.675	0.804	0.577	0.770	0.873	0.761	0.582
Model interpretation	+	+	+	+	+	+	+	+	+
Speed	++	++	++	++	++	++	++	++	++

Based on initial assumptions, several machine learning methods, which are used to create an algorithm for prediction of the response time, are analyzed.

The choice of the optimal machine learning method will be determined by the method's performances, including the accuracy and speed of predictions. However, an additional benefit is certainly the comprehensiveness of the resulting model, which can point to the plausible necessity of proactive database tuning and optimization.

A. Response time as continuous class

An important decision that influences the selection of the method is whether the target class (predicted query duration, response time) will be created as a discrete or a continuous value, due to the fact that most of the known machine learning methods are specialized just for one of these two types of target classes.

Of course, the response time is by its nature a continuous value - but there is a possibility of discretization. A large number of simulations were carried out, with the following conclusions:

- distribution of the collected response times during the learning phase in a number of fixed-size classes does not indicate the nature of the query or the system in any way;

- distribution of the collected response times in classes that would be based on knowledge of the system or of the application would require too much human time and cannot be successfully automated;

- the boundary that separates the acceptable from the unacceptable response time must be changed under production conditions - there is a possibility for it to be situated in the middle of a class;

- any dissatisfaction with the chosen classes, which is observed in the production environment, implies a return to the learning stage.

All the values mentioned previously, all attributes, as well as the class itself (response time), are primarily considered to be continuous numerical values - which significantly reduces the choice of available machine learning methods. The following machine learning methods are considered:

- M5P Model tree - a decision tree with linear regression models at its leaves [38-39];
- M5 Rules - a decision list generated using M5 Model trees [40];
- RepTree - a fast decision tree learner which builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning [41];
- IBk - a version of the k-nearest neighbours algorithm, used in k-nearest neighbours classification [42];
- linear regression;
- neural network - multilayer perceptron [43-44];
- SVMreg - support vector machines for regression [45].

A 10-fold cross-validation is used to test the learning algorithm’s ability to develop accurate models. For each cross-validation run, 10% of the examples are used for testing [46].

Table I shows a comparison of the selected methods. For each of the 9 reference queries the best results are noted. The M5P algorithm has the best result in the most cases - with a very good perspicuity of the resulting model and acceptable duration of the learning phase. The M5P algorithm follows M5 Rules, while other algorithms give results comparable to the M5P method only for certain data sets. The potential to understand and interpret the model, as well as the speed of the model, are evaluated approximately, relative to other algorithms. In this context, “++++” indicates the best result. Since learning by does not produce a comprehensible model, the algorithm does not have an appropriate evaluation.

Fig. 4 and 5 show the results of the M5P prediction method, where query execution instances are sorted according to the measured (actual) response time (Fig. 4). For several examples, the prediction error is rather large - as

a result of unexpected values of response time (outliers), due to significant and sudden changes in server load during query processing.

Method M5P, as an algorithm that relies on regression, is quite sensitive to occurrence of such examples. Therefore it is necessary to analyze the different procedures for the noise detection in the data, or atypical examples that can spoil the performance of the classifier. The first group of appropriate algorithms is based on the statistical analysis of the distribution of the data set [47]. The main problem of this approach lies in the fact that the actual distribution often cannot be successfully modelled by standard distributions.

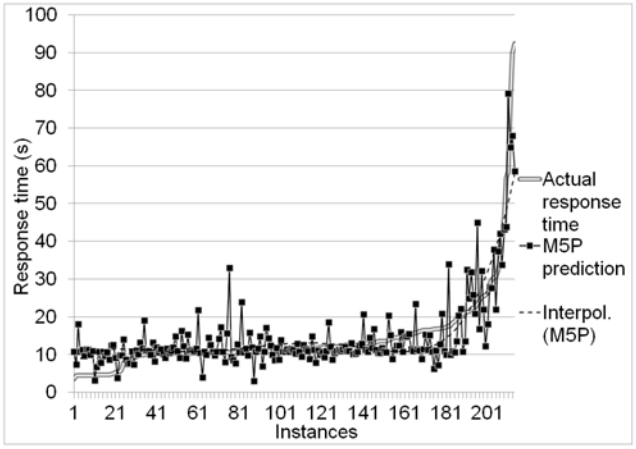

Figure 4. Results of the M5P method

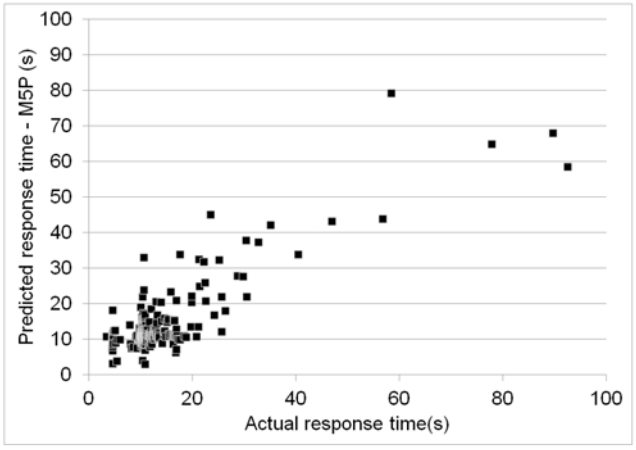

Figure 5. Prediction error for the M5P method

Another group of algorithms is based on the assumption that an extreme example would not have a significant number of "neighbours". In relation, the term "neighbourhood" is defined with mutual distance in k-dimensional space [48-49].

However, the removal of extreme values requires special attention. There is a risk that eliminated examples represent a separate phenomenon which is important to describe the observed process. It is also possible that the removal of such values in the training group reduces the effectiveness of the model when applied to the test group in which there is a similar level of noise.

