NARX-based multi-step ahead response time prediction for database servers

Amani, Payam; Kihl, Maria; Robertsson, Anders

Published in:
[Host publication title missing]

DOI:
10.1109/ISDA.2011.6121757

2011

NARX-based Multi-step Ahead Response Time Prediction for Database Servers

Payam Amani, Maria Kihl
Department of Electrical and Information Technology
Lund University, Sweden
Email: Payam.Amani@eit.lth.se, Maria.Kihl@eit.lth.se

Anders Robertsson
Department of Automatic Control
Lund University, Sweden
Email: Anders.Robertsson@control.lth.se

***Abstract*—Advanced telecommunication applications are often based on a *multi-tier architecture*, with application servers and database servers. With a rapidly increasing development of cloud computing and data centers, characterizations of the dynamics for database servers during changing workloads will be a key factor for analysis and performance improvements in these applications. We propose a multi-step ahead response time predictor for database queries based on a nonlinear autoregressive neural network model with exogenous inputs. The estimator shows many promising characteristics which make it a viable candidate for being implemented in admission control products for database servers. Performance of the proposed predictor is evaluated through experiments on a lab setup with a MySQL-server.**

***Index Terms*—response time prediction; database server; NARX neural network; modeling database dynamics.**

I. Introduction

Telecom and Internet operators need to provide their customers with a vast variety of services which are aimed at meeting their demands and desires. Multi-tier server clusters are used to host the service logic and user data. The optimization of resource allocation in server cluster systems has attracted much interest in recent years as it directly relates to the performance of these systems. Database servers, as important entities of these server clusters require secure, reliable and real-time activation, modification and deactivation of both new and current customers or services. These tasks should be performed fast and in an automated manner. Therefore, control mechanisms can be introduced, which enable the system to avoid the resource access conflict and protect it from becoming overloaded [1]–[3]. This control mechanism usually includes a feed-forward controller as it should predict the resource access conflict well before it happens and take action to avoid it. Therefore, there is a need for a multi-step ahead state predictor, which fairly represents the dynamics of the database in its whole operation range and also provides high precision state representation of the system near the overload region.

Many attempts to develop response time estimators or predictors for database queries have been presented in the literature. They can be divided into two categories namely analytical and experiment-driven methods. Analytical models [4]–[6], designed by experts, usually cover specific types of queries and database servers and assume some simplifying conditions. Thus they are not able to capture the complex dynamics of the database server. These models only support static cases and cannot be used in dynamic scenarios. Several instances of experiment-driven methods have been recently presented in the literature. Ganapathi *et. al* in [7] predict several metrics for database queries including the response time by means of Kernel Canonical Correlation Analysis (KCCA). Tozer in [8] used a linear regression model for the response time in order to throttle long running queries. Sheikh *et. al* in [9] have presented a Bayesian approach for on-line performance modeling of database appliances using Gaussian models. Their proposed model has the possibility of adaptation to changes in workload and configuration. The smallest prediction error of their method is 14%. In [10], we have presented a (nonlinear autoregressive neural network with exogenous inputs) NARX-based multi-step ahead response time predictor for single server queuing systems. We have shown via simulations that the suggested response time predictor is capable of predicting the response time of the single server queuing systems in multiple steps ahead with very small mean squared errors and mean absolute prediction errors respectively under both static and dynamic workload scenarios without adapting the model parameters to the changes in the workload.

The requirement for a nonlinear multi-step ahead query response time predictor that can work under stationary and steady state scenarios, as well as under time varying and non-stationary scenarios led us to a gray box approach to identification of database servers. Thus we have used the same type of response time predictor for database servers. By means of a NARX neural network, we have designed a predictor that covers all the aforementioned characteristics and is also able to very well predict the response times of queries of database servers with very good precision represented by very small mean absolute, mean squared and sum of squared prediction errors.

This paper is structured as follows: system description, the NARX neural network and the predictor are investigated in section II. Section III is dedicated to specifications of the experiment setup and scenarios. Experimental results are summarized in section IV and finally, section V concludes the paper.

