Database Performance Monitoring and Tuning Using Intelligent Agent Assistants

Sherif Elfayomy and Jigisha Patel
School of Computing, University of North Florida, Jacksonville, FL, USA

Abstract - Fast databases are a necessity in today's E-commerce environment. Information from the database has to be provided at fast rates to impatient customers, otherwise they will move on to competitors with more attentive services. Corporations needing up-to-date internal information cannot wait for long running queries that process numbers and produce detailed statistics. Instead, they need databases capable of providing data in fractions of seconds to compete in today's economy. Database performance tuning has always been one of the most important tasks for database administrators (DBAs). This research presents an intelligent agent assistant to aid DBAs in performance monitoring tasks and the automation of resolution actions. The assistant notifies DBAs when performance problems are detected and resolved. DBAs are expected to provide the agent assistant with definitions for the performance problem conditions and the possible resolution actions. They must also develop a notification communication mechanism for the problems the assistant monitors.

Keywords: Intelligent Agents; Performance Tuning; Database Performance.

1 Introduction

In today’s competitive business environments, application performance plays a major role in business success. Bad application performance can result in unhappy customers and revenue loss. Usually applications have their data stored in large database repositories. Database performance affects applications' response times, which eventually affect the end user's experience. Database performance is directly affected by the Database Management Systems (DBMS) resources allocation where many resource dependencies are involved, which makes performance troubleshooting a complex task.

Highly paid and experienced Database Administrators (DBAs) are typically needed to perform the database performance-tuning task. This task has to be performed on a constant basis to cope with the changes in performance due to database growth and workload changes. Achieving peak database performance is a difficult task but is not impossible. Database tuning requires DBAs to identify resources not properly tuned and then tune them to obtain better performance.

The intelligent agent assistant is a knowledge-based system that perceives its environment, reasons to interpret perceptions, draws interfaces, solves problems, and determines Actions; and acts upon that environment to realize a set of goals or tasks for which it was designed. There are different types of intelligent agents. For example, user or personal agents take actions on behalf of the user to perform tasks such as gathering information on a subject of interest, preparing customized news snippets, setting up and prioritizing e-mails according to the user’s preference, and playing games. Another type of intelligent agent is the monitoring and surveillance agent, which can observe and report on equipment, devices, and systems. This type of agent can automate tasks like tracking equipment inventory, planning, and scheduling order deliveries.

Intelligent agent assistant (IAA) can be employed to tune the performance of DBMSs on behalf of DBAs, which should substantially reduce the cost of database ownership. IAA will continuously provide effective monitoring and allow the DBMS to respond quickly to performance issues. It should also continue to improve the DBMS performance for both dynamic and static workloads. It also should be platform independent, easy to maintain, scalable, extensible, and able to communicate.

2 Background

Many efforts have already been made towards automating the control of DBMS resources. An automated system should be able to identify and readjust DBMS resources. Anolan et al., proposed a design and implementation of a global self-tuning architecture [1]. Their research considered performance tuning as a global issue, given changes of a single parameter can affect the performance of other operations. It described the proposed architecture and also discussed system’s integration within PostgreSQL in an implementation based on software agents. Software agents used to provide a more flexible approach to include automatic tuning features in PostgreSQL DBMS.

JP Bigus et al., described AutoTune, an agent-based approach to automated tuning [2]. This agent-based approach to automated tuning did not require prior knowledge of the controlled system being tuned. AutoTune enabled target systems to expose workload metrics (e.g., RPC arrival rates), configuration (e.g., processor speeds), and service levels (e.g., response times), as well as means to manipulate tuning controls (e.g., admission control parameters). This agent-based approach constructs a generic model of the target system (e.g., by training a neural network) and from this derive a controller. A prototype of AutoTune agent was...
implemented in the Agent Building and Learning Environment (ABLE)[2]. Also, Darcy G. Benoit demonstrated the automatic diagnosis of performance problems in DBMSs [3]. This dissertation discussed and demonstrated ideas for diagnosing database performance problems at small subset levels. The diagnosis model was tested on workload changes and database size changes. The model creates diagnosis trees for different types of databases and their workloads.

Chung et al., focused on goal-oriented buffer pool management [4]. In this paper, a performance index was calculated based on the response time of the buffer pool. They presented a goal-oriented approach in which the user can specify a buffer pool response time goal and total count for all buffer pools in the database. This goal-oriented approach dynamically changes each buffer pool size, while maintaining the total number of buffers. This dynamic tuning results in a great improvement in database performance. Also, Agrawal et al., investigated whether the selection of materialized views and indexes can be automated for DBMSs [6]. Database performance can be improved with a correct combination of indexes and materialized views. They developed algorithms and an architecture; and, explained how they were able to identify a small set of materialized views and indexes, despite a totally different structure.

Chaudhuri and Narasayya explained database technology as RISC-style data managers with self tuning capabilities [5]. Each individual component is self tuning and exhibits predictable performance by eliminating the need for manual tuning by a DBA. These components are capable of determining which database statistics are essential for boosting performance and make sure they are updated frequently. They explained that database tuning must consider the relationship between workload characteristics, knob settings, and the resulting performance, in a quantitative manner.

Oracle provides a database “grid control agent,” which can help DBAs monitor and maintain Oracle databases. Oracle’s agent provides a good mechanism for database administration tasks and sends warning/critical notices, depending on setup. Jobs can be setup as shown in Figure 1 by providing values for job name, description, and SQL script fields. User should also select one or more databases from the list on which this job will run.