Fig. 6 shows that by increasing the number of eliminated outliers above 30, which in this case represents 7.6% of the total number of test examples, both the correlation coefficient and the mean absolute error become worse than in the original test set without eliminated examples. This is due to the fact that this learning data set does not correctly describe the modelled phenomenon. Although the majority of outliers is removed, the procedure also removes a certain number of examples that represent the longest response times, as a result of increased server load.

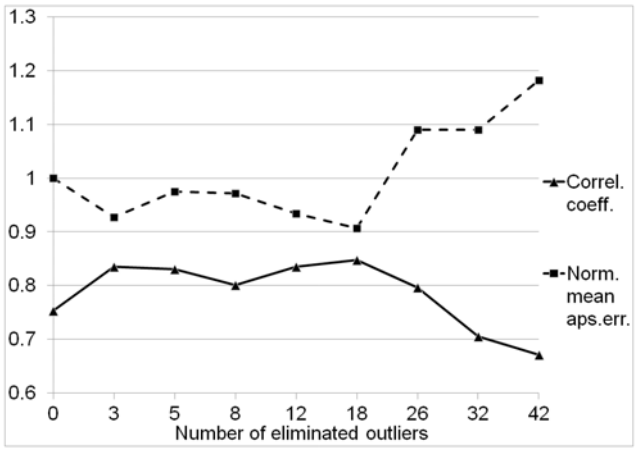

Figure 6. The impact of removed outliers on the reliability of the constructed model

Stability and reliability of the classifiers can be improved by methods using the initial set of examples to build multiple classifiers - after which the final prediction can be obtained by calculating the mean value (continuous class) or via voting (discrete classes). Examples of such algorithms are Bagging [50] and Boosting [51-52].

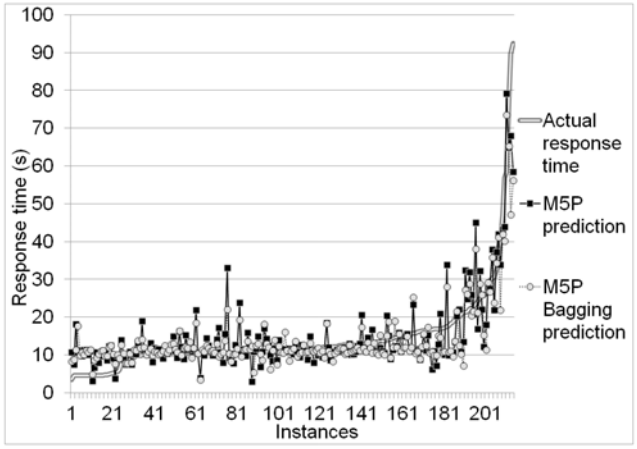

Figure 7. The implementation of Bagging algorithm on M5P method

Fig. 7 shows the results of the application of the Bagging algorithm to stabilize the results of M5P method. Query execution instances are again sorted according to the measured (actual) response time. Compared with the original algorithm, significant reductions in amplitudes were observed after 20 iterations, but at the cost of more complex models and prolonged time of calculation. Experiments have shown that increasing the number of iterations over 20 does not significantly contribute to the reliability of constructed models.

B. Class discretization

As mentioned above, an alternative approach to the prediction of SQL query duration - using machine learning methods - can be based on the target class discretization. In this case, it is possible to use additional specialized

algorithms to predict discrete values, but with a consequence of reduced usability of results interpretation.

Since the ultimate goal of predicting the response time is reliable identification of the different situations in which the response time may be unacceptable, it is reasonable to use a binary discretization of classes (e.g. 0 = acceptable response, 1 = unacceptable response). Experiments have confirmed the assumption that the discretization over the interval would not produce good results, due to unclear boundaries between the potential classes.

Of course, in the case of binary discretization, it is also necessary to pre-determine the boundaries between acceptable and unacceptable response times - which is probably the main drawback of this approach. In order to maintain a certain flexibility of this approach, the boundary response time can be determined as a multiple of the typical response time. A typical response time for a set of training examples can be determined from the respective histograms.

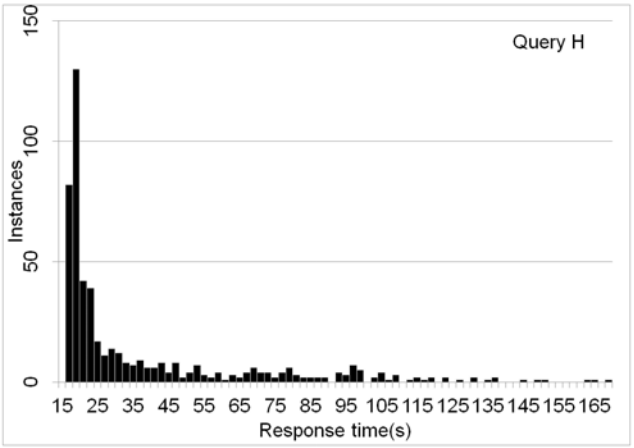

Figure 8. Response time distribution using histograms - for the reference query H

Fig. 8 shows the corresponding histogram for the reference query H, with the typical determined response time of 18s.