II. System Configuration

This section covers three sub-sections. In subsection (II-A) the pilot system for which a nonlinear multi-step ahead predictor is developed is introduced. Sub-section (II-B) is dedicated to the introduction of NARX recurrent neural networks. The proposed NARX multi-step ahead response time predictor is presented in sub-section (II-C).

A. System description

Figure 1 depicts a generic multi-tier server cluster. The system can correspond to a broad range of Telecom and Internet applications, as data centers, cloud networking systems, web shops, enterprise systems, or service management systems. Here, we focus on the database tier. The interactions between

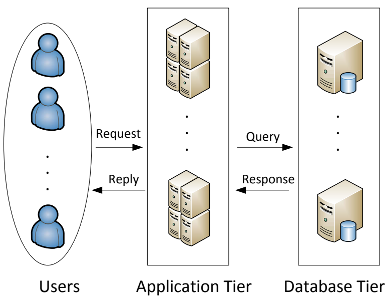

Fig. 1: A generic multi-tier server cluster.

the application tier and the database tier should not lead to the database servers to become overloaded. Therefore, control mechanisms should be implemented in the application servers that limit the traffic to the databases. The control system should be based on measurements which are available and which can be provided without a need for changing the current protocols and operating systems. In this paper, we use inter-arrival, inter-departure and response times of the queries sent to the database servers from the application servers. These measurements can easily be retrieved from the time-tagged logs of the queries traveling in the system. A high response time (compared to the reference response time) corresponds to a highly loaded database and a low response time to a lightly loaded one. Thus, response time can be used as an indicator of the databases' internal state. In this paper, we focus on the interaction of one application server with one database server.

As the control action should take place well before an overload occurs in the system, the control scheme will consist of not only a feedback loop but also a feed-forward part. The requirement for a feed-forward controller raises the need for a multi-step ahead query response time predictor for the databases. Figure 2 shows a controller scheme combining feedback and feed-forward, which requires response time prediction, presented by Kjaer *et. al* in [11].

Two main MySQL query types, Select and Update, are investigated in this paper. These requests have very different contributions to the response times of the queries sent to

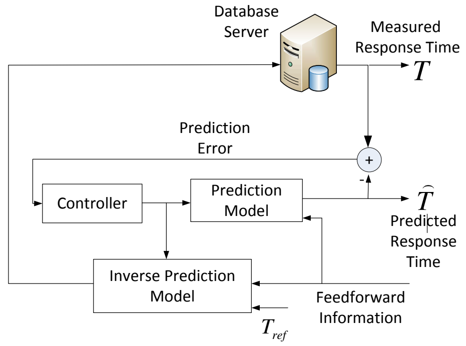

Fig. 2: A sample scheme with combined feedforward and feedback for control of database servers using response time prediction.

the database. Select queries are based on read actions while Update queries are based on write actions. Select queries are CPU restricted actions while Update queries are I/O restricted actions. As it can be seen in Figure 3, the nonlinear behavior of these two types of queries are very different. Processing of an Update query is much more time consuming compared to a Select query.

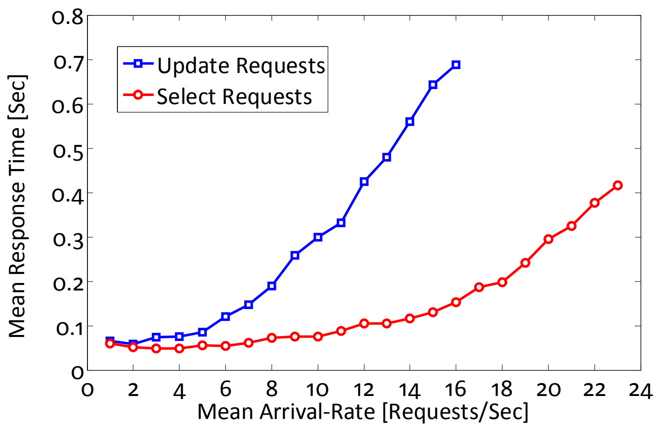

Fig. 3: Mean response times of Select and Update queries sent to a 1E7 tuples relation in a scalable Wisconsin benchmark table in a MySQL database server vs. mean arrival rates of the queries.

