Proactive Identification of Performance Problems

Songyun Duan
Duke University
syduan@cs.duke.edu

Shivnath Babu
Duke University
shivnath@cs.duke.edu

ABSTRACT

We propose to demonstrate *Fa*, an automated tool for timely and accurate prediction of Service-Level-Agreement (SLA) violations caused by performance problems in database systems. Fa periodically collects performance data at three levels: applications, database server, and operating system. This data is used to construct probabilistic models for predicting SLA violations. Fa currently uses graphical *Bayesian network* models because of their ability to support a wide range of inferences, including prediction and diagnosis, as well as their support for interactive visualization and presentation of complex system behavior in intuitive ways.

1. INTRODUCTION

The administration of enterprise database systems is often difficult and time-consuming. The performance of applications on these systems is affected by numerous factors like the availability of computing resources, characteristics of the stored data, concurrency of transactions in the workload, and values of database and system-level configuration parameters. Furthermore, these systems are usually very dynamic environments where data, workload, and system characteristics vary over time. For these reasons, database administrators (DBAs) have a hard time ensuring that application performance meets user demands.

While database administration has become hard, the reliability, availability, and serviceability of database systems have become more important than ever. These systems may now be required to meet *Service-Level-Agreements (SLAs)* specified by clients. For example, SLAs may be specified in terms of the maximum response time that clients can tolerate for transactions submitted to the system. The violation of SLAs because of system performance problems (e.g., resource contention, suboptimal execution plans) leads to user dissatisfaction and to the loss of commercial opportunities and revenue.

Accurate and timely prediction of potential SLA violations caused by performance problems can be very useful to DBAs. Such predictions enable DBAs to take remedial actions proactively and prevent the violations from occuring. For example, if we can predict in advance that a transaction will violate an SLA because of increased I/O, then the disk-block cache can be increased in size proactively (which may require reducing the size of another cache) to avoid the problem. As another example, if we can predict an SLA violation because of potential suboptimality of the execution plan for a query Q, then the DBA can proactively update the statistics required to choose the best plan for Q. Prediction of SLA violations is also very applicable in the new breed of *autonomic* or self-managing database systems [3].

Predicting SLA violations is challenging for several reasons. Database systems exhibit complex behaviors that stem from the interaction of workload, data, statistics, indexes, materialized views, optimizer, partitioning, concurrency, configuration parameters, software structure, and hardware. Pervasive instrumentation data from all component subsystems and applications may be necessary to model this behavior for the purpose of predicting SLA violations. However, the size and complexity of this data quickly overcome human ability to analyze it.

We propose to demonstrate *Fa*[1], an automated tool for timely and accurate prediction of SLA violations caused by performance problems. Fa collects performance data at the level of the database system, the operating system, and applications (transactions). Fa uses this data to build *probabilistic models* that can predict SLA violations. While many types of models are applicable for prediction, our work to date has focused on Bayesian networks [4] which are a form of graphical probabilistic models. In this proposal, we first give an overview of Fa (Section 2) and then describe a sample session from our proposed demonstration (Section 3). The snapshots from the sample demonstration session are available at [2]: `http://www.cs.duke.edu/ ~ shivnath/fa.html`

2. OVERVIEW OF FA

Figure 1 shows the current implementation of Fa. Clients send a workload of transactions to a MySQL database server. The workload is generated based on predefined scripts that enable us to create different types of performance problems at the server, causing SLA violations for transactions. While the system is running, Fa collects performance data periodically for transactions, the database system, and the operat-

[1] godchecker.com/pantheon/african-mythology.php?deity=FA

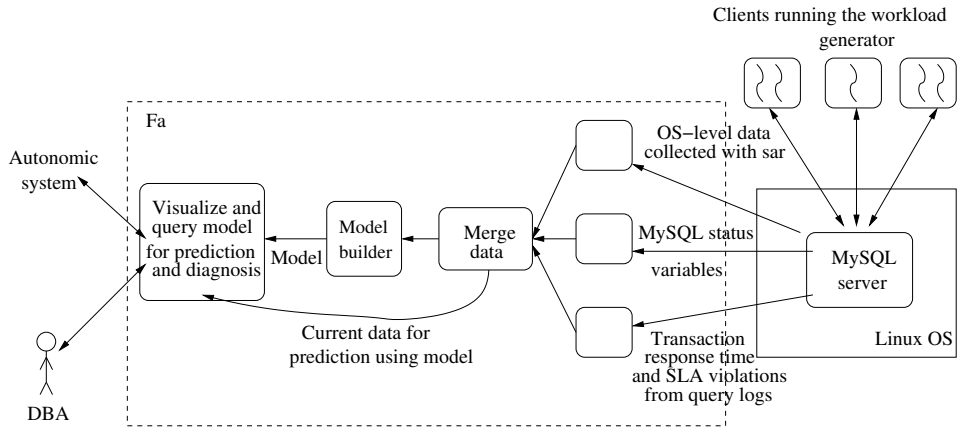

Figure 1: Overview of Fa

Measurement interval (1 minute)	OS−level data		MySQL status variables		Transaction−level	
	CPU util. (cpu_util)	#IOs (num_io)	#index accesses (idx_acc)		#SLA violations (sla_vio)	
10	16%	37	32	···	1	···
11	22%	41	36	···	0	···
12	72%	72	67	···	1	···
13	84%	70	65	···	0	···
14	75%	68	63	···	0	···
···	···	···	···	···	···	···