TABLE II. CLASSIFICATION ACCURACY FOR THE DISCRETIZED RESPONSE TIME (QUERY H)

Typical response time (18s)	Boundary between classes (0 ⇒ acceptable response time, 1 ⇒ unacceptable response time)			
	100% (36s)	200% (54s)	300% (72s)	400% (90s)
Class distribution	0 ⇒ 363 1 ⇒ 158	0 ⇒ 416 1 ⇒ 105	0 ⇒ 445 1 ⇒ 76	0 ⇒ 472 1 ⇒ 49
Method	Classification accuracy			
Naive Bayes	83.11%	84.84%	86.95%	88.86%
IBk	87.33%	87.14%	88.87%	91.75%
J48	85.80%	85.80%	92.32%	94.82%
Bagg.J48	**87.91%**	90.59%	**94.43%**	95.97%
AdaBoost M1 J48	86.37%	89.06%	93.47%	95.97%
Simple Cart	86.37%	90.98%	92.51%	94.24%
Mult. Percep.	84.84%	88.29%	89.44%	92.71%
Class.Via Regr.	86.76%	90.79%	92.51%	92.20%
Bagg.Class. Via Regr.	**87.91%**	**91.94%**	93.09%	**96.35%**

Discretization of the class can now be done with the boundary value which is a multiple of the typical response time. Table II shows the results (classification accuracy) of various machine learning methods used for predicting the value of discrete classes. Query H has been analyzed on 521 training examples, with the testing of boundaries between classes that represented an increase of the typical response time for 100%, 200%, 300% and 400%.

Method Classification Via Regression [53] - as the name suggests - performs the classification of discrete classes using regression methods that are otherwise appropriate for numeric classes. The authors were motivated for this approach by the good results of the already mentioned M5P algorithm. It is a meta-based method which involves class binarizing and building one regression model for each class value.

TABLE III. A COMPARISON OF THE TESTED MACHINE LEARNING ALGORITHMS FOR DISCRETE CLASSES (QUERY H)

Typical response time (18s)			Boundary between classes (0 ⇒ acceptable response time, 1 ⇒ unacceptable response time)							
			100% (36s)		200% (54s)		300% (72s)		400% (90s)	
Class distribution ⇒			0 ⇒ 363 1 ⇒ 158		0 ⇒ 416 1 ⇒ 105		0 ⇒ 445 1 ⇒ 76		0 ⇒ 472 1 ⇒ 49	
Method			Confusion matrix							
			Actual class							
			1	0	1	0	1	0	1	0
Naive Bayes	Predicted class	1	**131**	**61**	**95**	**69**	**71**	**63**	**46**	**55**
		0	**27**	**302**	**10**	**347**	**5**	**382**	**3**	**417**
IBk		1	**120**	**28**	**71**	**33**	**47**	**29**	**22**	**16**
		0	**38**	**335**	**34**	**383**	**29**	**416**	**27**	**456**
J48		1	**114**	**30**	**72**	**27**	**55**	**19**	**36**	**14**
		0	**44**	**333**	**33**	**389**	**21**	**426**	**13**	**458**
Bagg. J48		1	**121**	**26**	**77**	**21**	**59**	**12**	**38**	**10**
		0	**37**	**337**	**28**	**395**	**17**	**433**	**11**	**462**
AdaBoost M1 J48		1	**117**	**30**	**80**	**32**	**56**	**14**	**39**	**11**
		0	**41**	**333**	**25**	**384**	**20**	**431**	**10**	**461**
Simple Cart		1	**118**	**31**	**88**	**30**	**54**	**17**	**33**	**14**
		0	**40**	**332**	**17**	**386**	**22**	**428**	**16**	**458**
Mult. Percep.		1	**111**	**32**	**73**	**29**	**49**	**28**	**32**	**21**
		0	**47**	**331**	**32**	**387**	**27**	**417**	**17**	**451**
Class. ViaRegr.		1	**118**	**29**	**80**	**23**	**59**	**22**	**35**	**11**
		0	**40**	**334**	**25**	**393**	**17**	**423**	**14**	**461**
Bagging Class. ViaRegr.		1	**123**	**28**	**85**	**22**	**59**	**19**	**42**	**12**
		0	**35**	**335**	**20**	**394**	**17**	**426**	**7**	**460**

If we observe the criteria of the minimum number of false negative predictions (FN), the Naive Bayes classifier has the best characteristics. However, this algorithm is not ideal for use because it also produced the greatest number of false positive predictions (FP). So the best compromise - as it can be seen in Table III are classifiers based on the J48

algorithm (J48, J48 + Bagging), or M5P algorithm (Classification Via Regression, Classification Via Regression + Bagging).

An additional measure of model quality can be the Receiver Operating Characteristic (ROC) curve [54] - a method adapted from the signal detection theory. In a ROC curve, the true positive rate (Sensitivity) is plotted as function of the false positive rate. The Area Under the Curve (AUC) is equal to the probability that a classifier would rank a randomly chosen positive instance higher than a randomly chosen negative one.

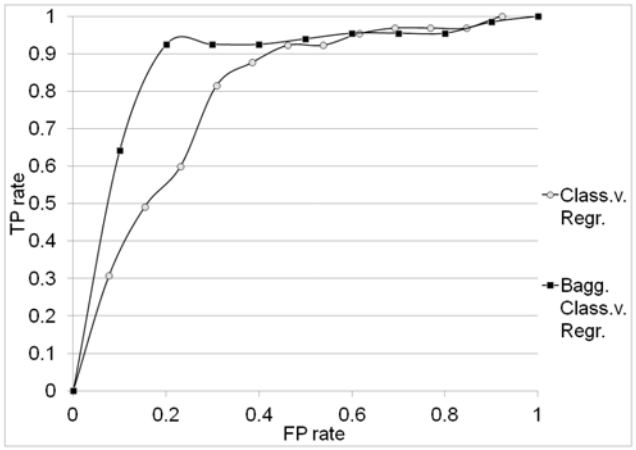

Figure 9. ROC curves for classifiers ClassificationViaRegression and Bagging ClassificationViaRegression

Fig. 9 shows the ROC curves for classifiers Classification ViaRegression and Bagging Classification ViaRegression. The examples are taken from the Table 2. The boundary value for the discretization is a response time increased by 200%, as compared to typical response times.