B. NARX Neural Network

Recurrent neural networks have been widely used for modeling of nonlinear dynamical systems [12], [13]. Among various types of the recurrent neural networks such as distributed time delay neural networks (TDNN) [12], layer recurrent networks [12] and NARX [12], the latest is of great interest in input output modeling of nonlinear dynamical systems and time series prediction [14]–[18].

NARX is a dynamical recurrent neural network based on the linear ARX model. The next value of the dependent output signal $y(t)$ is regressed over the latest n_x values of the independent input signal and n_y values of the dependent output signal. n_x and n_y respectively represent the dynamical order of the inputs and outputs of the NARX. A mathematical description of the NARX model is summarized in (1) in which f is a nonlinear function.

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), x(t-1), \\ , x(t-2), \dots, x(t-n_x)) \tag{1}$$

This network consists of three main layers namely input layer, hidden layer, and output layer. The input layer consists of the current and previous inputs and outputs. These are fed into the hidden layer. The hidden layer consists of one or several neurons resulting in a nonlinear mapping of affine weighted combination of the values from the input layer. The output layer consists of an affine combination of the values from the hidden layer. In this network, the dynamical order of inputs and outputs and number of neurons in each layer are pre-determined. Several methods for determination of these values are presented in [12]. A suitable training algorithm and performance measure should also be chosen. Finally, the type of the nonlinear map needs to be defined.

Some pre- and post processing on the input and target values should be performed in order to have a valid training set [12]. These processes include mapping of the input and target data to values in the range of $[-1, 1]$, normalization of the inputs and targets to have zero mean and unity variance and removal of constant inputs and outputs and processing of unknown inputs. As the measurements are very noisy, after normalization we filter both input and target values with a designed Butterworth low pass filter. The bandwidth of the filter is chosen so it suppresses noise as much as possible while not affecting the characteristics of in band part of input and output data sets.

C. NARX Multi-Step Ahead Response Time Predictor Set-up

Our application requires the prediction of response times of the queries sent to the database server in some time steps into the future, before they are processed in the database server. A gray box identification approach was chosen to predict the response times of such queries from three measured time values, namely inter-arrival, inter-departure, and response times of the queries. The predictor is designed by means of the Neural Networks Toolbox of MATLAB R2010b. The input vector consists of current inter-arrival times and inter-departure times as two inputs. Output of the neural predictor is the predicted response time. Measured response times are required for training and evaluation of the NARX multi-step ahead response time predictor and are fed back to the input layer of the proposed predictor. Measured data is divided into training, evaluation and test data sets. Prediction horizon m is defined as the shift between corresponding inputs and output values so that current input is used for prediction of output in m time steps in the future. The proposed multi-step ahead response time predictor is illustrated in Figure 4. The overload protection admission controller uses a gate for controlling the flow of queries to the database server. The flow of queries

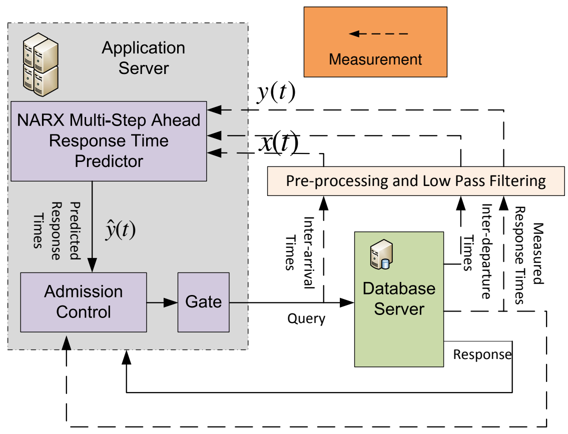

Fig. 4: Multi-Step ahead Response Time Predictor in Admission Control Set-up.

from the application server to the database server cannot get negative values as negative requests do not exist. Also, the gate cannot send more requests than the available requests in the application server. This imposes an input nonlinearity to the database server. We already know that database servers are nonlinear and stochastic computing systems under high load conditions. Thus, a NARX based predictor as a nonlinear predictor has a much better opportunity to capture the dynamics of the response time of the database server compared to linear predictors [19].