Figure 2: Performance data collected by Fa

ing system. The data collected in each measurement interval is merged based on the time of measurement. The combined data is input to Fa's *model builder* which builds and maintains a Bayesian network that represents the data.

The constructed Bayesian network can be interactively visualized in Fa using the *B-Course* tool [1]. This feature is designed for DBAs to visualize complex system behavior in simple and intuitive ways. The performance data collected during the current measurement interval is used by the Bayesian network to predict SLA violations for a window of time into the future. The Bayesian network can answer diagnostic queries on the predicted violations as well. Next, we give an overview of the main components of Fa.

2.1 Workload Generator

We have built a scripted workload generator that can generate workloads that cause different types of performance problems inside the database server. This tool gives us a controlled setting to study the relationships among various performance problems, their causes, their symptoms, and the accuracy and timeliness of models in predicting these problems. Currently, we inject performance problems and cause SLA violations through changes in the workload. We vary, e.g., the number and concurrency of transactions, the amount of data touched by each transaction, and the statistics of the stored data. Our sample demonstration session (Section 3) illustrates one instance of causing performance problems using the workload generator.

2.2 Data Collection

Figure 2 shows a snapshot in time of the data collected by Fa. (Detailed description and examples of the collected data are given along with snapshots from our sample demonstration session at [2].) During each measurement interval, Fa collects data at three levels:

- *Operating system*: Fa uses the *sar* utility [5] to collect performance data at the level of the operating system. While sar exports more than 100 attributes totally about the performance of various system components, we have so far used information about CPU utilization (6 attributes) and I/O transfers (5 attributes) only.
- *Database server*: MySQL maintains a set of status variables that provide information about server operation and performance. The version of MySQL that we use in our sample demonstration session maintains 133 status variables. (The latest version of MySQL maintains around 220 status variables.)
- *Transactions*: During each measurement interval, Fa records the number of transactions that completed during the interval, the average response time of these transactions, and the number of transactions that violated SLAs. This data is collected online from MySQL query logs.

The operating system, database server, and transaction data collected in each measurement interval are merged and input to the model builder. Currently, Fa does not collect any extra data apart from the data normally exported by the database server and operating system. For instance, we have not modified the source code of the systems to add extra instrumentation code. Our goal is to first study the effectiveness of prediction for off-the-shelf systems.

2.3 Model Builder

As mentioned earlier, Fa builds and uses Bayesian networks for predicting (and diagnosing) SLA violations caused by performance problems.

A Bayesian network $B(X)$ corresponding to a set of random variables $X = X_1, X_2, \ldots, X_n$ represents the *joint probabil-*

Measurement interval (1 minute)	OS–level data		MySQL status variables		Transaction–level	
	CPU util. (cpu_util)	#IOs (num_io)	#index accesses (idx_acc)		#SLA violations (sla_vio)	
10	16%	37	32	...	0	...
11	22%	41	36	...	0	...
12	72%	72	67	...	1	...
13	84%	70	65	...	1	...
14	75%	68	63	...	1	...
...	...	...	...	...	...	...

Figure 3: Shifted data used to build example model

ity distribution function of X, denoted *JPDF(X)* = $P(X_1, X_2, \ldots, X_n)$. $B(X)$ is a directed acyclic graph with n vertices corresponding to the n variables $X_1, X_2, \ldots, X_n$ comprising X. The vertex X_i in $B(X)$ is associated with a parameter $P(X_i|Parents(X_i))$ that represents the conditional probability distribution function of X_i given the values of X_i's parents $Parents(X_i)$ in $B(X)$. Since $B(X)$ represents *JPDF(X)*, the structure and parameters of $B(X)$ satisfy (approximately) the following equation for all $(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ [4]:

$$P(x_1, x_2, \ldots, x_n) = \Pi_{i=1}^{n} P(X_i = x_i|Parents(X_i))$$

As an illustration, consider a Bayesian network $B(X)$ to predict SLA violations three minutes in advance for the data in Figure 2, where X is the set of attributes in the figure. First, we shift the values in the transaction columns ahead by three rows to get the data in Figure 3. (Note that each row in Figures 2 and 3 corresponds to a one-minute measurement interval.) Then, we use a conventional Bayesian network construction algorithm [1] to build the network corresponding to the data in Figure 3.