It is important to notice that the shape of the curve is also determined by the distribution of test data. In cases where the ROC curve is used for the selection of optimal methods of machine learning, it is desirable to construct more curves based on different test data sets, and then do their averaging. On the example shown, one can notice that the ClassificationViaRegression Bagging algorithm can achieve better results, as proven by the fact that the corresponding curve is closer to the ideal.

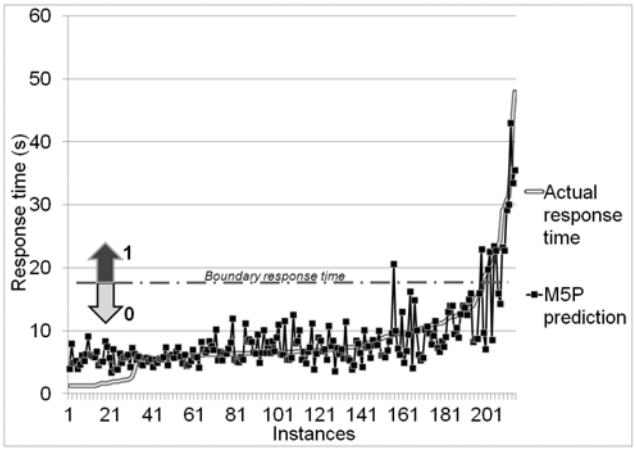

Figure 10. Results of the M5P method for the reference query F

For further evaluation of the impact of class discretization it is interesting to compare the results of these algorithms for discrete classes with the algorithm M5P, which is based on continuous class values. To make this possible, the M5P results are subsequently discretized based on set threshold.

Fig. 10 shows the results of predicting the response time to query F and the boundary chosen for the discretization. The obvious advantage of the discretization of the results for the continuous class is that decisions about the boundary response time value can be brought after the construction of models and prediction.

IV. IDENTIFICATION OF THE OPTIMAL SET OF ATTRIBUTES

In previous experiments, all available attributes describing the system status were used in order to analyze and incorporate into the final model as many of the mechanisms that influence the final response time as possible. However, it is evident that all these attributes are not mutually independent. Therefore, it is interesting to analyze the particular contribution of elimination of redundant and irrelevant attributes before the learning phase. This is done in order to improve the performance of the classifier, but also to reduce the dimensionality of some methods to simplify the resulting model. The following heuristic rule is applied: a good subset of features contains the attributes that should be highly correlated with target class variable and should not be correlated with each other [59].

A number of different techniques for finding the contributions of individual features and their subsets have been developed. The best results are often obtained with manual processing [41], assuming a detailed knowledge of the modelled problem and of the importance of each attribute is present. If there is insufficient knowledge about the problem, or the method of application is such that in an appropriate moment it is not possible to identify relevant attributes, the remaining choice to use one of the machine algorithms.

Currently, there are three types of feature selection methods: filters, wrappers and embedded methods. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. They can be computationally expensive and bear the risk of over fitting the model. Filters are similar to wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded methods perform feature selection as part of the training process in the prediction method.

The following approach is used:

- using the filter method (ReliefF) [55] [56], the attributes for each query (model) are ranked;
- for each query (model), the 10 highest ranked attributes are forwarded to the further procedure;
- for each forwarded attribute, its average contribution is calculated for all reference queries. On that basis, the final ranking of attributes is made.

Table IV shows the result of applying the algorithm ReliefF to all attributes and reference queries used in the learning stage, i.e. the 10 highest ranked attributes for each query, as well as the final order of attributes, considering their average weight when taking all the queries into consideration.

TABLE IV. ATTRIBUTE WEIGHTS AS A RESULTS OF APPLYING RELIEFF ALGORITHM TO ALL ATTRIBUTES AND QUERIES USED IN THE LEARNING STAGE

	SQL Query										
Atributte	A	B	C	D	E	F	G	H	I	Aver.	Rank
CPU_user	0.026		0.053	0.031		0.032		0.038	0.023	0.0225	8
CPU_sys	0.095	0.161	0.161	0.079	0.164	0.084	0.126	0.104	0.134	0.1232	1
CPU_iowait	0.037	0.107	0.076	0.048	0.052	0.044	0.049	0.063	0.079	0.0617	2
CPU_idle	0.050	0.081	0.069	0.044	0.077	0.062		0.037	0.045	0.0517	3
mem_fre	0.037	0.072	0.071				0.042	0.033	0.032	0.0318	4
pg_pi										0.0000	25
pg_po										0.0000	25
faults_in		0.065		0.027		0.014	0.047			0.0171	10
faults_sy		0.041	0.049	0.041	0.029	0.019	0.045	0.030		0.0283	6
faults_cs			0.049		0.033			0.029		0.0123	11
dskreads		0.067								0.0075	16
pagreads		0.063	0.055	0.034	0.029					0.0201	9
bufreads	0.030	0.049						0.027		0.0118	12
rdcached										0.0000	25
dskwrits					0.034				0.050	0.0093	14
pagwrits					0.032				0.024	0.0062	18
bufwrits	0.043									0.0048	20
wrcached	0.029									0.0033	23
bufwaits									0.021	0.0023	24
lokwaits										0.0000	25
lockreqs				0.025	0.035					0.0067	17
deadlks										0.0000	25
dltouts										0.0000	25
lchwaits	0.039	0.051	0.051				0.042	0.027		0.0234	7
ckpwaits									0.041	0.0045	22
compress										0.0000	25
tm_actDB	0.045		0.082	0.051	0.051	0.016		0.037		0.0313	5
Kbps_HD_DB						0.016	0.032			0.0054	19
tps_HD_DB				0.026		0.015	0.039			0.0088	15
Kb_rd_HD_DB							0.041			0.0046	21
Kb_wr_HD_DB						0.017	0.032		0.049	0.0109	13
tm_actLG										0.0000	25
Kbps_HD_LG										0.0000	25
tps_HD_LG										0.0000	25
Kb_rd_HD_LG										0.0000	25
Kb_wr_HD_LG										0.0000	25

As a result of the final order of highlighted attributes, a compromise has been made by selecting the next 10 attributes: CPU_sys, CPU_iowait, CPU_idle, mem_fre, tm_actDB, faults_sy, lchwaits, CPU_user, faults_in, and pagreads. These are the input data for testing the impact of reduction of the number of attributes on the success of the classification algorithm M5P (Table V).