The off-line training process is described as follows: The database server is stressed under high load conditions. The acquired data is then divided to training, validation and test data sets. The NARX multi-step ahead response time predictor is trained using the train data and training is validated and its performance is tested using validation and test data sets. This trained predictor is used for all static and dynamic load conditions containing various combinations of database queries. The performance of the predictor is investigated in the following sections.

III. DATABASE SERVER LAB SET-UP

The database server lab consists of two main computers: one hosting the database server and one hosting the traffic generator which represents the application tier. One objective of this lab is to test the performance of the proposed response time predictor for I/O constrained systems such as database servers. The mentioned computers are connected via an Ethernet switch. This is depicted in Figure 5.

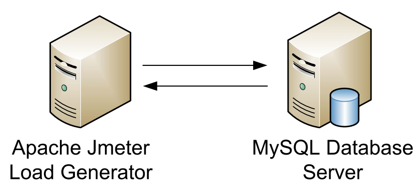

Fig. 5: Database lab set-up.

A. Hardware and Software

The database server is a Dell precision workstation 340 computer with an Intel Pentium 4 CPU running at 1.7 GHz, 768 MB RAM and a 72GB Hard Disk hosting MySQL Server 5.1.4.1. It has Solaris 11 Express as its operating system.

The application server, in this case, is represented by a Dell precision workstation 340 computer with an Intel Pentium 4 CPU running at 1.7 GHz, 512 MB RAM and a 36 GB Hard Disk hosting Apache Jmeter 2.4 as load generator sending queries to the database server. It has UBUNTU 10.04 LTS as its operating system.

B. Apache Jmeter

Apache Jmeter [20] is a java-based load generator with support for plugins that can be used to stress test various types of servers such as web servers, mail servers and database servers. Support for database queries is provided via java database connectivity, JDBC. Various load distributions can be generated by means of timer plugins. A timer plugin for generation of Poisson distributed database queries via JDBC has been used [21]. Apache Jmeter generates the load to the supported servers by means of blocking I/O, and a fixed number of threads. This imposes an upper limit for maximum number of concurrent requests which is equal to the number of Jmeter's threads. During the time intervals that all the threads are busy, no new queries can be sent out before the processing of an old query is finished. This will change the distribution of the load to the database server. Thus, all experiments that use all of the threads at the same time shall be invalidated.

C. Tracing and D-Trace

Tracing in software engineering terminology is a specialized use of logging for recording information about execution of an application. Dynamic Tracing, D-Trace [22], is a detailed dynamic tracing tool introduced by Oracle for Unix-like operating systems. D-Trace has the option to provide not only information regarding the whole application like CPU and memory demand, but also information regarding each function in the application. D-trace scripts are written in a C based programming language which is equipped with variables and functions required for tracing, called D. D programs include a set of one or more probes and each probe is associated with an action. When the condition of the probe is satisfied, the associated action is executed. We have used these probes to get exact time stamps of arrival of a Select or Update query to the MySQL database server and the time that the database server is done with processing of the mentioned queries. From these time stamps, we can calculate the inter-arrival and inter-departure times of the queries.

D. Structure of the Database

The database server has several relations all with the same structure from the Scalable Wisconsin Benchmark [23] with different number of tuples. Two types of MySQL queries which are most frequently used in the database servers, namely Select and Update, are taken into consideration in this paper. The structure of the queries are as follows:

Select queries:

SELECT unique2 from tenmil where unique1 equals ?;

Update queries:

UPDATE tenmil SET unique3=? where unique1=?;

In the above queries, *tenmil* is a ten million tuple relation from the Scalable Wisconsin Benchmark and ? represents a uniformly distributed random number between 0 and 1E7.

In order to test the performance of the multi-step ahead NARX response time predictor, we apply it to the described MySQL database server. Also, we consider 4 main test scenarios, consisting of two static and two dynamic scenarios.

E. Response Time Predictor's Parameters

The NARX multi-step ahead response time predictor is configured as follows: the two dimensional input vector $x(t)$ consists of inter-arrival and inter departure times. The one dimensional output $y(t)$ represents the predicted response times.

