QoS Control Based on Query Response Time Prediction

MARIO MILICEVIC¹, MIRTA BARANOVIC², VEDRAN BATOS¹
¹Department of Electrical Engineering and Computing
University of Dubrovnik
Cira Carica 4, Dubrovnik
CROATIA

²Department of Applied Mathematics
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, Zagreb
CROATIA

Abstract: - User oriented Quality of Service (QoS) of On-Line Transaction Processing (OLTP) systems (or Data Warehouse (DW)) is determined with a response time, availability, consistency and currency. In this paper we consider the influence of the system load and the system throughput on the response time, as well as a possibility of the accurate response time prediction - whereby that mechanism may be a foundation for (automatized) decision-making for a query (or DB procedure) scheduling. The possible application of proposed method can be the estimation of the optimal moment for the data transfer from OLTP system into DW or the appropriate moment for the complex query execution.

Key-Words: - quality of service, OLTP, data warehouse, data mining

1 Introduction
A diversity of the query duration is determined with the different aspects of the system load during the query execution. The complete description of the current state of the system is determined by numerous attributes associated with the hardware, operating system, internal DBMS mechanisms, DB design, application logic etc. The relationship between the system load/throughput and the query response time has been analyzed in the real OLTP system environment (dual-processor RISC configuration, RAID subsystem). Test query (a combination of select, update and insert statements) has been initiated in periods of the different system loads. CPU, memory and disk subsystem activity (especially partitions with DB data and logs) was monitored in 1s intervals.

Three typical cases are presented on Fig.1. In the case (A) query was started during the low system load level. Response time was 10s, the considerable activity growth can be noticed during the query execution. Cases (B) and (C) represent the test query execution during the moderate and increased system load. The response time was significantly prolonged (27s in the case (C)) and the other users' activities were jeopardized during this, or similar query executions.

2 Problem Formulation
A significant QoS improvement could be accomplished by the response time prediction – based on the current system state description. In that case the complex queries/procedures could be eventually postponed to the more appropriate moment. Of course, it must be mentioned in advance that prediction reliability is limited – because the query response time isn't consequence solely with the current system state, but also with the events initiated (by the users or system itself) during the afore mentioned query execution.

The proposed method collects numerous attributes needed for the detailed system state description – as an input for data mining algorithms generating classifications or actually query response time predictions.

During the learning phase (Fig.2.), a test query has been observed in the environment of real OLTP system. The system state description is recorded immediately - before the test query activation, as well as the corresponding response time. The system state is described with 37 attributes collected with available system tools or operating system and DB interfaces (CPU, memory and disk subsystem activity, physical and virtual memory availability, DB buffers use, etc.).

As mentioned above, the system state description is defined during 1s interval, immediately before the test query activation. There are at least two reasons for 1s
interval: (a) events during 1s adequately represent the current system state and (b) many system/DB [4] utilities (used for this purpose) have minimal sampling time also 1s.

- a model tree M5 [3] [5] - a regression tree (with the standard deviation used as a node impurity function) with linear regression functions at leaf nodes.

In order to perform classification experiments we used the WEKA platform [6] supporting chosen approaches:
- a classification tree construction using the method J48 - Weka's implementation of the C4.5 decision tree learning algorithm [2] [1]. This tree is built recursively, choosing the attribute with the highest information gain as the root. As this method implies nominal classes, all measured data are distributed (regarding response time) into three groups: A=favorable conditions, B=partly favorable conditions, C= unfavorable conditions;

- a model tree M5 [3] [5] - a regression tree (with the standard deviation used as a node impurity function) with linear regression functions at leaf nodes.

During the production phase (Fig.3.), using J48 method, the current system state before query activation will be classified into one of three anticipated classes (using rules generated during the learning phase). Based on that presumption, decision about the query activation can be made quickly. As opposed, the result of the method M5 is a numeric prediction of the query duration as a function of the current system state.
3 Case Study

The learning phase has been conducted during several days, where the same test query was executed more than 600 times - during all characteristic periods related to the rhythm of typical users activities.