TABLE V. IMPACT OF REDUCTION OF THE NUMBER OF ATTRIBUTES ON THE SUCCESS OF THE CLASSIFICATION METHOD M5P

Query	Number of attributes	Corr. coefficient	Mean absolute error	Root mean squared error
E	all attrib.	**0.82**	**4.11**	**6.56**
	10	0.81	3.86	6.65
	9	0.76	4.19	7.41
	8	0.74	4.24	7.63
	7	0.74	4.29	7.67
	6	0.74	4.30	7.67
	5	0.72	4.36	7.95
	4	0.71	4.37	7.97
G	all attrib.	**0.90**	**6.96**	**8.78**
	10	0.84	7.70	11.01
	9	0.85	7.51	10.72
	8	0.85	7.22	10.47
	7	0.85	7.30	10.46
	6	0.84	7.59	10.99
	5	0.84	7.44	10.78
	4	0.81	7.75	11.17
H	all attrib.	**0.83**	**9.79**	**16.79**
	10	0.84	9.73	16.23
	9	0.84	9.85	16.22
	8	0.84	10.01	16.35
	7	0.83	10.23	16.50
	6	0.82	10.23	16.53
	5	0.82	10.36	16.53
	4	0.80	10.87	17.04

For clarity, the best results in the table are outlined. The number of used attributes is reduced to the n highest ranked attributes (n = 10, 9, 8, ...).

Reducing the number of attributes (from the initial 36 to 10, 9, 8, etc.) does not have a drastic impact on the classification performance.

A similar observation about the optimal number of attributes was recognized by authors in [57]. According to their findings, the best ranked attributes for a classification problem can be determined from the data used in the learning stage. The authors also showed that a random attribute choice, without analysis of its contributions, significantly spoiled the performance of the classifier.

An alternative to the described ReliefF procedure, which analyzes the contribution of individual attributes, is to choose an optimal subset of attributes and take into account the relationships between certain attributes. This is done in order to eliminate problems arising from using redundant attributes.

The Correlation based Feature Selection (CFS) [58] [59] algorithm is based on the already mentioned hypothesis: a good subset of features contains the attributes that have a noticeable correlation with the target class. At the same time, those attributes are mutually uncorrelated.

The effects of selecting a subset of attributes with the CFS algorithm are analyzed on the data collected for queries D and H (Table VI). Greedy search strategies are used: forward selection and backward elimination. Forward selection begins with no features and it successively adds attributes into larger and larger subsets, while backward elimination starts with the set of all attributes and progressively eliminates the least promising ones. The

optimum does not change, no matter whether the search is interrupted after 5, 50 or 100 iterations without improvement. The number of analyzed subsets ranges between 300 and 3000.

TABLE VI. IMPACT OF REDUCTION OF THE NUMBER OF ATTRIBUTES WITH THE CFS ALGORITHM ON THE PERFORMANCE OF THE M5P AND IBK CLASSIFIERS

Query	Number of attributes	Corr. coeff.	Mean abs. error	Root mean sq. error	Rel. abs. error	Root rel. sq. error
D		algorithm **M5P**				
	all attributes (36)	0.825	11.12	16.49	50.88%	57.07%
	CFS forward (6)	0.777	12.75	18.07	58.31%	62.53%
	CFS backw. (29)	0.803	11.72	17.13	53.60%	59.28%
		algorithm **IBk**				
	all attributes (36)	0.733	13.51	19.77	61.80%	68.42%
	CFS forward (6)	0.781	12.77	18.28	58.42%	63.25%
	CFS backw. (29)	0.794	11.85	17.59	54.21%	60.88%
H		algorithm **M5P**				
	all attributes (36)	0.836	9.79	16.79	42.57%	55.03%
	CFS forward (5)	0.845	10.14	16.26	44.07%	53.30%
	CFS backw. (17)	0.836	10.01	16.77	43.51%	54.97%
		algorithm **IBk**				
	all attributes (36)	0.741	12.31	20.64	53.49%	67.63%
	CFS forward (5)	0.799	11.29	18.48	49.10%	60.56%
	CFS backw. (17)	0.755	11.94	20.09	51.92%	65.85%

V. MODEL IMPLEMENTATION

The models built on the basis of these findings are used to predict the response time of production queries (Fig. 11). The model implementation steps are illustrated on the case study example.

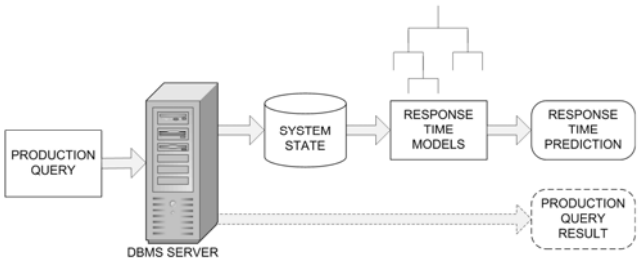

Figure 11. Production phase

We have already mentioned our experimental environment consisting of Informix and Oracle 10/11g database servers running under IBM AIX on IBM RISC architecture. In this case study, the learning phase is performed with reference queries D, E and F. Three queries with significantly different costs are chosen, with the assumption that this provides better conditions for predicting the response time of an unknown production query. The function Ψr is defined using the M5P algorithm for each query separately. Queries' costs are shown in Table VII.