The measured response times are fed back to the input layer for training. The tapped delay line in the hidden layer consists of three delays. Three neurons are considered in the hidden layer. The hidden layer neuron's activation function is considered to be tangential sigmoid function $tansig$. The output layer consists of one neuron with activation function chosen as linear function $purelin$. Several criteria such as sampling time, control structure and constraints affect the choice of the prediction horizon m. Since support for multi-step ahead prediction is required, we chose $m = 4$ to show that the NARX response time predictor is able to predict the response times of the queries sent to the MySQL database server in several time steps into the future.

The Bayesian Regularization algorithm [24] is chosen as the training algorithm and the performance metric is set to the sum of squared errors (SSE). The predictor is first trained with the data from the high load scenario then tested over high load, low load and two varying load scenarios.

F. Database load and queries specifications

Our test set includes an Apache Jmeter load generator with 30 concurrent threads. Duration of each experiment is set to 600 seconds. The effective mean arrival rate for which all the threads are busy in case of Update requests corresponds to 16 requests per second and for Select queries corresponds to 23 requests per second. This defines the maximum effective mean arrival rates in case of each type of the queries. The types of queries in real world applications are usually mixed of both the mentioned query types. In order to represent a more realistic case we have also considered a mix of 75% Select and 25% Update queries. The maximum allowed mean arrival rate in order to keep the Poisson distribution of the arrivals in this case is equal to 16 requests per second.

The static scenarios are designed for evaluation of performance of the NARX response time predictor in steady state under low ($\rho \approx 0.30$) and high ($\rho \approx 0.95$) load conditions. By load, here we mean the ratio between the mean arrival rate and the maximum effective mean arrival rate. The dynamical scenarios are meant to evaluate performance of the NARX response time predictor with arrival rates changing with a step function at time 200 seconds from low to high load (*Step1*) or vice versa (*Step2*). These are summarized in Table I.

TABLE I: Experiment Scenarios

	Scenario	Mean Arrival Rate [Req/Sec]			Predictor State
		Update	Select	Mixed	
Static	*S1*	15	22	15	Train,Test
	S2	5	7	5	Test
Dynamic	*S3*	*Step1*	*Step1*	*Step1*	Test
	S4	*Step2*	*Step2*	*Step2*	Test

IV. Experimental Results

Performance of the proposed predictor applied to the MySQL database server is summarized in Table II. In this section, MAE stands for mean absolute error, MSE for mean squared error and SSE stands for the sum of squared errors. It should be noted that the data used in these tests is normalized to its maximum value. This is the reason why the maximum value of the response times is equal to one.

As it can be seen from the results in Table II, the multi-step ahead NARX response time predictor is well trained and shows a promising performance under *S1* and *S2* considering MSE, MAE and also SSE. This shows that the proposed response time predictor is able to accurately predict the response times of the queries sent to the described MySQL server in 4 steps ahead under static and steady state load conditions.

Looking at the presented experimental results in Table II, one can observe a large difference between the performance of the proposed response time predictor under *S1* for update queries compared to the select and mixed queries. This can be related to the different nature of Select and Update queries. Select queries are CPU constrained while the Update queries are I/O constrained. This leads to a very different nonlinear behavior of the MySQL database server depending on the query types. As the response time predictor has been trained using the mixed queries, we can expect that the scenario *S1* for Update queries should have the worst prediction performance as it is the extreme case which has the longest distance from the mixed queries.

Performance of the response time predictor under dynamic load conditions especially its performance in transient load conditions is of interest in the following tests. Under both scenarios *S3* and *S4*, all the performance measures namely MSE, MAE and SSE indicate very good performance of

TABLE II: Performance (MAE, MSE and SSE) of NARX m step ahead response time predictor for MySQL database server in scenarios *S1*-*S4* with prediction horizon m set to 4.

Predictor's Performance Select Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.956$	MSE	$2.8716e-8$
		MAE	0.0061
		SSE	0.7964
S2	$\rho = 0.318$	MSE	$2.8582e-7$
		MAE	0.0055
		SSE	0.2056
S3	$\rho =$ *Step1*	MSE	$1.9117e-6$
		MAE	0.0094
		SSE	2.4692
S4	$\rho =$ *Step2*	MSE	$6.2844e-7$
		MAE	0.0071
		SSE	0.7352
Predictor's Performance Update Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.935$	MSE	$1.2331e-6$
		MAE	0.0073
		SSE	0.8717
S2	$\rho = 0.333$	MSE	$4.2261e-8$
		MAE	0.0065
		SSE	0.4208
S3	$\rho =$ *Step1*	MSE	$9.3608e-8$
		MAE	0.0055
		SSE	0.9220
S4	$\rho =$ *Step2*	MSE	$1.4715e-8$
		MAE	0.0055
		SSE	0.4
Predictor's Performance Mixed Queries			
Scenario	Server load ρ	Measure	Value
S1	$\rho = 0.935$	MSE	$8.1621e-9$
		MAE	0.0065
		SSE	0.7968
S2	$\rho = 0.333$	MSE	$4.3381e-7$
		MAE	$0.0.0049$
		SSE	0.1487
S3	$\rho =$ *Step1*	MSE	$9.3608e-8$
		MAE	0.0053
		SSE	0.4254
S4	$\rho =$ *Step2*	MSE	$2.2948e-7$
		MAE	0.0055
		SSE	0.3344

the predictor. Figure 6 depicts measured response times vs. estimated response times under *S3* for the mixed queries. As it can be seen in the upper diagram of Figure 6, the measured and predicted response time values are so close that it is really hard to distinguish between them. Thus an additional figure depicting the difference between the measured and predicted response time values or simply prediction error has been added to the lower part of Figure 6. As it can be seen in this Figure, the maximum prediction error for each mixed query is less than 5% which is a very promising performance under dynamic mean arrival rates.

Performance of the proposed predictor under some more query mixes such as (50% Select, 50% Update queries) and (25% Select and 75% Update queries) for the same sets of scenarios *S1*-*S4* has been investigated via experiments and very small MAE, MSE, and SSE for the prediction error has been confirmed. Due to the lack of space, we skipped presenting those results.

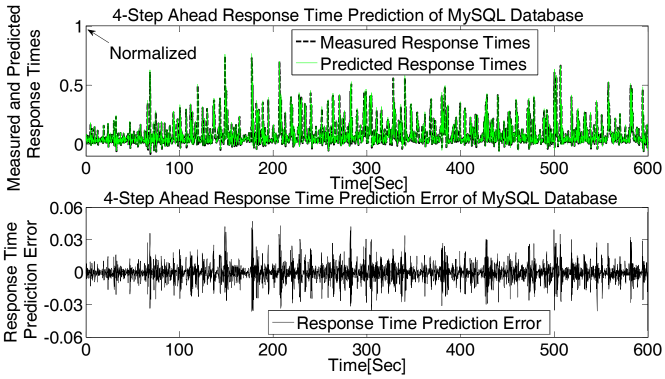

Fig. 6: NARX m step-ahead response time prediction of the MySQL Server Database with mean arrival rate changing with a step from 5 to 15 requests per second at time 200. The prediction horizon m is set to 4. (upper) Measured response times vs. estimated response times. (lower) Difference between measured and estimated response times.

V. Conclusion

A multi-step ahead NARX response time predictor for MySQL database server, has been proposed and its performance under several test scenarios has been studied. The proposed predictor benefits from several promising characteristics which turns it into a viable candidate for being implemented in admission control products for computing systems. It is nonlinear, it supports multi-step ahead prediction, its structure is simple and its required measurements can be obtained without any requirement on changing communication protocols or operating systems. It has been shown that with being trained in only one high load scenario, it still can predict the response times of queries in MySQL database server under both high and low load steady state scenarios with a high accuracy. Very good performance of the proposed predictor under time varying and non-stationary scenarios has been confirmed by very small MAE, MSE and SSE of the response time prediction.

Acknowledgment

Payam Amani, Maria Kihl and Anders Robertsson are members of Lund Center for Control of Complex Engineering Systems (LCCC), a Linnaeus Center at Lund University, funded by the Swedish Research Council. Maria Kihl is partly funded by the VINNMER program at the Swedish Governmental Agency for Innovation Systems (VINNOVA). Hereby, Anders Robertsson acknowledges the support by the Swedish Research Council (grant VR 2010-5864). The first author acknowledges valuable guidance from Gustav Cedersjö in setting up the lab.