As can be noticed in Fig.4., response times from 3.3s to 67s were achieved during the learning phase. The best response times are the consequence of successive query activations during periods of the low system activity (e.g. during the night) - so all selects, inserts and updates were accomplished using the data found in the cache memory (DB buffers). Another extreme (67s) is a consequence of the intensive system load - where a server may be facing contention for its resources (with an additional possibility of locking mechanisms used by the underlying database). All remaining cases representing common system load are placed between mentioned extremes - for example, it can be noticed that even 31% of these samples have the response time between 9s and 13s - as a result of the typical simultaneous users work (mainly without demanding transactions). The majority of data can be found in DB buffers, physical reads/writes are seldom (case (B) on Fig.1.).

During J48 method implementation, as mentioned before, all cases are divided into three groups: A (3s-12s), B(12s-20s) and C(+20s). Achieved classification accuracy (with recommended 10-fold cross-validation) is 78% - a quite satisfactory figure taking into account the nature of the problem (the system state description used for the prediction doesn't presume anything about processes initiated during the query execution).

The corresponding classification tree is shown on Fig.4., where \%idle is the percentage of the time when the CPU is idle, \%sys is the number of device interrupts, \%cached_rd is the number of free memory pages etc. The important fact can be noticed: only 1 of 621 samples is classified as (A) instead of (C) – what is, potentially, the most troublesome mistake.

Advantages of the classification into three fixed classes are unambiguity and clarity of results, but there are also several potential problems:
- it is hard to determine class borders. The distribution of samples can be influenced by the representativeness of sample collecting periods;
- sample belonging to the certain class still doesn't represent corresponding system state faithfully. For example, it can be important to distinguish two samples from (C) class with the response times 21s and 60s;
- if (during the production phase) it's required to change borders between classes, a new collection of rules must be generated (using data from the learning phase).

The proposed alternative to the classification tree (method J48) is the model tree according to the method M5. As mentioned above, the key difference is that the final result of M5 is a numerical prediction of the response time (instead of the classification in the fixed classes), i.e. a regression tree with the linear regression functions at the leaf nodes (Fig.5.). Expression for LM1 - for example - has a form:

\[
LM1 = -0.0099 \times \%user - 0.0106 \times \%sys - 0.0176 \times \%iowait - 0.012 \times \%idle + 0.0385 \times in - 0.0001 \times cs - 0.0267 \times dskreads + 0.0006 \times \%cached_{rd} + 0.0005 \times pagwrits + 0.0006 \times \%cached_{wr} + 0.0018 \times lchwaits + 0.0129 \times _tm_actDB - 0.0007 \times tps_{HD\_DB} + 0.0111
\]

It can be noticed (Fig.6.) that the prediction follows the distribution of samples quite confidently – except for response times below 7s. However, because the shortest response times are obtained during low activity periods of users activity, the mentioned error does not have essential influence.

The model tree (Fig.5.), generated during the learning phase, can be used in the production phase regardless the position of the border between classes (i.e. acceptable and unacceptable response times).

Confusion Matrix for two different border values (OK stands for the acceptable conditions) is:

<table>
<thead>
<tr>
<th>T_{\text{resp}} = 13s</th>
<th>OK</th>
<th>notOK</th>
<th>&lt;-- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>44%</td>
<td>5%</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>6%</td>
<td>45%</td>
<td>notOK</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T_{\text{resp}} = 17s</th>
<th>OK</th>
<th>notOK</th>
<th>&lt;-- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>61%</td>
<td>10%</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>5%</td>
<td>24%</td>
<td>notOK</td>
<td></td>
</tr>
</tbody>
</table>
Fig. 4. J48 Classification Tree

Fig. 5. M5 Model Tree
4 Conclusion
QoS of OLTP (or DW) system can be enhanced by using data mining algorithms aimed to predict the query/procedure response time, and to eventually postpone their initiation. Although prediction accuracy is limited by stochastic events initiated during the aforementioned query/procedure execution, QoS improvement can be considerable.

References: