Proactive Performance Monitoring Using Metric Extensions and SPA

Mughees A. Minhas
Oracle
Redwood Shores, CA, USA

Keywords:

Oracle, database, performance, proactive, fix, monitor, Enterprise manager, EM, developers, SQL Performance Analyzer, SPA, metric extensions

Introduction

Since introduced over 45 years ago, databases have become not only the foundation for the modern society, where governments store information for managing the municipals and counties, but also the corner stone for any company or organization allowing them to operate.

Because of the central role of the database makes it is obvious that the database performance will play an important role for any application or system that makes use of databases. Traditionally, discovering and addressing database related performance issues have been a very time consuming and often a user driven reactive activity, resulting in badly performing applications.

This paper describes a modern, pro-active approach to detect issues in the database performance before the users will begin to notice them as well as how to address these issues with a minimal effort.

Finding and addressing database issues

Traditionally application developers have had limited possibilities to identify and tune SQL queries for their applications and had to rely on daily checks or end user feedback in order to detect performance issues. After finding out that there could be a performance issue, the developers had to trouble shoot and diagnose the issue based on the information provided by the end users and their experience with the application or though database metric reports. Once a potential fix was implemented; the developer had to wait for end user feedback to validate the fix, followed by trying to determine if the issue might occur again, maybe in a different place, as well as creating custom scripts to monitor database metrics combined with complex alerting.

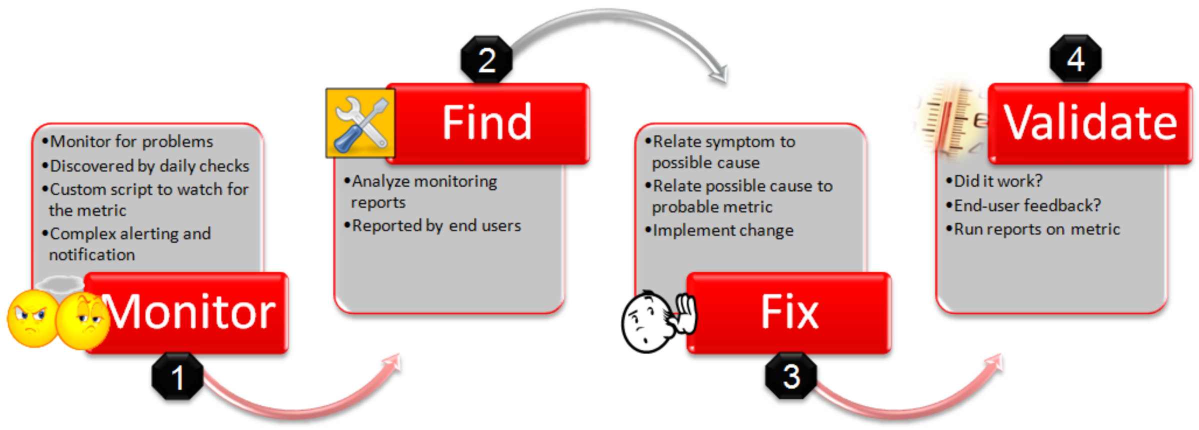

1
Monitor
•Monitor for problems
•Discovered by daily checks
•Custom script to watch for the metric
•Complex alerting and notification
2
Find
•Analyze monitoring reports
•Reported by end users
3
Fix
•Relate symptom to possible cause
•Relate possible cause to probable metric
•Implement change
4
Validate
•Did it work?
•End-user feedback?
•Run reports on metric

Illustration. 1: Monitor, find, fix, and validate database related issues

Oracle Enterprise Manager and a more efficient way

Oracle Enterprise Manager is the centralized management and monitoring solution for the whole Oracle product stack, offering a heterogeneous extendible solution though plug-ins. Enterprise Manager with its customizable notifications system, manages targets by their exceptions and can therefore be used in a lights-out datacenter scenario. Oracle Enterprise Manager can integrate with 3rd party help-desk solutions as well as non-Oracle management systems for enterprise deployments. Enterprise Manager also integrated with MyOracle Support, allowing customers to get direct access to patches, knowledge base articles that provide an easy and time saving way to manage and update the managed targets.