References

[1] M. Kihl, A. Robertsson, M. Andersson, and B. Wittenmark, "Control theoretic analysis of admission control mechanisms for web server systems," *The World Wide Web Journal*, vol. 11, no. 1, 2008.

[2] X. Chen, H. Chen, and P. Mohapatra, "Aces: an efficient admission control scheme for qos-aware web servers," *Computer Communication*, vol. 26, no. 14, 2003.

[3] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multitiered web applications using queuing predictor," in *10th IEEE/IFIP Network Operation Mangement Symposium*, 2006.

[4] Q. Zhang, L. Cherkasova, and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," in *Proceedings of the Fourth International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2007.

[5] N. Tomov, E. Dempster, M. H. Williams, A. Burger, H. Taylor, P. J. B. King, and P. Broughton, "Analytical response time estimation in parallel relational database systems," *Parallel Comput.*, vol. 30, pp. 249–283, February 2004.

[6] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang, "Probabilistic performance modeling of virtualized resource allocation," in *Proceeding of the 7th international conference on Autonomic computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 99–108.

[7] A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson, "Predicting multiple metrics for queries: Better decisions enabled by machine learning," in *Proceedings of the 2009 IEEE International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 592–603.

[8] S. Tozer, T. Brecht, and A. Aboulnaga, "Q-cop: Avoiding bad query mixes to minimize client timeouts under heavy loads," in *ICDE*, 2010, pp. 397–408.

[9] M. B. Sheikh, U. F. Minhas, O. Z. Khan, A. Aboulnaga, P. Poupart, and D. J. Taylor, "A bayesian approach to online performance modeling for database appliances using gaussian models," in *Proc. Int'l Conf. on Autonomic Computing*, June 2011.

[10] P. Amani, M. Kihl, and A. Robertsson, "Multi-step ahead response time prediction for single server queuing systems," in *Proceeding of the 16th IEEE symposium on Computers and Communications*, July 2011.

[11] M. Kjaer, M. Kihl, and A. Robertsson, "Responsetime control of single server queue," in *46th IEEE Conference on Decision and Control*, 2007.

[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1998.

[13] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall PTR, 1998.

[14] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent NARX neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 27, no. 2, pp. 208–215, 1997.

[15] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies is not as difficult with NARX recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 8, pp. 577–602, 1996.

[16] H. Xie, H. Tang, and Y.-H. Liao, "Time series prediction based on NARX neural networks: An advanced approach," in *Machine Learning and Cybernetics, 2009 International Conference on*, vol. 3, July 2009, pp. 1275 –1279.

[17] J. M. P. Menezes Jr. and G. A. Barreto, "A new look at nonlinear time series prediction with NARX recurrent neural network," in *Ninth Brazilian Symposium on Neural Networks*, Oct. 2006, pp. 160 –165.

[18] A. G. Parlos, O. T. Rais, and A. F. AtiyaParlos, "Multi-step-ahead prediction using dynamic recurrent neural networks," *Neural Networks*, vol. 13, no. 7, pp. 765 – 786, 2000.

[19] M. Kihl, A. Robertsson, and B. Wittenmark, "Analysis of admission control mechanisms using non-linear control theory," in *IEEE International Symposium on Computer Communications*, 2003.

[20] E. Halili, *Apache JMeter*. Packt Publishing, 2008.

[21] M. Kihl, G. Cedersjö, A. Robertsson, and B. Aspernäs, "Performance measurements and modeling of database servers," in *Sixth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID 2011), June 14*, 2011.

[22] J. Mauro, B. Gregg, and C. Mynhier, *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD*, ser. Oracle Solaris. PRENTICE HALL COMPUTER, 2011.

[23] D. J. DeWitt, "The Wisconsin benchmark: Past, present, and future," in *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*, J. Gray, Ed., 1993.

[24] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to bayesian learning," in *Proceedings of International Conference on Neural Networks ICNN97*, vol. 3. IEEE, 1997, pp. 1930–1935.