The desired schedule should be setup by clicking the schedule tab, as shown in Figure 2. Oracle’s agent provides several scheduling options such as one time only, monthly, yearly, and user-defined intervals. It also has options for allowing simple SQL statements or using complex SQL script stored in the database server for database monitoring. Despite these good features, it is missing some important features, such as a mechanism to take automatic actions and an option to add multiple SQL statements.

Microsoft SQL server also provides an agent tool to help DBAs monitor SQL server databases. New jobs can be scheduled by providing job name, schedule type, start time, and frequency to run the job, as shown in Figure 3. Job status notifications can be sent using email, pagers, or net send commands by providing email address, pager number or machine address. This agent provides several options for scheduling and notification, but is not flexible enough to support non-Microsoft SQL server databases.

A review of the published literature on automated performance tuning approaches and tools provided no evidence of the availability of a generic tool to assist DBAs in managing a variety of DBMSs in a heterogeneous environment. In this paper we present an intelligent agent assistant that was developed to work for a variety of databases such as Oracle, SQL Server, MySQL, to send SMS notifications to DBAs, to be easily manageable for most relational databases through the use of XML scripts, to use standard SQL queries to gather performance information from databases, rather than platform dependent queries such as T-SQL and PL-SQL, and to allow DBAs to provide any number of performance tuning queries.
In this research effort an intelligent agent assistant was designed and developed to operate on user defined rules. In the form of database performance tuning rules, SQL queries are used to measure certain performance metrics of the database server and/or to modify some of the performance parameters, if need be. Once the agent detects unsatisfactory performance measures, it performs the actions specified in its rule set, so it can rectify the performance issues. DBA needs to define IAA rules by specifying resources to monitor, queries to collect performance metrics, threshold levels of minimum acceptable values for performance metrics, corrective actions, if performance problems are identified, and frequency of performing these checks.

The IAA can run continuously in the background without any human intervention. The system was constructed to use multithreading, specifically Java threads. Multithreading provides for more responsive GUI, leveraging multiprocessing hardware, simpler modeling, and asynchronous processing of rules. Once the agent assistant has started, it starts the first thread, which is the main/parent thread. The main thread reads stored rules information from the knowledgebase, implemented in XML, and spawns off other child threads, according to a specified schedule, to monitor resources and perform corrective actions. The main thread keeps running until stopped through the user interface program (GUI). DBAs can add, edit, or delete rules through the GUI. Additions and changes are stored in the XML-based knowledgebase and take effect only after restarting the agent.

IAA uses an XML file to store all the information required to connect to the DBMS, including server name, instance name, user id, encrypted password, queries required to gather resource information, frequency at which resource information is gathered, formulas required to determine if performance problems exist, resolution actions, and phone number to notify when corrective actions are performed. XML was chosen to store agent’s rules because of its extensibility and ease of use to communicate between different system components, which otherwise would be unable to communicate or would be rather tightly coupled. Figure 4 shows IAA architecture and interaction between GUI, XML file, and agent. The XML file is validated against the DTD schema file exhibited in Figure 5.

Figure 3: New Job Schedule (SQL Server)

3 Intelligent Agent Assistant

Figure 4: IAA System Architecture

The rule node has a number of <query> elements; each is identified by a name parameter (Figure 7). These queries are used to gather database performance statistics.

Part of the XML rule node is shown in figure 8. The <interval> element is another node that stores the execution frequency of a rule, in minutes. The <condition> node contains a formula prepared by the DBA to determine if a performance problem exists. The condition may use some of the values returned by the queries, where each query is
identified by name. A condition can be a combination of queries with logical operators (AND and OR). If a condition is satisfied, the associated actions will be performed, as prescribed in the <action> element. The DBA can also specify multiple actions.

3.1 Use Case: Tuning Buffer Cache Hit Ratio

A use case was developed to demonstrate IAA’s operation and the ease of use to customize the performance rules. The Buffer Cache Hit Ratio is an Oracle metric used for the rate at which Oracle finds data blocks in main memory. A correctly tuned buffer cache can significantly improve overall database performance. The traditional trade-off between efficiency and performance exists here. A small buffer cache would result in more disks I/O operations, which reduces performance. On the other hand, a big buffer cache is a waste of the server’s scarce memory (less efficient). The hit ratio depends on physical and logical reads. A logical read occurs whenever the user requests data from the database, which is in memory or server hard disk. A physical read occurs only when the data must be read from hard disk. Oracle stores physical and logical read details in the V$SYSSTAT table. The hit ratio value can be determined with the following SQL query:

```
SELECT name, value FROM v$sysstat
WHERE name in ('db block gets', 'consistent gets', 'physical reads');
```

The cache hit ratio is calculated using the following formula:

```
Hit ratio = 1 - (PR / (DG + CG))
```

Where: PR = Physical reads,

DG = Db block gets, and

CG = Consistent gets

Ideally, the cache hit ratio should be greater than 90%. If it goes below the desired value, the database parameter “DB_CACHE_SIZE” will need to be increased. To create an agent rule, for tuning based on the buffer cache hit ratio, SQL statements should be converted into single column queries named Q1, Q2, etc. The condition and action can use these query names. The agent rule created for this use case is shown in Figure 9.
4 Conclusion

DBMS resources must be carefully allocated to achieve a good balance of performance versus efficiency. This task is complicated by the fluidity of DBMSs. Data and workloads keep changing and resource allocation must cope with these changes. The only solution to this performance dilemma is to keep monitoring the database continuously. The automation of some database monitoring and resource management aspects lessens the burden on expensive DBAs. The use of intelligent agent assistants is an effective approach to solving the database-tuning problem. We plan to continue testing the IAA using more use cases and enhance its operation by creating more flexible scheduling, create a web interface to manage rules, allow rule dependency, and allow for other types of notifications such as email.

5 References