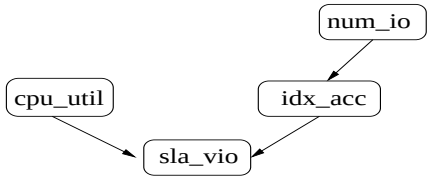

Figure 4: Example Bayesian network

An example network B for this data is shown in Figure 4. B shows that SLA violations (*sla_vio*) depend on CPU utilization (*cpu_util*), number of I/Os (*num_io*), and number of index accesses (*idx_acc*). Because $Parents(idx_acc) = \{num_io\}$ in B, *sla_vio* is independent of $\{num_io\}$ given the value of *idx_acc*, indicating that an SLA violation because of increased I/O is probably due to increased index accesses.

2.4 Using Bayesian Networks

Since a Bayesian network $B(X)$ captures *JPDF(X)* for attributes $X = X_1, X_2, \ldots, X_n$, $B(X)$ can answer any query of the form "$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = ?$," or combinations of such queries. This core functionality enables $B(X)$ to be used for a variety of purposes including prediction, diagnostic inferences, and intercausal inferences [4]; making $B(X)$ extremely useful in Fa. For example, given the measurements *cpu_util*=20%, *num_io*=40, and *idx_acc*=34 for the current interval, the probability of SLA violation three minutes ahead from the current interval can be computed from the Bayesian network in Figure 4 as "$P(sla_vio \geq 1|cpu_util = 20, num_io = 40, idx_acc = 34) = ?$"

In our current implementation, the model builder waits until data has been collected from a user-specified number of measurement intervals, and then builds the model. The model is not updated thereafter. (This limitation is mainly due to the lack of efficient off-the-shelf tools for dynamic maintenance of Bayesian networks.) We plan to address this limitation in future.

3. SAMPLE DEMONSTRATION SESSION

In this section we describe a sample session from our proposed demonstration of Fa. Snapshots of this session taken from our interactive visualizer are given at [2]. Section 3.1 describes the setup for this session, and Sections 3.2 and 3.3 outline how this session demonstrates Fa's model builder and the use of the constructed model for prediction and diagnosis of SLA violations.

3.1 Setup

In each individual session, we use our workload generator to cause a specific type of performance problem at the database server, creating occasional SLA violations. For the sample session considered in this section, we use a simple read-only workload composed of a single transaction type. Each transaction executes one parametrized range-search query on a table $R(A, B)$ with three million rows and an unclustered index on attribute A. Each query has the form "select avg(B) from R where A > \$1 and A < \$2," where parameters \$1 and \$2 are chosen randomly from the domain of A. The range [\$1, \$2] determines the minimum amount of data that has to be accessed to answer the query. By increasing this range, we can increase the transaction response time, potentially causing violations of SLAs specified on response time. For our sample session, a response time of over 20 seconds is an SLA violation.

3.2 Model Builder

As shown in Figure 1, Fa collects performance data and builds a Bayesian network for predicting SLA violations as the system runs. The Bayesian network built by Fa for the system and workload in Section 3.1 is shown in Figure 6. For readability, we have given intuitive names to the attributes in the performance data. Figure 6 also indicates the strength of the dependency represented by each edge in the Bayesian network. Because of space limitations, the snapshot of this network from our visualizer, the data used to build the network, and the complete description of the attributes are given in [2].

3.3 Prediction and Diagnosis

The first interesting thing about the network in Figure 6 to a DBA is that while the performance data contains 149 attributes, the network has only 21 attributes. The rest of the attributes were found to be irrelevant in this setting. Figure 5(a) shows a subgraph of Figure 6 around the *sla_vio* attribute. The value of this attribute gives the number of SLA violations in the measurement interval five minutes from the

current interval, for the single transaction type in the workload. *num_io* and *avg_time* in Figure 5(a) are respectively the number of disk transfers and the average response time of transactions in each measurement interval. Figure 5(a) also gives the prior (unconditional) distribution of *sla_vio*, e.g., showing that roughly 53% of the measurement intervals did not contain any SLA violations.

Figure 5(b) shows the probability distribution of *sla_vio* given that *num_io* is currently observed to be in $[1, 2.1]$. Note that this probability distribution is almost the same as in Figure 5(a). Figure 5(c) shows the probability distribution of *sla_vio* given that *num_io* is currently observed to be in $[3.1, 4.2]$. Now, *sla_vio* has a 95% of being > 0. This information can be used to predict SLA violations five minutes ahead of time when we observe that *num_io* is currently in $[3.1, 4.2]$. (Figures 5(a)–(c) illustrate the interactive querying of Bayesian networks in Fa.)