TABLE VII. REFERENCE QUERIES COSTS (CBO)

Query	Query cost (->CBO)
D	44144
E	8750
F	2881

Under the production conditions, it is determined that the cost of the forthcoming production query S is 12730. Also, the vector X_t is recorded, describing the status of the system within a one second interval, immediately prior to starting the query S.

Considering a known system status X_t, the response times of the reference queries can be estimated on the basis of these data. These figures are shown in Table VIII.

TABLE VII. RESPONSE TIMES PREDICTION FOR THE REFERENCE QUERIES, WHILE CONSIDERING THE CURRENT STATUS OF THE SYSTEM

Query	Predicted response time
D	187.5s
E	74.2s
F	31.4s

Considering the known costs for these queries, the polynomial interpolation can be obtained (3).

$$T_S = -1*10^{-7} * C_S^2 + 8.4*10^{-3} * C_S + 7.789 \qquad (3)$$

and it results in T_S amounting 98.6 seconds.

The accuracy of the method can be checked. The query can be initiated multiple times during the day under conditions of different server workloads, and each time both the actual response time and the estimated response times can be measured. The results are shown in Fig. 12, in which the query executions are sorted according to the measured (actual) response time.

Other standard indicators of classifier's performance have the following values:

Correlation coefficient	0.91
Mean absolute error	7.89
Root mean squared error	9.67

The results are very acceptable. They can even be better than response times predictions for individual reference queries, which can be explained by the fact that the final result is an interpolation of the predictions for the three reference queries (D, E and F). This reduces the impact of major errors that occasionally occur during the prediction of the response times for the individual queries. This effect can be compared with the results of the application of the Bagging algorithm.

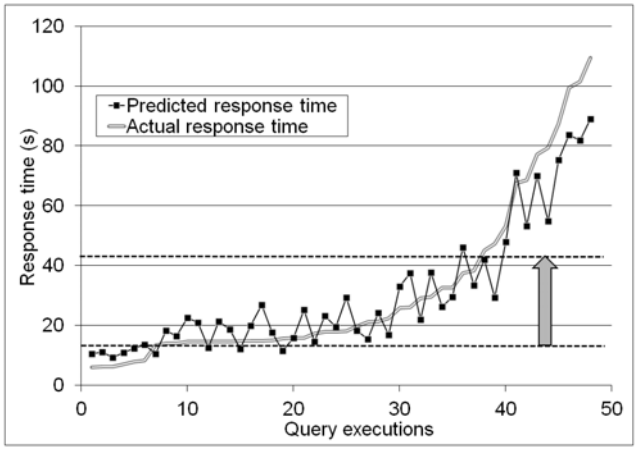

Figure 12. The results of response time predictions for the prod. query S

If the production query response time rises above a certain value, it can be declared unacceptable. As indicated in Fig. 12, that boundary response time is 3 times (i.e. 200%) higher than the typical response time. The following confusion matrix (Table IX) contains information about actual and predicted classifications.

TABLE IX. CONFUSION MATRIX FOR THE QUERY S - WITH 200% INCREASED BOUNDARY RESPONSE TIME

		Actual class	
		Unacceptable	Acceptable
Predicted class	Unacceptable	**9**	**1**
	Acceptable	**2**	**36**

Among eleven potentially undesirable situations when the response time exceeds the threshold, only two were not correctly identified.

VI. CONCLUSION

Building accurate online performance models for database servers is challenging because the workload and resource allocation change dynamically. The effectiveness of any applied algorithm depends on the unavoidable stochastic nature of the workload.

In this paper, we concentrate on the relationship between the SQL query response time and the database server's actual workload/throughput in the real OLTP system environment. This is particularly important when the OLTP system is used as a data source for operational reports, which often cannot be avoided. An experiment-driven modelling approach is used, where a system's profile is created using multiple reference queries during the learning phase. The system is viewed as a "black box", where information such as the type and number of processors, memory, etc. has no direct impact to the final profile. A systematic comparison of the performance of several machine learning algorithms is performed, using the same 36 features. Further improvements in model performance can be achieved with outlier detection and introduction of ensemble algorithms (bagging and boosting).

We consider a possibility of an accurate SQL query response time prediction, a mechanism which could be the foundation for adaptive resource allocation, QoS management or real-time dynamic query scheduling.

There are several open directions for future work - e.g. the presented results may enable development and implementation of a Service Level Management (SLM) system. Another possible approach is to install the appropriate mechanisms in the DBMS, but a probably more flexible solution, independent of the used version of the DBMS, is the implementation of external queues - External Queue Management System (EQMS).

REFERENCES

[1] M. Milicevic, M. Baranovic, V. Batos, "QoS control based on query response time prediction", WSEAS Transactions on Computers. 4 (2005), 882-889.

[2] M. Wimmer, V. Nicolescu, D. Gmach, M. Mohr, A. Kemper, H. Krcmar, "Evaluation of Adaptive Computing Concepts for Classical ERP Systems and Enterprise Services", Proceedings of IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC'06 and EEE'06), San Francisco, California, June 26-29, 352-355. [Online]. Available: http://dx.doi.org/10.1109/CEC-EEE.2006.45

[3] B. Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd ed., Addison-Wesley, Reading, MA, 1998.

[4] R.B. Miller, "Response time in man-computer conversational transactions", Proceedings of AFIPS Fall Joint Computer Conference, Vol. 33, 1968, 267-277. [Online]. Available: http://dx.doi.org/10.1145/1476589.1476628

[5] J. Nielsen, Usability Engineering, Morgan Kaufmann, San Francisco, 1994.