We will now take a look how Oracle Enterprise Manager can be used to address the challenges around finding, fixing, validating and tracking database issues.

1 Using Oracle Enterprise Manager to monitor database metrics

Enterprise Manager 12c Metric Extensions (ME) is a user extension to Enterprise Manager targets, replacing the User Defined Metrics (UDM) used in previous versions of the product (version 10g and 11g). Metric Extensions can be defined for any target managed by Enterprise Manager. The Metric Extensions supports SQL, OS as well as JMX fetchlets and can therefore make use of whatever technology that the target itself is already making use of.

Enterprise Manager also provides lifecycle support for Metric Extensions, covering the whole lifecycle from development, testing to deployment. In this way a developer can create a Metric Extension and test it against some available targets and once working as designed, he can save it as a ‘deployable draft’. The ‘deployable draft’ can then be deployed to targets in a test environment, where its behavior can be validated along with the metric data and notifications. When the testing is done, the

Metric Extension can be published and made available for general usage in the production environment.

Instead of developing custom scripts for monitoring the database for issues, developers or database administrators can utilize Oracle Enterprise Manager and Metric Extensions to monitor applications for activities that might have a negative impact of the application performance, such as sudden increase of sales orders in the order queue or application specific locking, resulting in slower performance for the end user.

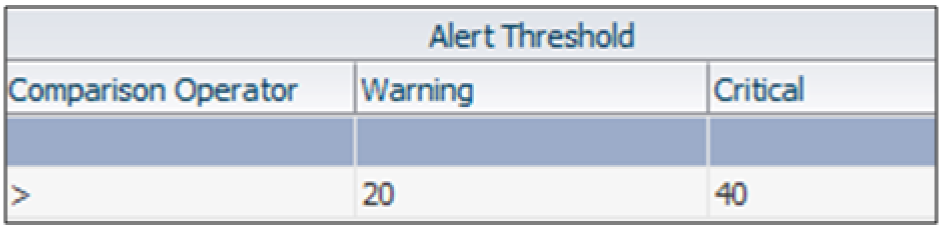
Example: How to monitor sales order queue in the Oracle Database using Metric Extensions

An easy way to monitor something like a sales order queue is to make use of Oracle Enterprise Manager and the Metric Extensions. The allows you to periodically monitor the database sales order queue and configure it to trigger an alert when your Metric Extensions threshold is violated.

Assuming we have a sales order queue and the sales orders have 3 different statuses (NEW, PROCESSING and COMPLETE) . Begin by creating a Metric Extension in Enterprise Manager that monitors the ORDERS_QUEUE table using a query like the one listed below

```
Select decode(status,1,’NEW’,2,’PROCESSING’,3,’COMPLETE’) as STATUS,
count(status) as ORDER_STATUS
from oe.order_queue
group by status
having STATUS in (1,2)
```

Next you define alert threshold for NEW and PROCESSING status in the queue based on your preferences.



Alert Threshold		
Comparison Operator	Warning	Critical
>	20	40

Once the metric extension has been deployed, it will coexist with existing out-of-the-box metrics or other metric extensions and it is now possible to view queue details like current numbers of orders in the queue, historical data on the orders as well as the alert history.

Since the metric extension has been deployed in Enterprise Manager, it can also make use of features like email notification in case any violation occurs.

2 Using Oracle Enterprise Manager Database diagnostics to find database issues

Oracle Enterprise Manager does not only provide monitoring and management capabilities, but also comprehensive database diagnostics solutions like Automatic Database Diagnostic Monitor (ADDM), that analyzes data in the Automatic Workload Repository (AWR) to identify performance bottlenecks. ADDM can locate the root cause for the reported issues and provides recommendations so that the identified issues can be resolved. The ADDM analysis task is performed every time an AWR snapshot is taken and all of its findings and recommendation are stored in the database. The ADDM analysis includes the following metrics

- CPU load
- Memory usage
- I/O usage
- Resource intensive SQL
- Resource intensive PL/SQL and Java
- RAC issues
- Application issues
- Database configuration issues
- Concurrency issues
- Object contention

The ADDM family of products also includes Compare Period ADDM, Real-time ADDM and Spot ADDM, that can all be used to find and identify database related issues.

