CS5226 Project – Semester 2, 2012/13

Database Performance Dashboard

Introduction

You can gather database statistics (e.g. Pin hit ratio for the library cache, Hit ratio for the buffer cache, number of physical reads and writes, etc) from an Oracle instance to facilitate performance tuning. Another aspect of performance tuning is to be able to define performance baselines and use them as the basis for performance comparison, and tuning. In Oracle, the Automatic Workload Repository (AWR) can be used for finding out various performance snapshots.

A **Database Performance Dashboard** is a useful tool for database administrators (DBAs) to monitor the health of the database and to support database performance investigation. The performance dashboard provides a one-stop overview of various key performance indicators (KPIs) on database performance. Users can then make use of the high-level information to further zoom in to identify the performance issues, and tune the system to improve performance.

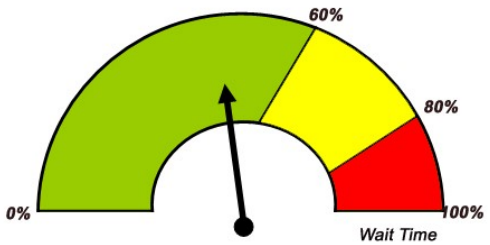

Figure 1 – Example of a scorecard-like visualization of wait time for database transactions

Requirements

In this project, you will develop a web-based application to help provide DBAs with a performance dashboard with key database statistics. The system will specify a user with the valid credentials to query the various dynamic performance views in order to obtain the statistics.

The application should monitor the following database parameters:

- Shared Pool
- Buffer Cache
- Redo Log Buffer/Files
- Memory area used for sorting

You are encouraged to define additional parameters that need to be monitored. You should provide a discussion in the report to explain the rationale for including these additional parameters.

The web-based application should provide the following features:

1. 3 levels of breakdown for each database parameter being monitored.
    a. Top-level breakdown: Shows the health of the specific database parameter being monitored (green: healthy, yellow: not so healthy, red: need DBA attention)
        i. In the visualization, you just need to show the three colours. It need not be similar to the scorecard-like visualization in Figure 1.

b. Second-level breakdown: Shows the aggregated values (e.g. wait-time) for the database parameter per x unit time block as specified by the user
c. Low-level breakdown: Shows the aggregated values for the database parameter per y unit time block (y < x) within each of the x unit time block.

   For example, a user can specify x to be 1 hour, and y 15 minutes. Thus after collecting data for 24 hours, besides seeing an average value for the 24 hours, the user can view the average for 0-1 hour block, 1-2 hour block, and so forth. User can also zoom in to look at a particular block, say 3-4 hour block, and (s)he gets to see the average values of the 4 15-minute blocks within that hour.

2. Provide a configuration page(s) to define the various thresholds used to determine whether a specific database parameter is healthy, not so healthy or need DBA attention.

3. For each of the performance issue identified that is in the "red" area, the application should provide information on the parameter in init.ora that should be modified in order to solve the performance issue.

4. On-demand reports – where the users can specify the date ranges for the database parameters being monitored.

5. A Debug interface - Allow the user to issue SQL commands to the database. The results should be displayed neatly on the webpage.

Note:
(a) To facilitate data collection, you can also write backend programs to query the database at specific time intervals and store it within the database. You should document this in the Project Report if you need to make use of backend programs.

(b) You can use any programming or scripting languages to develop the application.

(c) You can implement the web-based application on your PC/laptop using PC version of Oracle. Alternative, you may deploy your application in the Sun Solaris Zones environment. You can apply for a Solaris zone account for project development.

Project Team
Each project team will consist of 3-4 members. More effort will be expected of a 4-member team.

Deliverables
You should
(a) Upload a zipped file to IVLE Workbin. The filename should be the team number which will be assigned to you once the team is formed.
(b) Submit a hard-copy of the project report to the lecturer by the due date.

The zipped file should be clearly marked with the details of the team members, and should contain the following deliverables:

- Program source code
- **README.txt** – Provide instructions on how to setup the web application to be evaluated.
- Database Setup Scripts
    - **setup.sql** – database SQL scripts to create the tables that are used to support the application
    - **data.sql** – Used to pre-populate the table with the data used by the application
- Project Report, which consists of
  The report should not exceed 15 pages (font 12, single column, single spacing).
    - **Project Admin**
        - Breakdown of the key areas done by each member of the team
    - **System Design and Architecture**
        - The overall architecture and design of the application
        - Discussion on the database design for the tables used to support the application
        - Details on how the various statistics are computed, and the SQL commands used to obtain the statistics
        - Screenshots of the application
        - Highlight specific novel ideas that your team has used in the application
        - Insights of performance tuning that you have learnt in the course of doing the project.

You should also provide a hard-copy of the project report.

Submission Deadline

The submission deadline is **12 April 2013 (Friday), 5pm**.
Marks will be deducted for late submission.
Project demos will be scheduled during week 13 (15^{th}-19^{th} April 2013).

Resources
Oracle documentation - http://www.comp.nus.edu.sg/~oradoc,

Solaris Zone Accounts -
https://mysoc.nus.edu.sg/~wiki/index.php/Getting_started_with_SoC_Zone

Reporting Tools - http://jasperforge.org/