[6] R. McNab, Y. Wang, I.H. Witten, C. Gutwin, "Predicting query times", Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in information Retrieval SIGIR '98. ACM Press, New York, 1998, 355-356. [Online]. Available: http://dx.doi.org/10.1145/290941.291045

[7] S. Heisig, S. Moyle, "Using model trees to characterize computer resource usage", Proceedings of WOSS 2004, 80-84. http://dx.doi.org/10.1145/1075405.1075421

[8] P. Dinda, D. O'Hallaron, "The Statistical Properties of Host Load, Fourth Workshop on Languages", Compilers and Run-time Systems for Scalable Computers (LCR 98), Pittsburgh, 1998. [Online]. Available: http://dx.doi.org/10.1007/3-540-49530-4_23

[9] P. Dinda, D. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction", Proc. 8th IEEE Symposium on High-Performance, Distributed Computing (HPDC-8), Redondo Beach, 1999. Available: http://dx.doi.org/10.1109/HPDC.1999.805285

[10] P. Dinda, D. O'Hallaron, "Host load prediction using linear models", Cluster Computing, 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1019048724544

[11] P. Dinda, "A Prediction-based Real-time Scheduling Advisor", Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), 2002. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2002.1015480

[12] M.Andreolini, S. Casolari, "Load prediction models in web-based systems", Proceedings of the 1st international Conference on Performance Evaluation Methodolgies and Tools, New York: ACM Press, 2006. Available: http://dx.doi.org/10.1145/1190095.1190129

[13] W. Xu, X. Zhu, S. Singhal, Z. Wang, "Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers", 10th IEEE/IFIP In Network Operations and Management Symposium, (2006). 115-126.

[14] R. Vilalta, C.V. Apte, J.L. Hellerstein, S. Ma, S.M. Weiss, "Predictive algorithms in the management of computer systems", IBM Systems Journal Vol. 41, No 3, 2002. [Online]. Available: http://dx.doi.org/10.1147/sj.413.0461

[15] S. Cronen-Townsend, Y. Zhou, W.B. Croft, "Predicting query performance", Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 2002. Available: http://dx.doi.org/10.1145/564376.564429

[16] B. He, I. Ounis, "Query performance prediction. Information Systems", Special Issue for the String Processing and Information Retrieval (SPIRE2004), 2005.

[17] C. Hauff, L. Azzopardi, D. Hiemstra, "The combination and evaluation of query performance prediction methods", Lecture Notes in Computer Science, Volume 5478, 2009. 301–312. Available: http://dx.doi.org/10.1007/978-3-642-00958-7_28

[18] J. Perez-Iglesias, L. Araujo, "Evaluation of Query Performance Prediction Methods by Range", Proceedings of the 17th edition of the Symposium on String Processing and Information Retrieval, 2010. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16321-0_23

[19] N. Tomov, E.W. Dempster, M.H. Williams, J.B. King, A. Burger, "Approximate Estimation of Transaction Response Time", Comput. Journal, 42(3) (1999).241-250. Available: http://dx.doi.org/10.1093/comjnl/42.3.241

[20] D.A. Menascé, R. Dodge, D. Barbará, "Preserving QoS of E-Commerce Sites through Self-Tuning: A Performance Model Approach", Proceedings of 2001 ACM Conf. E-Commerce, ACM Press, 2001. 224–234. Available: http://dx.doi.org/10.1145/501158.501186

[21] D.A. Menascé, "Automatic QoS Control", IEEE Internet Computing 7(1) (2003). str. 92-95.

[22] P. Martin, W. Powley, H. Li, K. Romanufa, "Managing database server performance to meet QoS requirements in electronic commerce systems", International Journal on Digital Libraries 3(4), 2002, 316-324. [Online]. Available: http://dx.doi.org/10.1007/s007990100046

[23] B. Mozafari, C. Curino, A. Jindal, S.l Madden, "Performance and resource modeling in highly-concurrent OLTP workloads", Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). ACM, New York, 2013, 301-312. Available: http://dx.doi.org/10.1145/2463676.2467800

[24] B.Mozafari, C. Curino, S. Madden, "DBSeer: Resource and Performance Prediction for Building a Next Generation Database Cloud", CIDR, 2013.

[25] M. Ahmad, I. Bowman, "Predicting system performance for multi-tenant database workloads", Proceedings of the Fourth International

Workshop on Testing Database Systems (DBTest '11). ACM, New York, 2011. Available: http://dx.doi.org/10.1145/1988842.1988848

[26] C. Gupta, A. Mehta, U. Dayal, PQR: "Predicting Query Execution Times for Autonomous Workload Management", Proceedings of the 2008 International Conference on Autonomic Computing, 2008.13-22. [Online]. Available: http://dx.doi.org/10.1109/ICAC.2008.12

[27] A. Mehta, C. Gupta, U. Dayal, "BI batch manager: a system for managing batch workloads on enterprise data-warehouses", Proceedings of the 11th international conference on Extending database technology: Advances in database technology, 2008. Available: http://dx.doi.org/10.1145/1353343.1353420

[28] A. Ganapathi, H.A. Kuno, U. Dayal, J.L. Wiener, A. Fox, M.I. Jordan, D.A. Patterson, "Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning", Proceedings of the 2009 IEEE International Conference on Data Engineering, 2009. pp 592-603. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2009.130

[29] S. Krompass, A. Scholz, M. Albutiu, H. Kuno, J. Wiener, U. Dayal, A. Kemper, "Quality of Service-Enabled Management of Database Workloads", IEEE Data Engineering Bulletin, Special Issue on Testing and Tuning of Database Systems, 31(1), 2008.