Three other interesting queries and inferences based on the Bayesian network in Figure 6 are illustrated at [2] using snapshots from the visualizer. For example, we show how a rule-of-thumb used by MySQL DBAs when presented with the facts in Figures 5(a)–(c) can lead to severe performance degradation in this case. This rule-of-thumb suggests increasing the size of MySQL's cache for index blocks. However, it is easy to see from the Bayesian network that SLA violations in this case are negatively correlated with the number of index accesses: As the search range increases for transactions in the workload (causing SLA violations), the database server stops using the index and resorts to full table scan.

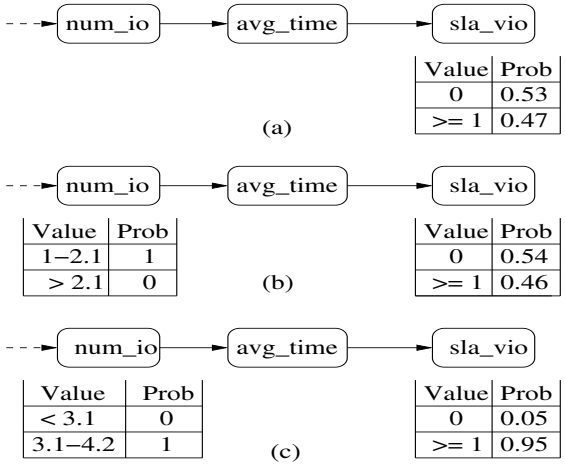

Figure 5: Using Bayesian network for prediction

4. CONCLUSION

We propose to demonstrate Fa, an automated tool for timely and accurate prediction of SLA violations caused by performance problems. Fa combines the power of Bayesian networks to support prediction, diagnosis, and other interesting inferences, with an interactive graphical interface for DBAs. In addition to demonstrating Fa, we hope to convey insights on the relationship among performance problems, their causes and symptoms, and the accuracy and timeliness of models in predicting these problems.

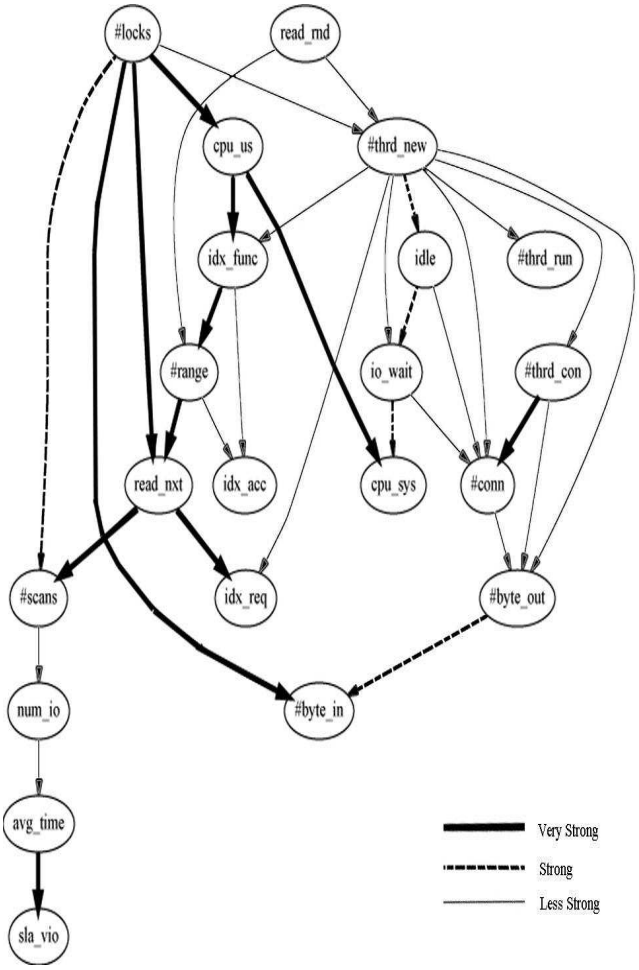

Figure 6: Bayesian network from sample session

5. ACKNOWLEDGMENTS

We are extremely grateful to the developers of B-Course for making their tool available online.

6. REFERENCES

[1] *The B-Course data analysis tool for Bayesian modeling.* `http://b-course.hiit.fi`.

[2] *Snapshots from the sample demonstration session.* `http://www.cs.duke.edu/ ~ shivnath/fa.html`.

[3] *IBM initiative on Autonomic Computing.* http://www.research.ibm.com/autonomic.

[4] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, Cambridge, United Kingdom, 2000.

[5] *Performance monitoring tools for Linux.* `http://perso.wanadoo.fr/sebastien.godard`.