By using ADDM you can quickly and easily find database related issues for your database and the applications relying on the database. Using ADDM allows you to be pro-active in finding issues and does not require you to rely on difficult to diagnose end-user feedback. ADDM will always be up to date and does also suggest solutions to your database issues.

3 Using Oracle Enterprise Manager to fix database issues

Seamlessly integrated with the database diagnostics, the Oracle Database SQL Tuning Advisor provides an in-depth analysis and recommendation for the fix recommended by ADDM.

So instead of manually trying to optimize any identified issues, you can make use of the SQL Tuning Advisor from Enterprise Manager.

The SQL Tuning Advisor uses one or more SQL statements and invokes the Database Automatic Tuning Optimizer to tune the statements. The output of the SQL Tuning Advisor is recommendation together with the rationale for each recommendation and the expected benefit.

Once the SQL Tuning Advisor provides its findings, you can use Enterprise Managers user interface to either ignore the recommendations or accept them in order to complete the tuning of the statements.

4 Using Oracle Enterprise Manager to validate database fixes

Once the database issues have been identified and addressed, you need to validate the fixes.

Oracle Real Application Testing is a unique database centric testing solution for Oracle Databases and can be used from command line or by using graphical user interface of Oracle Enterprise Manager.

Real Application Testing allows users to record production workloads, that later can be used for testing purposes. Real Application Testing offers two features; Database replay and SQL Performance Analyzer.

The SQL Performance Analyzer (SPA) can be used to test and predict impact of system changes on the SQL query performance and is seamlessly integrated with the diagnostics and tuning features earlier described. SPA also allows any tests to be re-executed in case any changes have been applied.

SPA offers a broad testing coverage and is an ideal solution when validating

- Optimizer statistics
- Database parameter changes
- Database schema changes
- Implementation of tuning recommendations
- I/O subsystem changes
- Consolidations
- Test, standby, production
- Testing of Oracle applications like EBS, Siebel or 3rd party applications like SAP as well as home grown applications
- Use as extensions to home-grown scripts, application specific database changes or 3rd party solutions that make use of STS compare analysis

You have now seen the individual components making up the solution and that it doesn't have to be difficult to monitor, find, fix or validate database related issues; it can be easy and simple by making use of Enterprise Manager and the Metric Extensions.

Let us explore two use cases on how the above can be applied in real world scenarios.

Use case: Detecting run-away queries using Metric Extensions

A run-away query is basically a SQL bug that consumes too much CPU or need too long execution times. It is critical to find and eliminate these type of issues from the application code to ensure good application performance.

By making use of the previously discussed possibilities in Enterprise Managers Metric Extension we can build a solution that detects issues like run-away queries.

Real-time SQL monitoring was introduced in Oracle 11g Database. This monitors long-running and parallel query SQL executions though its rich and user interface.

The real-time SQL monitoring is available in GV$SQL_MONITOR or V$SQL_MONITOR tables.

For our use case, we will create a Metric Extension using a GV$SQL_MONITOR query using Enterprise Managers guided wizard.