[30] S. Krompass, H.A. Kuno, K. Wilkinson, U. Dayal, A. Kemper, "Adaptive query scheduling for mixed database workloads with multiple objectives", Proceedings of the Third International Workshop on Testing Database Systems, DBTest 2010. Available: http://dx.doi.org/10.1145/1838126.1838127

[31] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, S. Zdonik, "Learning-based Query Performance Modeling and Prediction", Proceedings of the 2012 IEEE 28th International Conference on Data Engineering (ICDE '12). IEEE Computer Society, USA, 390-401. Available: http://dx.doi.org/10.1109/ICDE.2012.64

[32] M. Ahmad, A. Aboulnaga, S. Babu, "Query interactions in database workloads", Proceedingsof the Int. Workshop on Testing DatabaseSystems (DBTest), 2009. Available: http://dx.doi.org/10.1145/1594156.1594170

[33] M. Ahmad, S. Duan, A. Aboulnaga, S. Babu, "Interaction-aware prediction of business intelligence workload completion times", International Conference on Data Engineering (ICDE), 2010, 413-416. Available: http://dx.doi.org/10.1109/ICDE.2010.5447834

[34] J. Duggan, U. Cetintemel, O. Papaemmanouil, E. Upfal, "Performance prediction for concurrent database workloads", SIGMOD, 2011. Available: http://dx.doi.org/10.1145/1989323.1989359

[35] Y. Lingyun, I. Foster, J.M. Schopf, "Homeostatic and tendency-based CPU load predictions", Parallel and distributed processing Symposium, 2003. Available: http://dx.doi.org/10.1109/IPDPS.2003.1213129

[36] H. Li, D. Groep, L. Wolters, "Efficient response time predictions by exploiting application and resource state similarities", 6th International Workshop on Grid Computing (GRID 2005), 2005. [Online]. Available: http://dx.doi.org/10.1109/GRID.2005.1542747

[37] W. Smith, I.T. Foster, V.E. Taylor, "Predicting application run times with historical information", Journal of Parallel Distrib. Comput., 64(9) (2004).1007-1016. Available: http://dx.doi.org/10.1016/j.jpdc.2004.06.008

[38] J.R. Quinlan, "Learning with Continuous Classes", Proceedings of Fifth Australian Joint Conf. Artificial Intelligence, Australia,1992.

[39] Y. Wang, I.H. Witten, "Inducing model trees for continuous classes", Proceedings of Poster Papers, 9th European Conference on Machine Learning, Prague, Czech, 1997.

[40] G. Holmes, M. Hall, E. Frank, "Generating Rule Sets from Model Trees", Twelfth Australian Joint Conference on Artificial Intelligence, 1-12, 1999. Available: http://dx.doi.org/10.1007/3-540-46695-9_1

[41] I.H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005. Available: http://dx.doi.org/10.1145/507338.507355

[42] D. Aha,D. Kibler, M. Albert, "Instance-based learning algorithms", Machine Learning 6 (1991), 37-66. [Online]. Available: http://dx.doi.org/10.1007/BF00153759

[43] D. Rumelhart, G. Hinton, R. Williams, "Learning Internal Representations by Error Propagation", Parallel Distributed Processing Vol.1 (1986), Cambridge, MA, MIT Press. 318-362. Available: http://dx.doi.org/10.1016/B978-1-4832-1446-7.50035-2

[44] J. Han, M. Kamber, Data Mining. Morgan Kaufmann, San Francisco, CA, 2001.

[45] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, "Improvements to the SMO Algorithm for SVM Regression", IEEE Transactions on Neural Networks, 1999. Available: http://dx.doi.org/10.1109/72.870050

[46] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, 1995.

[47] V. Barnett, T. Lewis, Outliers in Statistical Data, 2nd ed., John Wiley & Sons, 1987.

[48] E. Knorr, R. Ng, "A unified notion of outliers: Properties and computation", Proceedings of 1997 Int. Conf. Knowledge Discovery and Data Mining (KDD'97), Newport Beach, CA, 1997.

[49] E. Knorr, R. Ng, "Algorithms for mining distance-based outliers in large datasets", Proceedings of 1998 Int. Conf. Very Large Data Bases (VLDB'98), New York, 1998.

[50] L. Breiman, "Bagging Predictors", Machine Learning, 24(2), 1996, 123-140. [Online]. Available: http://dx.doi.org/10.1007/BF00058655

[51] Y. Freund, R.E. Schapire, "Experiments with a new boosting algorithm", Proceedings of the Thirteenth International Conference on Machine Learning / editor L. Saitta. Bari, Italy. San Francisco: Morgan Kaufmann, 1996, 148–156.

[52] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods", Proceedings of the Fourteenth International Conference on Machine Learning / D. H. Fisher, editor. Nashville, TN. San Francisco: Morgan Kaufmann, 1997, 322–330. [Online]. Available: http://dx.doi.org/10.1023/A:1007421302149

[53] E. Frank, Y. Wang, S. Inglis, G. Holmes, I.H. Witten, "Using model trees for classification", Machine Learning, 32 (1998), 63-76.

[54] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers", Technical Report HPL-2003-4, HP Labs, 2003.

[55] I. Kononenko, "Estimating attributes: analysis and extensions of Relief", Proceedings of the European Conference on Machine Learning: ECML-94. / De Raedt, L., Bergadano, F., editors. Springer Verlag, 1994. 171-182. Available: http://dx.doi.org/10.1007/3-540-57868-4_57

[56] M. Robnik Sikonja, I. Kononenko, "An adaptation of Relief for attribute estimation on regression", Proceedings of 14th International Conference on Machine Learning ICML'97 / D.Fisher editor. Nashville, TN. 1997.

[57] H. Liu, J. Li, L. Wong, "A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns", Proceedings of 13th International Conference on Genome Informatics (GIW02), Tokyo, Japan, 2002.

[58] M.A. Hall, "Correlation-based feature selection machine learning", Ph.D. Thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1998.

[59] M.A. Hall, L.A. Smith, "Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper", Proceedings of the 22nd Australasian Computer Science Conference, 1999.