Metric Extension:

```
WITH SQLM as
(select sql_id,sql_exec_start,sql_exec_id
      ,MAX(M.user#)                                          as UserNum
      ,MAX(M.username)                                       as UserName
      ,MAX(NVL(PX_QCInst_ID,M.inst_id))                      as ExecInst
      ,MAX(NVL(PX_QCsid,M.sid))                              as ExecSid
      ,MAX(NVL(PX_QCsid,M.session_serial#))                  as ExecSerial
      ,MAX(CASE PX_QCsid WHEN null THEN null ELSE M.status END) as Status
      ,DECODE(count(distinct px_server#),0,'SERIAL','PARALLEL') as PQ_SERIAL
      ,COUNT(DISTINCT M.inst_id)                                  as instances
      ,MAX(NVL(PX_MAXDOP,1))                                      as MaxDOP
      ,MAX(last_refresh_time)                                          as last_refresh_time
      ,SUM(elapsed_time/1000000)                                       as DBtimeSecs
      ,ROUND((MAX(last_refresh_time)-MIN(sql_exec_start))*24*60*60,1)   as ExecElapsedSecs
      ,ROUND((MAX(last_refresh_time)-MIN(sql_exec_start))*24*60*60                            +
SUM(NVL(queuing_time,0)/1000000),1) as TotalElapsedSecs
      ,ROUND(SUM(NVL(queuing_time,0)/1000000),1)                       as QueuingSecs
      ,SUM(cpu_time/1000000)                                           as CPUsecs
      ,SUM(user_io_wait_time/1000000)                                  as IOsecs
      ,SUM((application_wait_time+concurrency_wait_time+cluster_wait_time)/1000000)          as
WaitSecs
      ,SUM((plsql_exec_time+java_exec_time)/1000000)                    as JavaPLSQLsecs
      ,SUM(buffer_gets)                    as BuffGets
      ,SUM(disk_reads)                     as DiskReads
      ,SUM(direct_writes)                  as DirectWrites
 from
      gv$sql_monitor M
group by sql_exec_id,sql_exec_start,sql_id
)
select sql_id              as SQL_ID
      ,ROUND(SUM(CPUSecs))  as TotalCPUSecs
      ,SUM(ExecElapsedSecs) as TotalElapsedSecs
--      ,IOSecs
--      ,WaitSecs
      ,SUM(BuffGets) as TotalBuffGets
      ,SUM(DiskReads) as TotalDiskReads
      ,MAX(UserName)  as SampleUser
      ,COUNT(*)       as NumExecs
from
      SQLM
where
    ( status like 'EXECUTING%'                                              -- currently
executing or
     OR (status like 'DONE%' AND last_refresh_time > SYSDATE - 15/(24*60))   -- finished last
15 minutes?
    )
group by sql_id
having upper(MAX(Username)) <> 'TOPSQL'
```

Once the metric Extension has been created you have to define the monitoring intervals and the alert configuration. We will execute the monitoring query every 15 minutes to ensure we are not missing anything and we will set alerting thresholds on CPU and Elapsed times.

	Warning	Critical
TotalCPUSecs (Total CPU -secs)	300	600
TotalElapsedSecs (Total Eapsed Time -secs)	1800	3600
TotalCPUSecs (Total CPU -secs)	50	100
TotalElapsedSecs (Total Eapsed Time -secs)	200	400

Then deploy the Metric Extension and Enterprise Manager will send a notification alert when it detects run-away queries.

Use case: Identifying High Risk SQL in Growing Data Volume Environments

Today the data volumes are rapidly growing at a rate greater than before. Rapidly growing data volume can often result in performance degradation and unplanned downtime.

Addressing the problem can be done either as a more traditional, reactive fix or as a proactive optimization, let’s explore these options more in detail.

Reactively addressing the issue would mean a more traditional approach with adjusting system or database configurations, tuning the system or storage, making more hardware resources available or to restrict user access.

With the solutions we earlier explored you would be able to take a more proactive approach by identifying SQL with potentially volatile execution plans in increasing data volume databases, followed by preempt the SQL performance in a reliable way. And last but not least, assess and plan for how you can handle data growth.

You can identify high risk SQL with growing data volumes by using SQL Performance Analyzer to establish a baseline. Then you can modify the statistics to simulate a data scale-up factor increase scenario. Then run SPA again to identify the high risk SQLs (this can be done periodically in the production environment to monitor the performance as it will not use any DML operations and therefore keep the data intact)

Contact address:

Mughees Minhas
Oracle
400 Oracle Parkway
Redwood City, CA 94065
USA

Phone:	+1 650 506 8661
Email	mughees.minhas@oracle.com
Internet:	http://www.oracle.com